GRUB

GRUB (https://www.gnu.org/software/grub/)—not to be confused with **GRUB Legacy**—is the next generation of the GRand Unified Bootloader. GRUB is derived from **PUPA (http://www.nongnu.org/pupa/)** which was a research project to develop the next generation of what is now GRUB Legacy. GRUB has been rewritten from scratch to clean up everything and provide modularity and portability **[1] (https://www.gnu.org/software/grub/grub-faq.html#q1)**.

Contents

Related articles

Arch boot process

Boot loaders

Master Boot Record

GUID Partition Table

Unified Extensible Firmware Interface

GRUB Legacy

GRUB/EFI examples

GRUB/Tips and tricks

Multiboot USB drive

Preface

A **boot loader** is the first software program that runs when a computer starts. It is responsible for selecting, loading and transferring control to an operating system kernel. The kernel, in turn, initializes the rest of the operating system. The name *GRUB* officially refers to version *2* of the software. If you are looking for the article on the legacy version, see **GRUB Legacy**.

GRUB has a few root file system-specific limitations:

- **F2FS** is not supported

If your root partition is on an unsupported file system, you must create a separate `/boot` partition with a supported file system. In some cases, the development version of GRUB **grub-git (https://aur.archlinux.org/packages/grub-git/)**[AUR] has native support.

BIOS systems

GUID Partition Table (GPT) specific instructions

On a BIOS/**GPT** configuration, a **BIOS boot partition (https://www.gnu.org/software/grub/manual/grub/html_node/BIOS-installation.html#BIOS-installation)** is required. GRUB embeds its `core.img` into this partition.

Note:

- Before attempting this method keep in mind that not all systems will be able to support this partitioning scheme. Read more on **GUID partition tables**.
- This additional partition is only needed on a GRUB, BIOS/GPT partitioning scheme. Previously, for a GRUB, BIOS/MBR partitioning scheme, GRUB used the Post-MBR gap for the embedding the `core.img`). GRUB for GPT, however, does not use the Post-GPT gap to conform to GPT specifications that require 1_megabyte/2048_sector disk boundaries.
- For **UEFI** systems this extra partition is not required, since no embedding of boot sectors takes place in that case. However, UEFI systems still require an **ESP**.

Create a mebibyte partition (`+1M` with **fdisk** or **gdisk**) on the disk with no file system and with partition type BIOS boot. Select `BIOS boot` and partition type number `4` for *fdisk*, `ef02` for *gdisk*, and `bios_grub` for *parted*. This partition can be in any position order but has to be on the first 2 TiB of the disk. This partition needs to be created before GRUB installation. When the partition is ready, install the bootloader as per the instructions below.

The post-GPT gap can also be used as the BIOS boot partition though it will be out of GPT alignment specification. Since the partition will not be regularly accessed performance issues can be disregarded, though some disk utilities will display a warning about it. In *fdisk* or *gdisk* create a new partition starting at sector 34 and spanning to 2047 and set the type. To have the viewable partitions begin at the base consider adding this partition last.

Master Boot Record (MBR) specific instructions

Usually the post-**MBR** gap (after the 512 byte MBR region and before the start of the first partition) in many MBR (or 'msdos' disklabel) partitioned systems is 31 KiB when DOS compatibility cylinder alignment issues are satisfied in the partition table. However a post-MBR gap of about 1 to 2 MiB is recommended to provide sufficient room for embedding GRUB's `core.img` (**FS#24103 (https://bugs.archlinux.org/task/24103)**). It is advisable to use a partitioning tool that supports 1 MiB partition alignment to obtain this space as well as to satisfy other non-512 byte sector issues (which are unrelated to embedding of `core.img`).

Installation

Install the **grub (https://www.archlinux.org/packages/?name=grub)** package. It will replace **grub-legacy (https://aur.archlinux.org/packages/grub-legacy/)**[AUR], where already installed. Then do:

```
# grub-install --target=i386-pc /dev/sdx
```

where `/dev/sdx` is the **partitioned** disk where grub is to be installed.

Now you must **#Generate the main configuration file**.

If you use **LVM** for your `/boot`, you can install GRUB on multiple physical disks.

Tip: See **GRUB/Tips and tricks#Alternative installation methods** for other ways to install GRUB, such as to a USB stick.

See **`grub-install(8)` (https://jlk.fjfi.cvut.cz/arch/manpages/man/grub-install.8)** and **[2] (https://www.gnu.org/software/grub/manual/grub/html_node/BIOS-installation.html#BIOS-installation)** for more details on the *grub-install* command.

UEFI systems

Note:

- It is recommended to read and understand the **UEFI**, **GPT** and **UEFI Bootloaders** pages.
- When installing to use UEFI it is important to start the install with your machine in UEFI mode. The Arch Linux install media must be UEFI bootable.

Check if you have GPT and an ESP

An **EFI System Partition** (ESP) is needed on every disk you want to boot using EFI. GPT is not strictly necessary, but it is highly recommended and is the only method currently supported in this article. If you are installing Arch Linux on an EFI-capable computer with an already-working operating system, like Windows 8 for example, it is very likely that you already have an ESP. To check for GPT and for an ESP, use `parted` as root to print the partition table of the disk you want to boot from.

```
# parted /dev/sdx print
```

For GPT, you are looking for "Partition Table: gpt". For EFI, you are looking for a small (512 MiB or less) partition with a vfat/fat32 file system and the *boot* flag enabled. On it, there should be a directory named "EFI". If these criteria are met, this is your ESP. Make note of the partition number. You will need to know which one it is, so you can mount it later on while installing GRUB to it. In the following of this section `esp` must be substituted by it in commands.

If you do not have an ESP, you will need to create one. See **EFI System Partition**.

Installation

Note: UEFI firmware are not implemented consistently by hardware manufacturers. The installation examples provided are intended to work on the widest range of UEFI systems possible. Those experiencing problems despite applying these methods are encouraged to share detailed information for their hardware-specific cases, especially where solving these problems. A **GRUB/EFI examples** article has been provided for such cases.

This section assumes you are installing GRUB for x86_64 systems (x86_64-efi). For 32-bit EFI systems (not to be confused with 32-bit CPUs), replace `x86_64-efi` with `i386-efi` where appropriate.

Make sure you are in a **bash** shell.

Install the packages **grub (https://www.archlinux.org/packages/?name=grub)** and **efibootmgr (https://www.archlinux.org/packages/?name=efibootmgr)**. *GRUB* is the bootloader, *efibootmgr* creates bootable `.efi` stub entries used by the GRUB installation script.

The following steps install the GRUB UEFI application to `esp/EFI/grub`, install its modules to `/boot/grub/x86_64-efi`, and place the bootable `grubx64.efi` stub in `esp/EFI/grub`.

First, tell GRUB to use UEFI, set the boot directory and set the bootloader ID. Mount the ESP partition to e.g. `/boot` or `/boot/efi` and in the following change `esp_mount` to that mount point (usually `/boot`):

```
# grub-install --target=x86_64-efi --efi-directory=esp_mount --bootloader-id=grub
```

The `--bootloader-id` is what appears in the boot options to identify the GRUB EFI boot option; make sure this is something you will recognize later. The install will create a directory of the same name under `esp/EFI/` where the EFI binary bootloader will be placed.

Tip: If you use the option `--removable` then GRUB will be installed to `esp/EFI/BOOT/BOOTX64.EFI` and you will have the additional ability of being able to boot from the drive in case EFI variables are reset or you move the drive to another computer. Usually you can do this by selecting the drive itself similar to how you would using BIOS. If dual booting with Windows, be aware Windows usually has a folder called boot inside the EFI folder of the EFI partition, but the only purpose this serves is to recreate the EFI boot option for Windows.

After the above install finished the main GRUB directory is located at `/boot/grub/`.

Remember to **#Generate the main configuration file** after finalizing **#Configuration**.

Note:

- While some distributions require a `/boot/efi` or `/boot/EFI` directory, Arch does not.
- `--efi-directory` and `--bootloader-id` are specific to GRUB UEFI. `--efi-directory` specifies the mountpoint of the ESP. It replaces `--root-directory`, which is deprecated.

- You might note the absence of a <device_path> option (e.g.: `/dev/sda`) in the `grub-install` command. In fact any <device_path> provided will be ignored by the GRUB install script, as UEFI bootloaders do not use a MBR or partition boot sector at all.

See **UEFI troubleshooting** in case of problems. Additionally see **GRUB/Tips and tricks#UEFI further reading**.

Generate the main configuration file

After the installation, the main configuration file `grub.cfg` needs to be generated. The generation process can be influenced by a variety of options in `/etc/default/grub` and scripts in `/etc/grub.d/`; see **#Configuration**.

If you have not done additional configuration, the automatic generation will determine the root filesystem of the system to boot for the configuration file. For that to succeed it is important that the system is either booted or chrooted into.

Note: Remember that `grub.cfg` has to be re-generated after any change to `/etc/default/grub` or files in `/etc/grub.d/`.

Use the *grub-mkconfig* tool to generate `grub.cfg`:

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

By default the generation scripts automatically add menu entries for Arch Linux to any generated configuration. See **Multiboot USB drive#Boot entries** and **#Dual-booting** for custom menu entries for other systems.

Tip: To have *grub-mkconfig* search for other installed systems and automatically add them to the menu, **install** the **os-prober (https://www.archlinux.org/packages/?name=os-prober)** package and **mount** the partitions that contain other systems.

Note:

- The default file path is `/boot/grub/grub.cfg`, not `/boot/grub/i386-pc/grub.cfg`. The **grub (https://www.archlinux.org/packages/?name=grub)** package includes a sample `/boot/grub/grub.cfg`; ensure your intended changes are written to this file.
- If you are trying to run *grub-mkconfig* in a chroot or *systemd-nspawn* container, you might notice that it does not work, complaining that *grub-probe* cannot get the "canonical path of /dev/sdaX". In this case, try using *arch-chroot* as described in the **BBS post (https://bbs.archlinux.org/viewtopic.php?pid=1225067#p1225067)**.

Configuration

This section only covers editing the `/etc/default/grub` configuration file. See **GRUB/Tips and tricks** for more information.

Remember to always **#Generate the main configuration file** after making changes to `/etc/default/grub`.

Additional arguments

To pass custom additional arguments to the Linux image, you can set the `GRUB_CMDLINE_LINUX` + `GRUB_CMDLINE_LINUX_DEFAULT` variables in `/etc/default/grub`. The two are appended to each other and passed to kernel when generating regular boot entries. For the *recovery* boot entry, only `GRUB_CMDLINE_LINUX` is used in the generation.

It is not necessary to use both, but can be useful. For example, you could use `GRUB_CMDLINE_LINUX_DEFAULT="resume=/dev/sdaX quiet"` where `sdaX` is your swap partition to enable resume after hibernation. This would generate a recovery boot entry without the resume and without *quiet* suppressing kernel messages during a boot from that menu entry. Though, the other (regular) menu entries would have them as options.

By default *grub-mkconfig* determines the **UUID** of the root filesystem for the configuration. To disable this, uncomment `GRUB_DISABLE_LINUX_UUID=true`.

For generating the GRUB recovery entry you have to ensure that `GRUB_DISABLE_RECOVERY` is not set to `true` in `/etc/default/grub`.

You can also use `GRUB_CMDLINE_LINUX="resume=UUID=uuid-of-swap-partition"`

See **Kernel parameters** for more info.

Dual-booting

The best way to add other entries is editing `/etc/grub.d/40_custom` or `/boot/grub/custom.cfg`. The entries in this file will be automatically added after rerunning `grub-mkconfig`.

"Shutdown" menu entry

```
menuentry "System shutdown" {
        echo "System shutting down..."
        halt
}
```

"Restart" menu entry

```
menuentry "System restart" {
        echo "System rebooting..."
        reboot
}
```

"Firmware setup" menu entry (UEFI only)

```
menuentry "Firmware setup" {
        fwsetup
}
```

GNU/Linux menu entry

Assuming that the other distro is on partition `sda2`:

```
menuentry "Other Linux" {
        set root=(hd0,2)
        linux /boot/vmlinuz (add other options here as required)
        initrd /boot/initrd.img (if the other kernel uses/needs one)
}
```

Alternatively let grub search for the right partition by *UUID* or *label*:

```
menuentry "Other Linux" {
        # assuming that UUID is 763A-9CB6
        search --set=root --fs-uuid 763A-9CB6

        # search by label OTHER_LINUX (make sure that partition label is unambiguous)
        #search --set=root --label OTHER_LINUX

        linux /boot/vmlinuz (add other options here as required, for example: root=UUID=763A-9CB6)
        initrd /boot/initrd.img (if the other kernel uses/needs one)
}
```

Windows installed in UEFI-GPT Mode menu entry

This mode determines where the Windows bootloader resides and chain-loads it after Grub when the menu entry is selected. The main task here is finding the EFI partition and running the bootloader from it.

Note: This menuentry will work only in UEFI boot mode and only if the Windows bitness matches the UEFI bitness. It will not work in BIOS installed GRUB. See **Dual boot with Windows#Windows UEFI vs BIOS limitations** and **Dual boot with Windows#Bootloader UEFI vs BIOS limitations** for more info.

```
if [ "${grub_platform}" == "efi" ]; then
        menuentry "Microsoft Windows Vista/7/8/8.1 UEFI-GPT" {
                insmod part_gpt
                insmod fat
                insmod search_fs_uuid
                insmod chain
                search --fs-uuid --set=root $hints_string $fs_uuid
                chainloader /EFI/Microsoft/Boot/bootmgfw.efi
        }
fi
```

where `$hints_string` and `$fs_uuid` are obtained with the following two commands.

The `$fs_uuid` command determines the UUID of the EFI partition:

```
# grub-probe --target=fs_uuid $esp/EFI/Microsoft/Boot/bootmgfw.efi
```

```
1ce5-7f28
```

Alternatively one can run `blkid` (as root) and read the UUID of the EFI partition from there.

The `$hints_string` command will determine the location of the EFI partition, in this case harddrive 0:

```
# grub-probe --target=hints_string $esp/EFI/Microsoft/Boot/bootmgfw.efi
--hint-bios=hd0,gpt1 --hint-efi=hd0,gpt1 --hint-baremetal=ahci0,gpt1
```

These two commands assume the ESP Windows uses is mounted at `$esp`. There might be case differences in the path to Windows's EFI file, what with being Windows, and all.

Windows installed in BIOS-MBR mode

Note: GRUB supports booting `bootmgr` directly and **chainloading (https://www.gnu.org/software/grub/manual/grub.html#Chain_002dloading)** of partition boot sector is no longer required to boot Windows in a BIOS-MBR setup.

Warning: It is the **system partition** that has `/bootmgr`, not your "real" Windows partition (usually C:). In `blkid` output, the system partition is the one with `LABEL="SYSTEM RESERVED"` or `LABEL="SYSTEM"` and is only about 100 to 200 MB in size (much like the boot partition for Arch). See **Wikipedia:System partition and boot partition** for more info.

Throughout this section, it is assumed your Windows partition is `/dev/sda1`. A different partition will change every instance of hd0,msdos1. Add the below code to `/etc/grub.d/40_custom` or `/boot/grub/custom.cfg` and regenerate `grub.cfg` with `grub-mkconfig` as explained above to boot Windows (XP, Vista, 7, 8 or 10) installed in BIOS-MBR mode:

Note: These menuentries will work only in Legacy BIOS boot mode. It will not work in UEFI installed GRUB. See **Dual boot with Windows#Windows UEFI vs BIOS limitations** and **Dual boot with Windows#Bootloader UEFI vs BIOS limitations** .

In both examples 69B235F6749E84CE *is the partition UUID which can be found with command* lsblk --fs.

For Windows Vista/7/8/8.1/10:

```
if [ "${grub_platform}" == "pc" ]; then
  menuentry "Microsoft Windows Vista/7/8/8.1/10 BIOS-MBR" {
    insmod part_msdos
    insmod ntfs
    insmod search_fs_uuid
    insmod ntldr
    search --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 69B235F6749E84CE
    ntldr /bootmgr
  }
fi
```

For Windows XP:

```
if [ "${grub_platform}" == "pc" ]; then
  menuentry "Microsoft Windows XP" {
    insmod part_msdos
    insmod ntfs
    insmod search_fs_uuid
    insmod ntldr
    search --fs-uuid --set=root --hint-bios=hd0,msdos1 --hint-efi=hd0,msdos1 --hint-baremetal=ahci0,msdos1 69B235F6749E84CE
    ntldr /ntldr
  }
fi
```

Note: In some cases, GRUB may be installed without a clean Windows 8, in which case you cannot boot Windows without having an error with `\boot\bcd` (error code `0xc000000f`). You can fix it by going to Windows Recovery Console (cmd from install disk) and executing:

```
x:\> "bootrec.exe /fixboot"
x:\> "bootrec.exe /RebuildBcd".
```

Do **not** use `bootrec.exe /Fixmbr` because it will wipe GRUB out. Or you can use Boot Repair function in the Troubleshooting menu - it will not wipe out GRUB but will fix most errors.

Also you would better keep plugged in both the target hard drive and your bootable device **ONLY**. Windows usually fails to repair boot information if any other devices are connected.

`/etc/grub.d/40_custom` can be used as a template to create `/etc/grub.d/nn_custom`. Where `nn` defines the precendence, indicating the order the script is executed. The order scripts are executed determine the placement in the grub boot menu.

Note: `nn` should be greater than 06 to ensure necessary scripts are executed first.

LVM

If you use **LVM** for your `/boot` or `/` root partition, make sure that the `lvm` module is preloaded:

/etc/default/grub

```
GRUB_PRELOAD_MODULES="lvm"
```

RAID

GRUB provides convenient handling of RAID volumes. You need to add `insmod mdraid09` or `mdraid1x` which allows you to address the volume natively. For example, `/dev/md0` becomes:

```
set root=(md/0)
```

whereas a partitioned RAID volume (e.g. `/dev/md0p1`) becomes:

```
set root=(md/0,1)
```

To install grub when using RAID1 as the `/boot` partition (or using `/boot` housed on a RAID1 root partition), on devices with GPT ef02/'BIOS boot partition', simply run *grub-install* on both of the drives, such as:

```
# grub-install --target=i386-pc --debug /dev/sda
# grub-install --target=i386-pc --debug /dev/sdb
```

Where the RAID 1 array housing `/boot` is housed on `/dev/sda` and `/dev/sdb`.

Note: GRUB currently (Sep 2015) supports booting from **Btrfs** RAID 0/1/10, but *not* RAID 5/6. You may use **mdadm** for RAID 5/6, which is supported by GRUB.

Multiple entries

For tips on managing multiple GRUB entries, for example when using both `linux (https://www.archlinux.org/packages/?name=linux)` and `linux-lts (https://www.archlinux.org/packages/?name=linux-lts)` kernels, see **GRUB/Tips and tricks#Multiple entries**.

Encryption

Root partition

To encrypt a root filesystem to be used with GRUB, add the `encrypt` hook or the `sd-encrypt` hook (if using systemd hooks) to **mkinitcpio**. See **dm-crypt/System configuration#mkinitcpio** for details, and **Mkinitcpio#Common hooks** for alternative encryption hooks.

If using the `encrypt` hook, add the `cryptdevice` parameter to `/etc/default/grub`. In the example below, the `sda2` partition has been encrypted as `/dev/mapper/cryptroot`:

```
/etc/default/grub
GRUB_CMDLINE_LINUX="cryptdevice=/dev/sda2:cryptroot"
```

If using the `sd-encrypt` hook, add `luks.uuid`:

```
/etc/default/grub
GRUB_CMDLINE_LINUX="luks.uuid=UUID"
```

where *UUID* is the UUID of the LUKS-encrypted device.

Be sure to **#Generate the main configuration file** when done.

For further information about bootloader configuration for encrypted devices, see **Dm-crypt/System configuration#Boot loader**.

Note: If you wish to encrypt `/boot` either as a separate partition or part of the `/` partition, further setup is required. See **#Boot partition**.

Tip: If you are upgrading from a working GRUB Legacy configuration, check `/boot/grub/menu.lst.pacsave` for the correct device/label to add. Look for them after the text `kernel /vmlinuz-linux`.

Boot partition

GRUB can be set to ask for a password to open a **LUKS** blockdevice in order to read its configuration and load any **initramfs** and **kernel** from it. This option tries to solve the issue of having an **unencrypted boot partition**. `/boot` is **not** required to be kept in a separate partition; it may also stay under the system's root `/` directory tree.

Warning: GRUB does not support LUKS2 headers. Make sure you do not specify `luks2` for the type parameter when creating the encrypted partition using `cryptsetup luksFormat`.

To enable this feature encrypt the partition with `/boot` residing on it using **LUKS** as normal. Then add the following option to `/etc/default/grub`:

/etc/default/grub

```
GRUB_ENABLE_CRYPTODISK=y
```

Be sure to **#Generate the main configuration file** while the partition containing `/boot` is mounted.

Without further changes you will be prompted twice for a passhrase: the first for GRUB to unlock the `/boot` mount point in early boot, the second to unlock the root filesystem itself as described in **#Root partition**. You can use a **keyfile** to avoid this.

Note:

- If you use a special keymap, a default GRUB installation will not know it. This is relevant for how to enter the passphrase to unlock the LUKS blockdevice.
- In order to perform system updates involving the `/boot` mount point, ensure that the encrypted `/boot` is unlocked and mounted before performing an update. With a separate `/boot` partition, this may be accomplished automatically on boot by using **crypttab** with a **keyfile**.
- If you experience issues getting the prompt for a password to display (errors regarding cryptouuid, cryptodisk, or "device not found"), try reinstalling grub as below appending the following to the end of your installation command:

```
# grub-install --target=x86_64-efi --efi-directory=$esp --bootloader-id=grub --modules="part_gpt part_msdos"
```

Chainloading an Arch Linux .efi file

If you have an .efi file generated from following **Secure Boot** or other means, `/etc/grub.d/40_custom` can be edited to add a new menu entry before regenerating `grub.cfg` with `grub-mkconfig`.

/etc/grub.d/40_custom

```
menuentry 'Arch Linux .efi' {
insmod part_gpt
insmod chain
set root='(hdX,gptY)'
chainloader /EFI/path/file.efi
}
```

Using the command shell

Since the MBR is too small to store all GRUB modules, only the menu and a few basic commands reside there. The majority of GRUB functionality remains in modules in `/boot/grub`, which are inserted as needed. In error conditions (e.g. if the partition layout changes) GRUB may fail to boot. When this happens, a command shell may appear.

GRUB offers multiple shells/prompts. If there is a problem reading the menu but the bootloader is able to find the disk, you will likely be dropped to the "normal" shell:

```
grub>
```

If there is a more serious problem (e.g. GRUB cannot find required files), you may instead be dropped to the "rescue" shell:

```
grub rescue>
```

The rescue shell is a restricted subset of the normal shell, offering much less functionality. If dumped to the rescue shell, first try inserting the "normal" module, then starting the "normal" shell:

```
grub rescue> set prefix=(hdX,Y)/boot/grub
grub rescue> insmod (hdX,Y)/boot/grub/i386-pc/normal.mod
rescue:grub> normal
```

Pager support

GRUB supports pager for reading commands that provide long output (like the `help` command). This works only in normal shell mode and not in rescue mode. To enable pager, in GRUB command shell type:

```
sh:grub> set pager=1
```

Using the command shell environment to boot operating systems

```
grub>
```

The GRUB's command shell environment can be used to boot operating systems. A common scenario may be to boot Windows / Linux stored on a drive/partition via **chainloading**.

Chainloading means to load another boot-loader from the current one, ie, chain-loading.

The other bootloader may be embedded at the starting of the disk(MBR) or at the starting of a partition or as an EFI file in the ESP in the case of UEFI.

Chainloading a partition

```
set root=(hdX,Y)
chainloader +1
boot
```

X=0,1,2... Y=1,2,3...

For example to chainload Windows stored in the first partiton of the first hard disk,

```
set root=(hd0,1)
chainloader +1
boot
```

Similarly GRUB installed to a partition can be chainloaded.

Chainloading a disk/drive

```
set root=hdX
chainloader +1
boot
```

Chainloading Windows/Linux installed in UEFI mode

```
insmod ntfs
set root=(hd0,gpt4)
chainloader (${root})/EFI/Microsoft/Boot/bootmgfw.efi
boot
```

insmod ntfs used for loading the ntfs file system module for loading Windows. (hd0,gpt4) or /dev/sda4 is my EFI System Partition (ESP). The entry in the *chainloader* line specifies the path of the .efi file to be chain-loaded.

Normal loading

See the examples in **#Using the rescue console**

Using the rescue console

See **#Using the command shell** first. If unable to activate the standard shell, one possible solution is to boot using a live CD or some other rescue disk to correct configuration errors and reinstall GRUB. However, such a boot disk is not always available (nor necessary); the rescue console is surprisingly robust.

The available commands in GRUB rescue include `insmod`, `ls`, `set`, and `unset`. This example uses `set` and `insmod`. `set` modifies variables and `insmod` inserts new modules to add functionality.

Before starting, the user must know the location of their `/boot` partition (be it a separate partition, or a subdirectory under their root):

```
grub rescue> set prefix=(hdX,Y)/boot/grub
```

where X is the physical drive number and Y is the partition number.

Note: With a separate boot partition, omit `/boot` from the path (i.e. type `set prefix=(hdX,Y)/grub`).

To expand console capabilities, insert the `linux` module:

```
grub rescue> insmod i386-pc/linux.mod
```

or simply

```
grub rescue> insmod linux
```

This introduces the `linux` and `initrd` commands, which should be familiar.

An example, booting Arch Linux:

```
set root=(hd0,5)
linux /boot/vmlinuz-linux root=/dev/sda5
initrd /boot/initramfs-linux.img
boot
```

With a separate boot partition (e.g. when using EFI), again change the lines accordingly:

Note: Since boot is a separate partition and not part of your root partition, you must address the boot partition manually, in the same way as for the prefix variable.

```
set root=(hd0,5)
linux (hdX,Y)/vmlinuz-linux root=/dev/sda6
initrd (hdX,Y)/initramfs-linux.img
boot
```

Note: If you experienced `error: premature end of file /YOUR_KERNEL_NAME` during execution of `linux` command, you can try `linux16` instead.

After successfully booting the Arch Linux installation, users can correct `grub.cfg` as needed and then reinstall GRUB.

To reinstall GRUB and fix the problem completely, changing `/dev/sda` if needed. See **#Installation** for details.

Troubleshooting

Intel BIOS not booting GPT

MBR

Some Intel BIOS's require at least one bootable MBR partition to be present at boot, causing GPT-partitioned boot setups to be unbootable.

This can be circumvented by using (for instance) fdisk to mark one of the GPT partitions (preferably the 1007 KiB partition you have created for GRUB already) bootable in the MBR. This can be achieved, using fdisk, by the following commands: Start fdisk against the disk you are installing, for instance `fdisk /dev/sda`, then press `a` and select the partition you wish to mark as bootable (probably #1) by pressing the corresponding number, finally press `w` to write the changes to the MBR.

Note: The bootable-marking must be done in `fdisk` or similar, not in GParted or others, as they will not set the bootable flag in the MBR.

With cfdisk, the steps are similar, just `cfdisk /dev/sda`, choose bootable (at the left) in the desired hard disk, and quit saving.

With recent version of parted, you can use `disk_toggle pmbr_boot` option. Afterwards verify that Disk Flags show pmbr_boot.

```
# parted /dev/sdx disk_toggle pmbr_boot
# parted /dev/sdx print
```

More information is available **here (http://www.rodsbooks.com/gdisk/bios.html)**

EFI path

Some UEFI firmwares require a bootable file at a known location before they will show UEFI NVRAM boot entries. If this is the case, `grub-install` will claim `efibootmgr` has added an entry to boot GRUB, however the entry will not show up in the VisualBIOS boot order selector. The solution is to place a file at one of the known locations. Assuming the EFI partition is at `/boot/efi/` this will work:

```
mkdir /boot/efi/EFI/boot
cp /boot/efi/EFI/grub/grubx64.efi /boot/efi/EFI/boot/bootx64.efi
```

This solution worked for an Intel DH87MC motherboard with firmware dated Jan 2014.

Enable debug messages

Note: This change is overwritten when **#Generate the main configuration file**.

Add:

```
set pager=1
set debug=all
```

to `grub.cfg`.

"No suitable mode found" error

If you get this error when booting any menuentry:

```
error: no suitable mode found
Booting however
```

Then you need to initialize GRUB graphical terminal (`gfxterm`) with proper video mode (`gfxmode`) in GRUB. This video mode is passed by GRUB to the linux kernel via 'gfxpayload'. In case of UEFI systems, if the GRUB video mode is not initialized, no kernel boot messages will be shown in the terminal (atleast until KMS kicks in).

Copy `/usr/share/grub/unicode.pf2` to ${GRUB_PREFIX_DIR} (`/boot/grub/` in case of BIOS and UEFI systems). If GRUB UEFI was installed with `--boot-directory=$esp/EFI` set, then the directory is `$esp/EFI/grub/` :

```
# cp /usr/share/grub/unicode.pf2 ${GRUB_PREFIX_DIR}
```

If `/usr/share/grub/unicode.pf2` does not exist, install **bdf-unifont (https://www.archlinux.org/packages/?name=bdf-unifont)**, create the `unifont.pf2` file and then copy it to `${GRUB_PREFIX_DIR}` :

```
# grub-mkfont -o unicode.pf2 /usr/share/fonts/misc/unifont.bdf
```

Then, in the `grub.cfg` file, add the following lines to enable GRUB to pass the video mode correctly to the kernel, without of which you will only get a black screen (no output) but booting (actually) proceeds successfully without any system hang.

BIOS systems:

```
insmod vbe
```

UEFI systems:

```
insmod efi_gop
insmod efi_uga
```

After that add the following code (common to both BIOS and UEFI):

```
insmod font
```

```
if loadfont ${prefix}/fonts/unicode.pf2
then
    insmod gfxterm
    set gfxmode=auto
    set gfxpayload=keep
    terminal_output gfxterm
fi
```

As you can see for gfxterm (graphical terminal) to function properly, `unicode.pf2` font file should exist in `${GRUB_PREFIX_DIR}` .

msdos-style error message

```
grub-setup: warn: This msdos-style partition label has no post-MBR gap; embedding will not be possible!
grub-setup: warn: Embedding is not possible. GRUB can only be installed in this setup by using blocklists.
            However, blocklists are UNRELIABLE and its use is discouraged.
grub-setup: error: If you really want blocklists, use --force.
```

This error may occur when you try installing GRUB in a VMware container. Read more about it **here (https://bbs.archlinux.org/viewtopic.php?pid=581760#p581760)**. It happens when the first partition starts just after the MBR (block 63), without the usual space of 1 MiB (2048 blocks) before the first partition. Read **#Master Boot Record (MBR) specific instructions**

UEFI

Common installation errors

- If you have a problem when running grub-install with sysfs or procfs and it says you must run `modprobe efivars`, try **Unified Extensible Firmware Interface#Mount efivarfs**.
- Without `--target` or `--directory` option, grub-install cannot determine for which firmware to install. In such cases `grub-install` will print `source_dir does not exist. Please specify --target or --directory`.
- If after running grub-install you are told your partition does not look like an EFI partition then the partition is most likely not `Fat32`.

Drop to rescue shell

If GRUB loads but drops you into the rescue shell with no errors, it may be because of a missing or misplaced `grub.cfg`. This will happen if GRUB UEFI was installed with `--boot-directory` and `grub.cfg` is missing OR if the partition number of the boot partition changed (which is hard-coded into the `grubx64.efi` file).

GRUB UEFI not loaded

An example of a working EFI:

```
# efibootmgr -v
```

```
BootCurrent: 0000
Timeout: 3 seconds
BootOrder: 0000,0001,0002
Boot0000* Grub HD(1,800,32000,23532fbb-1bfa-4e46-851a-b494bfe9478c)File(\efi\grub\grub.efi)
Boot0001* Shell HD(1,800,32000,23532fbb-1bfa-4e46-851a-b494bfe9478c)File(\EfiShell.efi)
Boot0002* Festplatte BIOS(2,0,00)P0: SAMSUNG HD204UI
```

If the screen only goes black for a second and the next boot option is tried afterwards, according to **this post (https://bbs.archlinux.org/viewtopic.php?pid=981560#p981560)**, moving GRUB to the partition root can help. The boot option has to be deleted and recreated afterwards. The entry for GRUB should look like this then:

```
Boot0000* Grub HD(1,800,32000,23532fbb-1bfa-4e46-851a-b494bfe9478c)File(\grub.efi)
```

Invalid signature

If trying to boot Windows results in an "invalid signature" error, e.g. after reconfiguring partitions or adding additional hard drives, (re)move GRUB's device configuration and let it reconfigure:

```
# mv /boot/grub/device.map /boot/grub/device.map-old
# grub-mkconfig -o /boot/grub/grub.cfg
```

`grub-mkconfig` should now mention all found boot options, including Windows. If it works, remove `/boot/grub/device.map-old`.

Boot freezes

If booting gets stuck without any error message after GRUB loading the kernel and the initial ramdisk, try removing the `add_efi_memmap` kernel parameter.

Arch not found from other OS

Some have reported that other distributions may have trouble finding Arch Linux automatically with `os-prober`. If this problem arises, it has been reported that detection can be improved with the presence of `/etc/lsb-release`. This file and updating tool is available with the package **lsb-release (https://www.archlinux.org/packages/?name=lsb-release)** in the **official repositories**.

Warning when installing in chroot

When installing GRUB on a LVM system in a chroot environment (e.g. during system installation), you may receive warnings like

```
/run/lvm/lvmetad.socket: connect failed: No such file or directory
```

or

```
WARNING: failed to connect to lvmetad: No such file or directory. Falling back to internal scanning.
```

This is because `/run` is not available inside the chroot. These warnings will not prevent the system from booting, provided that everything has been done correctly, so you may continue with the installation.

GRUB loads slowly

GRUB can take a long time to load when disk space is low. Check if you have sufficient free disk space on your `/boot` or `/` partition when you are having problems.

error: unknown filesystem

GRUB may output `error: unknown filesystem` and refuse to boot for a few reasons. If you are certain that all **UUIDs** are correct and all filesystems are valid and supported, it may be because your **BIOS Boot Partition** is located outside the first 2TB of the drive **[3] (https://bbs.archlinux.org/viewtopic.php?id=195948)**. Use a partitioning tool of your choice to ensure this partition is located fully within the first 2TB, then reinstall and reconfigure GRUB.

grub-reboot not resetting

GRUB seems to be unable to write to root BTRFS partitions **[4] (https://bbs.archlinux.org/viewtopic.php?id=166131)**. If you use grub-reboot to boot into another entry it will therefore be unable to update its on-disk environment. Either run grub-reboot from the other entry (for example when switching between various distributions) or consider a different file system. You can reset a "sticky" entry by executing `grub-editenv create` and setting `GRUB_DEFAULT=0` in your `/etc/default/grub` (do not forget `grub-mkconfig -o /boot/grub/grub.cfg`).

Old BTRFS prevents installation

If a drive is formatted with BTRFS without creating a partition table (eg. /dev/sdx), then later has partition table written to, there are parts of the BTRFS format that persist. Most utilities and OS's do not see this, but GRUB will refuse to install, even with --force

```
# grub-install: warning: Attempting to install GRUB to a disk with multiple partition labels. This is not supported yet..
# grub-install: error: filesystem `btrfs' does not support blocklists.
```

You can zero the drive, but the easy solution that leaves your data alone is to erase the BTRFS superblock with `wipefs -o 0x10040 /dev/sdx`

Windows 8/10 not found

A setting in Windows 8/10 called "Hiberboot", "Hybrid Boot" or "Fast Boot" can prevent the Windows partition from being mounted, so `grub-mkconfig` will not find a Windows install. Disabling Hiberboot in Windows will allow it to be added to the GRUB menu.

See also

- **Official GRUB Manual (https://www.gnu.org/software/grub/manual/grub.html)**
- **Ubuntu wiki page for GRUB (https://help.ubuntu.com/community/Grub2)**
- **GRUB wiki page describing steps to compile for UEFI systems (https://help.ubuntu.com/community/UEFIBooting)**
- **Wikipedia:BIOS Boot partition**
- **How to configure GRUB (http://web.archive.org/web/20160424042444/http://members.iinet.net/~herman546/p20/GRUB2%20Configuration%20File%20Commands.html#Editing_etcgrub.d05_debian_theme)**
- **Boot with GRUB (http://www.linuxjournal.com/article/4622)**

Retrieved from "https://wiki.archlinux.org/index.php?title=GRUB&oldid=510215"