Booting from GPT

by Rod Smith, rodsmith@rodsbooks.com

Last Web page update: 11/12/2012, referencing GPT fdisk version 0.8.5

I'm a technical writer and consultant specializing in Linux technologies. This Web page, and the associated software, is provided free of charge and with no annoying outside ads; however, I did take time to prepare it, and Web hosting does cost money. If you find GPT fdisk or this Web page useful, please consider making a small donation to help keep this site up and running. Thanks!

Donate $1.00	Donate $2.50	Donate $5.00	Donate $10.00	Donate $20.00	Donate another value
Donate	Donate	Donate	Donate	Donate	Donate

Note: This page is part of the documentation for my GPT fdisk program.

One of the challenges of GPT is that of booting from it. In late 2012, support for booting from GPT is limited compared to support for booting from MBR, although this is changing rapidly with the adoption of UEFI. This support varies by OS, as well. Following are my notes and general observations on this issue. Keep in mind, though, that these details are likely to change rapidly.

The rest of this page is broken down into three main sections: Information on EFI vs. BIOS booting; booting with EFI; and booting with BIOS.

General Comments: EFI vs. BIOS

GPT is part of the EFI specification, so of course booting from GPT disks becomes easier if your computer uses EFI. Unfortunately, EFI is still fairly rare on the installed base of mainstream PCs, but most new PCs are UEFI-capable. Among common computers, Apple Macintoshes are the systems that are most likely to use EFI. Many new motherboards and computers now include Unified EFI (UEFI) support, UEFI being essentially EFI 2.*x*. Most motherboards released in mid-2011 and later include UEFI support, and Microsoft requires that PCs bearing a Windows 8 logo ship with UEFI (and a specific UEFI feature known as *Secure Boot*) enabled. UEFI implementations usually include a BIOS support mode, so you may need to hunt for a firmware option to explicitly enable UEFI support or disable the BIOS (or *legacy*) support. One obscure product, the Developer's UEFI Environment (DUET), enables booting a BIOS-based computer using a disk-based EFI implementation. I have a separate Web page that describes how to set up DUET; however, the technology is still very "bleeding edge," and so is best used if you're an interested hobbyist or if you're desperate. The upcoming "Windows" section describes the DUET option, since Windows is the only OS that really needs it.

The EFI specification includes an EFI System Partition (ESP) as a storage place for EFI boot loaders, drivers, and other files. Thus, if you intend to boot a disk using an EFI-based computer, you should create an ESP. Different sources suggest different sizes, but 100-500 MiB seems to be the range. If you intend to boot multiple

Linux distributions in EFI mode and use the ELILO boot loader or a Linux kernel with EFI stub support, a larger ESP makes more sense; but if you'll be storing your kernels off the ESP, a size on the small end of the range should be sufficient.

Many Internet sources, particularly discussion groups, assert that it's impossible to boot a GPT disk on a BIOS computer. This is nonsense—or at least, it's true only of certain OSes. Windows, in particular, is behind the times on this score, as described shortly. I personally have successfully booted both Linux and FreeBSD on GPT-only computers with BIOSes. For the truly adventurous, it's even possible to get Mac OS running on GPT disks on conventional hardware, although this configuration is unsupported by Apple and may even be illegal. DUET, as already noted, can be used to boot any UEFI-capable OS, Windows included, on BIOS-based hardware. That said, there are some rare BIOS/GPT conflicts; see my Legacy BIOS Issues with GPT page for details.

BIOS-based computers, whether they use MBR or GPT, rely on a *boot loader* in the first sector of the disk to help get the computer booted. In fact, the first 440 bytes of the MBR data structure are devoted to this boot loader. DOS and Windows place a very simplistic boot loader in this space. Other OSes and third-party utilities enable placing more sophisticated boot loaders in the MBR, although these boot loaders usually rely on multiple stages—the boot loader code loads a secondary boot loader that's located elsewhere, and that boot loader may even load a third stage. In principle, these boot loaders can work just fine when the MBR is in fact a GPT protective MBR. In practice, the boot loader needs to be GPT-aware in order to work. The GRUB 2 boot loader, when used on a GPT disk, works best when you to have a BIOS Boot Partition (GPT fdisk code EF02) on the disk. Most boot loaders, including the patched versions of GRUB Legacy (version 0.97) that include GPT support, don't require a BIOS Boot Partition. If you do need it, the BIOS Boot Partition can be quite small—sometimes as small as 32 KiB, although I've seen reports that some configurations require more space than this—sometimes over 1 MiB. If you align your partitions to 1 MiB boundaries, 1 or 2 MiB is the logical size.

Old versions of GNU Parted could affect the BIOS-mode bootability of GPT disks by erasing the MBR's boot code (through at least version 1.7) or by removing the Legacy BIOS Bootable flag (through at least version 2.3) used by SYSLINUX's GPT support. Version 3.1 lacks these problems.

I've seen a few Web pages that suggest that booting from beyond the 2 TiB mark is iffy on BIOS-based systems. I haven't investigated this issue in detail, but I recommend creating the BIOS Boot Partition and any OS boot partitions below the 2 TiB mark. If necessary, create a small partition to hold the `/boot` directory below the 2 TiB mark.

Booting with EFI

Booting from GPT with EFI is, in theory, straightforward, since GPT is part of the EFI specification. There are, however, quirks related to specific OS boot loaders and installers. I describe some of these, as well as the process of booting OS X.

EFI Boot Loaders and Boot Managers

A *boot manager* is a program that enables you to select which OS to boot. A *boot loader,* on the other hand, loads an OS kernel and transfers control to it. Many programs perform both of these functions, but others handle just one of them. Under (U)EFI, the firmware usually includes a simple boot manager, and each OS provides its own boot loader. Most EFI boot loaders and boot managers reside in their own subdirectories of the `EFI`

directory on the ESP.

Rather than detail the available boot loaders and boot managers here, I direct you to my separate Web page on the topic, Managing EFI Boot Loaders for Linux. Briefly, though, most Linux distributions are gravitating toward an EFI version of GRUB 2. In my experience, though, the boot loader built into the Linux kernel since version 3.3.0 works better, and can be combined with a boot manager such as my own rEFInd or gummiboot to select which OS to boot.

Installing to (U)EFI

Every OS has installation quirks on any platform, and these can sometimes be frustrating, particularly if you're not familiar with them. I've done a number of test installations and conversions and have a few comments:

- Windows 7 insists that its ESP use the FAT32 filesystem. If a disk has a FAT16 ESP, Windows will try to create a new FAT32 ESP. If it can do so, installation will proceed to a point but then fail. Unfortunately, many Linux installers create FAT16 ESPs by default, so you may need to back it up, create a new FAT32 filesystem, and restore the files if you install Linux first.
- On the *x*86-64 platform, Windows 7 installs fine to most UEFI-based PCs, but not (in EFI mode) to Macs. To install Windows on a Mac, you must use a the Apple EFI's BIOS compatibility mode, which in turn requires a hybrid MBR because of Windows' limitations. Some Mac users have been trying to work around this limitation, and in some cases have managed to coax Windows into working on Macs. See this thread for one extended discussion of the issues, but be aware that the thread is very long and includes several false leads. Windows 8 appears to be more willing to install to Macs in EFI mode, but the betas that are available now (mid-2012) are still imperfect in this regard.
- Only the 64-bit versions of Windows Vista, 7, and 8 support UEFI booting on *x*86-64 systems; earlier versions and 32-bit versions of the OS can only install or boot in BIOS mode.
- I haven't yet tried installing Windows 8 on any computer, so I can't comment on its quirks, except to say that computers that ship with Windows 8 almost invariably ship with Secure Boot enabled. See this page for more information on this important feature. Note that Secure Boot is *not* required by Windows 8; you can disable the feature if you don't want it, or install to a computer that doesn't support Secure Boot.
- If you have a motherboard that supports both BIOS and UEFI booting, you can switch Windows from BIOS to UEFI booting if you follow this procedure. It's a bit involved, so I only recommend doing this if you have a real need for it.
- Mac OS X has specific requirements for partition sizing and placement. If they aren't met, the installer will refuse to install to your disk.
- Ubuntu 11.04 and 11.10 have an extremely serious bug that causes it to erase any existing ESP, thus wiping out any existing boot loaders or other files installed there. Be sure to back up your ESP before you install Ubuntu to a (U)EFI system! Thankfully, this bug has been fixed in Ubuntu 12.04.
- Through at least version 12.04, Ubuntu creates an undersized FAT16 ESP (well under 100 MiB, the precise value varying depending on the disk's size). I recommend creating the ESP manually before installing Ubuntu.
- OpenSUSE 12.1 creates a somewhat small FAT16 ESP (156 MiB in my test), and it insists on creating a new ESP even if one already exists on the disk. You can work around the size problem by using the manual partitioner, but you can't force it to use FAT32 for the ESP except by making it bigger than about 520 MiB. Earlier versions of OpenSUSE had far more serious EFI installation bugs, so I can't recommend you use them.

- Fedora has the best EFI installation support of any Linux distribution at the moment (Fedora 17 is current as I type), although it can be a bit quirky—it will install fine on one system but present unique challenges on another. Through at least Fedora 16, the ESP was too large. I'm not sure if that's been fixed in Fedora 17 (I partitioned manually on my EFI Fedora 17 installation).
- Any Linux distribution with appropriate EFI and GPT support in the kernel can be switched from BIOS-mode to UEFI-mode booting or vice-versa by installing a suitable boot loader and adjusting the firmware's boot mode. This fact can be used to work around some of the more serious installation problems: Install in BIOS mode, install an EFI boot loader, reconfigure the BIOS, and reboot. Unfortunately, many firmware implementations provide limited user control of the boot mode, and manufacturers seldom bother to document the algorithms the firmware uses to decide on the boot mode. Thus, you may need to scour the Internet or perform experiments to figure out how to switch from one boot mode to another.

Mac OS X

Intel-based Macs employ EFI and GPT by default. Thus, as you might expect, booting Mac OS from a GPT disk is not a problem. I've used `gdisk` to manipulate Mac boot disks, although my tests have been limited since I've got just one Mac Mini. The biggest caveat seems to be following Apple's recommendations for partition sizing and placement. The need for 128 MiB of unpartitioned space after each OS X partition is particularly important; without that space, the OS X installer will refuse to install or upgrade on the partition.

Booting from GPT on BIOS-Based Computers

Most OSes have GPT-specific boot quirks on BIOS-based computers. The best, such as Linux, install and boot fine on GPT systems, so long as you follow the advice for creating appropriate partitions. Others, such as Windows, are much more challenging to get booting in this way. With all OSes, be aware that there are some buggy BIOSes that have problems booting from GPT disks. These problems can usually be overcome, but you should be aware of the issue before you try to set up a system to boot from a GPT disk in BIOS mode.

Linux, GRUB, LILO, and SYSLINUX

Most modern Linux distributions install GRUB as the boot loader. Currently, GRUB 2 is the most common choice, although GRUB 0.97 (aka GRUB Legacy) remains available. Officially, GRUB Legacy is not GPT-aware and so can't boot anything from a GPT disk. In practice, though, patched versions of GRUB Legacy are common, and many distributions ship with them. (Ubuntu 8.04 later and Fedora 10 and later definitely have GPT-enabled versions of GRUB Legacy, but openSUSE 11.0 does not. Ubuntu switched to GRUB 2 with version 9.10, and Fedora switched to GRUB 2 for BIOS booting with version 16, although Fedora continues to use GRUB Legacy for EFI booting through version 17.) If your distribution's GRUB lacks GPT support, you can download the System Rescue CD and apply its version of GRUB to your GPT disk:

1. Mount your `/boot` partition over the SRCD's `/boot` directory, or copy the contents of your `/boot/grub` directory over the SRCD's directory.
2. If your drives' device filenames are different under SRCD than under a normal boot of your distribution, edit `/boot/grub/devices.map` appropriately.
3. Type **`grub-install /dev/sda`** or **`grub-install /dev/hda`** to re-install GRUB.

4. Reboot. Assuming GRUB is properly configured, your system should now boot.

Note that if you've converted an MBR disk to GPT format, booting will fail even if you were previously using a GPT-aware version of GRUB. This is because the MBR and GPT boot-time code for GRUB is different; in fact, GRUB installs part of itself just after the MBR on MBR-based disks (when you install it to the MBR), but that space becomes used by GPT on GPT disks, so converting MBR to GPT will wipe out part of GRUB. Re-installing a GPT-aware GRUB, as just described, will correct this problem.

Development on GRUB Legacy has officially ceased. A new boot loader, known as GRUB 2, is now under development instead. This boot loader includes GPT support. GRUB 2 works at least as well as GRUB Legacy on BIOS systems, although it's more complicated and poses some learning challenges for those already familiar with GRUB Legacy. The GRUB 2 configuration file format is different from (and more complex than) GRUB Legacy's. GRUB 2 includes scripts intended to help automate setup, so you shouldn't be editing its main configuration file (`/boot/grub/grub.cfg`) directly; instead, you should edit files in `/etc/grub.d` and then re-run `grub-mkconfig` or some other utility to regenerate the `grub.cfg` file. The two versions of GRUB identify partitions differently; GRUB numbers them starting from 0, whereas GRUB 2 numbers them starting from 1. Confusingly, GRUB 2 continues to number hard disks starting from 0, so that `(hd0,1)` is the first partition on the first hard disk.

There's a known bug with older versions of GRUB 2 and systems with multiple disks that use a mixture of GPT and MBR partition tables. On Ubuntu systems (and perhaps others), you can work around this by including the following line in `/etc/default/grub`:

```
GRUB_PRELOAD_MODULES="part_msdos"
```

Information on the old Linux Loader (LILO) and GPT is contradictory. Most sources say the two won't get along, but I've read others who opine that the combination does (or at least should) work fine, since LILO uses sector maps to point to the kernel file. My one attempt at this combination proved inconclusive. LILO was able to load and run the kernel, but the boot then failed with the kernel message `mount: could not find filesystem '/dev/root'`. This message followed messages that indicated that the computer's LVM configuration was working fine, but somehow handing off to the LVM-based root filesystem was a problem. A GRUB boot of this system worked fine.

The SYSLINUX boot loader is another Linux boot loader that includes GPT support. This support resides in the MBR and redirects the boot process to a partition with the Legacy BIOS Bootable flag set. AFAIK, no Linux distribution relies on SYSLINUX to boot from a GPT disk, but you could set yours up to use this boot method if you liked.

Beyond the boot loader, Linux requires GPT support in its kernel to work with GPT disks. This support is common, but there's no guarantee that any given kernel will have it. If you've compiled your own kernel, you can check on this detail by entering the kernel configuration utility (**`make xconfig`** or similar) and looking under File Systems -> Partition Types. Be sure that the EFI GUID Partition Support option is checked.

FreeBSD

FreeBSD supports GPT, and can boot from it; however, this support is very kludgy, as of FreeBSD 7.2. (I'm

afraid I haven't done much with FreeBSD on GPT recently, so I can't provide more up-to-date information.) To date, I have been unable to get FreeBSD to boot from a disklabel partition converted from an MBR disk, although the FreeBSD live CD can read that partition just fine. The FreeBSD installer also gets confused by GPT disks; it tends to treat them as MBR disks, which produces a corrupt MBR and a non-working installation.

I have successfully installed FreeBSD on GPT disks twice, but both times involved a conversion process. The simpler procedure, in broad outline, was as follows:

1. I installed FreeBSD as normal on an MBR disk. I ensured that no vital FreeBSD partitions were at the very start or very end of the disk, so as to avoid problems during the conversion stage. I also left a little free space on the disk.
2. I used `gdisk` to convert the disk to GPT format, including converting the BSD disklabel partitions. I then created a small (30-sector) GPT partition of type "FreeBSD boot" (0xA501 in `gdisk`). This partition would ultimately hold the GPT-aware FreeBSD boot loader. Note that this partition is not mounted (it's not for the `/boot` directory; it's more like FreeBSD's version of a BIOS Boot Partition). I created the boot partition immediately before the root partition in one test and immediately after it in another. I don't know if this ordering is critical, though.
3. I booted the FreeBSD live DVD and copied the `/boot/pmbr` and `/boot/gptboot` files from the FreeBSD installation (now in GPT form) to the temporary live DVD's filesystem. I then unmounted its filesystem.
4. I typed **`gpt boot /dev/ad0`**. You might need to change the device filename for your system and/or use the `-b` or `-g` options to point to the `pmbr` and `gptboot` files, respectively. This command installs FreeBSD's GPT-aware boot loader. The `pmbr` file is the MBR boot loader, while the `gptboot` file is the GPT-specific second-stage boot loader that ends up in the FreeBSD boot partition. (A key point is that this partition must be large enough to hold the `gptboot` file, but shouldn't be too much larger, since the whole thing is loaded into memory during boot.)
5. I edited `/etc/fstab` so that the system mounted the GPT partitions rather than the BSD disklabel partitions.

The system was then bootable and FreeBSD worked fine from its converted GPT partitions. To create a multi-booting system with FreeBSD and Linux (which I did on just one of my two tests), I had to jump through additional hoops:

1. Using a Linux live DVD, I copied the MBR from the GPT boot disk to a file, as in **`dd if=/dev/sda of=sda.mbr bs=512 count=1`**. I placed this file in my Linux `/boot` partition.
2. Using the Linux System Rescue CD, I re-installed GRUB on the boot disk. I also created an entry for FreeBSD:

```
title FreeBSD 7.1
    root (hd0,2)
    chainloader (hd1,5)/sda.mbr
```

   In this case, `(hd0,2)` was GRUB's identifier for the FreeBSD root partition and `(hd1,5)` was the idenfier for the Linux `/boot` partition where I'd stored the MBR copy.

The effect of this configuration is that, when I selected the FreeBSD entry from the GRUB menu, GRUB launched the copy of the MBR as if it were the boot sector on the disk. This code then took over and launched FreeBSD's native boot loader.

For more information, try this page, which provides FreeBSD-specific instructions on installing FreeBSD on a GPT disk.

Note that the `gpt` utility is FreeBSD's GPT-handling tool. It includes the ability to create GPT partitions and it has its own MBR-to-GPT facility. This facility can nominally split out a BSD disklabel into GPT partitions; however, when I've tried this it's produced overlapping GPT partitions, so it seems to be buggy (or at least, the version shipped with FreeBSD 7.1 for x86-64 is buggy).

NetBSD

I have not attempted to boot NetBSD from a pure-GPT disk. The NetBSD 5.01 installer assumes that the system is using MBR. (In fact, I had some install problems for a test installation that may have been related to some remnants of GPT data on the disk.)

I have discovered this page (see also another page on the same project), which describes a new (beta-level, as of July 2009) GPT-aware boot loader for NetBSD. In theory, it should be possible to follow a procedure similar to that described for FreeBSD, but using this new NetBSD boot loader, to get NetBSD to boot from a GPT disk.

Windows

According to Microsoft's Windows and GPT FAQ, no version of Windows through Windows 7 can boot from a GPT disk unless the computer uses UEFI. To boot from a GPT disk, you need a version of Windows for the Itanium CPU or Windows Vista or later on a UEFI-based system.

Microsoft's FAQ is a bit pessimistic. It *is* possible to boot Windows from a GPT disk on a BIOS-based computer, but the ways to do this are hacks. They fall into two categories:

- **Use a hybrid MBR**—If you create a hybrid MBR on a GPT disk, Windows will be able to boot from a hybridized partition. Note that, once booted, Windows will be able to see only the MBR side of the disk, so this is only useful in certain dual-boot configurations in which Windows needs to see no more than three of the disk's partitions.
- **Use DUET or Clover**—If you install DUET on the disk, you can make the system look like it's UEFI-based, enabling Windows to install to and boot from a GPT disk. The Clover boot loader (see here for a binary download link) is a Hackintosh boot loader that's based, in part, on DUET.

A hybrid MBR is a modified GPT protective MBR that uses up to three of the MBR's primary partitions to point to up to three GPT partitions. (The remaining MBR primary partition contains an EFI GPT partition entry.) You can create a hybrid MBR using the `h` option on GPT fdisk's recovery & transformation menu, or you can use the separate `gptsync` utility, which ships with Fedora's `anaconda` package and sometimes in other packages with other distributions.

Unfortunately, hybrid MBRs are ugly and dangerous hacks. Apple relies on them to get Windows booting on its EFI-based Macs. (Mac firmware includes a BIOS compatibility mode, so the Mac looks like a BIOS-based computer to Windows, which doesn't support EFI-mode booting from Macs.) Many things can go wrong with hybrid MBRs, so I strongly recommend avoiding them if at all possible.

The other option, DUET, is much more flexible and less dangerous than a hybrid MBR, but it's also much harder to set up. The DUET software was written as a software development tool and, until fairly recently, was very difficult to get working. Even today (late 2012), the best package I'm aware of for the job is tedious to set up, as described on my DUET Web page. If you want to convert an existing system to boot in this way, you can read this article—but be aware that there's a good chance you'll trash your Windows installation, particularly if you're not careful or don't understand what you're doing. If you want to go this route, I strongly recommend you begin by doing a test installation on a spare hard disk and then convert your existing system once you've gotten your test system running in UEFI mode. Even when everything works perfectly, the DUET path works only with 64-bit versions of Windows Vista or later; Windows XP and 32-bit versions of Windows can't be booted in this way. Despite these caveats, DUET or something like it may become extremely important in the future. An easier-to-install and more reliable version could provide a lifeline for those who need to upgrade their hard disks in the future.

Clover, being based on DUET, can be used much like DUET, but the installer is designed to run under OS X, which can make it hard to install if you're not running a Hackintosh configuration. Overall, you're probably better off using DUET if you must use this approach; but in some cases Clover might be as easy to install, if not easier.

I'm not sure why Microsoft has chosen to limit Windows by not supporting GPT boots on BIOS-based computers. This really is stunningly short-sighted of Microsoft. Do they really expect that nobody will need to replace a failed hard disk on a BIOS-based computer during Windows 7's lifetime, and opt to install an over-2TiB drive? Until a better solution than hybrid MBRs or the current state of DUET comes along, the combination of GPT, a legacy BIOS, and Windows just isn't a good one.

Go on to "Hybrid MBRs"

Return to "GPT fdisk" main page

If you have problems with or comments about this web page, please e-mail me at rodsmith@rodsbooks.com. Thanks.

Return to my main web page.