systemd-boot

systemd-boot, previously called **gummiboot**, is a simple UEFI boot manager which executes configured EFI images. The default entry is selected by a configured pattern (glob) or an on-screen menu. It is included with `systemd (https://www.archlinux.org/packages/?name=systemd)`, which is installed on Arch system by default.

It is simple to configure but it can only start EFI executables such as the Linux kernel **EFISTUB**, UEFI Shell, GRUB, the Windows Boot Manager.

Related articles

Arch boot process

Boot loaders

Secure Boot

Unified Extensible Firmware Interface

Contents

Installation

EFI boot

1. Make sure you are booted in UEFI mode.
2. Verify **your EFI variables are accessible**.
3. Mount your **EFI System Partition** (ESP) properly. `esp` is used to denote the mountpoint in this article.

Note: *systemd-boot* cannot load EFI binaries from other partitions. It is therefore recommended to mount your ESP to `/boot`. In case you want to separate `/boot` from the ESP see **#Manually** for more information.

4. If the ESP is **not** mounted at `/boot`, then copy your kernel and initramfs onto that ESP.

   Note: For a way to automatically keep the kernel updated on the ESP, have a look at **EFI System Partition#Using systemd** for some systemd units that can be adapted. If your EFI System Partition is using automount, you may need to add `vfat` to a file in `/etc/modules-load.d/` to ensure the current running kernel has the `vfat` module loaded at boot, before any kernel update happens that could replace the module for the currently running version making the mounting of `/boot/efi` impossible until reboot.

5. Type the following command to install *systemd-boot*:

   ```
   # bootctl --path=esp install
   ```

   It will copy the *systemd-boot* binary to your EFI System Partition (`esp/EFI/systemd/systemd-bootx64.efi` and `esp/EFI/Boot/BOOTX64.EFI` – both of which are identical – on x86-64 systems) and add *systemd-boot* itself as the default EFI application (default boot entry) loaded by the EFI Boot Manager.
6. Finally you must **configure** the boot loader to function properly.

BIOS boot

Warning: This is not recommended.

You can successfully install *systemd-boot* if booted with in BIOS mode. However, this process requires you to tell firmware to launch *systemd-boot*'s EFI file at boot, usually via two ways:

- you have a working EFI Shell somewhere else.
- your firmware interface provides a way of properly setting the EFI file that needs to be loaded at boot time.

If you can do it, the installation is easier: go into your EFI Shell or your firmware configuration interface and change your machine's default EFI file to `esp/EFI/systemd/systemd-bootx64.efi` (or `systemd-bootia32.efi` depending if your system firmware is 32 bit).

Note: the firmware interface of Dell Latitude series provides everything you need to setup EFI boot but the EFI Shell won't be able to write to the computer's ROM.

Updating

Unlike the previous separate *gummiboot* package, which updated automatically on a new package release with a `post_install` script, updates of new *systemd-boot* versions must now be done manually by the user. However the procedure can be automated using pacman hooks.

Manually

systemd-boot (bootctl(1) (https://jlk.fjfi.cvut.cz/arch/manpages/man/bootctl.1)) assumes that your EFI System Partition is mounted on `/boot`.

```
# bootctl update
```

If the ESP is not mounted on `/boot`, the `--path=` option can pass it. For example:

```
# bootctl --path=esp update
```

Note: This is also the command to use when migrating from *gummiboot*, before removing that package. If that package has already been removed, however, run `bootctl --path=esp install`.

Automatically

The **AUR** package systemd-boot-pacman-hook (https://aur.archlinux.org/packages/systemd-boot-pacman-hook/)AUR provides a **Pacman hook** to automate the update process. **Installing** the package will add a hook which will be executed every time the systemd (https://www.archlinux.org/packages/?name=systemd) package is upgraded.

Alternatively, place the following pacman hook in the `/etc/pacman.d/hooks/` directory:

```
/etc/pacman.d/hooks/systemd-boot.hook
---
[Trigger]
Type = Package
Operation = Upgrade
Target = systemd

[Action]
Description = Updating systemd-boot...
When = PostTransaction
Exec = /usr/bin/bootctl update
```

Configuration

Basic configuration

The basic configuration is stored in `esp/loader/loader.conf` file and it is composed by three options:

- `default` – default entry to select (without the `.conf` suffix); can be a wildcard like `arch-*`.
- `timeout` – menu timeout in seconds. If this is not set, the menu will only be shown on `Space` key (or most other keys actually work too) press during boot.
- `editor` – whether to enable the kernel parameters editor or not. `1` (default) is enabled, `0` is disabled; since the user can add `init=/bin/bash` to bypass root password and gain root access, it is strongly recommended to set this option to `0`.

Example:

```
esp/loader/loader.conf
```

```
default  arch
timeout  4
editor   0
```

Note: The first 2 options can be changed in the boot menu itself and changes will be stored as EFI variables.

Tip: A basic configuration file example is located at `/usr/share/systemd/bootctl/loader.conf`.

Adding boot entries

Note:

- *bootctl* will automatically check for "**Windows Boot Manager**" (`\EFI\Microsoft\Boot\Bootmgfw.efi`), "**EFI Shell**" (`\shellx64.efi`) and "**EFI Default Loader**" (`\EFI\Boot\bootx64.efi`) at boot time, as well as specially prepared kernel files found in `\EFI\Linux`. When detected, corresponding entries with titles `auto-windows`, `auto-efi-shell` and `auto-efi-default`, respectively, will be automatically generated. These entries do not require manual loader configuration. However, it does not auto-detect other EFI applications (unlike **rEFInd**), so for booting the Linux kernel, manual configuration entries must be created.

- If you dual-boot Windows, it is strongly recommended to disable its default **Fast Start-Up** option.
- Remember to load the intel **microcode** with `initrd` if applicable.
- You can find the `PARTUUID` for your root partition with the command `blkid -s PARTUUID -o value /dev/sdxY`, where `x` is the device letter and `Y` is the partition number. This is required only for your root partition, not `esp`.

bootctl searches for boot menu items in `esp/loader/entries/*.conf` – each file found must contain exactly one boot entry. The possible options are:

- `title` – operating system name. **Required.**
- `version` – kernel version, shown only when multiple entries with same title exist. Optional.
- `machine-id` – machine identifier from `/etc/machine-id`, shown only when multiple entries with same title and version exist. Optional.
- `efi` – EFI program to start, relative to your ESP (`esp`); e.g. `/vmlinuz-linux`. Either this or `linux` (see below) is **required.**
- `options` – command line options to pass to the EFI program or **kernel parameters**. Optional, but you will need at least `initrd=efipath` and `root=dev` if booting Linux.

For Linux, you can specify `linux path-to-vmlinuz` and `initrd path-to-initramfs`; this will be automatically translated to `efi path` and `options initrd=path` – this syntax is only supported for convenience and has no differences in function.

Tip: The available boot entries which have been configured can be listed with the command `bootctl list`.

An example entry file is located at `/usr/share/systemd/bootctl/arch.conf`. The **kernel parameters** for scenarios such as **LVM**, **LUKS** or **dm-crypt** can be found on the relevant pages.

EFI Shells or other EFI apps

In case you installed EFI shells and other EFI application into the ESP, you can use the following snippets:

```
esp/loader/entries/uefi-shell-v1-x86_64.conf
```

```
title  UEFI Shell x86_64 v1
efi    /EFI/shellx64_v1.efi
```

```
esp/loader/entries/uefi-shell-v2-x86_64.conf
```

```
title  UEFI Shell x86_64 v2
efi    /EFI/shellx64_v2.efi
```

Preparing kernels for EFI\Linux

EFI\Linux is searched for specially prepared kernel files, which bundle the kernel, the initrd, the kernel command line and `/etc/os-release` into one file. This file can be easily signed for secure boot.

Note: `systemd-boot` requires that the `os-release` file contain either `VERSION_ID` or `BUILD_ID` to generate an ID and automatically add the entry, which the Arch `os-release` does not. Either maintain your own copy with one of them, or make your bundling script generate it automatically.

Put the kernel command line you want to use in a file, and create the bundle file like this:

Kernel packaging command:

```
objcopy \
    --add-section .osrel="/usr/lib/os-release" --change-section-vma .osrel=0x20000 \
    --add-section .cmdline="kernel-command-line.txt" --change-section-vma .cmdline=0x30000 \
    --add-section .linux="vmlinuz-file" --change-section-vma .linux=0x40000 \
    --add-section .initrd="initrd-file" --change-section-vma .initrd=0x3000000 \
    "/usr/lib/systemd/boot/efi/linuxx64.efi.stub" "linux.efi"
```

Optionally sign *linux.efi* now (e.g. using *sbsigntools* from AUR).

Copying *linux.efi* into `esp\EFI\Linux`.

Support hibernation

See **Suspend and hibernate**.

Kernel parameters editor with password protection

Alternatively you can install `systemd-boot-password (https://aur.archlinux.org/packages/systemd-boot-password/)`AUR which supports `password` basic configuration option. Use `sbpctl generate` to generate a value for this option.

Install *systemd-boot-password* with the following command:

```
# sbpctl install esp
```

With enabled editor you will be prompted for your password before you can edit kernel parameters.

Keys inside the boot menu

The following keys are used inside the menu:

- `Up/Down` - select entry
- `Enter` - boot the selected entry
- `d` - select the default entry to boot (stored in a non-volatile EFI variable)
- `-/T` - decrease the timeout (stored in a non-volatile EFI variable)
- `+/t` - increase the timeout (stored in a non-volatile EFI variable)

- `e` - edit the kernel command line. It has no effect if the `editor` config option is set to `0`.
- `v` - show the gummiboot and UEFI version
- `Q` - quit
- `P` - print the current configuration
- `h/?` - help

These hotkeys will, when pressed inside the menu or during bootup, directly boot a specific entry:

- `l` - Linux
- `w` - Windows
- `a` - OS X
- `s` - EFI Shell
- `1-9` - number of entry

Troubleshooting

Manual entry using efibootmgr

If the `bootctl install` command failed, you can create a EFI boot entry manually using efibootmgr (https://www.archlinux.org/packages/?name=efibootmgr):

```
# efibootmgr -c -d /dev/sdX -p Y -l "\EFI\systemd\systemd-bootx64.efi" -L "Linux Boot Manager"
```

where `/dev/sdXY` is the **EFI System Partition**.

Note: The path to the EFI image must use the backslash (`\`) as the separator

Menu does not appear after Windows upgrade

See **UEFI#Windows changes boot order**.

See also

- **http://www.freedesktop.org/wiki/Software/systemd/systemd-boot/**

Retrieved from "https://wiki.archlinux.org/index.php?title=Systemd-boot&oldid=508162"