BootFromUSB

Unsupported Version
This article applies to an unsupported version of Ubuntu. More info...

Introduction

Contents

1. Introduction
2. Booting via GRUB
3. The minimal Linux method
    1. Using a CD
    2. Using an internal hard drive
4. See Also

PCs use a system called the BIOS (Basic Input/Output System) to start the machine. A modern BIOS (written after 2002) usually contains USB drivers and a boot from USB option but older computers often don't have these features. There are two common problems:

1. They may lack a BIOS setting to allow booting from USB.
2. They may not recognize USB drives initially, and may require operating system drivers to accomplish this.

An alternative for older machines is to let the BIOS start a minimal Linux system on a BIOS supported drive such as a floppy disk, CD, or HD, and then transfer control to the USB drive to continue booting the full operating system. A minimal Linux system contains the necessary USB drivers to continue the boot process.

Note: *If the computer was made between 2002 and 2005, it may contain USB drivers, but not an option to boot from USB. In this case the GRUB bootloader can do the job directly, without the need of an initial Linux system.*

Booting via GRUB

The easiest way to boot from a USB Drive is to boot via GRUB. To check if your BIOS is able to detect the USB drive and hand it over to GRUB, just run GRUB from your hard drive if it already installed, or from a GRUB boot floppy or CD. At the GRUB menu, hit the `C` key to enter command mode. Now search for your USB drive, using the `root` command to choose a drive/partition and the `find` command to see if you found the right one.

You can go through your devices like in this example:

```
grub> root (hd0,0)   # first harddrive, first partition
grub> find /[tab]    # type the slash then press [tab], and it will try to list files on this partition
Error 17: Cannot mount selected partition   # Oops no file system here
grub> root (hd0,1)   # first harddrive, second partition
grub> find /[tab]
 Possible files are: lost+found var etc media ...   # That was my hard drive with my linux install
grub> root (hd1,0)   # second hard drive usually is the USB drive if you have only one internal drive
grub> find /[tab]
 Possible files are: ldlinux.sys mydoc myfile mystick syslinux.cfg  # Bingo, that's the USB stick
```

Note: *If you have two internal drives including your CD/DVD drive, the USB drive probably is* `hd2,0` *and so on.*

Boot the drive by entering:

```
chainloader +1
boot
```

For convenience, add these commands to your GRUB configuration (usually in `/boot/grub/menu.lst`):

```
# to boot from a USB device
title    Boot USB drive
root     (hd1,0)
chainloader +1
boot
```

Of course, if you don't have GRUB installed on your hard drive, change the `menu.lst` on your GRUB floppy or CD.

If you are not able to find the drive with the help of GRUB, you have to use the Linux kernel as explained in following sections.

Warning!

The method described above constitutes the biggest security risk in Linux. Using GRUB, any OS can be booted from any USB or CD/DVD drive (as above), circumventing BIOS restrictions. Placing passwords or locking menu items (in the GRUB configuration files) does not prevent a user from booting manually using commands entered at the GRUB command-line. Once a foreign OS is booted, it can be used to access any part of an unencrypted hard drive.

The minimal Linux method

The most minimal system would be the Linux kernel itself. Though a stripped-down Linux kernel is actually small enough to fit onto a high density (1.44MB) floppy disk, the standard Ubuntu kernel is a little larger than this. Therefore a CD or HD boot is easier to create. Besides, the standard Ubuntu kernel does not have all of the drivers needed to boot a USB drive. To solve this problem, extra driver modules must be put into an "initial RAM disk" image. This image is called `initrd` and gets copied into the RAM during boot to enable the kernel to access the extra modules it contains.

Note: *Since the kernel and* `initrd` *are copied into RAM during the boot and run from there, it is not necessary for the CD to be in the drive after booting has finished. Thus, if you only have one CD-ROM drive, it will not be tied up like it would be with a live CD.*

Using a CD

To build your own boot CD, you can either use an Ubuntu system or the Live CD. Since an Ubuntu system could not be available, we will show how to build it from the Live CD.

Boot your Live CD (this procedure was tested with Ubuntu 9.04 beta) and wait for the whole system to load.

Open a terminal and create the CD folder structure:

```
mkdir -p iso/boot/grub
```

Copy `initrd` and `vmlinuz` to the boot folder:

```
cp /cdrom/casper/initrd.gz iso/boot/
cp /cdrom/casper/vmlinuz iso/boot/
```

We need to include some more modules to `initrd`, so we do:

```
gksudo gedit /etc/initramfs-tools/modules
```

Which will open the text editor with that file. You need to add these lines to the end of the file:

```
### This is a reminder that these modules have been added to allow a CD to boot a USB drive
usbcore
sd_mod
ehci_hcd
uhci_hcd
ohci_hcd
usb_storage
scsi_mod
```

It is always a good idea to put comments in any manually changed configurations so you don't get confused later. Now save the file and exit gedit.

There is one small problem before we carry on: USB drives take a few seconds before they are engaged properly by Linux, and your boot would fail before the drive becomes accessible. To sort this, we need to tell the `initrd` to wait a few seconds before it gets carried away and fails. You can do this by running:

```
gksudo gedit /etc/initramfs-tools/initramfs.conf
```

Now add these lines at the very top of the file:

```
### This makes the bootup wait until any USB drives are ready
WAIT=15
```

Then save the file and exit the text editor.

Now that we have corrected the `initrd`'s setup we must use this setup to rebuild the `initrd` using our new guidelines. To do this you must enter your system by running:

```
sudo mkinitramfs -o iso/boot/initrd.gz <kernelversion>
```

Where <*kernelversion*> is the version of Linux you have installed that you wish to reconfigure. The default for Ubuntu 9.04 is 2.6.28-11-generic.

Now that you have a kernel which can boot your USB drive, it is time to put it on a bootable CD. You will need to copy the `stage2_eltorito` file to the top folder of the live CD (the location varies slightly between 32bit and 64bit PCs):

```
cp /usr/lib/grub/i386-pc/stage2_eltorito iso/boot/grub/
```

Next we need to create GRUB's menu:

```
gedit iso/boot/grub/menu.lst
```

Add the following lines:

```
title Run Ubuntu 9.04 beta from USB DISK
root (cd)
kernel /boot/vmlinuz file=/cdrom/preseed/ubuntu.seed boot=casper noprompt cdrom-detect/try-usb=true persistent
initrd /boot/initrd.gz
boot
```

Note: *You can change some of the other options here if you want a hidden menu or a different timeout before the default entry is booted but never set this to zero or you will not be able to enter recovery mode.*

That is the entire contents of your bootable CD, so now we have to build it. Open a terminal (by default it will be in the home folder) and run (all on one line):

```
sudo mkisofs -R -b boot/grub/stage2_eltorito -no-emul-boot -boot-load-size 4 -boot-info-table -o UbuntuBootCDForUSB.iso iso/
```

You should now have a CD image called `UbuntuBootCDForUSB.iso` which you can write to a recordable CD using many common CD writing programs, such as Brasero. If you don't have your CD-RW drive accessible (because you are running the live CD from it) then you can copy the CD image somewhere to keep it safely stored on a real drive. You should also copy over the `iso` folder, like this:

```
sudo cp UbuntuBootCDForUSB.iso /somewhereelse
sudo cp -r iso /somewhereelse
```

Note: *Don't forget to replace* `/somewhereelse` *with a real path.*

Now you just have to boot off the CD you have written whilst your USB drive is plugged in and (depending on how you set up GRUB) you will be sent into Ubuntu's boot procedure.

Important!

Whenever Ubuntu's kernel is updated you will not notice. This is because you will still be running the older kernel from your CD. You can fix this easily by replacing the `vmlinuz` *and* `initrd.img` *files by the new ones in the* `boot` *folder and making a new image by rerunning the above* `sudo mkisofs` *command. New kernels will automatically contain our modifications since we added the changes to the system's configuration files. After writing the new image to disc you can use it to boot into your new kernel. I would not recommend using a single CD-RW disc and overwriting it every time, as you will be in trouble if it doesn't work. Instead I suggest using two CD-RW discs and updating one at a time.*

Using an internal hard drive

This section is only for those who have an internal drive that can be written to and has grub installed. It makes sense if you prefer to install Ubuntu to an external hard drive instead of the internal one, but still have the possibility to customize the internal hard drive a little in order to boot the Ubuntu installation on your external drive.

Basically, you just copy the `initrd` file and `vmlinuz` from your Ubuntu installation onto the internal disk. If you have the possibility to repartition the disk, you can make a new ext3 partition (anything more than 20MB should be enough) and use it as a "boot" partition.

Note: *You don't have to make an ext3 partition, but you need to use a partition type that GRUB can read, which is pretty much everything except NTFS. I.e. if your disk has a FAT32 partition already, you can copy* `vmlinuz` *and* `initrd` *there instead of making a new partition. In the case you only have an NTFS partition, there's still one solution: Install "GRUB for DOS" (*`grubldr.exe`*) on the NTFS partition. See the GRUB for DOS wiki for more information.*

Either boot from the Ubuntu Desktop Live CD or mount it in the filesystem. Suppose that Ubuntu is running and the live CD is mounted in `/media/cdrom` (as simple as `sudo mount karmic-desktop-i386.iso /media/cdrom`), let's make a new dir and store `vmlinuz` and `initrd` in it:

```
sudo mkdir /boot/usb-boot
sudo cp /media/cdrom/casper/vmlinuz /boot/usb-boot
sudo cp /media/cdrom/casper/initrd.lz /boot/usb-boot
```

Edit your GRUB's menu (with `gksudo gedit /boot/grub/menu.lst`) and add at the end:

```
title USB FLASH DRIVE
root (hd0,6)
kernel /boot/usb-boot/vmlinuz file=/cdrom/preseed/ubuntu.seed boot=casper noprompt cdrom-detect/try-usb=true persistent
initrd /boot/usb-boot/initrd.lz
boot
```

Now you can reboot your system with your Ubuntu Bootable usb device plugged in and choose **USB FLASH DRIVE** (tested with Ubuntu 9.10 Karmic Koala Alpha 3).

See Also

- Building a bootable GRUB CD - This is GNU GRUB's official documentation which I used when writing this guide
- https://wiki.ubuntu.com/Booting
- https://help.ubuntu.com/community/Installation/FromUSBStick

CategoryBootAndPartition CategoryUsb

BootFromUSB (last edited 2011-08-30 10:09:08 by https://login.launchpad.net/+id/sN8hMrQ @ sa.csbnet.se[95.80.51.237]:frankbooth)