Creating a UEFI Rescue DVD

Finnbarr P. Murphy
(fpm@fpmurphy.com)

Developing a UEFI rescue DVD containing a set of useful UEFI-related utilities is a personal project that I work on from time to time. I own a number of laptops with UEFI firmware and have encountered a myriad of strange UEFI-related booting issues on these laptops over the last few years. In fairness, some of these issues were due to early versions of UEFI firmware but a lot are not.

Here is my current build script for creating a UEFI-bootable ISO:

```
#!/bin/bash
ISONAME="urd-dvd.iso"
TOPDIR=$(pwd)
# preparation
[[ -d ./build ]] &amp;&amp; rm -rf ./build
[[ -f ${ISONAME} ]] &amp;&amp; rm -f ${ISONAME}
# create directories
mkdir -p ./build/EFI/BOOT
mkdir -p ./build/EFI/TOOLS
# populate directories
cp ${TOPDIR}/shell/*.EFI ./build/EFI/BOOT
cp ${TOPDIR}/tools/*.EFI ./build/EFI/TOOLS
# size El Torito image
cd ./build
ETSIZE=$(du -s EFI | cut -f 1)
let ETSIZE=($ETSIZE)/28
let ETSIZE=($ETSIZE)*320
# Create FAT32 filesystem and populate
dd if=/dev/zero of=urd-bin.img bs=1024 count=${ETSIZE}
mkdosfs -n "URD" urd-bin.img
mcopy -v -iurd-bin.img -s EFI ::/
# Make the ISO-9660 image
mkisofs -A "URD" -V "URD" -volset "URD" \
    -J -r -v -x ./lost+found -o ../${ISONAME} \
    -eltorito-alt-boot -efi-boot urd-bin.img \
    -no-emul-boot ./
#cd ${TOPDIR}
#rm -rf ./build
```

The *shell* directory contains the UEFI shell I want to include in the ISO image and the *tools* directory contains the UEFI tools I want included in the ISO image.

Here is what *dumpet,* a utility for debugging El Torito boot images, outputs:

```
$ dumpet -i urd-dvd.iso
Validation Entry:
    Header Indicator: 0x01 (Validation Entry)
    PlatformId: 0xef (EFI)
    ID: ""
    Checksum: 0x66aa
    Key bytes: 0x55aa
Boot Catalog Default Entry:
    Entry is bootable
    Boot Media emulation type: no emulation
    Media load address: 0 (0x0000)
```



```
    System type: 0 (0x00)
    Load Sectors: 21120 (0x5280)
    Load LBA: 39 (0x00000027)
$
```

Here is the contents of a DVD which I built today using the above script:

```
$ cd /run/media/fpm/URD
$ find . -ls
 1856     2 dr-xr-xr-x   3 fpm      fpm          2048 Dec 29 10:49 .
 1863     2 -r--r--r--   1 fpm      fpm          2048 Dec 29 10:49 ./boot.catalog
 1920     2 dr-xr-xr-x   4 fpm      fpm          2048 Dec 29 10:49 ./EFI
 2048     2 dr-xr-xr-x   2 fpm      fpm          2048 Dec 29 10:49 ./EFI/BOOT
 2054   754 -r--r--r--   1 fpm      fpm        771136 Dec 29 10:49 ./EFI/BOOT/BOOTX64.EFI
 1984     2 dr-xr-xr-x   2 fpm      fpm          2048 Dec 29 10:49 ./EFI/TOOLS
 1990    57 -r-xr-xr-x   1 fpm      fpm         58289 Dec 29 10:49 ./EFI/TOOLS/LISTCERTS.E
FI
 1994    48 -r-xr-xr-x   1 fpm      fpm         49083 Dec 29 10:49 ./EFI/TOOLS/SHOWEDID.EF
I
 1998    46 -r-xr-xr-x   1 fpm      fpm         46092 Dec 29 10:49 ./EFI/TOOLS/SHOWLENOVO.
EFI
 1870 10560 -r--r--r--   1 fpm      fpm      10813440 Dec 29 10:49 ./urd-bin.img
$
```

The three executables in */EFI/TOOLS* shown above are Lenovo T430 specific. You csn easily include your own tools. By the way, URD is shorthand for “UEFI Rescue DVD”.

UEFI firmware must support the El Torito multiple boot catalog support. El-Torito, initially released in January 1995, is a specification for creating bootable optical media like a CD-ROM or a DVD.

ON ISO 9660 media, sector sizes are 0x800. The first 15 sectors are unused and empty. The 16th sector contains the Primary Volume Descriptor which starts with “CD001”.

```
$ dd bs=1 count=200 skip=$((16 * 0X800 + 1)) if=urd-dvd.iso | xxd
0000000: 4344 3030 3101 004c 494e 5558 2020 2020  CD001..LINUX
0000010: 2020 2020 2020 2020 2020 2020 2020 2020
0000020: 2020 2020 2020 2055 5244 2020 2020 2020         URD
0000030: 2020 2020 2020 2020 2020 2020 2020 2020
0000040: 2020 2020 2020 2000 0000 0000 0000 0022         ........"
0000050: 1700 0000 0017 2200 0000 0000 0000 0000  ......".........
0000060: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000070: 0000 0000 0000 0001 0000 0101 0000 0100  ................
0000080: 0808 0030 0000 0000 0000 3015 0000 0000  ...0......0.....
0000090: 0000 0000 0000 1700 0000 0022 001d 0000  ..........."....
00000a0: 0000 0000 1d00 0800 0000 0008 0072 0c1d  .............r..
00000b0: 0a31 24ec 0200 0001 0000 0101 0055 5244  .1$.........URD
00000c0: 2020 2020 2020 2020
```

Booting begins by reading the Boot Record of the IS0 filesystem. The Boot Record resides at sector 17 and points to a Boot Catalog which is stored in one or more blocks in ISO filesystem. The Boot Catalog lists the available Boot Images (Catalog Entries), one per supported platforms. These Boot Images are marked as one of: *emulated floppy, emulated hard disk*, or *no-emulation*. In the build script, I specify *no-emulation* with the *-no-emul-boot* option.

```
$ dd bs=1 count=200 skip=$((17 * 0X800 + 1)) if=urd-dvd.iso | xxd
0000000: 4344 3030 3101 454c 2054 4f52 4954 4f20  CD001.EL TORITO
```



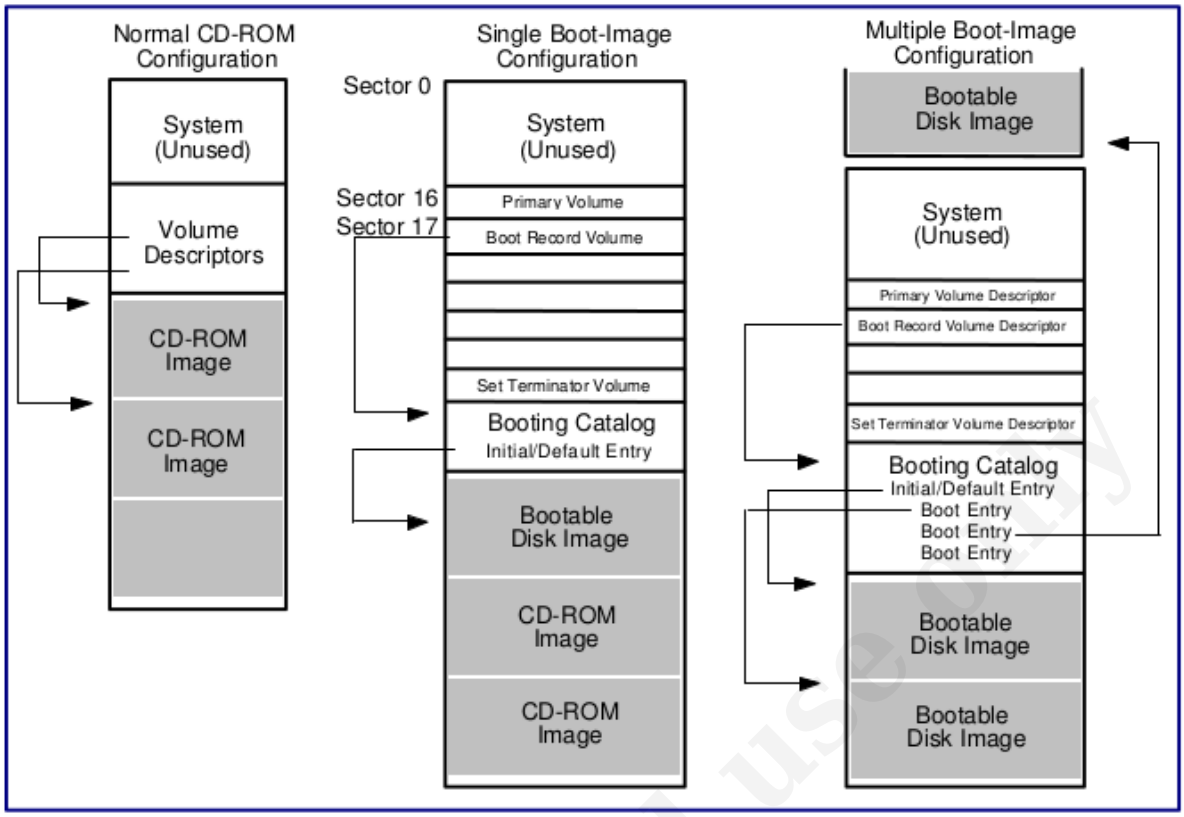
```
0000010: 5350 4543 4946 4943 4154 494f 4e00 0000  SPECIFICATION...
0000020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000040: 0000 0000 0000 2600 0000 0000 0000 0000  ......&.........
0000050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000060: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000070: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000080: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000090: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000a0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

The “CD001” string of the Boot Record indicates that this image is an ISO 9660. The “EL TORITO SPECIFICATION” string identifies the image as one that is potentially bootable. The DWORD at offset 47 (0x0026) is an absolute pointer to the first sector of the Boot Catalog.

```
$ dd bs=1 count=200 skip=$((0x26 * 0X800 + 1)) if=urd-dvd.iso | xxd
0000000: ef00 0000 0000 0000 0000 0000 0000 0000  ................
0000010: 0000 0000 0000 0000 0000 00aa 6655 aa88  ............fU..
0000020: 0000 0000 0080 5227 0000 0000 0000 0000  ......R'........
0000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000060: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000070: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000080: 0000 0000 0000 0000 0000 0000 0000 0000  ................
0000090: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000a0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000b0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

UEFI interprets a Boot Image with a Platform ID of 0xEF (see offset 0 above) as a FAT32 filesystem and uses a standardized filepath to find an executable to load. For URD, this is */EFI/BOOT/BOOTX64*. I probably should add a Manufacturer ID at offset 0x03. The maximum length of such an ID is 18. Note the “55AA” signature (which same as last two bytes in a MBR) in the second line of the output. The following byte is the Boot Indicator and indicates that the entry is bootable (88 – bootable, O – non-bootable). Offset Boot Indicator + 1 is the Boot Media Type byte (0 – no emulation, 4 – hard disk, etc.). Offset Boot Indicator + 8 contains the Load RBA DWORD which is the start address of the virtual disk.

Here is a diagram of the relationship between the various sections:



Note that you cannot boot this DVD on a non-UEFI system. Why not, you might ask? The reason is simple; the DVD contains no Boot Image with a Platform ID of 0x00. For example, this is what *dumpet* outputs for a dual boot Microsoft Vista SP1 X64 DVD:

```
Validation Entry:
    Header Indicator: 0x01 (Validation Entry)
    PlatformId: 0x00 (80x86)
    ID: "Microsoft Corporation"
    Checksum: 0x494c
    Key bytes: 0x55aa
Boot Catalog Default Entry:
    Entry is bootable
    Boot Media emulation type: no emulation
    Media load segment: 0x0 (0000:7c00)
    System type: 0 (0x00)
    Load Sectors: 4 (0x0004)
    Load LBA: 613 (0x00000265)
Section Header Entry:
    Header Indicator: 0x91 (Final Section Header Entry)
    PlatformId: 0xef (EFI)
    Section Entries: 1
    ID: ""
Boot Catalog Section Entry:
    Entry is bootable
    Boot Media emulation type: no emulation
    Media load address: 0 (0x0000)
    System type: 0 (0x00)
    Load Sectors: 1 (0x0001)
    Load LBA: 614 (0x00000266)
```

A Platform ID of 0x00 basically means a PC. The media load segment is 0000:7c00 which generally means a BIOS-based image. The load sectors are 4 which is always the case for a BIOS-based image. UEFI firmware must ignore catalog entries with a 0x00 Platform ID and read catalog entries with a 0xEF Platform ID.

Here is what Microsoft has to say about dual BIOS/UEFI boot media:

> The following guidelines apply for installation media on x64 platforms:
>
> – Windows installation media supports boot on both BIOS and UEFI platforms by taking advantage of multi-entry El Torito boot catalog support.
>
> – The default El Torito boot entry is a BIOS entry that includes the 80×86 Platform ID, which is defined as “0x00” in hexadecimal.
>
> – The second El Torito boot entry is an EFI entry that includes the Platform ID as “0xEF” in hexadecimal. The entry references a FAT partition that contains the bootable EFI application at \EFI\BOOT\BOOTX64.EFI.
>
> Firmware vendors must ensure that the following conditions exist:
>
> – The BIOS ignores boot entries that do not have the 80×86 Platform ID, which is defined as “0x00” in hexadecimal. Failure to ignore other boot entries results in the display of a confusing boot menu to the end user.
>
> – The BIOS boots based on the BIOS entry without prompt.
>
> – The UEFI boot manager ignores boot entries that do not have the “0xEF” Platform ID.
>
> – The UEFI boot manager boots based on the EFI entry without prompt.

The information provided in this post should be enough to get to started on creating your own custom UEFI-bootable DVDs. Happy experimenting and a Happy New Year!

