Upgrading Fedora 13 to GRUB2

Finnbarr P. Murphy
(fpm@fpmurphy.com)

While a number of other GNU/Linux releases such as Ubuntu have moved to GRUB2, Fedora continues to use GRUB Legacy. Some preliminary investigation work on migrating to GRUB2 was done by Jeremy Katz for the Fedora Project when he worked for Red Hat. but that work seems to have completely stopped when he left Red Hat as the last update of the project page was in July 2009.

This post will show you how to migrate your Fedora 13 system to boot using GRUB2 instead of GRUB Legacy. As always, back up your system before attempting the migration. Things can and do go wrong. Even for you! Yes you! You have been warned! If you have a serious problem with a bootloader that you are unfamiliar with, you can end up spending many frustrating hours attempting to get the system to successfully boot.

There currently is a GRUB2 package (RPM) available for Fedora 13 (Goddard). As of the date of this post, the RPM version is *grub2-1.98-2.fc13.*.rpm*. However, it is does not replace GRUB Legacy with GRUB2. Instead it co-exists with GRUB Legacy and adds itself to the GRUB Legacy menu upon installation so you are able to select GRUB2 from the GRUB Legacy menu when booting your system. Here is the README.Fedora that is included with this package:

```
Using GNU GRUB 2 in Fedora
==========================

Though GRUB 2 provides various feature enhancements over previous GRUB
version (referred to as "GRUB", or "GRUB Legacy"), it did not reach its
stability and feature completness yet, and thus is not ready to replace
it for the whole user base. This package is primarily intended to
encourage testing and accelerate distribution integration.

It is generally safe to install the package. It is able to co-exist with
existing GRUB installation and adds itself to the GRUB menu upon
installation, so you'll able to select GRUB 2 from GRUB menu during
the boot.

Utilities
---------

The GRUB 2 utilities are prefixed (or postfixed) with 'grub2':

grub2-editenv
grub2-mkimage
grub2-mkelfimage
grub2-mkrescue
grub2-emu
grub2-install
grub2-mkdevicemap
grub2-probe
grub2-setup
update-grub2

Documentation
-------------

The GRUB 2 lacks documentation. While you are encouraged to contribute
the documentation, you can use the GRUB 2 Wiki [1] as primary source of
information pertaining to this development snapshot.
```



```
[1] http://grub.enbug.org/

If you intend to install grub2 as your primary boot loader refer to
the GRUB 2 Wiki for information on how to set it up.

Support channels
----------------

If you find a bug in this package, report them to the Red Hat Bugzilla [2].
For talk about using grub2, use IRC channel #grub on freenode Network [3].
You can meet this package maintainer there (nick lkundrak).

[2] http://bugzilla.redhat.com/
[3] http://freenode.net/

--
Lubomir Rintel
Fedora Project
```

This is a rather strange way to add GRUB2 support to Fedora since it adds no value to the boot process. Instead it complicates it. First boot to GRUB Legacy, then select GRUB2 then boot Fedora. It does not supplant GRUB Legacy. It renames the GRUB2 scripts and binaries and thus is incompatible with other platforms that use GRUB2. In my opinion, users should simply avoid installing this package and the quicker this package is obsoleted the better.

We will build GRUB2 from the 1.98 release source tarball and replace GRUB legacy with GRUB2. After you are finished there will be no dependency on GRUB Legacy. You can download the 1.98 release from gnu.org. For the purpose of this post I am going to use */work/grub2* as the build directory.

```
$ mkdir -p /work/grub2
$ cd /work/grub2
$ wget ftp://alpha.gnu.org/gnu/grub/grub-1.98.tar.gz
$ wget ftp://alpha.gnu.org/gnu/grub/grub-1.98.tar.gz.sig
$ gpg --verify grub-1.98.tar.gz.sig
.....
$ tar xvf grub-1.98.tar.gz
$ cd grub-1.98
$ ls
cinclude.m4    DISTLIST           genterminallist.sh    mkinstalldirs
aclocal.m4      docs               gentrigtables.c       mmap
AUTHORS         efiemu             genvideolist.sh       NEWS
autogen.sh      font               gettext               normal
autom4te.cache  fs                 gfxmenu               partmap
boot            gencmdlist.sh      gnulib                parttool
bus             gendistlist.sh     grub-1.98.tar.gz.sig  po
ChangeLog       genfslist.sh       hello                 README
commands        genhandlerlist.sh  hook                  script
conf            geninitheader.sh   include               stamp-h.in
config.guess    geninit.sh         INSTALL               term
config.h.in     genkernsyms.sh.in  install-sh            tests
config.rpath    genmk.rb           io                    THANKS
config.sub      genmoddep.awk      kern                  TODO
configure       genmodsrc.sh       lib                   util
configure.ac    genpartmaplist.sh  loader                video
COPYING         genparttoollist.sh Makefile.in
disk            gensymlist.sh.in   missing
$ ls -d */
autom4te.cache/  disk/    gettext/  include/  mmap/      script/
boot/            docs/    gfxmenu/  io/       normal/    term/
bus/             efiemu/  gnulib/   kern/     partmap/   tests/
commands/        font/    hello/    lib/      parttool/  util/
conf/            fs/      hook/     loader/   po/        video/
```



```
$
```

You may be wondering why I downloaded a second file, i.e. *gpg –verify grub-1.98.tar.gz.sig<./em>. Each downloadable tarball at gnu.org has an accompanying .sig file. This is a digital signature created with GnuPG. Assuming you already have a suitable public key, you can verify the signature of the downloaded tarball using gpg –verify grub-1.98.tar.gz.sig.*

Next you need to configure the GRUB2 build environment using *configure*:

```
./configure --enable-efiemu=no
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking target system type... x86_64-unknown-linux-gnu
checking for cmp... cmp
checking for bison... bison
checking for gawk... (cached) gawk
checking whether make sets $(MAKE)... (cached) yes
.........
.........
config.status: creating Makefile
config.status: creating gensymlist.sh
config.status: creating genkernsyms.sh
config.status: creating stamp-h
config.status: creating config.h
config.status: config.h is unchanged
config.status: linking include/grub/i386 to include/grub/cpu
config.status: linking include/grub/i386/pc to include/grub/machine
config.status: executing depfiles commands
config.status: executing po-directories commands
****************************************************
GRUB2 will be compiled with following components:
Platform: i386-pc
With memory debugging: No
efiemu runtime: No (explicitly disabled)
grub-fstest: Yes
grub-mkfont: No (need freetype2 library)
****************************************************
$
```

Notice that I explicitly turned off EFI emulation via a *configure* command line option as you really do not need it. You can use *configure –help* to see what other options are available to customize the build.

Next build the GRUB2 binaries and modules. You can perform the build on either a 32-bit or a 64-bit system; it does not manner.

```
$ make
gcc -Ignulib -I./gnulib -I. -I./include -I./gnulib -I./include -Wall -W -DGRUB_LIBDIR=\"/u
sr/local/lib/`echo grub/i386-pc | sed 's,x,x,'`\" -DLOCALEDIR=\"\" -DGRUB_MACHINE_PCBIOS=1
 -DGRUB_UTIL=1 -DGRUB_KERNEL_MACHINE_LINK_ADDR=0x8200 -MD -c -o grub_mkimage-gnulib_progna
me.o gnulib/progname.c
gcc -Iutil -I./util -I. -I./include -I./gnulib -I./include -Wall -W -DGRUB_LIBDIR=\"/usr/l
ocal/lib/`echo grub/i386-pc | sed 's,x,x,'`\" -DLOCALEDIR=\"\" -DGRUB_MACHINE_PCBIOS=1 -DG
RUB_UTIL=1 -DGRUB_KERNEL_MACHINE_LINK_ADDR=0x8200 -MD -c -o grub_mkimage-util_grub_mkrawim
age.o util/grub-mkrawimage.c
.........
```



```
.........
/bin/mkdir -p $(dirname po/ru.mo)
/usr/bin/msgfmt -c --statistics -o po/ru.mo po/ru.po
268 translated messages, 1 fuzzy translation, 1 untranslated message.
/usr/bin/msgmerge -U po/sv.po po/grub.pot
..................................... done.
/bin/mkdir -p $(dirname po/sv.mo)
/usr/bin/msgfmt -c --statistics -o po/sv.mo po/sv.po
168 translated messages, 1 fuzzy translation, 101 untranslated messages.
/usr/bin/msgmerge -U po/zh_CN.po po/grub.pot
........................................... done.
/bin/mkdir -p $(dirname po/zh_CN.mo)
/usr/bin/msgfmt -c --statistics -o po/zh_CN.mo po/zh_CN.po
268 translated messages, 1 fuzzy translation, 1 untranslated message.
```

If the build is successful, you then install the build using *make install*.

```
# make install
/usr/bin/msgmerge -U po/ast.po po/grub.pot
..................................... done.
/usr/bin/msgmerge -U po/ca.po po/grub.pot
........................................... done.
/usr/bin/msgmerge -U po/de.po po/grub.pot
..................................... done.
/usr/bin/msgmerge -U po/fi.po po/grub.pot
..................................... done.
/usr/bin/msgmerge -U po/fr.po po/grub.pot
..................................... done.
/usr/bin/msgmerge -U po/hu.po po/grub.pot
..................................... done.
/usr/bin/msgmerge -U po/id.po po/grub.pot
..................................... done.
/usr/bin/msgmerge -U po/it.po po/grub.pot
................................ done.
/usr/bin/msgmerge -U po/nl.po po/grub.pot
..................................... done.
/usr/bin/msgmerge -U po/ru.po po/grub.pot
..................................... done.
/usr/bin/msgmerge -U po/sv.po po/grub.pot
..................................... done.
/usr/bin/msgmerge -U po/zh_CN.po po/grub.pot
..................................... done.
/bin/sh ./mkinstalldirs /usr/local/lib/`echo grub/i386-pc | sed 's,x,x,'`
mkdir -p -- /usr/local/lib/grub/i386-pc
rm -f /usr/local/lib/`echo grub/i386-pc | sed 's,x,x,'`/*
/bin/sh ./mkinstalldirs /usr/local/share/`echo grub | sed 's,x,x,'`
mkdir -p -- /usr/local/share/grub
/bin/sh ./mkinstalldirs /usr/local/bin /usr/local/share/man/man1
mkdir -p -- /usr/local/bin /usr/local/share/man/man1
/bin/sh ./mkinstalldirs /usr/local/sbin /usr/local/share/man/man8
mkdir -p -- /usr/local/sbin /usr/local/share/man/man8
/bin/sh ./mkinstalldirs /usr/local/etc/grub.d
mkdir -p -- /usr/local/etc/grub.d
/bin/sh ./mkinstalldirs /usr/local/lib/grub
/bin/sh ./mkinstalldirs /usr/local/share/info
```

If you thought that *make install* actually installed GRUB2 you are mistaken. Instead this command installs the GRUB2 build system into */usr/local*. You then use some of these scripts and binaries to install and configure GRUB2. *make install* simply populates a number of subdirectories in */usr/local*.

```
# ls /usr/local/bin
```



```
grub-bin2h    grub-mkelfimage  grub-mkpasswd-pbkdf2  grub-script-check
grub-editenv  grub-mkimage     grub-mkrelpath
grub-fstest   grub-mkisofs     grub-mkrescue
# ls /usr/local/sbin
grub-install   grub-mkdevicemap  grub-reboot       grub-setup
grub-mkconfig  grub-probe        grub-set-default
# ls /usr/local/etc/grub.d
00_header  10_linux  30_os-prober  40_custom  README
#
```

To actually install GRUB2 on your system, the easiest way is to run the *grub-install* shell script which in turn calls other scripts and binaries to do the actual work. The only parameter you need to pass to *grub-install* is a boot device. In our case it is */dev/sda*.

```
# cd /usr/local/sbin
# ./grub-install --help
Usage: grub-install [OPTION] install_device
Install GRUB on your drive.

  -h, --help              print this message and exit
  -v, --version           print the version information and exit
  --modules=MODULES       pre-load specified modules MODULES
  --root-directory=DIR    install GRUB images under the directory DIR
                          instead of the root directory
  --grub-setup=FILE       use FILE as grub-setup
  --grub-mkimage=FILE     use FILE as grub-mkimage
  --grub-mkdevicemap=FILE use FILE as grub-mkdevicemap
  --grub-probe=FILE       use FILE as grub-probe
  --no-floppy             do not probe any floppy drive
  --recheck               probe a device map even if it already exists
  --force                 install even if problems are detected
  --disk-module=MODULE    disk module to use

INSTALL_DEVICE can be a GRUB device name or a system device filename.

grub-install copies GRUB images into /boot/grub (or /grub on NetBSD and
OpenBSD), and uses grub-setup to install grub into the boot sector.

If the --root-directory option is used, then grub-install will copy
images into the operating system installation rooted at that directory.

Report bugs to .
# ./grub-install /dev/sda
Installation finished. No error reported.
#
```


So what does *grub-install* actually do? Among other things, it

- Copies all the module (*.*mod*) files from */work/grub* into */boot/grub*.
- Copies all the list (*.*lst*) files from */work/grub* into */boot/grub*. Simply put, a *.lst* file maps an entity to a module. For example, *moddep.lst* contains a list of all the other modules that a particular module depends on. When a module is loaded, the dependent modules are also automatically loaded.
- Generates a core image (*core.img*) which contains the basic GRUB2 image, plus the filesystem driver module needed to read the filesystem which contains the GRUB2 files and possibly one or more other modules. In our case, the appropriate filesystem module is *ext2.mod*.
- Modifies the contents of *core.img* to insert specific values for the boot device and the addresses of the other blocks of the file It then copies *core.img* to specific sectors on the boot device. In our case, the first 0x10000 sectors (32Kb) immediately following the MBR Master Boot Record).
- Generates and write an 512-byte MBR image to the appropriate area of the boot device which includes a pointer to the first block of *core.img*.

Here is a listing of the files that are installed in */boot/grub*:

```
acpi.mod          dm_nv.mod                   gfxterm.mod    msdospart.mod        search_
label.mod
affs.mod          drivemap.mod                gptsync.mod    multiboot2.mod       search.
mod
afs_be.mod        echo.mod                    grubenv        multiboot.mod        serial.
mod
afs.mod           efiemu32.o                  gzio.mod       normal.mod           setjmp.
mod
aout.mod          efiemu64.o                  halt.mod       ntfscomp.mod         setpci.
mod
ata.mod           efiemu.mod                  handler.lst    ntfs.mod             sfs.mod
ata_pthru.mod     elf.mod                     handler.mod    ohci.mod             sh.mod
at_keyboard.mod   example_functional_test.mod hashsum.mod    part_acorn.mod       sleep.m
od
befs_be.mod       ext2.mod                    hdparm.mod     part_amiga.mod       tar.mod
befs.mod          extcmd.mod                  hello.mod      part_apple.mod       termina
l.lst
biosdisk.mod      fat.mod                     help.mod       part_gpt.mod         termina
l.mod
bitmap.mod        font.mod                    hexdump.mod    partmap.lst          terminf
o.mod
bitmap_scale.mod  fshelp.mod                  hfs.mod        part_msdos.mod       test.mo
d
blocklist.mod     fs.lst                      hfsplus.mod    part_sun.mod         tga.mod
boot.img          functional_test.mod         iso9660.mod    parttool.lst         trig.mo
d
boot.mod          gcry_arcfour.mod            jfs.mod        parttool.mod         true.mo
d
bsd.mod           gcry_blowfish.mod           jpeg.mod       password.mod         udf.mod
bufio.mod         gcry_camellia.mod           kernel.img     password_pbkdf2.mod  ufs1.mo
d
cat.mod           gcry_cast5.mod              keystatus.mod  pbkdf2.mod           ufs2.mo
d
cdboot.img        gcry_crc.mod                linux16.mod    pci.mod              uhci.mo
d
chain.mod         gcry_des.mod                linux.mod      play.mod             usb_key
board.mod
charset.mod       gcry_md4.mod                lnxboot.img    png.mod              usb.mod
cmp.mod           gcry_md5.mod                loadenv.mod    probe.mod            usbms.m
od
command.lst       gcry_rfc2268.mod            locale         pxeboot.img          usbtest.
mod
configfile.mod    gcry_rijndael.mod           loopback.mod   pxecmd.mod           vbeinfo.
mod
core.img          gcry_rmd160.mod             lsmmap.mod     pxe.mod              vbe.mod
cpio.mod          gcry_seed.mod               ls.mod         raid5rec.mod         vbetest.
mod
cpuid.mod         gcry_serpent.mod            lspci.mod      raid6rec.mod         vga.mod
crc.mod           gcry_sha1.mod               lvm.mod        raid.mod             vga_tex
t.mod
crypto.lst        gcry_sha256.mod             mdraid.mod     read.mod             video_f
b.mod
crypto.mod        gcry_sha512.mod             memdisk.mod    reboot.mod           video.l
st
datehook.mod      gcry_tiger.mod              memrw.mod      reiserfs.mod         video.m
od
date.mod          gcry_twofish.mod            minicmd.mod    relocator.mod        videote
st.mod
datetime.mod      gcry_whirlpool.mod          minix.mod      scsi.mod             xfs.mod
device.map        gettext.mod                 mmap.mod       search_fs_file.mod   xnu.mod
diskboot.img      gfxmenu.mod                 moddep.lst     search_fs_uuid.mod   xnu_uui
d.mod
```


Now GRUB2 is nearly ready to use. The only thing missing is a working *grub.cfg* configuration file located in the */boot/grub/* subdirectory. You can edit the */etc/default/conf* configuration file and then use */usr/local/sbin/grub-mkconfig* to create */boot/grub/grub.cfg*. Note some distributions, including Ubuntu, specify the use of *update-grub* instead of *grub-mkconfig*. However *update-grub* is simply a link to *grub-mkconfig*. This is the official way of doing things. I, on the other hand, find it easier just to directly edit */boot/grub/grub.cfg* and skip the other steps. Why complicate things!

You can easily modify your GRUB Legacy configuration file to work with GRUB2. For example, here is a very basic *menu.lst* configuration file for GRUB Legacy which boots Fedora 13 from */dev/sda*:

```
default=0
timeout=15

title Fedora (2.6.33.5-124.fc13.x86_64)
    root (hd0,0)
    kernel /vmlinuz-2.6.33.5-124.fc13.x86_64 ro root=/dev/mapper/vg_ultra-lv_root rd_LVM_L
V=vg_ultra/lv_root rd_LVM_LV=vg_ultra/lv_swap rd_NO_LUKS rd_NO_MD rd_NO_DM LANG=en_US.UTF-
8 SYSFONT=latarcyrheb-sun16 KEYTABLE=us rhgb quiet
    initrd /initramfs-2.6.33.5-124.fc13.x86_64.img
```

and here is the same configuration modified to work with GRUB2:

```
default=0
timeout=15

menuentry "Fedora 13" {
   root=(hd0,1)
   linux /vmlinuz-2.6.33.5-124.fc13.x86_64 ro root=/dev/mapper/vg_ultra-lv_root rd_LVM_LV=
vg_ultra/lv_root rd_LVM_LV=vg_ultra/lv_swap rd_NO_LUKS rd_NO_MD rd_NO_DM LANG=en_US.UTF-8
SYSFONT=latarcyrheb-sun16 KEYTABLE=us rhgb quiet
   initrd /initramfs-2.6.33.5-124.fc13.x86_64.img
}
```

As you can see the structure of a menu stanza has changed. Menu stanzas are now delineated by means of the *menuitem "Name" { }* syntax rather than by Python-like indenting. The keyword *kernel* is replaced with a new keyword *linux*. GRUB2 partitions start from 1 rather than 0 – hence *root=(hd0,0)* becomes *root=(hd0,1)*.

If you want to display the GRUB2 menu at a higher resolution than VGA and display a background graphic, you have to modify your GRUB2 configuration file to support the higher resolution. The *vbe.mod* loadable module provides drivers for standard VESA BIOS Extentions 2.0 graphics. In addition you have to supply a suitable font.for the text glyphs and a background image of suitable format and resolution. For example, the following is a simple GRUB2 configuration file which uses the *vbe.mod* and *gfxterm.mod* modules to display the GRUB2 menu with a resolution of 1024 x 768 and a full screen background image (*f13rockets.tga*) using a *unicode* font.

```
timeout=15
default=0

if loadfont /grub/font/unicode.pf2
then
   set gfxmode="1024x768x32"
   set gfxpayload=keep
   insmod gfxterm
   insmod vbe
   terminal_output gfxterm
```



```
   terminal gfxterm
   insmod tga
   use_bg=true
   background_image /grub/image/f13rockets.tga
fi

set color_normal=yellow/black
set color_highlight=blue/yellow
set menu_color_normal=yellow/black
set menu_color_highlight=white/light-gray

menuentry "Fedora 13" {
   root=(hd0,1)
   linux /vmlinuz-2.6.33.5-124.fc13.x86_64 ro root=/dev/mapper/vg_ultra-lv_root rd_LVM_LV=
vg_ultra/lv_root rd_LVM_LV=vg_ultra/lv_swap rd_NO_LUKS rd_NO_MD rd_NO_DM LANG=en_US.UTF-8
SYSFONT=latarcyrheb-sun16 KEYTABLE=us rhgb quiet
   initrd /initramfs-2.6.33.5-124.fc13.x86_64.img
}
```

I suggest you start with a very simple GRUB2 configuration file such as the first example shown above for the first reboot of your system after installing GRUB2. Once you can successfully boot your system using that configuration, then you can experiment with more complex and sophisticated configuration file. Walk before you run! The rescue mode in GRUB2 is better than that in GRUB Legacy but you will quickly run into problems if you are unfamiliar with GRUB2.

By the way, if you want GRUB2 to install the build and configuration scripts and binaries in more expected location than */usr/local*, here is how I configure my GRUB2 build:

```
# ./configure --prefix=/usr --sysconfdir=/etc --disable-largefile --enable-efiemu=no --dis
able-grub-fstest
```

During install, the subdirectories */usr/sbin*, */usr/bin*, */usr/lib/grub*, */usr/share/grub*, */etc/default* and */etc/grub.* are populated with the relevant GRUB2 files.

–

This post should be enough to enable you to migrate to booting your Fedora system using GRUB2. Good luck with the migration! If you discover that I have left out any step that should be included in this post, please let me know.

