USING GRUB ON UEFI

TL;DR:
The content of this hint (expect "TODO") has been merged into BLFS
"GRUB-{version number} for EFI" section. Please refer to the BLFS book
instead of this hint.

AUTHOR: Dan McGhee, Kevin M. Buckley, and Xi Ruoyao

DATE: 2020-08-14

SYNOPSIS: Boot LFS by default in a UEFI Environment using GRUB

DESCRIPTION:
This hint contains the information to direct the OS Boot Manager to default
to the GRUB in a UEFI environment employing EFI Mode. This hint applies to
only x86_64 machines.

This version updates Dan McGhee and Kevin M. Buckley's original, dated
2017-02-07.

The 2018-04-09 update saw the UEFI packages built against an LFS 8.2 systemd
installation that was already being booted using the existing host system's
bootloaders.

ATTACHMENTS:
* None

PREREQUISITES:
* Base LFS system before or after Ch. 8
* Basic understanding of obtaining and building packages

HINT:

DISCLAIMER: The recipes in this hint neither supplant nor supersede the
build instructions in a stable version of either the LFS or BLFS books.
They merely augment them for newer firmware. If conflicts arise between
this hint and the instructions in the book, take the issue to the mailing
lists. Additionally, this hint applies to only x86_64 machines packaged
with Windows 7 or Windows 8. The recipes here can be used on Mac OS, but
have not been investigated at the initial writing of this hint.

The 2018-04-09 hint refers to an LFS 8.2 system, built onto an x86_64
machine from within a LFS 7.9 host, that had had never had a version of
windows installed on it, indeed the host contained one EFI directories
below `/boot/efi/EFI/`, namely `boot`, that having been installed by
the vendor of the computer.

USE OF TERMS: The following is a use of terms in this hint. Further
information for and amplification of them can be found in References 1-3.

```
  * Firmware Settings:  An interface accessed by the keyboard after power
    is applied but before boot.  In it a user can change the order and
    way of how the computer boots.

  * BIOS System:  Firmware with EFI Mode turned off.

  * EFI Mode:  A condition of the booted system in which the EFI partition is
    mounted and the uefi (efi) variable support in the kernel is working
    properly.  It results from enabling UEFI Mode in the firmware settings.

  * EFI Mount Point:  A user defined mount point for the EFI Partition.  In
    this hint, and in most distros, it is `/boot/efi`.

  * EFI Partition:  A small partition, usually before any other partitions;
    i.e., `/dev/sda1` of  200-250 Mb, formatted in FAT32 with the `boot` flag, in
    parted, or `ef00` (EF00) partition type in gdisk.  (NOTE: The `boot` flag has a
    different function and meaning in MBR partitioned disks.)

  * efi variables (synonymous: uefi variables):  variables through which the
    operating system can interact with the firmware.

  * Legacy Boot Option (Legacy Boot):  A boot process in the firmware settings
    where EFI Mode is ddisabled..

  * GUID Partition Table (GPT): A modern partitioning scheme that supports
    large disk drives and more flexibilty than older partitioning schemes.

  PRELIMINARY DISCUSSION: Additional information and more in depth
  discussion of the following concepts can be found using References 1-3.

  Booting LFS is no longer as simple as `grub-install  /dev/sda`.  There are
  more options and more considerations.  With the advent and proliferation of
  UEFI firmware, a user's knowledge and philosophy of the boot
  process requires expansion:

  1. GPT partitioning is different from MBR partitioning.
     Programs fdisk, cfdisk, parted, or gdisk (from
     gptfdisk) can be used.  Each has their pros and cons,
     supporters and detractors.
  2. UEFI mode uses Boot Managers to select Boot Loaders like GRUB or
     LILO.
  3. The Boot Loaders are placed on the EFI partition rather than the
     MBR.  This concept is similar and parallel to the LFS procedures of
     using a separate `/boot` partition.
  4. There are additional tools that LFS needs in order to accomplish
     this mode of booting.
  5. LFS can be built and booted as the instructions are written up to
     and including LFS-8.4. To do this, the firmware settings must be
     changed to disable EFI mode. Different hardware vendors may use
     different terminology or menu options to interface with EFI options.

     Note that the only operating system that requires EFI mode is Microsoft
     Windows. If that operating system is not installed, then it is generally
     easier to disable EFI mode and not use the rest of this hint.  Some
```

systems may also require removing the efi partition on the boot
drive in addition to changing firmware settings to completely disable
EFI mode.

One of the hugely discussed issues surrounding UEFI is Secure Boot. It is
necessary to understand that the terms "UEFI" and "Secure Boot" are NOT
synonymous. UEFI is firmware. Secure Boot is a process of using "keys" to
"guarantee" the safety and authenticity of a Boot Loader. NOTE: To use
the procedures in this hint, Secure Boot must be disabled in the BIOS Boot
Settings.

Please note that the recommended order for implementing these instructions is a
departure from the build order in LFS. The most convenient, and arguably the
most practical way, to implement the instructions here is to use them in the
build of an LFS System at the end of Ch. 6. Building the BLFS and non-BLFS
packages has been tested both inside and outside of the chroot environment.
Then, following the book, proceed through Ch. 7, returning to the instructions
in Ch. 8. The instructions are presented in that order.

The most inconvenient way to implement these procedures is in a completely
functional LFS-8.4, or earlier, system. This involves uninstalling
`GRUB-2.02`, removing it from its location as a result of `grub-install` and
implementing the instructions below. Migrating from Legacy Boot to UEFI boot is
possible. At the writing of this hint, however, it is not
included. References 1-3 contain more information on this subject.

The last consideration in implementing the instructions here is GRUB's
graphical terminal. In UEFI systems, if the GRUB video mode is not
initialized, no kernel boot messages will appear until the kernel video takes
over. The GRUB package does not supply fonts, and GRUB defaults to
`unicode.pf2`. There are two ways to supply this font. The first is to copy
`unicode.pf2` from the host system to `/boot/grub` on the LFS system. The
second method involves configuring grub to build grub-mkfont, and this creates
a build dependency of `FreeType` for GRUB. This hint addresses the second
situation.

Finally, as of the writing of this hint, there is no standard for
the use of UEFI and the implementation of Secure Boot. These are hugely
manufacturer dependent. This hint uses terms used in the original author's
hardware. They may be different in other manufacturers' implementations.
However, the capabilities to do the boot setup operations contained in this
hint will exist on each machine. The terms may differ, and more than one
operation might be needed to achieve a desired goal. For example, someone
may need to disable Secure Boot and remove Secure Keys.

PROCEDURES:
[NOTE] The instructions are written with the assumption that the packages are
being built in the chroot environment before the end of Ch. 8. They can be
modified, with little difficulty, to be used in a functional system.

CHECKING EFI-MODE
Before entering the chroot environment, check that the host booted in
EFI Mode.

```
    ls /sys/firmware/efi

  If this directory exists and is populated, the host booted in EFI Mode.

CREATE AND FORMAT NEW EFI PARTITION, IF NECESSARY
  Some firmware (for example those in Dell desktops) has a reserved,
  invisible EFI partition for its OEM Windows.  If it's the first time you
  install another EFI booted OS on the system, create a new EFI partition
  using `fdisk` or your prefered partition tool.  Then, mark it with the
  `EF00` flag.

  If you created a new EFI partition, install `dosfstools` from BLFS and
  format the new EFI partition with:

    mkfs.vfat /dev/sda(x)

  where sda(x) is the device containing the EFI partition.

MOUNT EFI PARTITION
  Determine which device is the EFI partition using gdisk or parted,
  enter the chroot environment, create `/boot/efi` if needed, and

    mount -vt vfat /dev/sda(x) /boot/efi

  where sda(x) is the device containing the EFI partition.

BUILD DEPENDENCIES:

Install BLFS packages `dosfstools` and `popt`, using the instructions in
the book.  Build and install `FreeType` if building grub with `grub-mkfont`
enabled.

The BLFS `FreeType` instructions recommend that it be built after
`which` and `libpng` have been installed, so it was, however, as the
recommendation for "HarfBuzz" notes that one builds `FreeType` without
it first, and then do a re-install, it wasn't thought necessary to do
the re-install.

The `libpng` install did include the "apng" patch.

EFIVAR-37

Download:
* https://github.com/rhboot/efivar/releases/download/37/efivar-37.tar.bz2
* Required patch:
http://svn.linuxfromscratch.org/patches/trunk/efivar/efivar-37-gcc_9-1.patch

Apply a patch to make some fixes required by gcc-9:

    patch -Np1 -i ../efivar-37-gcc_9-1.patch

Compile the package:

    make LIBDIR=/usr/lib BINDIR=/bin CFLAGS="-O2 -Wno-stringop-truncation"
```

  The meaning of the make parameter:

* `libdir=/usr/lib`: This option overrides the default library directory
of the package (`/usr/lib64`).

* `CFLAGS="..."`: Override the default compiler flags causing build failure.

  Despite the Makefile having a `test` target, albeit one which isn't run
  by default, you SHOULD NOT run that `make test`, as it has been found
  to cause firmware bugs. Here are the thoughts on, and the exhortation
  not to do, this from the `efivar` community:
  https://github.com/rhboot/efivar/issues/78 .

Install the package:

  Now as the `root` user:

```
    make LIBDIR=/usr/lib BINDIR=/bin install
```

  Move the shared libraries to the /lib directory, and fix the symbolic
  link in /usr/lib:

```
    mv -v /usr/lib/lib{efivar,efiboot}.so.* /lib
    ln -sfv ../../lib/$(readlink /usr/lib/libefivar.so) /usr/lib/libefivar.so
    ln -sfv ../../lib/$(readlink /usr/lib/libefiboot.so) /usr/lib/libefiboot.so
```

EFIBOOTMGR-17

Dependencies:
* Required: `popt` and `efivar`

Download:
* https://github.com/rhboot/efibootmgr/archive/17/efibootmgr-17.tar.gz

Fix an outdated hotfix declaration causing FTBFS:

```
    sed -e '/extern int efi_set_verbose/d' -i src/efibootmgr.c
```

Compile the package:

```
    make sbindir=/sbin EFIDIR=LFS EFI_LOADER=grubx64.efi
```

  The meaning of the make parameters:

* `EFIDIR=LFS`: This is the distro's subdirectory name under
  `/boot/efi/EFI`. `Make.default` file of `efibootmgr` need this variable
  to be set.
* `EFI_LOADER=grubx64.efi`: This variable is set to the default EFI boot
  loader. It affects the default of parameter `--loader`. You can skip
  this if you won't invoke `efibootmgr` directly and only `grub-install`
  runs it.

Install the package:

```
    Now as the `root` user:

      make sbindir=/sbin EFIDIR=LFS EFI_LOADER=grubx64.efi install

  Multiple Partitions:

    Linking efibootmgr dynamically against popt preduces runtime dependency
    to libpopt.so, which resides in /usr hierarchy.  If /usr is a seperate
    mount point and you wish to use efibootmgr in case /usr is not avaliable
    (for example, to rescue a broken system), move popt libraries as follows:

      mv -v /usr/lib/libpopt.so.* /lib
      ln -sfv ../../lib/$(readlink /usr/lib/libpopt.so) /usr/lib/libpopt.so

  UNIFONT-13.0.03

  Download:
  * http://unifoundry.com/pub/unifont/unifont-13.0.03/font-builds/unifont-13.0.03.pcf.gz

  Install the package:

    [NOTE] This package is not a tarball.  So DON'T (and you can't)
    `tar -xf` it and change to the unzipped directory like normal LFS/BLFS
    packages.

    As the `root` user:

      mkdir -pv /usr/share/fonts/unifont &&
      gunzip -c unifont-13.0.03.pcf.gz > \
       /usr/share/fonts/unifont/unifont.pcf

  GRUB-2.04

  Dependencies:
  * Optional: `FreeType` (for `grub-mkfont` and `unicode.pf2`)
  * Optional: `unifont` (for `unicode.pf2`)

    [NOTE] The 2017-02-07 hint installs `unicode.pf2` manually.
    However, if `freetype` and `unifont` are both installed, GRUB
    will build `unicode.pf2`, and `grub-install` will install the
    file automatically.  This is recommended in 2018-04-09 hint.

  Download:
  * https://ftp.gnu.org/gnu/grub/grub-2.04.tar.xz

  Prepare for compilation:

    Unset custom optimization flags, since speeding up GRUB is not very useful
    and they are known to cause FTBFS:

      unset {C,CXX,LD}FLAGS

    [NOTE] Some options in 2017-02-07 hint are no longer necessary.
      ./configure --prefix=/usr  \
          --sbindir=/sbin        \
          --sysconfdir=/etc      \
          --disable-efiemu       \
          --enable-grub-mkfont   \
          --with-platform=efi    \
          --disable-werror
```

  The meaning of configure options:

* `--enable-grub-mkfont`: This ensures `grub-mkfont` to be built.
If `Freetype` is not installed, remove this option and then you
have to get `unicode.pf2` from other sources (either the host or
Internet).

* `--with-platform=efi`: This ensures `grub` to be built for EFI.

  If the optional dependencies are installed, `configure` should
  output the following information at last:

```
    grub-mkfont: Yes
    Build-time grub-mkfont: Yes
    With unifont from /usr/share/fonts/unifont/unifont.pcf
```

  That means `unicode.pf2` would be built and used.

Compile the package:

```
    make
```

Install the package:

  Now as the `root` user:

```
    make install
    mv -v /etc/bash_completion.d/grub /usr/share/bash-completion/completions
```

MODIFICATION OF /etc/fstab

  When constructing `/etc/fstab` in LFS chapter 8, add a line to
  mount EFI partition:

```
    /dev/<name of EFI partition>     /boot/efi    vfat     defaults       0     1
```

  Systemd would mount `efivarfs` automatically. If using sysvinit,
  add another line to mount `efivarfs`:

```
    efivarfs       /sys/firmware/efi/efivars  efivarfs  defaults  0      1
```

  Notes:

  a) If you are going to be booting your UEFI-aware LFS system using a
  non-LFS GRUB from your host AND if that GRUB is one (eg Fedora)

```
    that allows for the kernel to be specified using that GRUB's
    `linuxefi` attribute, so

       linuxefi  /path/to/kernel root=/path/to/root ro

    then you don't appear to need the `/etc/fstab` line, and indeed,
    you'll get told during the boot that the mounter knows nothing
    about the efivars filesystem type. However, LFS's efibootmgr will
    still be capable of interrogating your UEFI environment.

    b) If the LFS system is booted from the LFS+Hint's grub, which doesn't
    appear to know about the "linuxefi" attribute so using

      linux  /path/to/kernel root=/path/to/root ro

    then, unless you have the efivars filesystem mounted, and you are
    able to, then LFS's efibootmgr will be **not** capable of interrogating
    your UEFI environment, and you'll be told that there is no `efivars`
    filesystem.

  KERNEL CONFIGURATION OPTIONS FOR EFI

  The LFS kernel build's `make defconfig` populated a good number of
  the EFI-related options on my UEFI-enabled hardware, however, so as to
  make the 2014-10-16 hint's list of settings easier to find when coming
  to alter/set things, here is the list of the options along with the
  location of the various checkboxes and the settings they should have,
  as seen when starting from a `make menuconfig`:

      -> Enable the block layer
        -> Partition Types
          [*] Advanced partition selection      [CONFIG_PARTITION_ADVANCED]
          ...
          [*] EFI GUID Partition support        [CONFIG_EFI_PARTITION]

      -> Processor type and features
        [*] EFI runtime service support         [CONFIG_EFI]
          [*] EFI stub support                  [CONFIG_EFI_STUB]



      -> Device Drivers
        -> Graphics support
          -> Frame buffer Devices
            [*] EFI-based Framebuffer Support   [CONFIG_FB_EFI]



      -> Device Drivers
        -> Graphics support
          -> Console display driver support
            Framebuffer Console support         [CONFIG_FRAMEBUFFER_CONSOLE]

      -> Firmware Drivers
        -> EFI (Extensible Firmware Interface) Support
          < > EFI Variable Support via sysfs    [CONFIG_EFI_VARS]
          [*] Export efi runtime maps to sysfs  [CONFIG_EFI_RUNTIME_MAP]


      -> File systems
        -> Pseudo filesystems
          <*/M> EFI Variable filesystem         [CONFIG_EFIVAR_FS]
```

Note:

The only Kernel Config setting that a `make defconfig` didn't set on
the UEFI-enabled host was this one:

```
        [*] EFI stub support                  [CONFIG_EFI_STUB]
```

and without that setting in the kernel, attempts to boot the LFS system
tell you that:

```
    Kernel doesn't support EFI handover
```

however, adding just that one Kernel Config setting sees you able to
boot into the LFS system using the host system's Grub.

Don't select `CONFIG_EFI_VARS`, despite some outdated references may
suggests to select it. It is deprecated because of an 1024-byte variable
size limit. `efivarfs` (`CONFIG_EFIVAR_FS`) replaces its functionality,
and doesn't suffer from the same limit.

`CONFIG_FB_EFI=y` seems not necessary if you have other FB devices
avaliable. But without it you will lose some boot messages along
with the Tux logos on the FB.

USING GRUB TO SET UP THE BOOT PROCESS

INSTALLING GRUB TO THE EFI PARTITION

Installing GRUB to the EFI partition and creating an OS Boot Manager
entry is the major difference between the instructions in this hint and the
procedures in the LFS book. In concept, it is not actually a divergence
from the concepts of the book. The instructions there install GRUB to
the MBR, the MBR protected layer of a GPT disk or to a dedicated /boot
partition. The instructions here install GRUB to the EFI partition and
generate an entry in the system's Boot Manager. It is for the single
command here that this hint was written and for which all the non-LFS
packages were installed.

```
grub-install --bootloader-id=LFS --recheck --debug &> grub.log
```

`--bootloader-id=<some name>` is the directory on the EFI partition to
which the GRUB image is written.

Running this command generates lots of output (redirected to `grub.log`).
But at the end it will indicate that it was successful. This command
installs the GRUB image to `/boot/efi/EFI/LFS/grubx64.efi` and creates
the entry `LFS` in the system's Boot Manager.

To check it, inspect the contents of `/boot/efi/EFI/LFS` and, as root, run `efibootmgr`. The results of this command will list the Boot Order and all the Boot Entries. If the entry "LFS" does not appear, read the efibootmgr man page, create an entry and change the Boot Order to what is desired.

If GRUB was built with `freetype` and `unifont`, `unicode.pf2` should be installed automatically now. Issue:

```
grep "unicode.pf2" grub.log
```

You should see something like

```
copying `/usr/share/grub/unicode.pf2' -> `/boot/grub/fonts/unicode.pf2'
```

If not, you should get `unicode.pf2` from the host system or Internet, and install it into `/boot/grub/fonts`.

CONFIGURING GRUB

Generate `grub.cfg`:

```
cat > /boot/grub/grub.cfg << "EOF"
# Begin /boot/grub/grub.cfg
set default=0
set timeout=5

insmod gzio
insmod part_gpt
insmod ext2
set root=(hd[x], gpt[y])
# hd[x] is the drive of the LFS partion and gpt[y] is the partition

insmod efi_gop
insmod efi_uga
insmod font
if loadfont /boot/grub/fonts/unicode.pf2; then
  loadfont /boot/grub/fonts/unicode.pf2
  set gfxmode=auto
  insmod gfxterm
  set gfxpayload=keep
  terminal_output gfxterm
fi

menuentry "GNU/Linux, Linux <kernel name>"  {
  linux   /boot/vmlinuz-<kernel name> root=/dev/sda[x] ro
}
EOF
```

Note that in `menuentry`, `/dev/sda[x]` is the device of the LFS partition.

[NOTE] From GRUB's perspective, the kernel files are relative

```
    to the partition used. If you used a separate `/boot` partition,
    remove `/boot` from the above linux line and path of `unicode.pf2`.
    You will also need to change the `set root` line to point to the
    boot partition.
```

FINAL DISCUSSION:

As stated before, the implementation of UEFI firmware and its manipulation depends on the manufacturer. As of the writing of this hint, there is no standard approach. Therefore, while the instructionss here all do what is advertised, regrettably the system may not default to the grub boot loader "out of the box." In that case, reviewing References 1-3, will provide information that will lead users to a solution to the situation. As always, one of the best resources is the {,B}LFS mailing lists.

At this point, it is worth stating that there are other helpful tools: systemd-boot and rEFInd are two of them. They are described as Boot Managers, but in fact are a user space layer between the OS Boot Manager and the Boot Loader. Information about both is in the references.

REFERENCES:

1. Rod's Books - A collection of web page articles that goes into great detail about the concepts of UEFI booting, partitioning and tools. The below URL goes right to the efi information. www.rodsbooks.com is the main page and has many, many good articles.
    URL: http://www.rodsbooks.com/efi-bootloaders/index.html

2. "Unified Extensible Firmware Interface - ArchWiki"
    URL: https://wiki.archlinux.org/index.php/Unified_Extensible_Firmware_Interface

3. "GRUB - ArchWiki"
    URL: https://wiki.archlinux.org/index.php/GRUB

4. Google. URL: https://google.com

ACKNOWLEDGEMENTS:
* Craig Magee for comments and testing.
* Pierre Labastie for testing, font manipulation and comments.
* Lei Niu for comments on efivar and grub.cfg issues.
* Bruce Dubbs for word changes and comments.

TODO:
* Merge the content of this hint into BLFS-10.1
* Add paragraph and section numbers and TOC to make searchable
* Add appendix for migration from Legacy Boot to UEFI boot
* Add appendix for more options to default to GRUB
* Add appendix for LVM
* Add appendix for "standalone" GRUB on EFI partition independent
  from distro

CHANGELOG:
[2020-09-14]

```
    Fix efibootmgr-17 FTBFS

[2020-08-14]
    Remove the reference to pciutils
    Update to efibootmgr-17
    Update to unifont-13.0.03
        Install executables and shared libraries into /, instead of /usr
    Warn about deprecated `CONFIG_EFI_VARS`

[2019-11-26]
    Update to grub-2.04
    Update to unifont-12.1.03
    Add efivar-37 patch to fix FTBFS with gcc-9

[2019-03-29]
    Always use spaces instead of tabs
    Fix a typo in unifont instruction
    "gummiboot" is now named "systemd-boot"
    Merge Bruce's changes

[2019-03-28]
    Use linuxfromscratch.org URL for grub patch
    Cleanup dependencies
    Update to efibootmgr-16
    Update to efivar-37
    Add instructions creating new EFI partition

[2019-01-10]
    Added grub patch for binutils-2.31

[2018-04-09]
    Updated for LFS 8.2
    Use BLFS-like format for package dependency list
    Use package name in BLFS (FreeType instead of Freetype2)
    Deleted dosfstools-3.0.28 (replaced by dosfstools-4.1 in BLFS)
    Updated efivar-30 to efivar-34
    Updated unifont-9.0.06 to unifont-10.0.07
    Updated GRUB-2.02~beta3 to GRUB-2.02
    Modified the hint to let GRUB install unicode.pf2 automatically
    Removed some options not necessary now
    Fixed path to unicode.pf2 in grub.cfg for those without /boot partition
    Copied note about seperated /boot partition for grub.cfg from LFS book
    Removed incorrect personal email addresses (they are LFS mail lists)
    Added URL of Google
    Adapted for checkHint script (AUTHORS to AUTHOR)

[2017-01-22]
    Updated for LFS 7.10 and "extra package" updates
    dosfstools-3.0.26 -> dosfstools-3.0.28
    efivar-0.12       -> efivar-30
    efibootmgr-0.9.0  -> efibootmgr-14
    unifont-7.0.05    -> unifont-9.0.06
```

```
  [2014-10-16]
      Initial hint.
```