Unified Extensible Firmware Interface (Русский)


Эта страница нуждается в сопроводителе


Статья не гарантирует актуальность информации. Помогите русскоязычному сообществу поддержкой подобных страниц. См. **Команда переводчиков ArchWiki**


Перевод этой статьи или раздела не отражает оригинальное содержание.


Причина: Перевод устарел (**Discuss**)

Unified Extensible Firmware Interface (или UEFI для краткости) представляет из себя новый тип прошивки, который изначально был разработан Intel (известно было тогда как просто EFI) в основном для систем на базе Itanium. Он вводит новый способ загрузки операционных систем, который отличается от обычного "загрузочного кода MBR" использующийся в BIOS системах. Она началась как Intel EFI в версии 1.x и после группы компании UEFI Forum взяла на себя её дальнейшее развитие и начиная с версии 2.x стала именоваться как Unified EFI. По состоянию на 23 мая 2012 года, спецификация UEFI 2.3.1 является самой последней.

Примечание: Если явно не указана спецификация EFI 1.x то термины EFI и UEFI используются как взаимозаменяемые для обозначения прошивок UEFI 2.x. Также, если не указано явно, эти указания являются общими и не Mac специфичными. Часть из них может не работать и может быть быть различными в Mac. Кроме того, реализация EFI компании Apple является ни EFI версии 1.x, ни UEFI версии 2.x, она смешивает обе версии. Такая прошивка не попадает ни под одну из версий спецификаций UEFI и поэтому она не является стандартной прошивкой UEFI.

Contents

[hide]

Загрузка ОС с помощью BIOS

BIOS или Базовая Система Ввода-Вывода - это первая программа, которая выполняется после включения компьютера. После инициализации всего оборудования и завершения самотестирования, BIOS выполняет первый загрузочный код на первом устройстве в списке загрузочных устройств.

Если список начнается с устройства CD/DVD, выполняется запись El-Torito на CD/DVD. Так работают загрузочные CD/DVD. Если список начинается с HDD, BIOS выполняет загрузочный код MBR, находящийся на первых 440байтах. Загрузочный код в свою очередь загружает(chainloads or bootstraps) гораздо более большой и сложный загрузчик, а тот уже загружает ОС

В основном BIOS понятия не имеет как читать таблицу разделов файловой системы. Все, что он делает - это инициализирует оборудование и загружает 440байт загрузочного кода

Мультизагрузка с помощью BIOS

Since very little can be achieved by a program which fits into the 440-byte boot code area, multi-booting using BIOS requires a multi-boot capable bootloader (multi-boot refers to booting multiple operating systems, not to booting a kernel in the Multiboot format specified by the GRUB developers). So usually a common bootloader like **GRUB** or **GRUB2** or **Syslinux** or **LILO** would be loaded by the BIOS, and it would load an operating system by either chain-loading or directly loading the kernel.

Загрузка ОС с помощью UEFI

UEFI не поддерживает загрузку вышеописанным методом, который только BIOS-ом и поддерживается. UEFI умеет работать как с таблицами разделов, так и с файловыми системами.

Стандартно используемые версии UEFI имеют поддержку таблиц разделов MBR и GPT. EFI в компьютерах Apple на базе процессоров Intel поддерживают кроме того ещё и Apple Partition Map(Таблицу разделов Apple). Большинство прошивок UEFI поддерживают работу с файловыми системами FAT12(флоппи диски), FAT16 FAT32 на жестких дисках и с файловой системой ISO9660 на CD/DVD дисках. EFI на компьютерах Apple имеют возможность работать кроме описанных ещё и с HFS/HFS+.

UEFI не выполняет никакой код из MBR даже если он есть. Вместо этого используется специальный раздел на жестком диске называемый "EFI SYSTEM PARTITION", на которой и располагаются файлы, которые необходимо запустить для загрузки. Каждый, кому необходимо может хранить необходимые ему загрузочные файлы по следующему пути: <EFI SYSTEM PARTITION>/EFI/<ИМЯ ВЛАДЕЛЬЦА>/. Если Вы имеете вот такую вот директорию, то у Вас появляется уникальная возможность загрузить свои файлы из консоли(UEFI shell). Среди таких файлов может быть Вам приятный загрузчик операционных систем. Обычно UEFI system partition отформатирована с файловой системой FAT32.

Under UEFI, every program whether they are OS loaders or some utilities (like memory testing apps) or recovery tools outside the OS, should be a UEFI Application corresponding to the EFI firmware architecture. Most of the UEFI firmware in the market, including recent Apple Macs use x86_64 EFI firmware. Only some older macs use i386 EFI firmware while no non-Apple UEFI system is known to use i386 EFI firmware.

A x86_64 EFI firmware does not include support for launching 32-bit EFI apps unlike the 64-bit Linux and Windows which include such support. Therefore the bootloader must be compiled for that architecture correctly.

Мультизагрузка с помощью UEFI

Так как каждая операционная система или поставщик, никому не мешая, может сохранять свои собственные файлы в системный раздел EFI, мульти-загрузка с использованием UEFI является лишь вопросом запуска приложения UEFI, соответствующего загрузчику конкретной ОС. Это избавляет от необходимости полагаться на механизм цепочной загрузки (chainloading), заключающейся в передаче управления от boot-менеджера к boot-сектору диска с загружаемой ОС, для переключения операционных систем.

Linux Windows x86_64 UEFI-GPT мультизагрузка

Windows Vista (SP1+) and 7 pr 8 x86_64 versions support booting natively using UEFI firmware. But for this they need **GPT** partitioning of the disk used for UEFI booting. Windows x86_64 versions support either UEFI-GPT booting or BIOS-MBR booting. Windows 32-bit versions support only BIOS-MBR booting. Follow the instructions provided in the forum link given in the references sections as to how to do this. See **http://support.microsoft.com/default.aspx?scid=kb;EN-US;2581408** for more info.

This limitation does not exist in Linux Kernel but rather depends on the bootloader used. For the sake of Windows UEFI booting, the Linux bootloader used should also be installed in UEFI-GPT mode if booting from the same disk.

Процесс загрузки в UEFI

1. Система включена - POST проверка.
2. Прошивка UEFI загружена.
3. Прошивка запускает диспетчер загрузки чтобы определить, какие приложения UEFI будут запущены и откуда (т.е., с каких дисков и разделов).
4. Прошивка запускает UEFI приложение с файловой системой FAT32 раздела UEFISYS как это определено в загрузочной записи менеджера загрузки микропрограммы.
5. UEFI приложение может запустить другое приложение (в случае UEFI консоли или менеджера загрузки, как rEFInd) или ядро и initramfs (в случае загрузчика как GRUB2) в зависимости от того, как приложение UEFI было настроено.

Определение разрядности прошивки UEFI

Не-Mac системы

Если каталог `/sys/firmware/efi` существует, то ядро загружено в режиме EFI. В этом случае UEFI имеет такую же разрядность, как у ядра. (т.е. i686 или x86_64)

Перевод этой статьи или раздела не отражает оригинальное содержание.

Причина: секция устарела (**Discuss**)

Примечание: Системы с SoC (System-on-a-Chip) Intel Atom поставляются с 32-битным UEFI (по состоянию на 2 ноября 2013). Дополнительную информацию можно получить на этой странице.

Apple Mac

Mac, выпущенные до 2008 года, в основном имеют прошивку i386-efi, а выпущенные в 2008 или позднее - x86_64-efi. Все Mac, способные работать с 64-битным ядром Mac OS X Snow Leopard имеют прошивку x86_64 EFI 1.x.

Чтобы узнать разрядность прошивки EFI в Mac, введите следующую команду в терминале Mac OS X:

```
ioreg -l -p IODeviceTree | grep firmware-abi
```

Если команда возвращает значение EFI32, то прошивка IA32 EFI (32-битная), а если значение EFI64 - прошивка x86_64 EFI (64-битная). Большинство Mac не имеют прошивку UEFI 2.x, так как реализация EFI от Apple не полностью совместима со спецификацией UEFI 2.x.

Поддержка UEFI в ядре Linux

Параметры конфигурации ядра Linux для UEFI

The required Linux Kernel configuration options for UEFI systems are :

```
CONFIG_EFI=y
CONFIG_EFI_STUB=y
CONFIG_RELOCATABLE=y
CONFIG_FB_EFI=y
CONFIG_FRAMEBUFFER_CONSOLE=y
```

UEFI Runtime Variables/Services Support - 'efivars' kernel module . This option is important as this is required to manipulate UEFI Runtime Variables using tools like **efibootmgr**.

```
CONFIG_EFI_VARS=m
```

Примечание: This option is compiled as module in Arch core/testing kernel.

Примечание: For Linux to access UEFI Runtime Services, the UEFI Firmware processor architecture and the Linux kernel processor architecture must match. This is independent of the bootloader used.

Примечание: If the UEFI Firmware arch and Linux Kernel arch are different, then the "**noefi**" kernel parameter must be used to avoid the kernel panic and boot successfully. The "noefi" option instructs the kernel not to access the UEFI Runtime Services.

GUID Partition Table **GPT** config option - mandatory for UEFI support

```
CONFIG_EFI_PARTITION=y
```

Примечание: All of the above options are required to boot Linux via UEFI, and are enabled in Archlinux kernels in official repos.

Retrieved from **http://git.kernel.org/?p=linux/kernel/git/torvalds/linux.git;a=blob_plain;f=Documentation/x86/x86_64/uefi.txt;hb=HEAD** .

Переменные для поддержки UEFI

UEFI defines variables through which an operating system can interact with the firmware. UEFI Boot Variables are used by the boot-loader and used by the OS only for early system start-up. UEFI Runtime Variables allow an OS to manage certain setting sof the firmware like the UEFI Boot Manager or managing the keys for UEFI Secure Boot Protocol etc.

Примечание: The below steps will not work if the system has been booted in BIOS mode and will not work if the UEFI processor architecture does not match the kernel one, i.e. x86_64 UEFI + ix86 32-bit Kernel and vice-versa config will not work. This is true only for efivars kernel module and efibootmgr step. The other steps (ie. upto setting up <UEFISYS>/efi/arch/grub.{efi,cfg}) can be done even in BIOS/Legacy boot mode.

Access to UEFI Runtime services is provided by "efivars" kernel module which is enabled through the `CONFIG_EFI_VAR=m` kernel config option. This module once loaded exposes the variables under the directory `/sys/firnware/efi/vars`. One way to check whether the system has booted in UEFI boot mode is to load the "efivars" kernel module and check for the existence of `/sys/firnware/efi/vars` directory with contents similar to :

```
Sample output (x86_64-UEFI 2.3.1 in x86_64 Kernel):
```

```
# ls -1 /sys/firmware/efi/vars/
Boot0000-8be4df61-93ca-11d2-aa0d-00e098032b8c/
BootCurrent-8be4df61-93ca-11d2-aa0d-00e098032b8c/
BootOptionSupport-8be4df61-93ca-11d2-aa0d-00e098032b8c/
BootOrder-8be4df61-93ca-11d2-aa0d-00e098032b8c/
ConIn-8be4df61-93ca-11d2-aa0d-00e098032b8c/
ConInDev-8be4df61-93ca-11d2-aa0d-00e098032b8c/
ConOut-8be4df61-93ca-11d2-aa0d-00e098032b8c/
ConOutDev-8be4df61-93ca-11d2-aa0d-00e098032b8c/
ErrOutDev-8be4df61-93ca-11d2-aa0d-00e098032b8c/
Lang-8be4df61-93ca-11d2-aa0d-00e098032b8c/
LangCodes-8be4df61-93ca-11d2-aa0d-00e098032b8c/
MTC-eb704011-1402-11d3-8e77-00a0c969723b/
MemoryTypeInformation-4c19049f-4137-4dd3-9c10-8b97a83ffdfa/
PlatformLang-8be4df61-93ca-11d2-aa0d-00e098032b8c/
PlatformLangCodes-8be4df61-93ca-11d2-aa0d-00e098032b8c/
RTC-378d7b65-8da9-4773-b6e4-a47826a833e1/
del_var
new_var
```

The UEFI Runtime Variables will not be exposed to the OS if you have used "noefi" kernel parameter in the boot-loader menu. This parameter instructs the kernel to completely ignore UEFI Runtime Services.

Пользовательские приложения

There are few tools that can access/modify the UEFI variables, namely

1. efibootmgr - Used to create/modify boot entries in the UEFI Boot Manager - **efibootmgr** or **efibootmgr-git**AUR[ссылка недействительна: package not found]
2. uefivars - simply dumps the variables - **uefivars-git**AUR - uses efibootmgr library
3. Ubuntu's Firmware Test Suite - fwts - **fwts-git**AUR - uefidump command - `fwts uefidump`

Не-Mac UEFI системы

efibootmgr

Важно: Using `efibootmgr` in Apple Macs will brick the firmware and may need reflash of the motherboard ROM. There have been bug reports regarding this in Ubuntu/Launchpad bug tracker. Use bless command alone in case of Macs. Experimental "bless" utility for Linux by Fedora developers - **mactel-boot**AUR.

`efibootmgr` command will work only if you have booted the system in UEFI mode itself, since it **requires access to UEFI Runtime Variables** which are **available only in UEFI boot mode** (with "noefi" kernel parameter NOT being used).

Initially the user may be required to manually launch the boot-loader from the firmware itself (using maybe the UEFI Shell) if the UEFI boot-loader was installed when the system is booted in BIOS mode. Then `efibootmgr` should be run to make the UEFI boot-loader entry as the default entry in the UEFI Boot Manager.

To use efibootmgr, first load the 'efivars' kernel module:

```
# modprobe efivars
```

If you get **no such device found** error for this command, that means you have not booted in UEFI mode or due to some reason the kernel is unable to access UEFI Runtime Variables (noefi?).

Verify whether there are files in */sys/firmware/efi/vars/* directory. This directory and its contents are created by "efivars" kernel module and it will exist only if you have booted in UEFI mode, without the "noefi" kernel parameter.

If */sys/firmware/efi/vars/* directory is empty or does not exist, then `efibootmgr` command will not work. If you are unable to make the ISO/CD/DVD/USB boot in UEFI mode try **https://gitorious.org/tianocore_uefi_duet_builds/pages/Linux_Windows_BIOS_UEFI_boot_USB**.

Примечание: The below commands use grub2-efi-x86_64 boot-loader as example.

Assume the boot-loader file to be launched is `/boot/efi/efi/arch_grub/grubx64.efi`. `/boot/efi/efi/arch_grub/grubx64.efi` can be split up as `/boot/efi` and `/efi/arch_grub/grubx64.efi`, wherein `/boot/efi` is the mountpoint of the UEFI System Partition, which is assumed to be /dev/sdXY (here X and Y are just placeholders for the actual values - eg:- in /dev/sda1 , X=a Y=1).

To determine the actual device path for the UEFI System Partition, try :

```
# cat /proc/self/mounts | grep /boot/efi | awk '{print $1}'
/dev/sdXY
```

Then create the boot entry using efibootmgr as follows :

```
# efibootmgr --create --gpt --disk /dev/sdX --part Y --write-signature --
label "Arch Linux (GRUB2)" --loader '\EFI\arch_grub\grubx64.efi'
```

In the above command `/boot/efi/efi/arch_grub/grubx64.efi` translates to `/boot/efi` and `/efi/arch_grub/grubx64.efi` which in turn translate to drive `/dev/sdX` -> partition Y -> file `/EFI/arch_grub/grubx64.efi`.

UEFI uses backward slash as path separator (similar to Windows paths).

The 'label' is the name of the menu entry shown in the UEFI boot menu. This name is user's choice and does not affect the booting of the system. More info can be obtained from **efibootmgr GIT README** .

FAT32 filesystem is case-insensitive since it does not use UTF-8 encoding by default. In that case the firmware uses capital 'EFI' instead of small 'efi', therefore using `\EFI\arch_grub\grubx64.efi` or `\efi\arch_grub\grubx64.efi` does not matter (this will change if the filesystem encoding is UTF-8).

Загрузчики Linux для UEFI

Смотреть **UEFI загрузчики**.

Создание UEFI раздела в Linux

Примечание: Раздел UEFISYS может быть любого размера, который поддерживается файловой системой FAT32. В соответствии с документацией пресловутой Microsoft, минимальный размер раздела с FAT32 - 512Мб. В соответствии с вышесказанным рекомендуется устанавливать размер UEFISYS раздела больше 512Мб. Особенно, если Вы используете несколько UEFI загрузчиков или несколько ОС, загружаемых с помощью UEFI. В

общем, места должно быть достаточно, чтобы хранить сопутствующие файлы всех загружаемых операционных систем. Если в качестве загрузчика Вы решили использовать Linux Kernel **EFISTUB** то Вы должны выделить достаточно места, чтобы в разделе хранились файлы Kernel и Initramfs.

Для GPT разметки диска

Two choices:

- Using GNU Parted/GParted: Create a FAT32 partition. Set "boot" flag on for that partition.
- Using GPT fdisk (aka gdisk): Create a partition with gdisk type code "EF00". Then format that partition as FAT32 using `mkfs.vfat -F32 /dev/<THAT_PARTITION>`

Примечание: Setting "boot" flag in parted in a MBR partition marks that partition as active, while the same "boot" flag in a GPT partition marks that partition as "UEFI System Partition".

Важно: Do not use util-linux fdisk, cfdisk or sfdisk to change the type codes in a GPT disk. Similarly do not use gptfdisk gdisk, cgdisk or sgdisk on a MBR disk, it will be automatically converted to GPT (no data loss will occur, but the system will fail to boot).

Для MBR разметки диска

Two choices:

- Using GNU Parted/GParted: Create FAT32 partition. Change the type code of that partition to 0xEF using fdisk, cfdisk or sfdisk.
- Using fdisk: Create a partition with partition type 0xEF and format it as FAT32 using `mkfs.vfat -F32 /dev/<THAT_PARTITION>`

Примечание: It is recommended to use always GPT for UEFI boot as some UEFI firmwares do not allow UEFI-MBR boot.

UEFI консоль

The UEFI Shell is a shell/terminal for the firmware which allows launching uefi applications which include uefi bootloaders. Apart from that, the shell can also be used to obtain various other information about the system or the firmware like memory map (memmap), modifying boot manager variables (bcfg), running partitioning programs (diskpart), loading uefi drivers, editing text files (edit), hexedit etc.

Ссылки для скачивания UEFI консоли

You can download a BSD licensed UEFI Shell from Intel's Tianocore UDK/EDK2 Sourceforge.net project.

- **x86_64 UEFI Shell 2.0 (Beta)**
- **x86_64 UEFI Shell 1.0 (Old)**
- **i386 UEFI Shell 2.0 (Beta)**
- **i386 UEFI Shell 1.0 (Old)**

Shell 2.0 works only in UEFI 2.3+ systems and is recommended over Shell 1.0 in those systems. Shell 1.0 should work in all UEFI systems irrespective of the spec. version the firmware follows. More info at **ShellPkg** and **this mail**

Запуск UEFI консоли

Few Asus and other AMI Aptio x86_64 UEFI firmware based motherboards (from Sandy Bridge onwards) provide an option called `"Launch EFI Shell from filesystem device"` . For those motherboards, download the x86_64 UEFI Shell and copy it to your UEFI SYSTEM PARTITION as `<UEFI_SYSTEM_PARTITION>/shellx64.efi` (mostly `/boot/efi/shellx64.efi`) .

Systems with Phoenix SecureCore Tiano UEFI firmware are known to have embedded UEFI Shell which can be launched using either F6, F11 or F12 key.

Примечание: If you are unable to launch UEFI Shell from the firmware directly using any of the above mentioned methods, create a FAT32 USB pen drive with Shell.efi copied as (USB)/efi/boot/bootx64.efi . This USB should come up in the firmware boot menu. Launching this option will launch the UEFI Shell for you.

Важные команды UEFI консоли

More info at **http://software.intel.com/en-us/articles/efi-shells-and-scripting/**

bcfg

BCFG command is used to modify the UEFI NVRAM entries, which allow the user to change the boot entries or driver options. This command is described in detail in page 83 (Section 5.3) of "UEFI Shell Specification 2.0" pdf document.

To dump a list of current boot entries -

```
Shell> bcfg boot dump -v
```

To add a boot menu entry for grub2's grubx64.efi (for example) as 4th (numbeering starts from zero) option in the boot menu

```
Shell> bcfg boot add 3 fs0:\EFI\arch\grubx64.efi "Arch Linux (GRUB2)"
```

where fs0: is the mapping corresponding to the UEFI System Partition and \EFI\arch\grubx64.efi is the file to be launched.

To remove the 4th boot option

```
Shell> bcfg boot rm 3
```

To move the boot option #3 to #0 (i.e. 1st or the default entry in the UEFI Boot menu)

```
Shell> bcfg boot mv 3 0
```

For bcfg help text

```
Shell> help bcfg -v -b
```

or

```
Shell> bcfg -? -v -b
```

edit

EDIT command provides a basic text editor with an interface similar to nano text editor, but slightly less functional. It handles UTF-8 encoding and takes care or LF vs CRLF line endings.

To edit, for example grub2's grub.cfg in the UEFI System Partition (fs0: in the firmware)

```
Shell> fs0:
FS0:\> cd \efi\grub
FS0:\efi\grub\> edit grub.cfg
```

Исправление проблем

Windows изменяет порядок загрузки

Допустим, если вы обновились с Windows 8 до Windows 8.1 и вы больше не видите загрузочного меню после этого обновления, (то есть сразу грузится Windows):

- Убедитесь, что как Secure Boot (настраивается в UEFI), так и Fast Startup (настраивается в опциях питания в Windows) отключены.
- Убедитесь, что в вашем UEFI Linux Boot Manager первичнее, чем Windows Boot Manager (настраивается в UEFI настройках, таких как Hard Disk Drive Priority).

Примечание: Windows 8.x+, включая Windows 10, будут перезаписывать любой выбор порядка загрузки, который вы делаете и устанавливать себя в качестве приоритетной загрузки после каждого запуска. Изменение порядка загрузки в прошивке UEFI продлится только до следующей Windows 10 загрузки. Вы должны знать *Change Boot Option* ключ для вашей материнской платы.

Для того, чтобы Windows 8.X не изменяли порядок загрузки, вы должны настроить групповые политики для Windows и выполнить скрипт (*.bat*) при запуске. В Windows:

1. Откройте командную строку с правами администратора. Выполните `bcdedit /enum firmware`
2. Найдите приложение, в котором в описании "Linux", например "Linux Boot Manager"
3. Скопируйте ID, включая скобки, например `{31d0d5f4-22ad-11e5-b30b-806e6f6e6963}`.
4. Создайте bat-файл (например, `bootorder.bat`) со следующим содержанием: `bcdedit /set {fwbootmgr} DEFAULT {`*`скопированный_ID`*`}` (например, `bcdedit /set {fwbootmgr} DEFAULT {31d0d5f4-22ad-11e5-b30b-806e6f6e6963}`).
5. Откройте *gpedit* и в *Local Computer Policy* > *Computer Configuration* > *Windows Settings* > *Scripts(Startup/Shutdown)*, выберите *Startup*. Должно открыться окно с заголовком *Startup Properties*.
6. Во вкладке *Scripts*, нажмите кнопку *Add*
7. Нажмите *Browse* и выберите созданный bat-файл.

См. также

- Wikipedia's page on **UEFI**
- Wikipedia's page on **UEFI SYSTEM Partition**
- **Linux Kernel UEFI Documentation**
- **UEFI Forum** - contains the official **UEFI Specifications** - GUID Partition Table is part of UEFI Specification
- **Intel's Tianocore Project** for Open-Source UEFI firmware which includes DuetPkg for direct BIOS based booting and OvmfPkg used in QEMU and Oracle VirtualBox
- **Intel's page on EFI**
- **FGA: The EFI boot process**
- **Microsoft's Windows and GPT FAQ** - Contains info on Windows UEFI booting also
- **Convert Windows Vista SP1+ or 7 x86_64 boot from BIOS-MBR mode to UEFI-GPT mode without Reinstall**
- **Create a Linux BIOS+UEFI and Windows x64 BIOS+UEFI bootable USB drive**
- **Rod Smith - A BIOS to UEFI Transformation**
- **UEFI Boot problems on some newer machines (LKML)**
- **EFI Shells and Scripting - Intel Documentation**
- **UEFI Shell - Intel Documentation**
- **UEFI Shell - bcfg command info**
- **Some useful 32-bit UEFI Shell utilities**

Categories:

- Boot process (Русский)
- Русский