Documentation 0.1

Fedora Multiboot Guide

Booting Fedora and other operating systems.

Fedora Documentation Project

Abstract

The Fedora Multiboot Guide explains using Fedora to enable booting of one or more additional operating systems. The Guide covers the basic principles of GRUB, Fedora's bootloader, and demonstrates configuration of GRUB on both BIOS and UEFI systems.

1. Introduction

Fedora 21 can be used as the only operating system on your computer, or you can *dual boot* Fedora with another operating system. Fedora can enable selection of other options at boot, often with little to no user configuration. Systems that can boot more than two operating systems are referred to as *multiboot* systems.

Installing Fedora on a system where another operating system is installed requires *unallocated* drive space. *Section 5, “Making Room for Fedora”* explains options for creating this free space.

When your system starts, it first performs self tests, then loads a bootloader, **GRUB**. GRUB provides a menu so you can select the Fedora kernel or other operating system you would like to boot, and GRUBs configuration file stores options and settings required for initial booting of the selected operating system. Read *Section 6, “The GRUB Bootloader”* for more information on GRUB.

Changing Fedora's boot options can be helpful, especially when troubleshooting. *Section 6.4, “Configuring the GRUB Bootloader”* explains how to change these options once or permanently.

The system's firmware allows basic configuration, performs initial startup, initializes hardware, and brings up the bootloader. Until recently, this firmware was referred to as the *BIOS*, or **`Basic Input Output System`**, which is described in *Section 2, “Bootloader Basics”*. Newer systems, such as those sold with Windows 8, use a newer type called *UEFI* or **`Universal Extensible Firmware Interface`**, which is described in *Section 2, “Bootloader Basics”*

In most cases, the Fedora installer will recognize other operating systems on your computer and create boot menu entries for them. If an operating system such as Windows is installed *after* Fedora, GRUB may be overwritten and require reinstallation. Some circumstances, such as missing menu entries, require refreshing of the menu entries. These tasks are explained in *Section 7, “Multiboot on BIOS Systems”* or *Section 8, “Multiboot on UEFI Systems”*

The Guide also covers concerns regarding booting of specific operating systems. For information regarding booting of Fedora with Windows 8, refer to *Section 8.2, “Booting on Windows Systems”*. To learn about using Fedora on Apple hardware, read *Section 8.3, “Booting on Apple systems”*

Using Root Safely

Many of the procedures in this guide involve editing of files or execution of commands that can only be performed as `root`. The `root` account should be used with educated caution and *only when required*. Your system can break or fail to boot if the account is misused.

If your normal user account is configured as an administrator, you can use **sudo** to enter your user password and execute privileged commands:

```
[fedorauser@localhost ~]$ sudo <command>
[sudo] password for fedorauser:
```

You can also open a root shell using the root password **su**. The hyphen (-) in the example is important because it ensures that you will work in a root environment, and not act on normal user files with root privileges.

```
[fedorauser@localhost ~]$ su -
Password:
[root@localhost ~]#
```

The **`root prompt`** is always a hash (#) and a normal user prompt is always a dollar sign ($). Example commands in this guide will include one of these prompts to show the privileges required for the action.

To keep your system healthy and safe, do not execute user applications as root or log into a desktop environment as root.

2. Bootloader Basics

To boot a modern operating system, a computer must identify the kernel, storage, and various options at the time of booting. Software called a bootloader keeps track of these parameters, manages settings for different operating systems, and loads the kernel.

Bootloaders on BIOS systems

Historically, bootloaders have been installed into the first part of a drive, known as the Master Boot Record (MBR) or Boot Sector. The computer's firmware, or BIOS, would check the boot sector for bootable code on startup, and load whatever it found. As bootloaders became more complex, and therefore required more space, the MBR came to contain only a pointer to the second stage of the bootloader.

The second stage of the bootloader performs the actual work. Because the BIOS does not know how to open and read from filesystems, the bootloader finds the operating system, presents a menu, interacts with the user as required, and launches the OS. It was installed in the part of the drive between the MBR and the beginning of the first filesystem, known as the MBR Gap. Less featureful bootloaders could also be installed in the gap between the start of the partition and the beginning of the partition's filesystem, but because this can damage the filesystem if the gap is too small the practice is no longer supported.

BIOS is short for Basic Input Output System, and the software is indeed very simple. BIOS systems had a number of technical limitations, such as being unable to boot from disks larger than 2 TB and extremely limited interfaces. GUID Partition Table (GPT) partitioning schemes also came with larger drives, overcoming the MBR partitioning scheme's 4 partition limit. To overcome these issues and add new features, a newer firmware implementation was developed, called the Unified Extensible Firmware Interface.

Bootloaders on EFI systems

Instead of reading the bootloader from the drive, UEFI systems store information about available bootloaders right on the system firmware. Each boot entry is persistently stored on the firmware, and describes the location of the entry's bootloader.

The bootloaders themselves, and sometimes small applications such as memory testing utilites, are stored on the UEFI system partition. This partition contains a FAT filesystem, has a standardized partition identifier, and is mounted at **`/boot/efi`**. Each operating system places files required for booting in a dedicated directory of this partition, and the boot entry in the firmware points to these files. UEFI systems will often also support BIOS style booting for compatibility reasons, using a feature called the Compatibility Support Module (CSM).

Bootloader data

On Linux systems, the data used by the bootloader is traditionally contained in a `boot partition`, mounted at and known as **`/boot`**. The boot partition contains the kernel, a read only filesystem that holds tools used by the kernel during bootup, called an `initramfs`, and files for the menus and for the bootloader itself. These files were traditionally placed on a different partition because the bootloader did not support complex storage arrangements, and could not read the kernel and initramfs from them. Because a simple, separate boot also allows for easier disaster recovery, the practice has continued to this day.

Fedora supports both UEFI and BIOS systems, using the GRUB bootloader, which is short for **`GNU GRand Unified Bootloader`**. GRUB provides a boot menu and support for many filesystems, as well as software for scanning the system for available operating systems and adding them to the menu.

3. Frequently Asked Questions

Q: Can I install Fedora on my computer, and keep Windows?

A: Yes! Fedora can coexist with Windows, other Linux distributions, and more. Fedora provides a menu that lets you choose the operating system to use when you turn on your computer.

Q: Can I access my files from Windows from Fedora? What about accessing my Fedora files from Windows?

A: You can easily access your Windows files from Fedora. The **Files** application in Fedora Workstation, and most other graphical file browsers, will show Windows NTFS volumes for you to browse.

Unlike Fedora, Windows does not have native support for most filesystems. There are third party drivers available that allow Windows to read some filesystems, like ext3 and ext4. Windows does not currently support virtual block devices such as LVM, which is used for Fedora installations unless you specify otherwise.

Q: Can I replace my unsupported Windows XP installation with Fedora?

A: Of course! Fedora releases a new version roughly every six months, so keeping up with Fedora means you'll have a secure operating system with the latest open source software.

If your computer came with Windows XP, try out alternative desktop environments from *https://spins.fedoraproject.org*. Fedora has something to offer for older, less powerful computers that might be more suitable for you.

Q: Can I install my favorite software on Fedora?

A: Fedora is more than just an operating system, it's a robust ecosystem of open source software. Word processing applications, spreadsheet programs, games, music and movie players, image viewers and editors, and email clients are just the beginning of the software Fedora provides.

Some of the software will be the same as on Windows, like Firefox. Some, like LibreOffice Writer or Calc, do the job you're looking for in a familiar way. Fedora's repositories offer solutions for almost every computing task, but it might not do it with the same program you're used to.

Q: Should I install Windows first, or Fedora?

A: It's best to install Windows first. Fedora should detect your Windows installation, and set up a bootloader entry for it. Windows will overwrite the Fedora bootloader, and does not set up a menu option for Fedora.

On newer computers, this usually isn't as important as it was in the past. The system doesn't need to rely on the bootloader from one operating system or another, because they have information about each installed operating system stored in the UEFI firmware. You can simply choose your OS from the system boot menu.

Q: Can I have Windows installed in UEFI mode (the default for Windows 8 systems) and disable it to install Fedora?

A: No, you should not do that. UEFI's job is to boot operating systems, and if you disable that, you will also disable the ability to boot OSes that need it.

You should be consistent about firmware settings when multibooting. If you have a UEFI system, use it for every OS you install. Fedora supports UEFI systems, with or without SecureBoot. If you use backwards compatibility settings to emulate BIOS installations, you should do so for every OS you install.

Q: Can I use universal USB creation tools to create Fedora installation media?

A: In short, no.

To fully answer that, you have to understand what these tools do. In times past, Linux distributions produced ISO images in a format that made them work when burned to an optical disc. The same format doesn't work when the image is directly written to a USB drives, so tools were created to modify the images for use on USB sticks. Typically, this involved transferring the contents of the image to the USB drive, then installing a **syslinux** bootloader to the drive and configuring it to boot the drive's new contents.

Today, Fedora images are created in a **`hybrid`** format that's directly usable as an optical disk or USB disk, for both legacy and UEFI systems. A **syslinux** bootloader alone isn't compatible with UEFI booting. When the universal installation tools replace the hybrid booting configuration with **syslinux**, the image can't be used on UEFI systems. You end up with a USB drive that's only bootable in legacy mode, which conflicts with dual booting of other operating systems installed in UEFI mode, such as any laptop preinstalled with Windows 8.

For best results, follow the instructions in the *Fedora Installation Guide*[1] for a **`direct write`** method of creating USB media.

Q: The Fedora installer says I don't have enough free space, but Windows says my **`C:`** drive has plenty. What's going on?

A: Fedora needs it's own space, it cannot use free space on **`C:`**. You can use the installer to resize existing partitions and make room for Fedora.

4. Is your system UEFI or BIOS?

Because commands and file locations differ between BIOS and UEFI systems, it is important to identify which you have before attempting advanced boot configuration.

Identifying a BIOS system

- Older computers are more likely to use BIOS. UEFI systems did not become commonplace until after 2010.
- 32 bit systems are almost always BIOS. (UEFI booting of 32 bit systems is not supported by Fedora)
- Your computer originally shipped with Windows Vista or XP.
- The computer's manual and the system setup menu do not mention UEFI, EFI, or SecureBoot.

The term `BIOS` is still used with UEFI systems

Because BIOS systems have been around for so long, the term **`BIOS`** is often used to describe UEFI systems as well. Manufacturers might list system firmware updates as **`BIOS updates`** or provide directions to **`enter the BIOS setup menu`**. The word has come to represent the pre-OS menu on your computer as much as the actual software, but your system may still be UEFI capable.

To avoid confusion, this guide uses the term **`firmware`** when describing interactions with BIOS or UEFI menus.

Identifying a UEFI system

- Newer systems are more likely to use UEFI. If you bought your computer new in 2013 or after, it probably has UEFI.

[1] https://docs.fedoraproject.org/install-guide

- Your computer shipped with Windows 8. The terms of service for Windows 8 *require* SecureBoot, a UEFI feature.
- Your system setup menu has a graphical interface or mouse support. UEFI menus can be more elaborate.
- The system's boot menu gives you the option of booting media via UEFI, or has boot options describing operating systems instead of just physical drives. The boot order menu might look like this:

```
UEFI: Generic USB Stick
Generic USB Stick
UEFI: DVD-RW Drive
DVD-RW Drive
Fedora
Windows
```

5. Making Room for Fedora

Fedora requires a volume of *unallocated* storage space for installation. To make room for Fedora, both the existing filesystem and the partition it resides on must be resized.

Be cautious when resizing!

Resizing partitions and filesystems is always a potentially destructive endeavor. While the methods described in this section are consistently reliable when properly executed, you should always be prepared for problems.

Before installing a new operating system on your computer, make sure you have current backups of any crucial data.

The advanced methods described in this section offer a greater degree of flexibility and control, but are more difficult to implement. You should resize partitions using the original operating system's native utilities, or using the Fedora installer, unless you are confident that you understand the advanced methods.

5.1. Using the Fedora Installer to resize partitions

The Fedora installer provides a guided graphical method for resizing partitions and reclaiming space. This method is simpler and easier than the others discussed in this section. For most simple dual boot situations, using the installer to reclaim space will be the most expedient and foolproof method.

Resizing partitions with the Fedora installer is covered in the *Fedora Installation Guide*[2]

5.2. Resizing an NTFS filesystem the command line

This section explains using tools provided by Fedora to resize an NTFS partition. A terminal window from a liveCD can be used.

[2] http://docs.fedoraproject.org/en-US/Fedora/20/html/Installation_Guide/reclaim_space-x86.html

Disable Windows fast reboot feature!

Newer versions of Windows use a strategy of suspending some system processes to disk to speed startup times. Modifying a filesystem in this state can corrupt or damage the data, so the linux NTFS drivers will not mount it. If the filesystem were to be mounted, the cached view if the files may differ from the changes you made in Fedora, with the most likely result of loosing your changes - or worse.

Before attempting any operation on an NTFS volume from Fedora, make sure you have disabled this feature in Windows.

Procedure 1. Selecting and resizing a partition.

1. Show available filesystems:

```
# blkid
/dev/sda1: SEC_TYPE="msdos" UUID="32AE-E651" TYPE="vfat" PARTLABEL="EFI System
 Partition" PARTUUID="0315942d-8c2c-414f-a560-cfa499494a72"
/dev/sda2: UUID="593153ae-2b67-4a5b-9efa-fa3954953abd" TYPE="ext4"
 PARTUUID="68cadad8-6de2-4ef7-96ff-f58e5114fdcc"
/dev/sda3: UUID="P2xKTQ-aQWG-z2Uv-jSw5-kkUK-SN5Q-cNf3PI" TYPE="LVM2_member"
 PARTUUID="7ba7ed40-b43f-4e71-b83e-51629bf7db47"
/dev/sda5: UUID="44B6BAD1B6BAC2AA" TYPE="ntfs" PARTUUID="9c62d1dc-
dedd-4d4c-9728-5f2ef69f2b42"
```

2. `ntfs` partitions indicate the existing Windows installation. Examine the partitions on the **`/dev/sda`** drive.

```
# parted /dev/sda print
Model: ATA ST9320328CS (scsi)
Disk /dev/sda: 320GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags: pmbr_boot

Number  Start   End    Size    File system  Name                          Flags
 1      1049kB  211MB  210MB   fat16        EFI System Partition          boot
 2      211MB   735MB  524MB   ext4
 3      735MB   119GB  118GB
 4      119GB   119GB  134MB                Microsoft reserved partition  msftres
 5      119GB   215GB  96.2GB  ntfs
```

3. Partition number 5 is a large filesystem, and probably has some free space to share. Check how much of that space is in use.

```
# ntfsresize --info /dev/sda5
ntfsresize v2013.1.13 (libntfs-3g)
Device name        : /dev/sda5
NTFS volume version: 3.1
Cluster size       : 4096 bytes
Current volume size: 96234107392 bytes (96235 MB)
Current device size: 96234110976 bytes (96235 MB)
Checking filesystem consistency ...
```

```
100.00 percent completed
Accounting clusters ...
Space in use       : 29222 MB (30.4%)
Collecting resizing constraints ...
You might resize at 29221265408 bytes or 29222 MB (freeing 67013 MB).
Please make a test run using both the -n and -s options before real resizing!
```

4. Resize the filesystem. Make sure to balance the available space, despite the suggestion from **ntfsresize** to shrink to the bare minimum. A very full filesystem can cause problems for any operating system.

```
# ntfsresize --size 40G /dev/sda5
ntfsresize v2013.1.13 (libntfs-3g)
Device name        : /dev/sda5
NTFS volume version: 3.1
Cluster size       : 4096 bytes
Current volume size: 96234107392 bytes (96235 MB)
Current device size: 96234110976 bytes (96235 MB)
New volume size    : 39999996416 bytes (40000 MB)
Checking filesystem consistency ...
100.00 percent completed
Accounting clusters ...
Space in use       : 29222 MB (30.4%)
Collecting resizing constraints ...
Needed relocations : 0 (0 MB)
WARNING: Every sanity check passed and only the dangerous operations left.
Make sure that important data has been backed up! Power outage or computer
crash may result major data loss!
Are you sure you want to proceed (y/[n])? y
Schedule chkdsk for NTFS consistency check at Windows boot time ...
Resetting $LogFile ... (this might take a while)
Updating $BadClust file ...
Updating $Bitmap file ...
Updating Boot record ...
Syncing device ...
Successfully resized NTFS on device '/dev/sda5'.
You can go on to shrink the device for example with Linux fdisk.
IMPORTANT: When recreating the partition, make sure that you
  1)  create it at the same disk sector (use sector as the unit!)
  2)  create it with the same partition type (usually 7, HPFS/NTFS)
  3)  do not make it smaller than the new NTFS filesystem size
  4)  set the bootable flag for the partition if it existed before
Otherwise you won't be able to access NTFS or can't boot from the disk!
If you make a mistake and don't have a partition table backup then you
can recover the partition table by TestDisk or Parted's rescue mode.
```

Procedure 2. Resizing the partition

1. To make sure that the partition is recreated accurately, work using sectors as units instead of bytes. The first figure we need is the sector the filesystem starts on.

```
# parted /dev/sda unit s print
Model: ATA ST9320328CS (scsi)
Disk /dev/sda: 625142448s
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags: pmbr_boot

Number  Start       End         Size        File system  Name
 Flags
 1      2048s       411647s     409600s     fat16        EFI System Partition
 boot
 2      411648s     1435647s     1024000s    ext4
 3      1435648s    232122367s   230686720s
 4      232122368s  232384511s   262144s                  Microsoft reserved partition
 msftres
 5      232384512s  420341759s   187957248s  ntfs
 6      420341760s  420343807s   2048s
 bios_grub
 7      420343808s  421367807s   1024000s    ext4
 8      421367808s  625141759s   203773952s
```

The partition starts on sector **`232384512`**, and the sector size is **`512 bytes`**

2. Calculate the size of the filesystem in sectors, and the end of the new partition
   - After resizing, the filesystem is marked as *dirty* so it will be checked on the next Windows boot. Clear this flag so that we can run further commands.

```
#ntfsfix -d /dev/sda5
Mounting volume... OK
Processing of $MFT and $MFTMirr completed successfully.
Checking the alternate boot sector... OK
NTFS volume version is 3.1.
NTFS partition /dev/sda5 was processed successfully.
```

   - Find the cluster size and number of clusters.

```
# ntfsinfo -m /dev/sda5|grep Cluster
    Cluster Size: 4096
    Volume Size in Clusters: 9765624
    Compression Block Clusters: 0
    Free Clusters: 2631945 (27.0%)
```

   - Calculate the size in sectors of the new filesystem.

Equation 1. Filesystem size in sectors

4096 bytes per cluster × 9765624 total clusters ÷ 512 bytes per sector = 78124992 sectors

   - Find the end of the new partition.

Equation 2. End sector of new partition

232384512 start + 78124992 filesystem = 310509504 end sector

Adding onto the end sector value costs little free space and provides a margin of error. Consider padding the end of your partition for best results.

3. Use parted to resize the `ntfs` filesystem on **`/dev/sda`**. Using the **`s`** after the number ensures **`parted`** uses sectors as the unit.

```
# parted /dev/sda
GNU Parted 3.1
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
(parted) rm 5
(parted) mkpart
Partition name?  []? windowsdisk
File system type?  [ext2]? ntfs
Start? 232384512s
```

```
End? 310509504s
(parted) quit
```

4. Check the partition table to confirm the free space is available, and exit **`parted`**.

```
(parted) print free
Model: ATA ST9320328CS (scsi)
Disk /dev/sda: 320GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags: pmbr_boot

Number  Start   End     Size    File system  Name                          Flags
        17.4kB  1049kB  1031kB  Free Space
 1      1049kB  211MB   210MB   fat16        EFI System Partition          boot
 2      211MB   735MB   524MB   ext4
 3      735MB   119GB   118GB
 4      119GB   119GB   134MB                Microsoft reserved partition  msftres
 5      119GB   159GB   40.0GB  ntfs
        159GB   215GB   56.2GB  Free Space

        (parted) quit
```

6. The GRUB Bootloader

6.1. GRUB Basics

Fedora uses the GRUB bootloader. Short for GNU *GR*and *U*nified *B*ootloader. GRUB is a powerful and versatile bootloader that meets most needs. For other platforms such as IBM System Z or cloud images, Fedora may use other bootloaders, but because they are not likely to multiboot, this document does not address them.

The primary configuration file for GRUB describes the menu presented to the user, including the options passed to each operating system definition it boots. It also contains options for GRUB itself, such as the default option, timeout before continuing to the default option, and the visual presentation of the menu.

On BIOS systems, this file is located on the boot partition, as **`/boot/grub2/grub.cfg`**.

On UEFI systems, this file is located on the UEFI system partition, as **`/boot/efi/EFI/fedora/grub.cfg`**.

Do not edit GRUB configurations directly!

Any changes made directly to the GRUB configuration files will be overwritten when the file is regenerated, such as after a kernel update. Instead of editing these files directly, edit the templates that are used to generate them. This process is covered in *Section 6.4, “Configuring the GRUB Bootloader”*

GRUB can identify and directly boot most Unix type operating systems. You only need one GRUB configuration to boot them all, but each should have a dedicated **`/boot`** partition. Operating systems such as Windows cannot be booted directly, but GRUB can identify and chainload them.

6.2. Changing GRUB entries at boot

Changing selection or adding arguments to a GRUB menu entry at boot time can be useful for testing or troubleshooting purposes. You might have a graphics issue to troubleshoot by booting temporarily into a non-graphical **`target`**, a hardware issue to work around with a `kernel parameter`, or a problem to resolve by selecting an alternate kernel. This section covers the procedure for adding boot arguments, see *Section 9, “Boot Options”* for some arguments you might use.

The Rescue option

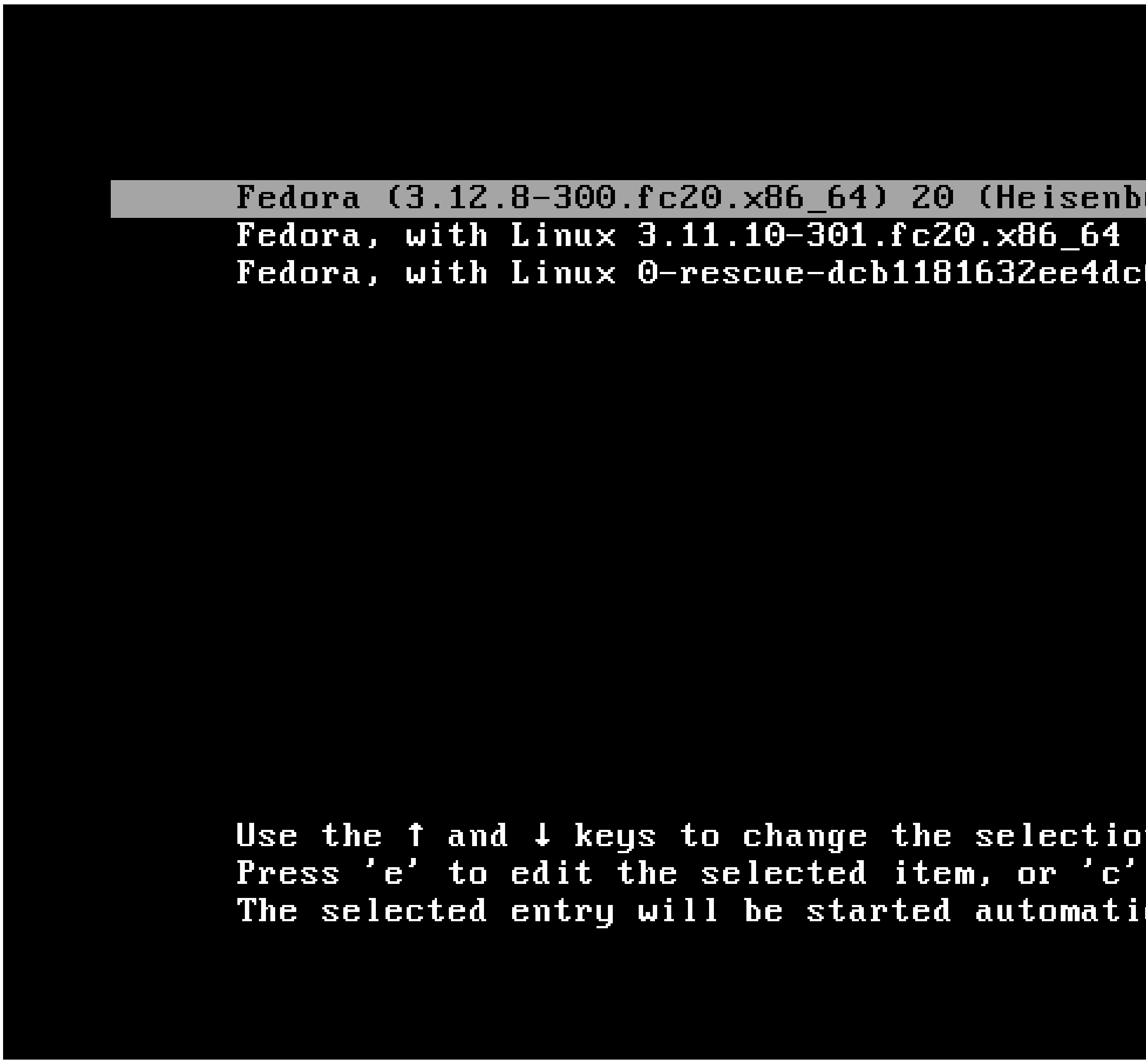
The normal boot entries for Fedora use a trimmed down `initramfs`, or initial boot filesystem. This **`host-only`** `initramfs` had information only about the hardware on your computer, allowing the system to boot more quickly by avoid loading unneeded drivers.

One of the default menu entries for Fedora is a **`rescue`** entry. This will load the same Fedora environment as the other entries, but loads a full `initramfs`.

The rescue option is useful if you have problems after adding new hardware. You can use the rescue option to regenerate the `initramfs` for all boot entries using this command:

```
# dracut --regenerate-all --force
```

1. Turn on or reboot your computer. When the GRUB menu (*Figure 1, “GRUB menu during countdown”*) appears, press the **`Esc`** key to stop the countdown.



```
Fedora (3.12.8-300.fc20.x86_64) 20 (Heisenb
Fedora, with Linux 3.11.10-301.fc20.x86_64
Fedora, with Linux 0-rescue-dcb1181632ee4dc

Use the ↑ and ↓ keys to change the selectio
Press 'e' to edit the selected item, or 'c'
The selected entry will be started automati
```

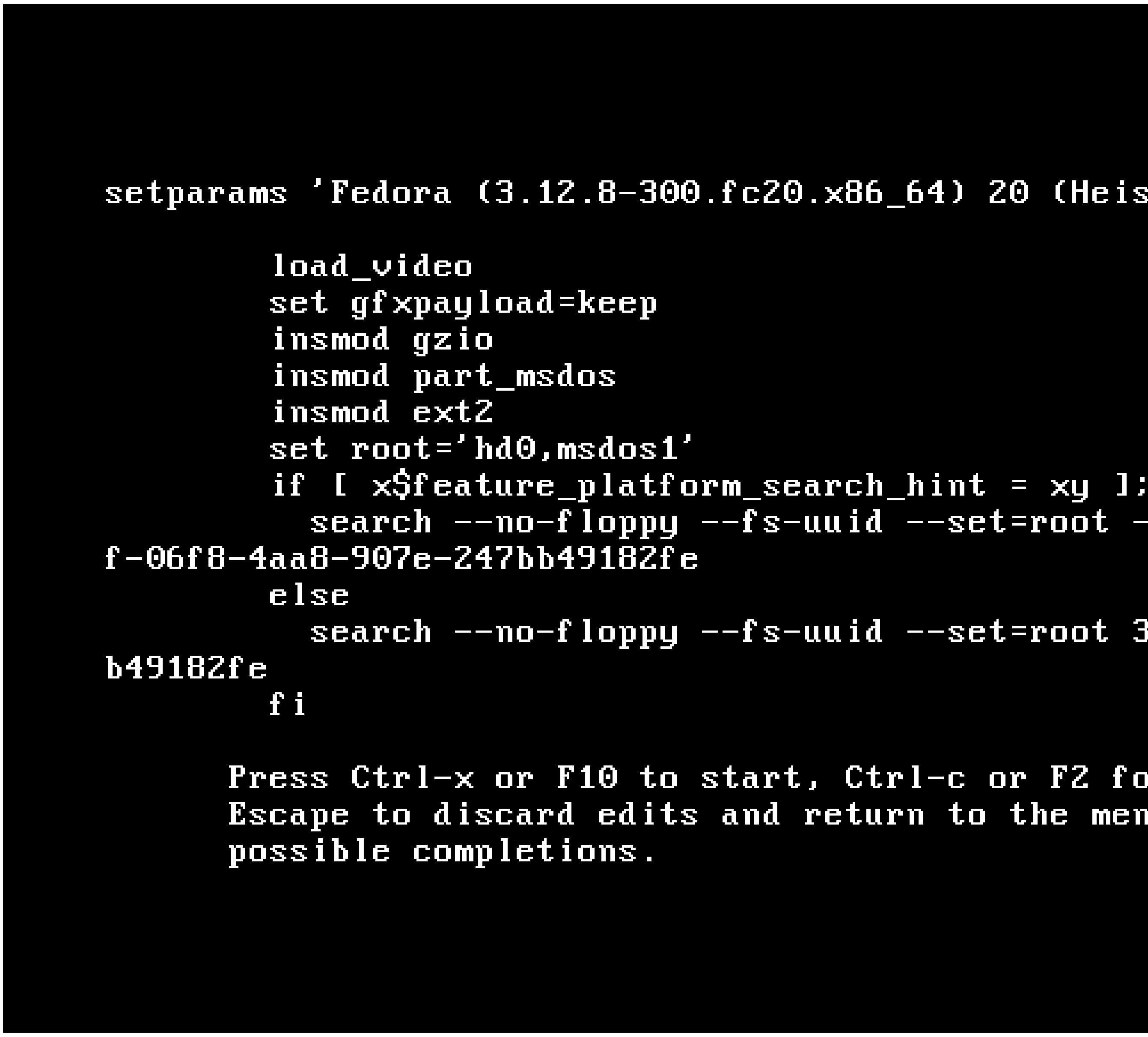
Figure 1. GRUB menu during countdown

2. Use the arrow keys to highlight the desired boot entry.

   a. If you want to boot an alternate menu entry without editing, press **`Enter`** to continue.

   b. If you want to edit the entry before booting, press **`e`** to edit.

3. a. The initial screen displayed for editing shows information GRUB needs to find and boot the operating system, as pictured in *Figure 2, “The GRUB edit screen, Part 1”*. These lines should not be changed.

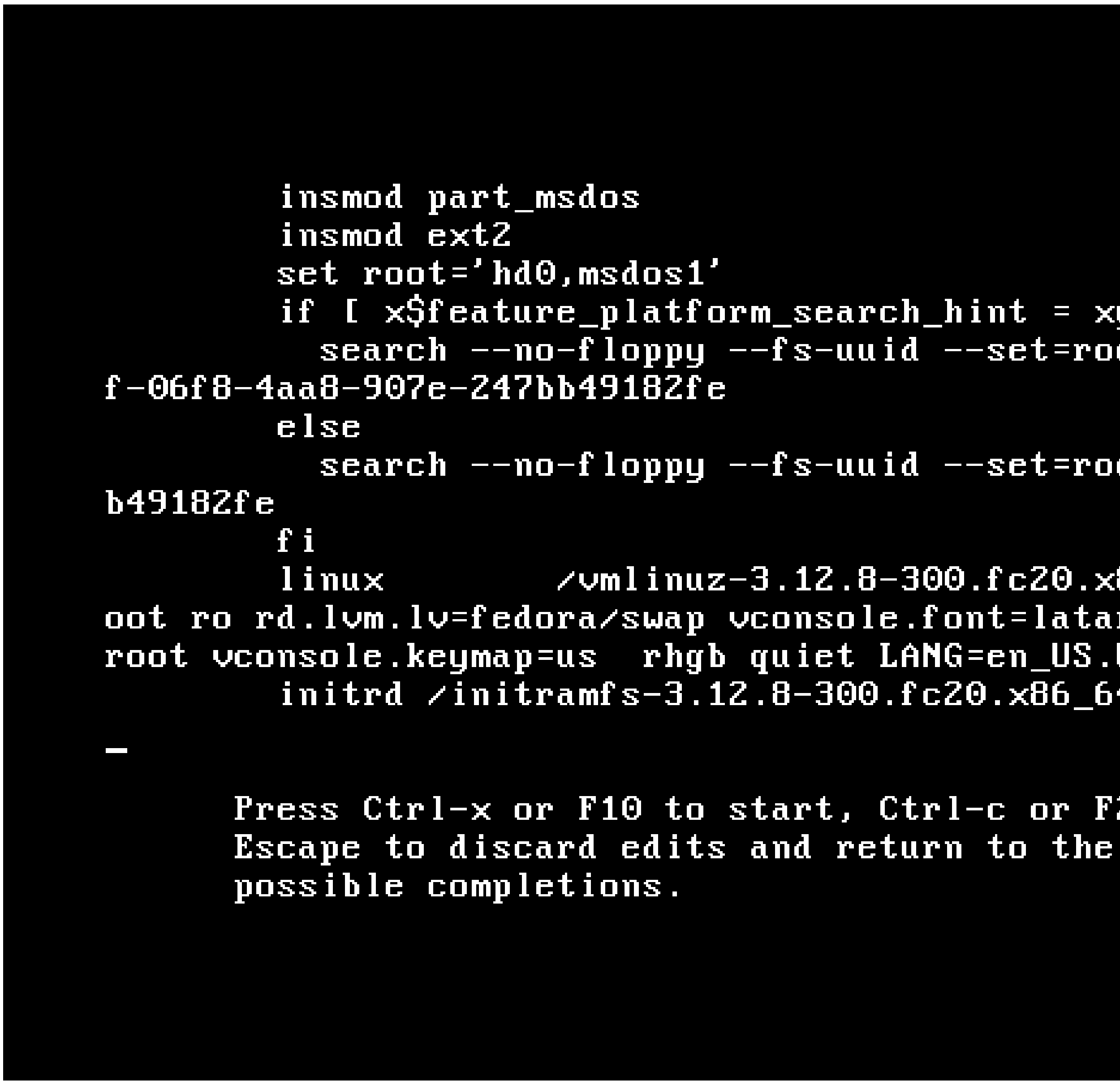


```
setparams 'Fedora (3.12.8-300.fc20.x86_64) 20 (Heis

        load_video
        set gfxpayload=keep
        insmod gzio
        insmod part_msdos
        insmod ext2
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = xy ];
          search --no-floppy --fs-uuid --set=root -
f-06f8-4aa8-907e-247bb49182fe
        else
          search --no-floppy --fs-uuid --set=root 3
b49182fe
        fi

      Press Ctrl-x or F10 to start, Ctrl-c or F2 fo
      Escape to discard edits and return to men
      possible completions.
```

Figure 2. The GRUB edit screen, Part 1

b. Using the arrow keys, move down to the line that contains the boot arguments. On UEFI systems, this line will begin with *`linuxefi`*, or on BIOS systems the line will begin with *`linux`*. The next argument on the line will typically be the kernel, a string beginning with **`/vmlinuz-`**. This is shown in *Figure 3, “The GRUB edit screen, Part 1”*.

Press the **End** key to move to the end of the line.



```
        insmod part_msdos
        insmod ext2
        set root='hd0,msdos1'
        if [ x$feature_platform_search_hint = x
          search --no-floppy --fs-uuid --set=ro
f-06f8-4aa8-907e-247bb49182fe
        else
          search --no-floppy --fs-uuid --set=ro
b49182fe
        fi
        linux        /vmlinuz-3.12.8-300.fc20.x
oot ro rd.lvm.lv=fedora/swap vconsole.font=lata
root vconsole.keymap=us  rhgb quiet LANG=en_US.
        initrd /initramfs-3.12.8-300.fc20.x86_6
_

      Press Ctrl-x or F10 to start, Ctrl-c or F
      Escape to discard edits and return to the
      possible completions.
```

Figure 3. The GRUB edit screen, Part 1

Getting more information during boot

The default boot parameters for Fedora include *`rghb`* and *`quiet`*. These enable the boot splash screen instead of showing details about services, mountpoints, and other units as the sytem boots. If you are troubleshooting a boot issue or want to see more information when Fedora loads, removing these parameters will disable the splash screen.

4. Type in your desired parameters and press **`Ctrl`**+**`x`** to boot. See *Section 9, “Boot Options”* for some parameters you might find useful.

6.3. Recreating GRUB

This section describes the process for reinstalling the GRUB bootloader and recreating the configuration. Unlike legacy GRUB, which required users to manually create entries, GRUB2 will scan the system for bootable systems and automatically create the menu configuration. There are a number of reasons to reinstall GRUB:

The boot menu does not include an available operating systems.
The bootloader has been overwritten during installation of Windows or another operating system.
You have chosen not to use **anaconda** to install grub.

6.3.1. Refreshing GRUB configuration

To recreate the GRUB configuration from a booted system, use the **`grub2-mkconfig`** utility. The program will scan your system and create menu entries for what it finds. You should also use **`grub2-mkconfig`** after customizing menu entries, a process described in *Section 6.4, "Configuring the GRUB Bootloader"* If Fedora does not boot because of an invalid GRUB configuration, you can perform this procedure to repair it after following the instructions in *Section A.1, "Setting up a chroot from a live image"*.

Refresh GRUB configuration on a BIOS system:

```
# grub2-mkconfig -o /boot/grub2/grub.cfg
```

Refresh GRUB configuration on UEFI systems:

```
#grub2-mkconfig -o /boot/efi/EFI/fedora/grub.cfg
```

In most cases, this operation will detect any operating system available on your computer and create entries for them in the GRUB menu. Actually reinstalling the GRUB bootloader isn't required, just reconfiguring.

6.3.2. Reinstalling GRUB

In some cases, you may need to reinstall the GRUB bootloader itself. It might be because another operating system has overwritten GRUB, or because of some inadvertent action. This section explains how to do that, but if you do have a GRUB menu or GRUB prompt, you probably don't need to perform the operations. This section assumes that you have tried all of the firmware boot options, and are sure you do not have a functioning GRUB bootloader on your system.

Be careful not to mix up the different instructions for UEFI and legacy systems. Installing the wrong implementation of GRUB can cause problems that are difficult to resolve

6.3.2.1. Reinstalling GRUB on UEFI systems.

Procedure 3. Reinstalling GRUB on UEFI systems.

1. Enter your installed system in a chroot using the instructions in *Section A.1, "Setting up a chroot from a live image"*. Be sure to boot the live image in UEFI mode.

2. Reinstall the packages that provide bootloader files. This will recreate any Fedora files missing from the EFI system partition.

```
# yum reinstall grub2-efi shim
```

3. Optionally, recreate the firmware boot entry. You only need to do this if the Fedora entry is missing, so check to see if the Fedora entry is present before continuing.

   a. Check the existing firmware boot entries.

   ```
   # efibootmgr -v
    BootCurrent: 0015 Timeout: 1 seconds BootOrder: 0015,0000 Boot0000* Windows
    Boot Manager HD(3,2e9a5000,32000,0d13443c-6bf1-4952-960c-c05ba2b3fd8c)File(\EFI
   \Microsoft\Boot\bootmgfw.efi)WINDOWS.........x...B.C.D.O.B.J.E.C.T.=.
   {.9.d.e.a.8.6.2.c.-.5.c.d.d.-.4.e.7.0.-.a.c.c.1.-.f.3.2.b.3.4.4.d.4.7.9.5.}...a...............
    Boot0015* Fedora HD(1,800,64000,211be689-9d4f-4034-bbc9-4e03372165db)File(\EFI
   \fedora\grubx64.efi) Boot0018* SATA : PORT 4 : HL-DT-ST BD-RE WH14NS40
    BIOS(3,0,00)AMBO Boot0019* SATA : PORT 6G 0 : ST31500341AS : PART 0 : Boot Drive
    BIOS(2,0,00)AMBO
   ```

   b. If the Fedora entry is missing, check the installation logs to get the command to put it back.

   ```
   # grep efibootmgr /var/log/anaconda/anaconda.program.log
    05:43:07,548 INFO program: Running... efibootmgr 05:43:07,566 INFO program:
    Running... efibootmgr -c -w -L Fedora -d /dev/sda -p 2 -l \EFI\fedora\shim.efi
   ```

   c. Run the **`efibootmgr`** invocation again. If needed, adjust the arguments to fit your EFI system partition's location. In this example, **`-d /dev/sda`** places that partition on the **`/dev/sda`**, and **`-p 2`** designates the second partition on the drive.

   ```
   # efibootmgr -c -w -L Fedora -d /dev/sda -p 2 -l \EFI\fedora\shim.efi
   ```

4. Exit the chroot and reboot your system into Fedora.

6.3.2.2. Reinstalling GRUB on BIOS systems.

6.4. Configuring the GRUB Bootloader

GRUB configuration files can be found in several places:

`/etc/default/grub`
Configuration for GRUB itself is defined in **`/etc/default/grub`**. The default options used when creating new entries for Fedora can also be found here.

`/etc/grub.d/`
Files in **`/etc/grub.d`** are used as templates for creating new GRUB entries as well as custom boot entries.

`/boot`
The **`/boot`** directory contains Fedora's kernel and `initramfs`.

On BIOS systems, this directory also contains the configuration for GRUB itself.

`/boot/efi/EFI`
This directory is only found on UEFI systems, and is within the UEFI System Partition. Both GRUB configuration files and the GRUB executable can be found in **`/boot/efi/EFI`**.

6.4.1. Permanently adding to Fedora boot entries

The options that GRUB passes to Fedora when booting are generated from the discovered filesystems and from the value of *`GRUB_CMDLINE_LINUX`* in **`/etc/default/grub`**.

Adding a parameter to the end of *`GRUB_CMDLINE_LINUX`* and regenerating **`grub.cfg`** will apply the parameter to all current and future Fedora boot entries.

Example 1. In practice: Fixing backlight issues with kernel parameters

Some laptops have problems with the display backlight using the default configuration. The screen might be too bright, too dim, flicker, or even appear to be completely black.

The issue can often be resolved by directing the system to prefer vendor-specific drivers for the backlight using the *`acpi_backlight=vendor`* parameter.

1. With root permissions, open the file **`/etc/default/grub`**.

   ```
   # nano /etc/default/grub
    GRUB_TIMEOUT=0 GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
    GRUB_DEFAULT=saved GRUB_DISABLE_SUBMENU=true GRUB_TERMINAL_OUTPUT="console"
    GRUB_CMDLINE_LINUX="vconsole.font=latarcyrheb-sun16 $([ -x /usr/sbin/rhcrashkernel-
   param ] && /usr/sbin/rhcrashkernel-param || :) vconsole.keymap=us rhgb quiet"
    GRUB_DISABLE_RECOVERY="true"
   ```

2. Add the parameter to the end of *`GRUB_CMDLINE_LINUX`* and save the file.

   ```
    GRUB_TIMEOUT=0 GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
    GRUB_DEFAULT=saved GRUB_DISABLE_SUBMENU=true GRUB_TERMINAL_OUTPUT="console"
    GRUB_CMDLINE_LINUX="vconsole.font=latarcyrheb-sun16 $([ -x /usr/sbin/rhcrashkernel-
   param ] && /usr/sbin/rhcrashkernel-param || :) vconsole.keymap=us rhgb quiet
    acpi_backlight=vendor" GRUB_DISABLE_RECOVERY="true"
   ```

3. Regenerate the GRUB configuration to apply the new changes.

   a. For BIOS systems:

      ```
      # grub2-mkconfig -o /boot/grub/grub2.cfg
      ```

   b. For UEFI systems:

      ```
      # grub2-mkconfig -o /boot/efi/EFI/fedora/grub.cfg
      ```

7. Multiboot on BIOS Systems

7.1. BIOS Basics

8. Multiboot on UEFI Systems

8.1. UEFI Basics

This section of the guide covers the UEFI boot process in general and how Fedora successfully handles a Secure Boot environment.

We hope that you will find this information both interesting and helpful. With the information outlined in this and other sections of this guide, you will have a better understanding of what is happening 'behind the scenes'. That should prove useful when it comes time to work your way through troubleshooting a particular issue related to booting using UEFI. This guide represents our best effort at explaining a very technical and difficult to understand specification in terms that everyone can understand. For this reason, some parts of the explanation have been over-simplified for ease in understanding the overall concept or process that is taking place. For those that desire more information that what is available in this ~~article~~, extensive information about UEFI is available here *http://www.uefidk.com/learn* and here *http://www.uefi.org/*

8.1.1. EFI Boot Sequence Explained

In this section we will consider two scenarios:

1. How EFI handles a normal boot cycle - no intervention.

2. What happens when you press the appropriate key to enter the EFI Settings prior to a successful boot of an operating system.

First, EFI attempts each entry in the order listed in its BootOrder variable. It will boot the first entry that 'works' with the data listed for that entry. A typical entry is in the format of: ACPI(a0341d0,0)PCI(1f,2)SATA(0,0,0)HD(1,800,64000,12029cda-8961-470d-82ba-aeb17dba91a5) File(\EFI\fedora\shim.efi)

The EFI Boot Manager begins the process of loading the file (end result) by initializing each preceeding device in order. Thus, in the example above, it starts by initializing ACPI(a0248d0,0) which then provides access to PCI(1d,2) which then provides access to SATA(0,0,0) etc.. EFI just goes down the line until it can load the file called for in that entry. If anything in that chain is broken or missing the boot fails. If it cannot boot the first entry in the BootOrder list, it will go to the second, and failing there, will move on to the third entry, etc.

If it gets to the end of the BootOrder list and still has not been able to transfer execution, it will begin to initialize every device connected to the system (fixed and removable) and begins to look specifically for removable devices. Remember, this is what happens automatically -- its what happens if you do not go into the "EFI Settings" and thereby stop the process at the end of initial enumeration of all connected devices. By hitting the 'magic' key, you alter the sequence of events and are, in effect, "Skip the boot process for now and just enumerate everything connected to the motherboard." Full enumeration is not conducted until after the BootOrder sequence has been exhausted during normal boots. Right now, we are looking at what happens automatically if EFI cannot find a bootable device by traversing its BootOrder entries.

As EFI Boot Manger moves on to find a bootable, removable device, it looks for an EFI partition, formatted in Fat32, Fat16 or Fat12 with an \EFI\Boot\ directory structure and that bootx64.efi is in that directory.

When that file is found and loaded, execution is passed to it. In Fedora, that means that bootx64.efi is going to check for the presence of fallback.efi; and if it finds it, will pass execution to it. Fallback.efi will then enumerate all of the boot.csv's it can find in its own partition, create and append an entry for each one to the EFI NVRAM, change the NextBoot variable and pass execution back to the EFI for processing. Having a valid entry, EFI will boot to that device and hand over execution.

At some point, either grubx64.efi or the kernel itself will issue a command to EFI to terminate its boot support processes and standby for a reboot, standby, hibernate or power down command. In effect, we are saying to EFI: "Ok. We got it from here."

Ubuntu, for example, has chosen to make that call just prior to loading the kernel, while Fedora continues to use EFI support services until after the kernel has verified the signatures on all boot files; thus continuing the 'chain of trust' a little further.

In any event, EFI will boot to the first device it can either find on its own, or to the one its told to.

When you enter the "EFI Settings" of your computer and look at the menu, all of that enumeration is complete before the menu is displayed. All you then have to do is select which device you wish to boot from and hit Enter.

8.1.2. What is Default Boot Behavior?"

Default Boot Behavior is an EFI process that is initiated when the EFI boot process cannot find a suitable boot manager or boot loader to pass execution to after traversing the BootOrder list.

It begins by enumerating all removable devices and then passing execution to the first instance of **`bootx64.efi`** it finds.

`bootx64.efi` uses **`fallback.efi`** to scan the entire EFI partition looking for **`boot.csv`** files in each sub-directory within the EFI partition. Everytime it finds one, **`fallback.efi`** creates and appends an entry in the EFI NVRAM. It then changes the BootNext variable to point to the first one it found. When finished, it directs execution back to EFI Boot Manager to boot using the NextBoot variable.

8.1.3. "What is Fallback and How Does it Work?"

The reason this process is referred to as "fallback" is because, as noted above, **`bootx64.efi`** checks to see if a file named **`fallback.efi`** is in the same directory as itself, and if so, will execute it as an EFI application.

It is the **`fallback.efi`** application that enumerates the various **`boot.csv`** files that it finds, creates and appends the EFI NVRAM entries, and finally, passes control back to the EFI Boot Manager to boot the first entry it created; which, as mentioned earlier, it does by changing the BootNext variable.

The Default Boot Behavior, which initiates fallback, happens in the event that any of the following conditions are met:

1. There aren't any existing NVRAM entires for any installed operating systems

2. That the entries that were listed in the BootOrder resulted in a 'no boot' situation from all installed, fixed devices

3. That your removable device settings in your EFI Boot Order Priority are such that removable devices (Live media, etc) are listed above your installed Operating Systems menu entries.

Before moving on, I would like to add a few notes to help clariy the above information:

One caveat to all that has been said about "Default Boot Behavior": If you already have installed Operating Systems on your system that will boot normally, and you want to use "Default Boot Behavior" to boot your removable media, then please read on.

Your EFI implementation has to be 'compliant' and your EFI Boot Settings have to indicate that removable devices are higher in the list than your installed Operating Systems. This allows the EFI to find and boot the removable media (if inserted) prior to attempting to run down the normal BootOrder list.

IF that is the case, then "Default Boot Behavior" will occur on that removable device and it will boot.

IF the implementation of EFI on your system is not compliant it may never get to the point where it loads **`fallback.efi`** - it still might not boot.

The only way to know for sure, is to plug in a USB, boot using the 'magic' key to enter your EFI Settings, change your boot priorites in the EFI to put the inserted USB at the top of the list, save and exit. Leave in the USB and power on without pressing any keys. If it boots to the USB automatically then everything is working as it should. If it won't boot, and/or it is not showing up on the list of devices provided by the EFI Boot Manager, then check that the device has an \EFI\Boot directory with bootx64.efi and fallback.efi in it. If all of that is there, then either the USB is 'bad', the port you plugged it into is 'bad' (something in the device path is 'bad') or your EFI implementaton is non-compliant.

If you do not have any installed OS's on your system... Just plug in removable media that meets the criteria for EFI "Default Boot Behavior" and turn it on.

8.2. Booting on Windows Systems

Fedora is a great choice for all aspects of multibooting your system using UEFI! Fedora has gone to great lengths to ensure that you can install it on many different platforms and in several different environments; including desktops, laptops (with or without Secure Boot enabled) as well as Apple/ Macintosh based computers.

This section of the Multiboot Guide begins by covering a few important definitions as well as the more general aspects of the Windows and Fedora UEFI boot process. Windows and Fedora specific boot files, their purposes, and locations are then listed. After that, this section deals with the similarities and the differences in the UEFI boot process for both operating systems. The final section covers how to use **`BCD`** to dual boot. Please be sure to read the few, but very important caveats to modifying that critical Windows configuration file.

We hope that you will find this information both interesting and helpful. With the information outlined in this, and other, sections of this guide, you will have a better understanding of what is happening 'behind the scenes'. That should prove useful when it comes time to work your way through troubleshooting a particular issue related to booting using UEFI.

This guide represents our best effort at explaining a very technical and difficult to understand specification in terms that everyone can understand. For those that desire more information than what is available in this guide, extensive information about UEFI is available here *http://www.uefidk.com/learn* and here *http://www.uefi.org/*

8.2.1. Boot Managers and Boot Loaders Defined

Knowing the difference between a Boot Manager and Boot Loader is a good start to understanding the UEFI boot process for both Windows and Fedora.

1. **`Boot Manager:`** A utility that allows multiple operating systems to be booted from the same computer

2. **`Boot Loader:`** The program that calls the operating system into memory

As you will see, **`GRUB`*** acts as both a Boot Manager and a Boot Loader

* The word **GRUB** will be used in this section of the guide to indicate all versions of GRUB, including GRUB2, without differentiation.

8.2.2. Windows EFI Partition Files

Microsoft uses **bootmgr.efi** and **bootmgrfw.efi** as Boot Managers. **bootmgr.efi** is the 32 bit version and **bootmgrfw.efi** is the 64 bit version. Both of these files are located in the root directory of the Microsoft-created EFI partition (usually found at **/dev/sda1**). For EFI compatibility, they are both located at **/EFI/Microsoft/Boot/** as well.

Other files found within the **/EFI/Microsoft/Boot** directory structure are the **BCD** (a registry formatted configuration file), sub-directories for language files, a memory test efi application, and the **BCD** log files. The **BOOTSTAT.DAT** file is used to flag whether a recovery mode is required.

Mui stands for 'Multilingual User Interface' and the ***.mui** files located in the **bg-BG** sub-directory are used to provide language support in other-than-english installations.

There doesn't seem to be any publicly available information on what **boot.stl** is or what functions it may perform.

The Windows boot loader, **winload.efi**, is located at **/WINDOWS/System32/Boot** on whatever drive and partition Windows was installed on.

8.2.3. Fedora EFI Partition Files

Within a Fedora-created EFI partition you will find the following files at the designated locations. The purpose of each file is also given:

1. **./EFI/fedora/MokManager.efi**: Mok stands for Machine Owner Key. To launch a locally-compiled kernel, you must sign it with a MOK and register that MOK with the system. You would need to do this, for example, if you recompiled the kernel with third-party kernel drivers, such as those needed by Nvidia's or AMD/ATI's proprietary video drivers. This file manages that process.

2. **./EFI/fedora/gcdx64.efi**: Used to boot live media; called by shim.

3. **./EFI/fedora/grubx64.efi**: The main GRUB executable (signed with Fedora's key for Secure Booting).

4. **./EFI/fedora/shim-fedora.efi**: Performs the signing mechanism for grubx64.efi in a Secure Boot scenario when using a MoK

5. **./EFI/fedora/shim.efi**: This a Microsoft signed stub to allow booting using Secure Boot on 'Windows Ready' certified computers.

8.2.4. Windows Boot Files

Windows 64 bit installations use **bootmgrfw.efi** to load **BCD** and execute **winload.efi**. 32 bit installations use **bootmgr.efi** to initiate the same chain of events.

Windows uses the same boot process whether the Secure Boot mechanism is enabled or not.

On a legacy, BIOS-based, PC the **BCD** is located at **\Boot\BCD** On an EFI firmware-based PC it is located at **\EFI\Microsoft\Boot\**

BCD is a formatted as a registry hive and is therefore totally Windows-centric and proprietary. While **BCD** can be configured to boot other operating systems, it cannot be done without a certain measure of difficulty and an even greater level of un-dependability after reboots.

`BCD` contains configuration data that controls the operation of the Windows boot loader. Since there is only one defined mechanism for booting, if **`bootmgr.efi`**, **`bootmgrfw.efi`** or the **`BCD`** is missing, or corrupt, you cannot boot into Windows.

8.2.5. Fedora Boot Files

Fedora uses **`shim.efi`** during Secure Boot to load and execute **`grubx64.efi`**; which then reads **`grub.cfg`** and displays a boot selection menu. With Secure Boot disabled, **`grubx64.efi`** is called directly.

In a Secure Boot environment, each link in the boot chain must be signed by a trusted authority. On Windows Ready certified computers, the key that is trusted is provided by Microsoft and is pre-loaded on the computer prior to shipping. Thus, in order to boot (without manually changing the key structure in the EFI firmware) a boot manager and boot loader has to be signed by the pre-loaded key in order to execute.

Fedora's **`shim.efi`** is a code stub that has been signed by Microsoft. This allows the EFI firmware, in Secure Boot mode, to load and execute **`shim.efi`**. **`shim.efi`** then checks the signature on **`gurbx64.efi`** and, if correct, passes execution over to it. **`grubx64.efi`** has been signed by Fedora and that is the signature that **`shim.efi`** is looking for. Thus, **`shim.efi`** effectively transfers the chain of trust from Microsoft to Fedora as early in the boot sequence as possible.

`grubx64.efi` continues the chain of trust by checking the signature of the kernel files prior to passing execution to them.

The only real difference in the boot process between having Secure Boot enabled and having it disabled is the signature checking that takes place. If a file in the chain has a missing or invalid signature, then that file will not be allowed to execute. This provides another 'brick-in-the-wall' for protecting your computer against executing malware prior to loading the operating system.

It is important to note that Microsoft is not 'in charge' of this process. The whole issue of having to use Microsoft signed keys comes from the desire of operating system software companies (including Fedora) to have thier software installed on hardware that has complied with the certification process for a "Windows Ready" computer. Many computers do not go through the certification process and other-than-Windows operating systems can be installed on them without a Microsoft signed boot manager/loader. Additionally, the end user can also completely disable Secure Boot at any time or even add their own keys to be used in a Secure Boot environment. So, as you can see, Microsoft does not have a monopoly on the UEFI Secure Boot process and simply provides a process for all other operating system developers to boot on machines that have been specifically designed for running Microsoft Windows.

In the event that the normal files cannot be loaded, with or without Secure Boot enabled, Fedora provides a 'second chance' boot sequence that uses a process called "fallback".

In fallback, **`bootx64.efi`** is loaded and executed by the UEFI firmware; which in turn loads **`fallback.efi`**. Another file called **`Boot.csv`** contains entries that will be added 'on-the-fly' to the EFI NVRAM. **`fallback.efi`** searches the entire EFI partition looking for **`Boot.csv`** files. It read their data, and creates entries in NVRAM for each one. After that, it changes the **`BootNext`** variable and passes execution back to the firmware to boot from that entry. The fallback process is executed as a part of 'Default Boot Behavior'. Default Boot Behavior is initiated when the EFI firmware is unable to load any boot managers/loaders after traversing the entire list of operating system entries indicated in the **`BootOrder`** variable. More information about the UEFI Boot Process and Default Boot Behavior is available here: *Section 8.1, “UEFI Basics”*.

If your **`grub.cfg`** file is missing you will go to a GRUB command prompt where (using various GRUB commands) you can gather the information you need to boot your Fedora installation. Additional information on using the GRUB command prompt can be found here: *Section 6.2, “Changing GRUB entries at boot”*

8.2.6. Similarities and Differences Between `bootmgrfw.efi`, `grubx64.efi` and `bootx64.efi`

While you could say that **`bootmgrfw.efi`** acts in the same way as the boot manager files provided by GRUB, there are technical differences that show this not to be the case.

`bootmgrfw.efi` is roughly equivilent to both of the GRUB files **`grubx64.efi`** and **`bootx64.efi`** (which will be covered in more detail later). What happens as each is called (and the format of the files involved) is where the differences start to show up...

They are similar in that they all look for a configuration file: **`bootmgrfw.efi`** looks for **`BCD`**, **`grubx64.efi`** looks for **`grub.cfg`**, and **`bootx64.efi`** (using **`fallback.efi`**) looks for **`Boot.csv`** files.

A major difference between the two systems is that GRUB provides an additional method of booting using **`bootx64.efi`**. This file is used when the EFI firmware goes into 'Default Boot Behavior' as a result of not finding any boot managers to load on the first time through the **`BootOrder`** listing in NVRAM.

In a Windows based system, **`bootmgrfw.efi`** is called every time and no other mechanism is provided. The end result is that if either the **`bootmgrfw.efi`** or the **`BCD`** is corrupt, or otherwise inaccessible, then Windows will not boot. Its game over.

GRUB, on the other hand, has alternative methods for booting, which are implemented using **`bootx64.efi`**, as well as a GRUB command prompt. This is all explained, in detail, below and in other sections of this guide.

8.2.7. BCD Limitations vs GRUB Universality

`BCD`, by default, only knows about ONE installation (Windows); which it boots by loading **`winload.efi`**

While, in theory, **`BCD`** can be configured to allow for other entries, including pointing its default loader to a particular GRUB installation, in reality, it doesn't work very well. **`bootmgrfw.efi`**, using **`BCD`**, is designed to handle Windows. Period. It provides a straight shot to the Windows installation and that's it.

Adding additional entries to the **`BCD`** is accomplished through a commandline tool, provided by Microsoft, called **`bcdedit`**. Unfortunately, **`bcdedit`** is poorly documented, not intuitive to use, and the little information available on the Internet about it is usually incorrect and confusing. It is also difficult to properly modify primarilly becuase it is formatted as a proprietary database file (registry hive); in other words you must use **`bcdedit`** and that can only be done from within Windows.

GRUB, on the other hand, is designed from the gound up to handle all types of operating systems and makes it relatively easy to make changes to its configuration file.

Under normal boot operations, the GRUB manager/loader (**`grubx64.efi`**) is called directly (via **`shim.efi`** in Secure Boot), which then reads the **`grub.cfg`** file and displays its entries for user intervention. Adding additional entries to **`grub.cfg`** can be done with any simple text editor. More specific nformation on GRUB is available here: *Section 6.1, “GRUB Basics”*

8.2.8. Using BCD to dual boot

Modifying **`BCD`** to dual boot can be accomplished by using an administrators command prompt and entering:

```
C:\Windows\System32> bcdedit /set {bootmgr} path \EFI\fedora\shim.efi
```

This command causes the **`BCD`** to point to the Fedora **`shim.efi`**, as the default boot manager for Windows. **`shim.efi`** will in turn call **`grubx64.efi`** and present the GRUB menu. The only caveat with this, is that the Fedora EFI partition files have to be moved to the windows-created EFI partition (usually **`/dev/sda1`**). Specifically, **`/dev/sda1/boot/EFI/`**

During an installation of Fedora, where Anaconda is allowed to create the partitions automatically, there will be a separate EFI partition for the Fedora installation. To make **`BCD`** the boot configuration default for booting into your Fedora installation, you will need to copy all of the files from the Fedora-created EFI partition (starting at the fedora sub-directory) to the EFI partition created by Microsoft during its installation. You can also choose, during installation, to manually set up your partitions; in which case you can simply indicate a mount point of **`/boot/efi`** for the Microsoft created partition and tell it not to format that partition. GRUB will then create a **`/dev/sda1/EFI/fedora`** directory and populate it with the necessary files. Upon restart, **`/dev/sda1`** will be mounted at **`/boot/efi/`**

`However! Please keep reading!`

`Note!:` There are reports that Windows, upon seeing a change to "its" EFI partition, will (either inadvertantly or deliberately; no one knows which one for sure) destroy the directory structure for the other operating system; thus disallowing **`bootmrgfw.efi`** to access the GRUB file pointed to by the modified **`BCD`**. This will completely render all of your installations non-bootable.

It is highly recommended to maintain a separate EFI partition for all installations other than Windows. Simply let Anaconda do the work of setting up all of the partitions, and then tweak their sizes as desired. This will ensure that what has happened to others will not happen to you.

8.3. Booting on Apple systems

9. Boot Options

Boot options come in several categories:

Systemd arguments
Kernel parameters
Dracut options

A. Common Operations

This section includes common operations you might have to perform when following various instructions in this guide.

A.1. Setting up a chroot from a live image

To operate on a Fedora installation that will not boot, you must use a **`chroot`**. The process initially uses a Fedora live image, but sets up an environment that uses the files, executables, and libraries from the installed system.

The Fedora Netinstall image has a system recovery option that will set up the chroot for a Fedora system automatically. Use the instructions here for a live image.

Be consistent about your booting methods. If you have Fedora installed Fedora in UEFI mode, boot the live image in the same way.

Procedure A.1. Creating chroot of a Fedora installation

1. Boot a Fedora live image and open a root terminal window.

2. Create a directory to work in.

```
# mkdir /mnt/sysimage
```

3. List the available partitions. We'll use this list to identify all the partitions your Fedora installation needs.

```
           # blkid
            /dev/sda1: LABEL="WINRE_DRV" UUID="C812A5BC12A5AFBC" TYPE="ntfs"
 PARTLABEL="Basic data partition" PARTUUID="3f4f9bb5-6551-4411-9cea-d06dd1d10aff" /
dev/sda2: LABEL="SYSTEM_DRV" UUID="32A7-BBCC" TYPE="vfat" PARTLABEL="EFI
 System Partition" PARTUUID="f540b7b9-782a-48d2-89b4-4f0a720121cf" /dev/sda3:
 LABEL="LRS_ESP" UUID="F2C1-30B7" TYPE="vfat" PARTLABEL="Basic data partition"
 PARTUUID="cdbadfc7-e84b-4fe0-935c-28419b0082f7" /dev/sda4: PARTLABEL="Microsoft
 reserved partition" PARTUUID="bb3d61f2-652b-4793-a8e9-8b42e819f6bb" /dev/
sda5: LABEL="Windows8_OS" UUID="7898C3F398C3ADC6" TYPE="ntfs" PARTLABEL="Basic
 data partition" PARTUUID="111e1bb0-dafd-439b-8af3-9d6b86bff9e9" /dev/sda6:
 LABEL="LENOVO" UUID="16E2DDF6E2DDD9D7" TYPE="ntfs" PARTLABEL="Basic data
 partition" PARTUUID="45dc466a-929d-4862-91ed-396afe2e4f59" /dev/sda7:
 LABEL="PBR_DRV" UUID="40A4E1E7A4E1E000" TYPE="ntfs" PARTLABEL="Basic data partition"
 PARTUUID="29b2cac6-ff56-4cf2-b62e-d04230a72294" /dev/sda8: UUID="5958dcb4-dfeb-4f6c-
ba07-8b2ada8b1602" TYPE="ext4" PARTUUID="cacf80f0-3376-443d-bfb5-5d3fe689885e" /
dev/sda9: UUID="PRueOy-TXAC-mKDV-NRLC-k4gt-C63H-D0zfYd" TYPE="LVM2_member"
 PARTUUID="555251b9-438f-4f16-9011-754620526ebe" /dev/mapper/fedora-swap:
 UUID="28d71a2c-1b34-4115-aa19-083373ec4d8a" TYPE="swap" /dev/mapper/fedora-root:
 UUID="125425e2-dd7b-46e9-8c64-6d2c5b192c76" TYPE="ext4" /dev/mapper/fedora-home:
 UUID="1cec1d42-8750-4746-bd36-145c1ebb2d89" TYPE="ext4"
```

4. Identify Fedora's root partition.

```
# mount /dev/sda8 /mnt/sysimage
# cat /mnt/sysimage/etc/fedora-release
 cat: /mnt/sysimage/etc/fedora-release: No such file or directory
# ls /mnt/sysimage
 config-3.17.7-300.fc21.x86_64 config-3.17.8-300.fc21.x86_64
 config-3.19.0-0.rc5.git2.1.fc22.x86_64 efi elf-memtest86+-5.01
 extlinux grub2 initramfs-0-rescue-39995d7d26a74b1784e7bd9a7bfefdf1.img
 initramfs-3.17.7-300.fc21.x86_64.img initramfs-3.17.8-300.fc21.x86_64.img
 initramfs-3.19.0-0.rc5.git2.1.fc22.x86_64.img initrd-plymouth.img
 lost+found memtest86+-5.01 System.map-3.17.7-300.fc21.x86_64
 System.map-3.17.8-300.fc21.x86_64 System.map-3.19.0-0.rc5.git2.1.fc22.x86_64
 vmlinuz-0-rescue-39995d7d26a74b1784e7bd9a7bfefdf1 vmlinuz-3.17.7-300.fc21.x86_64
 vmlinuz-3.17.8-300.fc21.x86_64 vmlinuz-3.19.0-0.rc5.git2.1.fc22.x86_64
# umount /dev/sda8
```

   `/dev/sda8` wasn't the root partition, since it did not contain the **`/etc/fedora-release`** file, but we did find other things inside. The **`vmlinuz`** and **`initramfs`** files are the kernels and initramfs, and they always live on **`/boot`**. Remember that for later.

```
# mount /dev/mapper/fedora-root /mnt/sysimage
# cat /mnt/sysimage/etc/fedora-release
Fedora release 21 (Twenty One)
```

That's the Fedora root partition, so we will leave it mounted there. Because the other filesystems use mount points inside the root partition, we mount it first.

5. Next, identify and mount the **/boot** partition. We found it as **/dev/sda8** already, but we can confirm by checking Fedora's **/etc/fstab**, the file that tells the system where to mount filesystems.

```
#  grep boot /etc/fstab
 UUID=5958dcb4-dfeb-4f6c-ba07-8b2ada8b1602 /boot ext4 defaults 1 2 UUID=32A7-BBCC /boot/
efi vfat umask=0077,shortname=winnt 0 0
```

   /etc/fstab uses **Universally Unique Identifiers** to identify partitions. This makes sure that the right partitions get mounted in the right place, even if drive orders change. The UUID listed for **/boot** matches what **blkid** reports for **/dev/sda8**, so we know we have the right partition.

```
# mount /dev/sda8 /mnt/sysimage/boot
```

6. If this is a UEFI installation, mount the EFI system partition next. In this example, we found it in the previous step, and can match the UUID with **blkid** output. You can confirm by looking at the contents once mounted.

```
# mount /dev/sda2 /mnt/sysimage/boot/efi
# ls /mnt/sysimage/boot/efi
 /boot/efi/BOOTSECT.BAK /boot/efi/EFI: BOOT fedora Microsoft
```

7. Next mount the virtual filesystems that the system uses to communicate with processes and devices. These are always the same, so we can do it in one command.

```
#for dir in /dev /proc /sys; do mount --bind $dir /mnt/sysimage/$dir ; done
```

8. Enter the chroot to continue, and begin working on your Fedora installation.

```
# chroot /mnt/sysimage
```

9. When you are finished, use the **exit** command to leave the chroot.

B. Revision History

Revision 0-7	**Mon Jan 5 2015**	**Pete Travis** *immanetize@fedoraproject.org*

Adding an introductory FAQ

Revision 0-2	**Mon Jul 29 2013**	**Chris Roberts** *chris.roberts@croberts.org*
Adding UEFI Windows content		
Revision 0-3	**Sat Oct 4 2014**	**Roger Baran** *rogerbaran@fedoraproject.org*
Add lots of Windows content.		
Revision 0-1	**Sun Jul 14 2013**	**Pete Traviss** *immanetize@fedoraproject.org*
Initial creation of book by publican		

Index