gentoo linux (/) **Wiki**

GRUB2

From Gentoo Wiki

GRUB2 (the **GR**and **U**nified **B**ootloader version **2**) is a replacement for the original GRUB (/wiki/GRUB) boot loader, now referred to as "GRUB Legacy". GRUB2 has an entirely separate code base from GRUB Legacy, and features a new shell-like syntax for advanced scripting capabilities.

For a shotgun approach, see GRUB2 Quick Start (/wiki/GRUB2_Quick_Start).

If migrating from GRUB Legacy to GRUB2, see GRUB2 Migration (/wiki/GRUB2_Migration).

Resources
Home (http://gnu.org/software/grub/)
Official documentation (http://gnu.org/software/grub/grub-documentation.html)
Package information (https://packages.gentoo.org/packages/sys-boot/grub)
Wikipedia (https://en.wikipedia.org/wiki/GNU_GRUB%23GRUB%202)
GitWeb (http://git.savannah.gnu.org/cgit/grub.git/)

Contents

Installation

The sys-boot/grub (https://packages.gentoo.org/packages/sys-boot/grub) package is slotted. Both **grub-0.97** (GRUB Legacy (/wiki/GRUB)) and **grub-2.02** may be installed at the same time; however, only one version at a time may be installed in the Master Boot Record (MBR) of a hard drive.

Prerequisites

To control which platforms GRUB2 will install for, set the *GRUB_PLATFORMS* variable in `make.conf`. The **amd64** architecture includes a profile default which works for most systems.

FILE `/etc/portage/make.conf` **Example of setting the *GRUB_PLATFORMS* variable for EMU, EFI, and PC platforms**

```
GRUB_PLATFORMS="emu efi-32 efi-64 pc"
```

The following platforms are supported depending on the target CPU:

	Target						
Platform	**i386**	**ia64**	**mips**	**mipsel**	**powerpc**	**spac64**	**x86_64**
arc	✖ No	✖ No	✖ No	✔ Yes	✖ No	✖ No	✖ No
coreboot	✔ Yes	✖ No	✖ No	✖ No	✖ No	✖ No	32-bit
efi	✔ Yes	✔ Yes	✖ No	✖ No	✖ No	✖ No	✔ Yes
emu	✔ Yes	✔ Yes	✔ Yes	✔ Yes	✔ Yes	✔ Yes	✔ Yes
ieee1275	✔ Yes	✖ No	✖ No	✖ No	✔ Yes	✔ Yes	32-bit
loongson	✖ No	✖ No	✖ No	✔ Yes	✖ No	✖ No	✖ No

multiboot	✔ Yes	✖ No	✖ No	✖ No	✖ No	✖ No	32-bit
qemu	✔ Yes	✖ No	✖ No	✖ No	✖ No	✖ No	32-bit
qemu-mips	✖ No	✖ No	✔ Yes	✖ No	✖ No	✖ No	✖ No
pc	✔ Yes	✖ No	✖ No	✖ No	✖ No	✖ No	32-bit

Note
Whenever the values in the *GRUB_PLATFORMS* variable are adjusted GRUB2 will need to be re-emerged in order to build the changed binary. Be sure to use the `--newuse --deep` options as shown in the emerge section below.

The **amd64** profiles enable support for (U)EFI functionality by default. When using a BIOS-based system, set *GRUB_PLATFORMS* variable to `pc` to avoid unneeded dependencies.

USE flags

USE flags for sys-boot/grub (https://packages.gentoo.org/packages/sys-boot/grub) GNU GRUB boot loader

(https://packages.gentoo.org/useflags/debug)	meaningful backtraces see https://wiki.gentoo.org/wiki/Project:Quality_Assurance/Backtraces	
device-mapper (https://packages.gentoo.org/useflags/device-mapper)	Enable support for device-mapper from sys-fs/lvm2	local
doc (https://packages.gentoo.org/useflags/doc)	Add extra documentation (API, Javadoc, etc). It is recommended to enable per package instead of globally	global
efiemu (https://packages.gentoo.org/useflags/efiemu)	Build and install the efiemu runtimes	local

More information about USE flags

Data provided by the Gentoo Package Database (https://packages.gentoo.org) · Last update: 2018-02-04 00:55

(/wiki/Handbook:AMD64/Working/USE)

Emerge

To install GRUB2 use the normal **emerge** syntax:

```
root # emerge --ask --newuse --deep sys-boot/grub:2
```

Additional software

Optionally, install the **os-prober** utility (provided through the sys-boot/os-prober (https://packages.gentoo.org/packages/sys-boot/os-prober) package) to have GRUB2 probe for other operating systems when running the **grub-mkconfig** command. In most instances, this will enable GRUB2 to automatically detect other operating systems including Windows 7, 8.1, 10, other distributions of Linux, etc.

```
root # emerge --ask --newuse sys-boot/os-prober
```

The GRUB2 (and optionally sys-boot/os-prober (https://packages.gentoo.org/packages/sys-boot/os-prober)) installations do not automatically enable the boot loader. These only install the software on the operating system, but to install the boot loader to the system itself (so that it is used when booting the system), additional steps need to be taken, which are covered in the Configuration section.

Configuration

There are two important aspects to the configuration of GRUB2:

1. Installation of GRUB2 software as the boot loader of the system.
2. Configuration of the GRUB2 boot loader.

The installation of GRUB2 software is specific to the type of system, and is covered in Installing the boot loader. First we cover the configuration of the boot loader itself.

Main configuration file

The **grub-mkconfig** script is used to generate a grub configuration. It uses the scripts under `/etc/grub.d/*` together with the `/etc/default/grub` configuration file to generate the final `/boot/grub/grub.cfg` - the only configuration file used by GRUB2 itself.

File	Format	Edits recommended?	Description
`/usr/sbin/grub-mkconfig`	POSIX shell script	✖ No	Installed as part of the sys-boot/grub (https://packages.gentoo.org/packages/sys-boot/grub):2 package. Run this script to generate `/boot/grub/grub.cfg` after configuring the files described below.
`/boot/grub/grub.cfg`	GRUB2 shell script		The file generated by **grub-mkconfig**. This file is evaluated by GRUB2's built-in script interpreter and doesn't necessarily support all POSIX commands or

		✖ No	syntax. See the scripting reference (https://www.gnu.org/software/grub/manual/grub.html#Shell_002dlike-scripting) in the GRUB manual for supported features. Be aware that modifications to this file won't persist to the next time grub-mkconfig is run.
`/etc/grub.d/*`	POSIX shell scripts	➖ Maybe	Each script under `/etc/grub.d/*` that has its execute bit set is evaluated in sequence, and the stdout is concatenated to form the final `/boot/grub/grub.cfg` (or whatever file is given to the **`grub-mkconfig`** `-o` option). These scripts use the current system shell and therefore can use any supported syntax. Ideally they should be POSIX-compatible scripts, and the output script must be compatible with the GRUB2 interpreter. It may be necessary to disable or add scripts. For instance, to add menu items that couldn't be automatically generated.
`/boot/grub/custom.cfg`	GRUB2 shell script	➖ Maybe	The /etc/grub.d/41_custom script will reference this file to be read in at boot time if it exists. This file provides a place to add additional entries or commands and does not require regeneration of the main grub.cfg file.
`/etc/default/grub`	POSIX shell script	✔ Yes	In most cases this is the only file that should be modified directly. It is mainly used to assign variables used by the scripts in `/etc/grub.d` to generate a working configuration file. See GRUB2 configuration variables (/wiki/GRUB2/Config_Variables) or the official reference (https://www.gnu.org/software/grub/manual/grub.html#Simple-configuration) for supported variables.

GRUB2 does not require the administrator to manually maintain a boot option configuration (as is the case with boot loaders such as GRUB Legacy (/wiki/GRUB) and LILO (/wiki/LILO)). Instead it can generate its configuration file (`/boot/grub/grub.cfg`) using the **`grub-mkconfig`** command. This utility will use the scripts in `/etc/grub.d/` and the settings in `/etc/default/grub`.

> **⚠ Warning**
> The **`grub-mkconfig`** utility does not work properly when using software RAID. Manual configuration of the scripts in `/etc/grub.d/` is necessary, as otherwise after installation the system will be left in a non-bootable state.

After modifying one or more settings, run the **`grub-mkconfig`** utility with the `-o` option pointing to the output file located at `/boot/grub/grub.cfg` (this is GRUB2's default output location):

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

```
Generating grub.cfg ...
Found linux image: /boot/vmlinuz-3.3.0-gentoo
done
```

```
done
```

Each time the **grub-mkconfig** utility is called a new configuration will be generated.

> **⚠ Warning**
> If **grub-mkconfig** does not report any found entries then no entries were found. In this case GRUB2 will offer no boot selections when upon system restart which may be a tricky, time consuming situation to resolve. Make sure the output is satisfactory before restarting the system.

Setting configuration parameters

The following variables in `/etc/default/grub` are the most common ones to set to control how GRUB2 will function:

Variable	Explanation	Default value
GRUB_DEFAULT	Defines the default menu entry selected on boot. May be a numeric index, a menu title, or "saved".	Defaults to first detected entry.
GRUB_TIMEOUT	Delay (in seconds) before booting default menu entry. Set to `0` to boot immediately or `-1` to wait indefinitely.	The default is 5 seconds.
GRUB_CMDLINE_LINUX	Parameters to be passed on the kernel command line for all Linux menu entries. For instance, to support hibernation, users will need to add `GRUB_CMDLINE_LINUX="resume=/dev/sdXY"` with `/dev/sdXY` being the swap partition.	
GRUB_CMDLINE_LINUX_DEFAULT	Parameters to be passed on the kernel command line for non-recovery Linux menu entries.	
GRUB_DEVICE	The initial root device (i.e. the kernel's `root=` parameter). Set this to override the **grub-mkconfig** command's root device auto-detection. For example, `GRUB_DEVICE=/dev/ram0` will force `root=/dev/ram0` to be used in the kernel command line.	

For a more complete list, please refer to the GRUB2 configuration variables (/wiki/GRUB2/Config_Variables) sub-page.

After modifying the parameters, regenerate the GRUB2 configuration file with **grub-mkconfig**.

Enabling or disabling configuration scripts

The directory `/etc/grub.d/` contains the scripts that **`grub-mkconfig`** uses to generate a `grub.cfg` file. By default the contents of this directory should be similar to the following:

```
user $ ls /etc/grub.d/
```

```
00_header  10_linux  20_linux_xen  30_os-prober  40_custom  41_custom README
```

GRUB2 will use all installed scripts that are marked as executable (which by default, they all are). To disable any of the scripts simply remove the executable bit from the script's file permissions using the **`chmod`** command. In the following example every script but `00_header` and `10_linux` are disabled:

```
root # chmod -x /etc/grub.d/{20_linux_xen,30_os-prober,40_custom,41_custom}
```

After modifying the scripts (or removing the executable bit), regenerate the GRUB2 configuration file using **`grub-mkconfig`**.

Manipulating configuration scripts

Some features or GRUB2 functionalities are only possible to be exploited by modifying the configuration scripts. For instance, to support dual-booting with FreeBSD, the following manipulation needs to be done.

Change the `/etc/grub.d/40_custom` script to:

FILE **`/etc/grub.d/40_custom`** **Adding an entry for dual booting**

```
menuentry "FreeBSD" --class freebsd --class bsd --class os {
 insmod ufs2
 insmod bsd
 set root=(hd0,1)
 kfreebsd /boot/kernel/kernel
 kfreebsd_loadenv /boot/device.hints
 set kFreeBSD.vfs.root.mountfrom=ufs:/dev/ada0s1a
 set kFreeBSD.vfs.root.mountfrom.options=rw
 set kFreeBSD.hw.psm.synaptics_support=1
}
```

`/dev/sda1` or `(hd0,1)` is the partition in which FreeBSD resides. If the normal UFS install was used for the FreeBSD partition then `/dev/sda1` is a container (something like a logical partition). It consists of the swap and root partition. Verify the `40_custom` script is executable by running **`ls -la /etc/grub.d/40_custom`**. If the executable bit is not set then set it using the **`chmod u+x 40_custom`** command.

Note
Users familiar with how GRUB Legacy numbered partitions should note that GRUB2 numbers partitions starting from 1, not 0.

Next install GRUB2 using the **grub-install** command and update GRUB2's configuration file:

```
root # grub-install /dev/sda
root # grub-mkconfig -o /boot/grub/grub.cfg
```

Installing the boot loader

Installing GRUB2 as the system's boot loader depends on how the system is meant to boot (through BIOS (/wiki/BIOS) or UEFI) and how the disk on which the boot loader should be installed is partitioned (using MBR or GPT partition layout).

This article covers the following situations:

- BIOS with MBR
- BIOS with GPT
- UEFI with GPT

Select the installation instructions appropriate for the system.

BIOS with MBR

> **Note**
> When the system is meant to dual-boot with Microsoft Windows, make sure that the system itself does *not* have an UEFI firmware. Even when such systems are booted in 'legacy BIOS' mode, Microsoft Windows will refuse to boot.

Make sure that the `/boot` location is available - if this uses a separate partition, make sure that it is mounted:

```
root # mount /boot
```

Run the **grub-install** command to copy the relevant files to `/boot/grub`. On the PC platform, this also installs a boot image to the Master Boot Record (MBR) or a partition's boot sector. If all goes well, after running **grub-install** an output such as the one below is to be expected:

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

```
Installation finished. No error reported.
```

grub-install accepts a `--target` option to set the CPU architecture and system platform. If unspecified, **grub-install** will attempt to guess the proper values; on an **amd64**/**x86** system it will use `i386-pc` by default. **grub-install** also accepts a `--boot-directory` option to tell the GRUB2 installer which directory to look for GRUB2's boot files. This defaults to the current `/boot` but is useful when trying to move a root partition.

Partitioning for BIOS with MBR

Be sure to leave enough free space before the first partition. Starting the first partition at sector 2048 leaves at least 1 MiB of disk space for the master boot record. It is recommended (but not mandatory) to create an additional partition for GRUB called the *BIOS boot partition*. This partition just needs to be defined, but not formatted. It is only needed if the system is later migrated to the GPT partition layout. When sticking with MBR, this is not needed.

If the Gentoo installation instructions (/wiki/Handbook:Main_Page) were followed, this BIOS boot partition will already be available.

BIOS with GPT

> **Note**
> When the system is meant to dual-boot with Microsoft Windows, make sure that the system itself does *not* have an UEFI firmware. Even when such systems are booted in 'legacy BIOS' mode, Microsoft Windows will refuse to boot. Also, older Microsoft Windows systems might not support GPT. It is possible to use a hybrid MBR-GPT approach; see Hybrid partition table (/wiki/Hybrid_partition_table).

If a `/boot` partition is needed, start by mounting the `/boot` partition:

```
root # mount /boot
```

If all goes well, after running the **grub-install** command an output such as the one below is to be expected:

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

```
Installation finished. No error reported.
```

grub-install accepts a `--target` option to set the CPU architecture and system platform. If unspecified, **grub-install** will attempt to guess the correct values; on an **amd64**/**x86** system it will use `i386-pc` by default. **grub-install** also accepts a `--boot-directory` option to tell the GRUB2 installer which directory to look in for GRUB2's boot files. This defaults to the current `/boot` but is useful when trying to move a root partition.

Partitioning for BIOS with GPT

When a GPT partition table is present on the system, a small *BIOS boot partition* with type `EF02` (which is different from the *EFI System Partition (ESP)* which has type `EF00`) will need to be available. 1 MiB will be enough to work, but 2-4 MiB is a safer option. This BIOS boot partition will hold the stage 2 of the bootloader. BIOS boot partitions do not need to be formatted with a filesystem; the **grub-install** command will overwrite any existing filesystem with one of its own.

> **Important**
> The BIOS boot partition is *not* the same partition that is commonly mounted at `/boot`. The `/boot` and BIOS boot are different partitions and should be handled separately. The BIOS boot partition should *not* be regularly mounted on the system (i.e., it should *not* be defined in `/etc/fstab`). The `/boot` partition *can* be regularly mounted with no issues and therefore can be present in the `/etc/fstab` file.

To set a partition as a BIOS partition use the command-line tool **parted** (sys-block/parted (https://packages.gentoo.org/packages/sys-block/parted)) by typing (change `1` to the number of the partition to mark as a BIOS boot partition!):

```
(parted) set 1 bios_grub on
```

With sys-apps/gptfdisk (https://packages.gentoo.org/packages/sys-apps/gptfdisk)'s **cgdisk** utility, this is accomplished by setting the partition type to `0xEF02` and giving it a label of `gptbios`.

An EFI System Partition is not required, but it would be sensible to make sure that the BIOS boot partition is large enough to be converted to one, should the system motherboard later be upgraded to an UEFI board.

The following is the output of pressing the p key using the **gdisk** utility on a GPT-partitioned disk with both a BIOS boot [0xEF02] partition and an EFI [0xEF00] partition:

```
root # gdisk /dev/sdc
```

```
GPT fdisk (gdisk) version 0.8.1

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.

Command (? for help): p
Disk /dev/sdc: 976773168 sectors, 465.8 GiB
Logical sector size: 512 bytes
Disk identifier (GUID): AA369F4D-37A4-4C0D-A357-DC24B99A6337
Partition table holds up to 128 entries
First usable sector is 34, last usable sector is 976773134
Partitions will be aligned on 2048-sector boundaries
Total free space is 2014 sectors (1007.0 KiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048       828377087   395.0 GiB   8E00  Linux LVM
   2       828377088       891291647   30.0 GiB    0700  Microsoft basic data
   3       891291648       975177727   40.0 GiB    0700  Microsoft basic data
   4       975177728       976754687   770.0 MiB   8300  Linux filesystem
   5       976754688       976756735   1024.0 KiB  EF02  BIOS boot partition
   6       976756736       976773134   8.0 MiB     EF00  EFI System

Command (? for help):
```

Note
The `0x` hexadecimal prefix does not need to be entered for GPT when using **fdisk**.

Using the same setup, the **parted** utility gives output with slightly different syntax:

```
root # parted /dev/sdc
```

```
GNU Parted 3.0
Using /dev/sdc
(parted) print
...
Sector size (logical/physical): 512B/512B
Partition Table: gpt

Number  Start   End    Size    File system  Name                  Flags
 1      1049kB  424GB  424GB                Linux LVM             lvm
 2      424GB   456GB  32.2GB               Microsoft basic data
 3      456GB   499GB  42.9GB               Microsoft basic data
 4      499GB   500GB  807MB   ext2         Linux filesystem
 5      500GB   500GB  1049kB               BIOS boot partition   bios_grub
 6      500GB   500GB  8396kB               EFI System            boot

(parted)
```

Creating partitions in **gdisk** is straightforward for users familiar with the **fdisk** partitioning utility. After starting **gdisk**, type n (for new) in the main menu, provide beginning and end sectors (if needed), and set the partition type to `EF00` for an EFI system partition.

Users who have followed the Gentoo installation instructions (/wiki/Handbook:Main_Page) will already have the proper partitioning layout set up.

UEFI with GPT

Make sure that the `/boot` location is available - if this uses a separate partition, make sure that it is mounted:

```
root # mount /boot
```

Run the **grub-install** command to copy the relevant files to `/boot/grub`. This should install GRUB2 in `/boot/grub`, copy the core image to `/boot/efi/EFI/gentoo/grubx64.efi`, and call efibootmgr (/wiki/Efibootmgr) to add a boot entry.

```
root # grub-install --efi-directory=/boot/efi
```

```
Installation finished. No error reported.
```

The above command assumes the vfat EFI System Partition (/wiki/EFI_System_Partition) (ESP) is mounted at `/boot/efi`. If the ESP is mounted directly at `/boot`, use `--efi-directory=/boot` .

grub-install also accepts a `--target` option to set the CPU architecture and system platform. If unspecified, **grub-install** will attempt to guess the proper values; on an AMD64 UEFI-booted system it will use `x86_64-efi` by default. **grub-install** also accepts a `--boot-directory` option to tell the GRUB2 installer which directory to look for GRUB2's boot files. This defaults to `/boot` but is useful when trying to move a root partition.

Partitioning for UEFI with GPT

For UEFI GPT boot using GRUB2, the system *must* have a dedicated EFI partition containing a FAT filesystem.

The EFI partition can *replace* having a `/boot` partition on `/dev/sda1` by having a `/boot/efi` partition on `/dev/sda1`. This is to say a successful UEFI boot scenario using GRUB2 *can* operate with two partitions total (three total if a swap partition is needed): a root partition and an EFI partition. Using this configuration, the `/boot` *folder* will be located in the root `/` *partition* (at `/boot`) and the EFI *partition* will mount *in* the boot *folder* (at `/boot/efi`). For further clarification, see the example `/etc/fstab (/wiki/Handbook:AMD64/Installation/System#About_fstab)` file below.

FILE `/etc/fstab` **Example of an UEFI capable `/etc/fstab` file with a swap partition:**

```
/dev/sda1               /boot/efi       vfat            noauto,noatime  1 2
/dev/sda2               none            swap            sw              0 0
/dev/sda3               /               ext4            noatime         0 1
```

Generating a 100MB partition for `/boot/efi` should provide plenty of space for holding multiple

- .efi files (multiple entries will most likely not be needed; most systems will only use one).

Create the partition using the partitioning tool of choice (/wiki/Partition#GUID_Partition_Table_.28GPT.29). The **gdisk** (sys-apps/gptfdisk (https://packages.gentoo.org/packages/sys-apps/gptfdisk)) and **parted** (sys-block/parted (https://packages.gentoo.org/packages/sys-block/parted)) tools fit nicely for this purpose. When using the **gdisk** utility, be sure to use type `EF00`.

Proceed to create a FAT filesystem on the EFI system partition using **mkfs.fat** and add it to `/etc/fstab` by following the example below:

```
root # mkfs.fat -F 32 -n efi-boot /dev/sda1
root # mkdir /boot/efi
```

FILE `/etc/fstab` **Adding the /boot/efi mount entry**

```
/dev/sda1               /boot/efi       vfat            noauto,noatime  1 2
```

```
root # mount /boot/efi
```

Note
It is helpful to set the *GRUB_PLATFORMS* variable in `/etc/portage/make.conf`. This will assist GRUB2 in determining what options to use when detecting the proper EFI target. For 32-bit UEFI systems use `efi-32`. For 64-bit use `efi-64`.

Important
In order for GRUB2 to install properly, the EFI directory *must* be mounted *and* the `efivars` kernel module must be loaded before the **grub-install** command will complete successfully.

Alternative: using the default UEFI firmware location

Alternative: using the default UEFI firmware location

If the system's UEFI firmware fails to find GRUB2's EFI bootloader file, using the default boot loader location should provide a working solution. This circumvents the boot menu managed by efibootmgr (/wiki/Efibootmgr) and thus offers reduced functionality, but is less error prone. To do this, verify the EFI partition is mounted at `/boot/efi` then copy the file `grubx64.efi` located at `/boot/efi/EFI/gentoo/grubx64.efi` to `/boot/efi/EFI/BOOT/BOOTX64.EFI`. This example assumes a 64-bit UEFI system, adjust accordingly for 32-bit UEFI systems.

Extended features

GRUB2 has many features that make it a very powerful boot loader. It supports:

- Booting from UEFI platforms.
- Booting from GPT partitioned drives without needing a hybrid MBR (hybrid MBR can enabled as needed for compatibility or portability).
- Booting from a btrfs (/wiki/Btrfs) formatted `/boot` partition.
- Booting from a ZFS pool.
- Booting directly from a btrfs (/wiki/Btrfs) raid set without needing an initramfs (/wiki/Initramfs) for early mount setup.
- Booting directly from logical volume management (such as LVM2 (/wiki/LVM)).
- Booting with support for DM-RAID (RAID 0, 1, 4, 5, 6, 9 and 10).
- Booting from encrypted devices (LUKS).

Some specific features are explained in more detail next.

Chainloading

GRUB2 was built with a truly improved chainload mode when compared to GRUB Legacy. To chainload another boot loader, use the `chainloader` option.

FILE `/etc/grub.d/40_custom` **Chainloading another bootloader**

```
menuentry "Custom Super-bootloader example" {
     insmod part_msdos
     insmod chain
     chainloader (hd1,1)+1
}
```

For more information on chainloading, please see the Chainloading (/wiki/GRUB2/Chainloading) sub-page.

Using framebuffer display

To have GRUB2 use a framebuffer (/wiki/Framebuffer) graphical display, re-emerge GRUB with the `truetype` USE flag enabled. This will install a default True Type font as well as a font conversion utility.

```
root # emerge --ask --newuse sys-boot/grub:2
```

Proceed to configure the default GRUB2 configuration file located at `/etc/default/grub`. For example:

Proceed to configure the default GRUB2 configuration file located at `/etc/default/grub`. For example:

FILE `/etc/default/grub` **Framebuffer related settings**

```
# Set resolution and color depth
GRUB_GFXMODE=1366x768x32

# Keep resolution when loading the kernel
GRUB_GFXPAYLOAD_LINUX=keep

# Set a background image
GRUB_BACKGROUND="/boot/grub/bg.png"

# Use a custom font, converted using grub-mkfont utility
GRUB_FONT="/boot/grub/fonts/roboto.pf2"
```

In order to find out what display modes the system's graphics card supports, use the following commands on the GRUB2 shell:

```
(grub) insmod all_video
(grub) videoinfo
```

Troubleshooting

Most of the issues can be resolved by ensuring that the partition layout is correct. Make sure enough space is available before the first partition of the disk, or optionally make sure that a *BIOS boot partition* is available. Also verify that `/boot/grub/grub.cfg` was correctly generated with **grub-mkconfig**, or generate one with a custom menu entry.

For more GRUB2 troubleshooting, please refer to the Troubleshooting (/wiki/GRUB2/Troubleshooting) sub-article.

Motherboard firmware not finding the .EFI file

Some motherboard manufacturers seem to only support one location for the .EFI file in the EFI System Partition (ESP). If this seems to be the case, simply move GRUB's default file to the `/efi/boot/` location. First, make sure the ESP is mounted. Presuming the ESP is mounted at `/boot/efi` (as suggested in the Handbook (/wiki/Handbook)), execute:

```
root # mkdir -p /boot/efi/efi/boot
root # cp /boot/efi/efi/gentoo/grubx64.efi /boot/efi/efi/boot/bootx64.efi
```

This should aid the motherboard firmware in loading the GRUB executable. Reboot the system to see if the firmware now correctly loads GRUB.

os-prober and UEFI in chroot

The sys-boot/os-prober (https://packages.gentoo.org/packages/sys-boot/os-prober) utility is used to discover alternate installs, such as Microsoft Windows. To function properly, it needs to have access to information from the live environment's udev to test for the EFI System Partition.

Run these commands in the host environment to provide the required files (example shows Gentoo mounted on `/mnt/gentoo` like in the Handbook (/wiki/Handbook:AMD64)):

```
root # mkdir -p /mnt/gentoo/run/udev
root # mount -o bind /run/udev /mnt/gentoo/run/udev
root # mount --make-rslave /mnt/gentoo/run/udev
```

Installing a new kernel

Whenever a new kernel is installed, GRUB2 must be reconfigured to recognize it. This can be done using **`grub-mkconfig`**, as shown below, or can be done manually (/wiki/GRUB2_Quick_Start#Manual_configuration).

> **Note**
> Make sure the `/boot` partition is mounted for this step.

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

```
Generating grub.cfg ...
Found linux image: /boot/kernel-3.3.8-gentoo
Found initrd image: /boot/initramfs-genkernel-x86_64-3.3.8-gentoo
Found linux image: /boot/kernel-3.2.12-gentoo
Found initrd image: /boot/initramfs-genkernel-x86_64-3.2.12-gentoo
done
```

Note that GRUB2 only needs to be reconfigured, not *reinstalled* to the boot drive's Master Boot Record (MBR). On the other hand, when GRUB2 itself has been upgraded it does need to be reinstalled on the boot drive, but usually does not need to be reconfigured.

See also

There are a few specific GRUB2 resources available:

- In Chainloading (/wiki/GRUB2/Chainloading) the use of GRUB2 to boot other boot loaders is described. This is important to read when dual-booting systems, or when GRUB2 needs to be configured to boot ISO files.
- In Advanced storage (/wiki/GRUB2/AdvancedStorage) the necessary steps are documented on how to install and use GRUB2 on more advanced storage situations, such as software RAID, logical volumes or encrypted file systems.
- In Configuration variables (/wiki/GRUB2/Config_Variables) an exhaustive list of GRUB2 configuration variables, as used by `/etc/default/grub`, is documented.
- In Troubleshooting (/wiki/GRUB2/Troubleshooting) a list of common GRUB2 errors (with their solutions) is presented.
- In Hybrid partition table (/wiki/Hybrid_partition_table) the use of a mixed MBR/GPT setup is documented, as well as how to use such hybrid partition layout with GRUB2.

External resources

For more information, please see:

- GNU GRUB 2 manual page (https://www.gnu.org/software/grub/manual/grub.html)
  - Network (PXE) section of GRUB2 (https://www.gnu.org/software/grub/manual/grub.html#Network)
- Legacy BIOS issues with GPT article (http://www.rodsbooks.com/gdisk/bios.html)
- GPT and Hybrid MBR article (http://www.rodsbooks.com/gdisk/hybrid.html)
- GPT fdisk utility page (http://www.rodsbooks.com/gdisk/)
- Arch Linux GRUB2 wiki article (https://wiki.archlinux.org/index.php/GRUB2)
- Fedora GRUB2 wiki article : Encountering the dreaded GRUB2 boot prompt (https://fedoraproject.org/wiki/GRUB_2?rd=Grub2#Encountering_the_dreaded_GRUB_2_boot_prompt)
- ubuntu UEFI booting help (https://help.ubuntu.com/community/UEFIBooting)
- http://unix.stackexchange.com/questions/109272/dualboot-freebsd-gentoo-with-grub2-mbr (http://unix.stackexchange.com/questions/109272/dualboot-freebsd-gentoo-with-grub2-mbr)
- A blog post entry on locking specific GRUB2 boot entries with a password (http://daniel-lange.com/archives/75-Securing-the-grub-boot-loader.html)

Retrieved from "http://wiki.gentoo.org/index.php?title=GRUB2&oldid=667696 (http://wiki.gentoo.org/index.php?title=GRUB2&oldid=667696)"

Category (/wiki/Special:Categories): Bootloaders (/wiki/Category:Bootloaders)

- This page was last modified on 16 August 2017, at 08:34.