OpenCore

Reference Manual (0.7.3.4)

[2021.09.13]

figure. Available entries include:

- `BOOTx64.efi` or `BOOTIa32.efi`
  Initial bootstrap loaders, which load `OpenCore.efi`. `BOOTx64.efi` is loaded by the firmware by default consistent with the UEFI specification. However, it may also be renamed and put in a custom location to allow OpenCore coexist alongside operating systems, such as Windows, that use `BOOTx64.efi` files as their loaders. Refer to the `LauncherOption` property for details.
- `boot`
  Duet bootstrap loader, which initialises the UEFI environment on legacy BIOS firmware and loads `OpenCore.efi` similarly to other bootstrap loaders. A modern Duet bootstrap loader will default to `OpenCore.efi` on the same partition when present.
- `ACPI`
  Directory used for storing supplemental ACPI information for the `ACPI` section.
- `Drivers`
  Directory used for storing supplemental `UEFI` drivers for `UEFI` section.
- `Kexts`
  Directory used for storing supplemental kernel information for the `Kernel` section.
- `Resources`
  Directory used for storing media resources such as audio files for screen reader support. Refer to the `UEFI Audio Properties` section for details. This directory also contains image files for graphical user interface. Refer to the OpenCanopy section for details.
- `Tools`
  Directory used for storing supplemental tools.
- `OpenCore.efi`
  Main booter application responsible for operating system loading. The directory `OpenCore.efi` resides in is called the `root directory`, which is set to `EFI\OC` by default. When launching `OpenCore.efi` directly or through a custom launcher however, other directories containing `OpenCore.efi` files are also supported.
- `config.plist`
  `OC Config`.
- `vault.plist`
  Hashes for all files potentially loadable by `OC Config`.
- `vault.sig`
  Signature for `vault.plist`.
- `SysReport`
  Directory containing system reports generated by `SysReport` option.
- `nvram.plist`
  OpenCore variable import file.
- `opencore-YYYY-MM-DD-HHMMSS.txt`
  OpenCore log file.
- `panic-YYYY-MM-DD-HHMMSS.txt`
  Kernel panic log file.

Note: It is not guaranteed that paths longer than `OC_STORAGE_SAFE_PATH_MAX` (128 characters including the `0`-terminator) will be accessible within OpenCore.

3.2 Installation and Upgrade

To install OpenCore, replicate the Configuration Structure described in the previous section in the EFI volume of a GPT partition. While corresponding sections of this document provide some information regarding external resources such as ACPI tables, UEFI drivers, or kernel extensions (kexts), completeness of the matter is out of the scope of this document. Information about kernel extensions may be found in a separate Kext List document available in the OpenCore repository. Vaulting information is provided in the Security Properties section of this document.

The `OC config` file, as with any property list file, can be edited with any text editor, such as nano or vim. However, specialised software may provide a better experience. On macOS, the preferred GUI application is Xcode. ~~For a lightweight~~The ProperTree editor is a lightweight, cross-platform and open-source alternative~~, the ProperTree editor can be utilised~~.

It is strongly ~~advised not to use any software that is~~ recommended to avoid configuration creation tools that are aware of the internal ~~configration structure as it constantly gets out of date and will cause incorrect configuration to be~~

Available flags are:

- 0x00000001 (bit 0) — LINUX_BOOT_SCAN_ESP, Allows scanning for entries on EFI System Partition.
- 0x00000002 (bit 1) — LINUX_BOOT_SCAN_XBOOTLDR, Allows scanning for entries on Extended Boot Loader Partition.
- 0x00000004 (bit 2) — LINUX_BOOT_SCAN_LINUX_ROOT, Allows scanning for entries on Linux Root filesystems.
- 0x00000008 (bit 3) — LINUX_BOOT_SCAN_LINUX_DATA, Allows scanning for entries on Linux Data filesystems.
- 0x00000080 (bit 7) — LINUX_BOOT_SCAN_OTHER, Allows scanning for entries on file systems not matched by any of the above.

  The following notes apply to all of the above options:

  Note 1: Apple filesystems APFS and HFS are never scanned.

  Note 2: Regardless of the above flags, a file system must first be allowed by Misc/Security/ScanPolicy before it can be seen by OpenLinuxBoot or any other OC_BOOT_ENTRY_PROTOCOL driver.

  Note 3: It is recommended to enable scanning LINUX_ROOT and LINUX_DATA in both OpenLinuxBoot flags and Misc/Security/ScanPolicy in order to be sure to detect all valid Linux installs.

- 0x00000100 (bit 8) — LINUX_BOOT_ALLOW_AUTODETECT, If set allows autodetecting and linking vmlinuz* and init* ramdisk files when loader/entries files are not found.
- 0x00000200 (bit 9) — LINUX_BOOT_USE_LATEST, When a Linux entry generated by OpenLinuxBoot is selected as the default boot entry in OpenCore, automatically switch to the latest kernel when a new version is installed.

  When this option is set, an internal menu entry id is shared between kernel versions from the same install of Linux. Linux boot options are always sorted highest kernel version first, so this means that the latest kernel version of the same install always shows as the default, with this option set.

  Note: This option is recommended on all systems.

- 0x00000400 (bit 10) — LINUX_BOOT_ADD_RO, This option applies to autodetected Linux only (i.e. to Debian-style distrubutions, not to BLSpec and Fedora-style distributions with /loader/entries/*.conf files). Some distrubtions run a filesystem check on loading which requires the root filesystem to initially be mounted read-only via the ro kernel option. Set this bit to add this option on autodetected distros; should be harmless but very slightly slow down boot time (due to requried remount as read-write) on distros which do not require it. To specify this option for specific distros only, use partuuidopts:{partuuid}+=ro instead of this flag.
- 0x00004000 (bit 14) — LINUX_BOOT_LOG_VERBOSE, Add additional debug log info about files encountered and autodetect options added while scanning for Linux boot entries.
- 0x00008000 (bit 15) — LINUX_BOOT_ADD_DEBUG_INFO, Adds a human readable file system type, followed by the first eight characters of the partition's unique partition uuid, to each generated entry name. Can help with debugging the origin of entries generated by the driver when there are multiple Linux installs on one system.

Flag values can be specified in hexadecimal beginning with 0x or in decimal, e.g. flags=0x80 or flags=128.

- partuuidopts:{partuuid}[+]="{options}" - Default: not set.

  Allows specifying kernel options for a given partition only. If specified with += then these are used in addition to autodetected options, if specified with = they are used instead. Used for autodetected Linux only. Values specified here are never used for entries created from /loader/entries/*.conf files.

  Note: The partuuid value to be specified here is typically the same as the PARTUUID seen in root=PARTUUID=... in the Linux kernel boot options (view using cat /proc/cmdline) for autodetected Debian-style distros, but is NOT the same for Fedora-style distros booted from /loader/entries/*.conf files.

  Typically you should not need this option in the latter case, but in case you do, to find out the unique partition uuid to use, look for LNX: entries in the OpenCore debug log file. Alternatively, and for more advanced scenarios, you may wish to examine how your drives are mounted using the Linux mount command, and then find out the partuuid of relevant mounted drives by examining the output of ls -l /dev/disk/by-partuuid.

- autoopts[+]="{options}" - Default: None specified. The kernel options to use for autodetected Linux only. The value here is never used for entries created from /loader/entries/*.conf files. partuuidopts may be more suitable where there are multiple distros, but autoopts with no PARTUUID required is more convenient