GRUB Legacy

Related articles

- Arch boot process
- Boot loaders
- Boot debugging
- grub-gfx
- Kernel parameters

GRUB Legacy (http://www.gnu.org/software/grub/grub-legacy.html) is a **multiboot (http://www.gnu.org/software/grub/manual/multiboot/)** bootloader previously maintained by the **GNU Project**. It was derived from GRUB, the GRand Unified Bootloader, which was originally designed and implemented by Erich Stefan Boleyn.

Briefly, the *bootloader* is the first software program that runs when a computer starts. It is responsible for loading and transferring control to the Linux kernel. The kernel, in turn, initializes the rest of the operating system.

Warning: GRUB Legacy is no longer maintained upstream and is not officially supported in Arch (see the news **here (https://www.archlinux.org/news/grub-legacy-no-longer-supported/)**). Users are recommended to switch to **GRUB**(2) or **Syslinux** instead. See **Upgrading to GRUB2**.

Note: If you have grub-legacy installed as grub-0.97 pkg in your system, it will be updated to `grub (https://www.archlinux.org/packages/?name=grub)`-2.xx pkg (GRUB2) during pkg updates. If you want to keep using grub-legacy install `grub-legacy (https://aur.archlinux.org/packages/grub-legacy/)`AUR so that pkgname does not conflict. During this update only the files in `/usr/lib/grub/` are updated, and grub-legacy files installed to `/boot/grub` and the `MBR` are not removed. You can boot back into grub-legacy by simply renaming `/boot/grub/menu.lst.pacsave` to `/boot/grub/menu.lst` .

Contents

Installation

GRUB Legacy has been dropped from the **official repositories** in favor of **GRUB version 2.x** but is still available from the `grub-legacy (https://aur.archlinux.org/packages/grub-legacy/)`AUR package.

Additionally, GRUB must be installed to the boot sector of a drive or partition to serve as a bootloader. This is covered in the **Bootloader installation** section.

Upgrading to GRUB2

Is upgrading necessary?

The short answer is No. GRUB legacy will not be removed from your system and will stay fully functional.

However, as any other packages which are not supported anymore, bugs are unlikely to be fixed. So you should consider upgrading to **GRUB version 2.x**, or one of the other supported **Boot loaders**.

GRUB legacy does not support **GPT** disks, **Btrfs** filesystem and **UEFI** firmwares.

How to upgrade

Upgrade from GRUB Legacy to **GRUB version 2.x** is much the same as installing GRUB on a running Arch Linux. Detailed instructions is covered **here**.

Differences

- There are differences in the commands of GRUB Legacy and GRUB. Familiarize yourself with **GRUB commands (https://www.gnu.org/software/grub/manual/grub.html#Commands)** before proceeding (e.g. "find" has been replaced with "search").
- GRUB is now *modular* and no longer requires "stage 1.5". As a result, the bootloader itself is limited -- modules are loaded from the hard drive as needed to expand functionality (e.g. for **LVM** or RAID support).
- Device naming has changed between GRUB Legacy and GRUB. Partitions are numbered from 1 instead of 0 while drives are still numbered from 0, and prefixed with partition-table type. For example, `/dev/sda1` would be referred to as `(hd0,msdos1)` (for MBR) or `(hd0,gpt1)` (for GPT).

- GRUB is noticeably bigger than GRUB legacy (occupies ~13 MB in `/boot`). If you are booting from a separate `/boot` partition, and this partition is smaller than 32 MB, you will run into disk space issues, and pacman will refuse to install new kernels.

Backup important data

Although a GRUB installation should run smoothly, it is strongly recommended to keep the GRUB Legacy files before upgrading to GRUB v2.

```
# mv /boot/grub /boot/grub-legacy
```

Backup the MBR which contains the boot code and partition table (replace `/dev/sdX` with your actual disk path):

```
# dd if=/dev/sdX of=/path/to/backup/mbr_backup bs=512 count=1
```

Only 446 bytes of the MBR contain boot code, the next 64 contain the partition table. If you do not want to overwrite your partition table when restoring, it is strongly advised to backup only the MBR boot code:

```
# dd if=/dev/sdX of=/path/to/backup/bootcode_backup bs=446 count=1
```

If unable to install GRUB2 correctly, see **Restore GRUB Legacy**.

Converting GRUB Legacy's config file to the new format

If `grub-mkconfig` fails, convert your `/boot/grub/menu.lst` file to `/boot/grub/grub.cfg` using:

```
# grub-menulst2cfg /boot/grub/menu.lst /boot/grub/grub.cfg
```

Note: This option works only in BIOS systems, not in UEFI systems.

For example:

```
/boot/grub/menu.lst
-------------------
default=0
timeout=5

title  Arch Linux Stock Kernel
root   (hd0,0)
kernel /vmlinuz-linux root=/dev/sda2 ro
initrd /initramfs-linux.img

title  Arch Linux Stock Kernel Fallback
root   (hd0,0)
kernel /vmlinuz-linux root=/dev/sda2 ro
initrd /initramfs-linux-fallback.img
```

```
/boot/grub/grub.cfg
-------------------
set default='0'; if [ x"$default" = xsaved ]; then load_env; set default="$saved_entry"; fi
set timeout=5

menuentry 'Arch Linux Stock Kernel' {
  set root='(hd0,1)'; set legacy_hdbias='0'
  legacy_kernel   '/vmlinuz-linux' '/vmlinuz-linux' 'root=/dev/sda2' 'ro'
  legacy_initrd '/initramfs-linux.img' '/initramfs-linux.img'
}
```

```
menuentry 'Arch Linux Stock Kernel Fallback' {
  set root='(hd0,1)'; set legacy_hdbias='0'
  legacy_kernel   '/vmlinuz-linux' '/vmlinuz-linux' 'root=/dev/sda2' 'ro'
  legacy_initrd '/initramfs-linux-fallback.img' '/initramfs-linux-fallback.img'
}
```

If you forgot to create a GRUB `/boot/grub/grub.cfg` config file and simply rebooted into GRUB Command Shell, type:

```
sh:grub> insmod legacycfg
sh:grub> legacy_configfile ${prefix}/menu.lst
```

Boot into Arch and re-create the proper GRUB `/boot/grub/grub.cfg` config file.

Restore GRUB Legacy

- Move GRUB v2 files out of the way:

```
# mv /boot/grub /boot/grub.nonfunctional
```

- Copy GRUB Legacy back to `/boot`:

```
# cp -af /boot/grub-legacy /boot/grub
```

- Replace MBR and next 62 sectors of sda with backed up copy

Warning: This command also restores the partition table, so be careful of overwriting a modified partition table with the old one. It **will** mess up your system.

```
# dd if=/path/to/backup/first-sectors of=/dev/sdX bs=512 count=1
```

A safer way is to restore only the MBR boot code use:

```
# dd if=/path/to/backup/mbr-boot-code of=/dev/sdX bs=446 count=1
```

Configuration

The configuration file is located at `/boot/grub/menu.lst`. Edit this file to suit your needs.

- `timeout #` -- time to wait (in seconds) before the `default` operating system is automatically loaded.
- `default #` -- the default boot entry that is chosen when the `timeout` has expired.

An example configuration (`/boot` is on a separate partition):

/boot/grub/menu.lst

```
# Config file for GRUB - The GNU GRand Unified Bootloader
# /boot/grub/menu.lst
# DEVICE NAME CONVERSIONS
#
#  Linux           GRUB
# -------------------------
#  /dev/fd0        (fd0)
#  /dev/sda        (hd0)
#  /dev/sdb2       (hd1,1)
#  /dev/sda3       (hd0,2)
#

#  FRAMEBUFFER RESOLUTION SETTINGS
#     +-------------------------------------------------+
#          | 640x480    800x600    1024x768   1280x1024
#      ----+--------------------------------------------
#      256 | 0x301=769  0x303=771  0x305=773   0x307=775
#      32K | 0x310=784  0x313=787  0x316=790   0x319=793
#      64K | 0x311=785  0x314=788  0x317=791   0x31A=794
#      16M | 0x312=786  0x315=789  0x318=792   0x31B=795
#     +-------------------------------------------------+
#  for more details and different resolutions see
#  https://wiki.archlinux.org/index.php/GRUB#Framebuffer_Resolution

# general configuration:
timeout   5
default   0
color light-blue/black light-cyan/blue

# boot sections follow
# each is implicitly numbered from 0 in the order of appearance below
#
# TIP: If you want a 1024x768 framebuffer, add "vga=773" to your kernel line.
#
#-*

# (0) Arch Linux
title  Arch Linux
root   (hd0,0)
kernel /vmlinuz-linux root=/dev/sda3 ro
initrd /initramfs-linux.img

# (1) Windows
#title Windows
#rootnoverify (hd0,0)
#makeactive
#chainloader +1
```

Finding GRUB's root

GRUB must be told where its files reside on the system, since multiple instances may exist (i.e., in multi-boot environments). GRUB files always reside under `/boot`, which may be on a dedicated partition.

Note: GRUB defines storage devices differently than conventional kernel naming does.

- Hard disks are defined as *(hdX)*; this also refers to any USB storage devices.
- Device and partitioning numbering begin at zero. For example, the first hard disk recognized in the BIOS will be defined as *(hd0)*. The second device will be called *(hd1)*. This also applies to partitions. So, the second partition on the first hard disk will be defined as *(hd0,1)*.

If you are unaware of the the location of `/boot`, use the GRUB shell `find` command to locate the GRUB files. Enter the GRUB shell as root by:

```
# grub
```

The following example is for systems *without* a separate `/boot` partition, wherein `/boot` is merely a directory under `/`:

```
grub> find /boot/grub/stage1
```

The following example is for systems *with* a separate `/boot` partition:

```
grub> find /grub/stage1
```

GRUB will find the file, and output the location of the `stage1` file. For example:

```
grub> find /grub/stage1

 (hd0,0)
```

This value should be entered on the `root` line in your configuration file. Type `quit` to exit the shell.

Dual booting with Windows

Add the following to the end of your `/boot/grub/menu.lst` (assuming that your Windows partition is on the first partition of the first drive):

```
/boot/grub/menu.lst

 title Windows
 rootnoverify (hd0,0)
 makeactive
 chainloader +1
```

Note:

- If you are attempting to dual-boot with Windows 7, you should comment out the line `makeactive`.

- Windows 2000 and later versions do NOT need to be on the first partition to boot (contrary to popular belief). If the Windows partition changes (i.e. if you add a partition before the Windows partition), you will need to edit the Windows `boot.ini` file to reflect the change (see **this article (http://vlaurie.com/computers2/Articles/bootini.htm)** for details on how to do that).

If Windows is located on another hard disk, the `map` command must be used. This will make your Windows install think it is actually on the first drive. Assuming that your Windows partition is on the first partition of the second drive:

```
/boot/grub/menu.lst
---
title Windows
map (hd0) (hd1)
map (hd1) (hd0)
rootnoverify (hd1,0)
makeactive
chainloader +1
```

Note: If you are attempting to dual-boot with Windows 7, you should comment out the line `makeactive`.

Dual booting with GNU/Linux

This can be done the same way that an Arch Linux install is defined. For example:

```
/boot/grub/menu.lst
```

```
title Other Linux
root (hd0,2)
kernel /path/to/kernel root=/dev/sda3 ro
initrd /path/to/initrd
```

Note: There may be other options that are required, and an initial RAM disk may not be used. Examine the other distribution's `/boot/grub/menu.lst` to match boot options, or see **chainloader and configfile** (recommended).

chainloader and configfile

To facilitate system maintenance, the `chainloader` or `configfile` command should be used to boot another Linux distribution that provides an "automagic" GRUB configuration mechanism (e.g. Debian, Ubuntu, openSUSE). This allows the distribution to manage its own `menu.lst` and boot options.

- The `chainloader` command will load another bootloader (rather than a kernel image); useful if another bootloader is installed in a partition's boot sector (GRUB, for example). This allows one to install a "main" instance of GRUB to the **MBR** and distribution-specific instances of GRUB to each partition boot record (PBR).

- The `configfile` command will instruct the currently running GRUB instance to load the specified configuration file. This can be used to load another distribution's `menu.lst` without a separate GRUB installation. The caveat of this approach is that other `menu.lst` may not be compatible with the installed version of GRUB; some distributions heavily patch their versions of GRUB.

For example, GRUB is to be installed to the MBR and some other bootloader (be it GRUB or LILO) is already installed to the boot sector of `(hd0,2)`.

```
---------------------------------------------
|   |           |           |   %           |
| M |           |           | B %           |
| B |  (hd0,0)  |  (hd0,1)  | L %  (hd0,2)  |
| R |           |           |   %           |
|   |           |           |   %           |
---------------------------------------------
  |                           ^
  |       chainloading        |
  -----------------------------
```

One can simply include in `menu.lst`:

```
title Other Linux
root (hd0,2)
chainloader +1
```

Or, if the bootloader on `(hd0,2)` is GRUB:

```
title Other Linux
root (hd0,2)
configfile /boot/grub/menu.lst
```

The `chainloader` command can also be used to load the MBR of a second drive:

```
title Other drive
rootnoverify (hd1)
```

```
chainloader +1
```

Dual booting with GNU/Linux (GRUB2)

If the other Linux distribution uses GRUB2 (e.g. Ubuntu 9.10+), and you installed its boot loader to its root partition, you can add an entry like this one to your `/boot/grub/menu.lst`:

/boot/grub/menu.lst

```
# other Linux using GRUB2
title Ubuntu
root (hd0,2)
kernel /boot/grub/core.img
```

Selecting this entry at boot will load the other distribution's GRUB2 menu assuming that the distribution is installed on `/dev/sda3`.

Bootloader installation

Manual recovery of GRUB libs

The `*stage*` files are expected to be in `/boot/grub`, which may not be the case if the bootloader was not installed during system installation or if the partition/filesystem was damaged, accidentally deleted, etc.

Manually copy the GRUB libs like so:

```
# cp -a /usr/lib/grub/i386-pc/* /boot/grub
```

Note: Do not forget to mount the system's boot partition if your setup uses a separate one! The above assumes that either the boot partition resides on the root filesystem or is mounted to /boot on the root file system!

General notes about bootloader installation

GRUB may be installed from a separate medium (e.g. a LiveCD), or directly from a running Arch install. The GRUB bootloader is *seldom* required to be reinstalled and installation is *not* necessary when:

- The configuration file is updated.
- The `grub-legacy (https://aur.archlinux.org/packages/grub-legacy/)`AUR package is updated.

Installation is *necessary* when:

- A bootloader is not already installed.
- Another operating system overwrites the Linux bootloader.
- The bootloader fails for some unknown reason.

Before continuing, a few notes:

- Be sure that your GRUB configuration is correct (`/boot/grub/menu.lst`) before proceeding. Refer to **Finding GRUB's root** to ensure your devices are defined correctly.
- GRUB must be installed on the **MBR** (first sector of the hard disk), or the first partition of the first storage device to be recognized by most BIOSes. To allow individual distributions the ability to manage their own GRUB menus, multiple instances of GRUB can be used, see **chainloader and configfile**.
- Installing the GRUB bootloader may need to be done from within a `chroot` ed environment (i.e. from installed environment via a separate medium) for cases like RAID configurations or if you forgot/broke your GRUB installation. You will need to **Change root** from a LiveCD or another Linux installation to do so.

First, enter the GRUB shell:

```
# grub
```

Use the `root` command with the output from the `find` command (see **Finding GRUB's root**) to instruct GRUB which partition contains stage1 (and therefore, `/boot`):

```
grub> root (hd1,0)
```

Tip: The GRUB shell also supports tab-completion. If you type 'root (hd' then press `Tab` twice you will see the available storage devices, this can also be done for partitions. Tab-completion also works from the GRUB boot menu. If there is an error in your configuration file you can edit in the boot menu and use tab-completion to help find devices and partitions. See **#Edit GRUB entries in the boot menu**.

Installing to the MBR

The following example installs GRUB to the **MBR** of the first drive:

```
grub> setup (hd0)
```

Installing to a partition

The following example installs GRUB to the first partition of the first drive:

```
grub> setup (hd0,0)
```

After running `setup`, enter `quit` to exit the shell. If you chrooted, **exit your chroot and unmount partitions**. Now reboot to test.

Alternate method (grub-install)

Note: This procedure is known to be less reliable, the recommended method is to use the GRUB shell.

Use the `grub-install` command followed by the location to install the bootloader. For example to install the GRUB bootloader to the MBR of the first drive:

```
# grub-install /dev/sda
```

GRUB will indicate whether it successfully installs. If it does not, you will have to use the GRUB shell method.

Tips and tricks

Additional configuration notes.

Graphical boot

For those desiring eye candy, see **grub-gfx**. **GRUB** also offers enhanced graphical capabilities, such as background images and bitmap fonts.

Framebuffer resolution

One can use the resolution given in the `menu.lst`, but you might want to use your LCD wide-screen at its full native resolution. Here is what you can do to achieve this:

On Wikipedia, there is a list of extended framebuffer resolutions (i.e. beyond the ones in the VBE standard). But, for example, the one I want to use for 1440x900 (`vga=867`) does not work. This is because the graphic card manufacturers are free to choose any number they wish, as this is not part of the VBE 3 standard. This is why these codes change from one card to the other (possibly even for the same manufacturer).

So instead of using that table, you can use one of the tools mentioned below to get the correct code:

GRUB recognized value

This is an easy way to find the resolution code using only GRUB itself.

On the kernel line, specify that the kernel should ask you which mode to use.

```
kernel /vmlinuz-linux root=/dev/sda1 ro vga=ask
```

Now reboot. GRUB will now present a list of suitable codes to use and the option to scan for even more.

You can pick the code you would like to use (do not forget it, it is needed for the next step) and boot using it.

Now replace `ask` in the kernel line with the correct one you have picked.

e.g. the kernel line for `[369] 1680x1050x32` would be:

```
kernel /vmlinuz-linux root=/dev/sda1 ro vga=0x369
```

hwinfo

1. **Install** the **hwinfo (https://www.archlinux.org/packages/?name=hwinfo)** package.
2. Run `hwinfo --framebuffer` as root.
3. Pick up the code corresponding to the desired resolution.
4. Use the 6 digit code with 0x prefix in `vga=` kernel option in `menu.lst`. Or convert it to decimal to avoid the use of 0x prefix.

Example output of **hwinfo**:

```
Mode 0x0364: 1440x900 (+1440), 8 bits
Mode 0x0365: 1440x900 (+5760), 24 bits
```

And the kernel line:

```
kernel /vmlinuz-linux root=/dev/sda1 ro vga=0x0365
```

Naming partitions

By Label

If you alter (or plan to alter) partition sizes from time to time, you might want to consider defining your drive/partitions by a label. You can label ext2, ext3, ext4 partitions by:

```
e2label /dev/drive|partition label
```

The label name can be up to 16 characters long but cannot have spaces for GRUB to understand it. Then define it in your `menu.lst`:

```
kernel /boot/vmlinuz-linux root=/dev/disk/by-label/Arch_Linux ro
```

By UUID

The **UUID** (Universally Unique IDentifier) of a partition may be discovered with `blkid` or `ls -l /dev/disk/by-uuid`. It is defined in `menu.lst` with either:

```
kernel /boot/vmlinuz-linux root=/dev/disk/by-uuid/uuid_number
```

or:

```
kernel /boot/vmlinuz-linux root=UUID=uuid_number
```

Boot as root (single-user mode)

At the boot loader, select an entry and edit it (`e` key). Append the following parameters to the kernel options:

```
[...] single init=/bin/bash
```

This will start in single-user mode (init 1), i.e. you will end up to a root prompt without being asked for password. This may be useful for recovery features, like resetting the root password. However, this is a huge security flaw if you have not set any **#Password protection** for grub.

Password protection

You can enable password protection in the GRUB configuration file for operating systems you wish to have protected. Bootloader password protection may be desired if your BIOS lacks such functionality and you need the extra security.

First, choose a password you can remember and then encrypt it:

```
# grub-md5-crypt
```

```
Password:
Retype password:
$1$ZOGor$GABXUQ/hnzns/d5JYqqjw
```

Then add your password to the beginning of the GRUB configuration file at `/boot/grub/menu.lst` (the password must be at the beginning of the configuration file for GRUB to be able to recognize it):

```
# general configuration
timeout   5
default   0
color light-blue/black light-cyan/blue

password --md5 $1$ZOGor$GABXUQ/hnzns/d5JYqqjw
```

Note: Remember that Grub uses the standard QWERTY layout for input.

Then for each operating system you wish to protect, add the `lock` command:

```
# (0) Arch Linux
title  Arch Linux
lock
root   (hd0,1)
kernel /boot/vmlinuz-linux root=/dev/disk/by-label/Arch_Linux ro
initrd /boot/initramfs-linux.img
```

Warning: If you disable booting from other boot devices (like a CD drive) in the BIOS's settings and then password protect all your operating system entries, it could be difficult to re-enable booting back into the operating systems if the password is forgotten.

It is always possible to reset your BIOS settings by setting the appropriate jumper on the motherboard (see your motherboard's manual, as it is specific to every model). So in case other have access to the hardware, there is basically no way to prevent boot breakthroughs.

Restart with named boot choice

If you realize that you often need to switch to some other non-default OS (e.g. Windows) having to reboot and wait for the GRUB menu to appear is tedious. GRUB offers a way to record your OS choice when restarting instead of waiting for the menu, by designating a temporary new default which will be reset as soon as it has been used.

Supposing a simple `menu.lst` setup like this:

```
/boot/grub/menu.lst

# general configuration:
timeout 10
default 0
color light-blue/black light-cyan/blue

# (0) Arch
title  Arch Linux
root (hd0,1)
kernel /boot/vmlinuz-linux root=/dev/disk/by-label/ARCH ro
initrd /boot/initramfs-linux.img

# (1) Windows
title Windows XP
rootnoverify (hd0,0)
makeactive
chainloader +1
```

Arch is the default (0). We want to restart in to Windows. Change `default 0` to `default saved` -- this will record the current default in a `default` file in the GRUB directory whenever the **savedefault** command is used. Now add the line `savedefault 0` to the bottom of the Windows entry. Whenever Windows is booted, it will reset the default to Arch, thus making changing the default to Windows temporary.

Now all that is needed is a way to easily change the default manually. This can be accomplished using the command `grub-set-default`. So, to reboot into Windows, enter the following commands:

```
# grub-set-default 1
```

Then reboot.

For ease of use, you might to wish to implement the "**Allow users to shutdown** fix" (including `/sbin/grub-set-default` amongst the commands the user is allowed to issue without supplying a password).

LILO and GRUB interaction

If the **LILO** package is installed on your system, **remove** it. As some tasks (e.g. kernel compilation using `make all`) will make a LILO call, and LILO will then be installed over GRUB. LILO may have been included in your base system, depending on your installer media version and whether you selected/deselected it during the package selection stage.

Note: Removing `lilo (https://aur.archlinux.org/packages/lilo/)`AUR will not remove LILO from the MBR if it has been installed there; it will merely remove the `lilo (https://aur.archlinux.org/packages/lilo/)`AUR package. The LILO bootloader installed to the MBR will be overwritten when GRUB (or another bootloader) is installed over it.

GRUB boot disk

First, format a floppy disk:

```
# fdformat /dev/fd0
# mke2fs /dev/fd0
```

Now mount the disk:

```
# mount -t ext2 /dev/fd0 /mnt/fl
```

Install GRUB to the disk:

```
# grub-install --root-directory=/mnt/fl '(fd0)'
```

Copy your `menu.lst` file to the disk:

```
# cp /boot/grub/menu.lst /mnt/fl/boot/grub/menu.lst
```

Now unmount your floppy:

```
# umount /mnt/fl
```

Now you should be able to restart your computer with the disk in the drive and it should boot to GRUB. Make sure that your floppy disk is set to have higher priority than your hard drive when booting in your BIOS first, of course.

See also: **Super GRUB Disk (http://www.supergrubdisk.org/)**.

Hide GRUB menu

The `hiddenmenu` option can be used in order to hide the menu by default. That way no menu is displayed and the default option is going to be automatically selected after the timeout passes. Still, you are able to press `Esc` and the menu shows up. To use it, just add to your `/boot/grub/menu.lst`:

```
hiddenmenu
```

Advanced debugging

See dedicated **article**.

Troubleshooting

GRUB Error 17

Note: the solution below works also for GRUB Error 15

The first check to do is to unplug any external drive. Seems obvious, but sometimes we get tired ;)

If your partition table gets messed up, an unpleasant "GRUB error 17" message might be the only thing that greets you on your next reboot. There are a number of reasons why the partition table could get messed up. Commonly, users who manipulate their partitions with **GParted** -- particularly logical drives -- can cause the order of the partitions to change. For example, you delete `/dev/sda6` and resize `/dev/sda7`, then finally re-create what used to be `/dev/sda6` only now it appears at the bottom of the list, `/dev/sda9` for example. Although the physical order of the partitions/logical drives has not changed, the order in which they are recognized has changed.

Fixing the partition table is easy. Boot from your Arch CD/DVD/USB, login as root and fix the partition table:

```
# fdisk /dev/sda
```

Once in disk, enter e[x]tra/expert mode, [f]ix the partition order, then [w]rite the table and exit.

You can verify that the partition table was indeed fixed by issuing an `fdisk -l`. Now you just need to fix GRUB. See the **Bootloader installation** section.

Basically you need to tell GRUB the correct location of your `/boot` then re-write GRUB to the **MBR** on the disk.

For example:

```
# grub
```

```
grub> root (hd0,6)
grub> setup (hd0)
grub> quit
```

See **[1] (http://stringofthoughts.wordpress.com/2009/05/24/grub-error-17-debianubuntu)** for a more in-depth summary of this section.

/boot/grub/stage1 not read correctly

If you see this error message while trying to set up GRUB, and you are not using a fresh partition table, it is worth checking it.

```
# fdisk -l /dev/sda
```

This will show you the partition table for `/dev/sda`. So check here, whether the "Id" values of your partitions are correct. The "System" column will show you the description of the "Id" values.

If your boot partition is marked as being "HPFS/NTFS", for example, then you have to change it to "Linux". To do this, go to fdisk,

```
# fdisk /dev/sda
```

change a partition's system id with `t`, select you partition number and type in the new system id (Linux = 83). You can also list all available system ids by typing `L` instead of a system id.

If you have changed a partitions system id, you should [v]erify your partition table and then [w]rite it.

Now try to set up GRUB again.

See also the **forum post (https://bbs.archlinux.org/viewtopic.php?pid=799930)** reporting this problem.

Accidental install to a Windows partition

If you accidentally install GRUB to a Windows partition, GRUB will write some information to the boot sector of the partition, erasing the reference to the Windows bootloader. (This is true for NTLDR the bootloader for Windows XP and earlier, unsure about later versions).

To fix this you will need to use the Windows Recovery Console for your Windows release. Because many computer manufacturers do not include this with their product (many choose to use a recovery partition) Microsoft has made them available for download. If you use XP, look at **this page (http://tips.vlaurie.com/2006/05/recovery-console-for-those-without-an-xp-disk/)** to be able to turn the floppy disks to a Recovery CD. Boot the Recovery CD (or enable Windows Recovery mode) and run `fixboot` to repair the partition boot sector. After this, you will have to install GRUB again---this time, to the MBR, not to the Windows partition---to boot Linux.

See **Dual boot with Windows#Restoring a Windows boot record** for more information.

Edit GRUB entries in the boot menu

Once you have selected and entry in the boot menu, you can edit it by pressing key `e`. Use tab-completion if you need to to discover devices then `Esc` to exit. Then you can try to boot by pressing `b`.

Note: These settings **will not be saved**.

device.map error

If an error is raised mentioning `/boot/grub/device.map` during installation or boot, run:

```
# grub-install --recheck /dev/sda
```

to force GRUB to recheck the device map, even if it already exists. This may be necessary after resizing partitions or adding/removing drives.

KDE reboot pull-down menu fails

If you have opened a sub-menu with the list of all operating systems configured in GRUB, selected one, and upon restart, you still booted your default OS, then you might want to check if you have the line:

```
default saved
```

in `/boot/grub/menu.lst`.

GRUB fails to find or install to any virtio */dev/vd** or other non-BIOS devices

I had trouble installing GRUB while installing Arch Linux in an virtual KVM machine using a virtio device for hard drive. To install GRUB, I figured out the following: Enter a virtual console by typing `Ctrl+Alt+F2` or any other F-key for a free virtual console. This assumes that your root file system is mounted in the folder `/mnt` and the boot file system is either mounted or stored in the folder `/mnt/boot`.

1. Assure that all needed GRUB files is present in your boot directory (assuming it is mounted in `/mnt/boot` folder), by issuing the command:

```
# ls /mnt/boot/grub
```

2. If the `/mnt/boot/grub` folder already contains all the needed files, jump to step 3. Otherwise, do the following commands (replacing `/mnt`, `your_kernel` and `your_initrd` with the real paths and file names). You should also have the `menu.lst` file written to this folder:

```
# mkdir -p /mnt/boot/grub                # if the folder is not yet present
# cp -r /boot/grub/stage1 /boot/grub/stage2 /mnt/boot/grub
# cp -r your_kernel your_initrd /mnt/boot
```

3. Start the GRUB shell with the following command:

```
# grub --device-map=/dev/null
```

4. Enter the following commands. Replace `/dev/vda`, and `(hd0,0)` with the correct device and partition corresponding to your setup.

```
device (hd0) /dev/vda
root (hd0,0)
setup (hd0)
quit
```

5. If GRUB reports no error messages, then you probably are done. You also need to add appropriate modules to the ramdisk. For more information, please refer to **QEMU#Preparing an (Arch) Linux guest**.

See also

- **GNU GRUB (http://www.gnu.org/software/grub/)**
- **GRUB Grotto (http://www.troubleshooters.com/linux/grub/index.htm)**
- **Boot debugging** - Debugging with GRUB, set module values

Retrieved from "https://wiki.archlinux.org/index.php?title=GRUB_Legacy&oldid=502209"