Boot Fedora 15 Using UEFI and GRUB2

Finnbarr P. Murphy
(fpm@fpmurphy.com)

With the release of Fedora 15 (Lovelock), I decided to have another look at the state of UEFI booting Fedora using GRUB2. In this post I show you how to build and install an UEFI version of GRUB2 which can be used to boot Fedora 15 instead of using the UEFI-enabled version of Legacy GRUB that comes with Fedora 15. I do not go into details of how to install that particular version of Fedora 15 as I assume that you already know how to do that if you are reading this post. Suffice to say that UEFI installing Fedora 15 is simple, hassle free and much better than in previous releases.

Note that not all UEFI firmware implementations will work with Fedora 15 or even with GRUB2. There are a number of reasons for this. Probably the two most common are ACPI issues and an assumption that virtual memory in GNU/Linux is contiguous. As Matthew Garrett so eloquently put it:

> Some firmware implementations assume that physically contiguous regions will be contiguous in virtual address space. This assumption is, obviously, entirely unjustifiable. Said firmware implementations lack the good grace to handle their failings in a measured and reasonable manner, instead tending to shit all over address space and oopsing the kernel.
>
> In an ideal universe these firmware implementations would simultaneously catch fire and cease to be a problem, but since some of them are present in attractively thin and shiny metal devices vanity wins out and some poor developer spends an extended period of time surrounded by a growing array of empty bottles until the underlying reason becomes apparent. Said developer presents this patch, which simply merges adjacent regions if they happen to be contiguous and have the same EFI memory type and caching attributes.

A number of patches are making their way into the Linux kernel which should fix some of the problems.

The version of GRUB2 that I used is 1.99. This particular version was released on May 15th, 2011 and is the most current version of GRUB2. A tarball of the 1.99 sources can be downloaded here. If *autoconf* and *automake* are not already installed on your system, you need to install the appropriate packages as they are required to build GRUB2. If you want to build the *grub-mkfont* utility, required if you want to create fonts for GRUB2, you also need the *freetype-devel* package.

Here is a simple script that will download and build a UEFI-enabled version of GRUB2:

```
$ wget http://ftp.gnu.org/gnu/grub/grub-1.99.tar.gz
$ gunzip grub-1.99.tar.gz
$ tar xvf grub-1.99.tar
$ cd grub-1.99
$ ./autogen.sh
$ ./configure –with-platform=efi –enable-grub-mkfont
…..
*****************************************************
```



```
GRUB2 will be compiled with following components:
Platform: x86_64-efi
With devmapper support: No (need devmapper header)
With memory debugging: No
efiemu runtime: No (not available on efi)
grub-mkfont: Yes
*****************************************************
$ make
```

If you want a version of GRUB2 that supports the GNU *gettext* form of localized message catalogs add –with-included-gettext to the configuration options. Read the included *ABOUT-NLS* text file for more options.

The next few scripts assume that you are going to place the required GRUB2 files in the */boot/efi/EFI/grub2* subdirectory. You will need to change these scripts if you want to put GRUB2 somewhere else. So long as GRUB2 is on the ESP (EFI System Partition) which is mounted on */boot/efi*, it does not really matter where it is located.

Invoke the following script as root from *../grub-1.99/grub-core* directory to build the final GRUB2 image (*grub.efi*) and install the image plus all the loadable (*.mod*) modules and their associated dependency (*.lst*) files.

```
#!/bin/bash

../grub-mkimage -O x86_64-efi -d . -o grub.efi -p "" \
     boot normal part_gpt fat ext2 lvm configfile lspci \
     ls reboot datetime loadenv search help video efi_gop

cp grub.efi *.mod *.lst /boot/efi/EFI/grub2/
```

Note the two additional modules *video* and *efi_gop* which were not required in previous releases of GRUB2. The *efi_gop* module is for use with UEFI firmware that implements the Graphic Output Protocol which pretty much all UEFI implementations do. If your system firmware implements the older UGA ((Universal Graphics Adapter) Protocol which was part of the EFI 1.1 specification (older Apple Macs for example), then use the *efi_uga* module instead. If you use the wrong module, typically you will get an *Error: no suitable mode found* message but your system will continue to boot and you may or may nor be able to see anything on your screen.

You will also need to create a GRUB2 configuration file called *grub.cfg* in the */boot/efi/EFI/grub2* subdirectory. I do not bother with creating an */etc/default/grub* file and all that nonsense about editing several */etc/grub.d/** files. etc. Seems like the famous camel design committee somehow got involved in the design of the GRUB2 configuration workflow while they were high on camel dung! I like to keep it simple and edit *grub.cfg* directly.

Here is a simple fairly minimum GRUB2 configuration file. It assumes that your GNU/Linux partition is the second partition on the first disk – which it is if you do a default installation. Change this if necessary. The kernel version will probably need to be changed to match your particular kernel.

```
timeout=30
pager=1
default=0
set color_normal=white/black
set color_highlight=yellow/blue
set prefix=(hd0,gpt1)/efi/grub2
```



```
menuentry "Fedora 15 (2.6.38.6-27.fc15.x86_64)" {
   set root=(hd0,gpt2)
   linux /vmlinuz-2.6.38.6-27.fc15.x86_64 ro root=/dev/mapper/vg_ultra-lv_root rd_LVM_LV=v
g_ultra/lv_root rd_LVM_LV=vg_ultra/lv_swap rd_NO_LUKS rd_NO_MD rd_NO_DM LANG=en_US.UTF-8 S
YSFONT=latarcyrheb-sun16 KEYTABLE=us rhgb quiet
   initrd /initramfs-2.6.38.6-27.fc15.x86_64.img
}
```

Note the *set prefix=(hd0,gpt1)/efi/grub2* directive. That is there to make it easier to load modules, etc. without having to input the full path. The *pager=1* directive simply tells GRUB2 to pause after each “page” of information is displayed on the screen when you are in GRUB2 command mode.

You should now be able to reboot your system and use your UEFI boot manager or UEFI shell to invoke *\efi\grub2\grub.efi*. When loaded, the regular GRUB menu screen should be displayed with a single menu option.

With GRUB2 you can optionally display a background image behind the menu and change the font used to display text. For this, you need to install the *gfxterm* graphical output terminal. To display text in *gfxterm*, at least one font must be loaded so that suitable glyphs are available to be painted onto the screen. GRUB2 uses a custom bitmap font format called PFF2. The format was designed to provide a bitmap font format that is simple to use, compact, and cleanly supports Unicode. The standard file extension for PFF2 font files is *.pf2*. However, no pre-built PFF2 files are supplied with the GRUB2 sources. Instead you have to source a suitable Unicode font file and use the *grub-mkfont* utility to convert the source format file into the PFF2 format.

A GRUB2 font must include at least the ASCII character set, since most GRUB2 messages are encoded in ASCII and should also include a number of other glyphs such as ISO 8859-1 (Latin-1), Latin Extended A, Latin Extended B, arrows, box and block characters. All fonts must use Unicode codes. Currently only monochrome bitmap fonts are supported.

By the way, the UnicodeData.txt file in the *unicode* directory is an outdated listing of the semantics of every encoded character in Unicode, i.e. it’s name, how it combines with other characters, it’s directionality and type, and other normative properties in a format (See Unicode Standard Annex #44 – Unicode Character Database) that can be read by applications. Most algorithms specified by Unicode like normalization and capitalization use this data. It and the other files in the *unicode* directory are used by *autogen.sh* to generate *../grub-core/unidata.c*.

A good source of suitable glyphs for GRUB2 are the GNU Unifont glyphs. These are available as BCD (Bitmap Distribution Format), PCF (Portable Compiled Format) and TTF (True Type Format) files. This particular font is included as the *xfonts-unifont* package in the Debian and Ubuntu distributions but not in Fedora.

The *grub-mkfont* utility supports every font format that the *freetype* library supports including PCF, BDF and TTF. Note that bitmap font formats like PCF and BDF are more suitable for use with GRUB2 than outline fonts like TTF.

```
$ ./grub-mkfont –help
Usage: ./grub-mkfont [OPTIONS] FONT_FILES

Options:
-o, –output=FILE_NAME set output file name
–ascii-bitmaps save only the ASCII bitmaps
–width-spec create width summary file
-i, –index=N set face index
-r, –range=A-B[,C-D] set font range
-n, –name=S set font family name
-s, –size=N set font size
-d, –desc=N set font descent
-c, –asce=N set font ascent
```



```
-b, —bold convert to bold font
-a, —force-autohint force autohint
—no-hinting disable hinting
—no-bitmap ignore bitmap strikes when loading
-h, —help display this message and exit
-V, —version print version information and exit
-v, —verbose print verbose messages
```

Here is an example of how to use *grub-mkfont* to create glyphs for ASCII, Latin-1, Latin A, Latin B, arrows, box and block from *unifont.bdf.gz* and store them in *unifont.pf2*:

```
$ wget http://unifoundry.com/unifont-5.1.20080820.bdf.gz
$ gunzip unifont-5.1.20080820.bdf.gz
$ grub-mkfont —output=unifont.pf2 —range=0x0000-0x0241,0x2190-0x21FF,0x2500-0x259f
unifont-5.1.20080820.bdf
```

Loading a font into GRUB2 is a two stage operation:

```
# load font module
insmod font
# load a font file
loadfont $prefix/unifont.pf2
```

Internally, text strings appear to be stored as C character strings. A *paint_char()* routine is used to "paint" a glyph representing a specific character onto the screen via a simple look-up table. I see no evidence that this code currently supports multibyte or wide character strings.

To display a background image, the image must first, if necessary, be converted into one of the image formats currently supported by GRUB2. These are JPG, TGA and PNG. See *../grub/grub-core/video/readers/*.c* for more information. A suitable image size is 640 x 480 or larger. If necessary, you can use the *background_image -m stretch* mode option to scale an image to fill the whole screen.

A number of additional steps are necessary to display a background image:

```
# load the image file reading module
insmod tga
# set the use_bg variable to true
use_bg=true
# invoke the background_image command with the pathname of the image to load
background_image $prefix/fedora15.tga
```

Here is an example GRUB2 configuration file which displays a background image *fedora15.tga* and uses the *unifont.pf2* font file which we created above.

```


timeout=30
default=0
set prefix=(hd0,gpt1)/efi/grub2

if loadfont /efi/grub2/unifont.pf2
then
   set gfxmode="1024x768x32"
   set gfxpayload=keep
   insmod tga
   insmod gfxterm
   terminal_output gfxterm
   use_bg=true
   background_image /efi/grub2/fedora15.tga
```



```
fi

set color_normal=white/black
set color_highlight=yellow/black

menuentry "Fedora 15 (2.6.38.6-27.fc15.x86_64)" {
   set root=(hd0,gpt2)
   linux /vmlinuz-2.6.38.6-27.fc15.x86_64 ro root=/dev/mapper/vg_ultra-lv_root rd_LVM_LV=v
g_ultra/lv_root rd_LVM_LV=vg_ultra/lv_swap rd_NO_LUKS rd_NO_MD rd_NO_DM LANG=en_US.UTF-8 S
YSFONT=latarcyrheb-sun16 KEYTABLE=us rhgb quiet
   initrd /initramfs-2.6.38.6-27.fc15.x86_64.img
}

menuentry "Reboot" {
   reboot
}
```

Note that the *terminal* command which appears in most of the current examples of *grub.cfg* was replaced by the *terminal_input* command.

The last time I wrote about GRUB2, I stated that:

> Fairly major changes where made to the EFI code in GRUB2 since the 1.97 release in March 2010. In my humble opinion most of these changes were not for the good as far as GRUB2 on EFI is concerned. In particular video mode setting is significantly worse. Furthermore. the developers in many cases need to update the GRUB2 documentation to incorporate these changes as at present the documentation is woefully out of sync with the 1.98 codebase.

However things have changed for the better since I wrote the above paragraph. The GRUB2 codebase has greatly stabilized and development of core functionality is complete. The (U)EFI video subsystems appear to work as intended. Much work has been done on the documentation front but much work remains to be done.

Note that GRUB2 always loads the kernel at *GRUB_LINUX_BZIMAGE_ADDR*. This may be problematic on some platforms which have EFI regions that exist in the address region used by the kernel. There is no easy solution to this issue. It requires fairly extensive work in the GRUB2 sources to enable the kernel to be loaded at a different address with the correct alignment.

The good news for Fedora users is that there are plans in hand to include GRUB2 in Fedora 16. The bad news is that somehow the Fedora kernel is not being passed the correct video mode when a video mode other than 80×25 is chosen, e.g. as in the second *grub.cfg* example above. As a result the GNOME shell defaults to fallback mode. This could be due to some defect in my systems or their configuration. However, I do not think this is the case. When I get time, I plan to investigate the issue and try and discover the root reason.

P.S. – I have placed a tarball of my GRUB2 directory as described above on my public download area. If you install the tarball in */boot/efi/EFI/GRUB2*, unpack it, and fix up *grub.cfg* to match your kernel name and location, you should be able to invoke GRUB2 from UEFI and use it to boot into Fedora 15. By the way, if you do not have a UEFI-enabled system, you can always use Tianocore DUET on a USB stick to emulate one. I often do that.

