

## Multiboot CDs with Boot Scriptor

# Burn Baby Burn!

Boot floppies may be a thing of the past, but you still tend to find them heaped next to any modern PC. A CD writer, Linux, Boot Scriptor and the following guidelines are all you need to prepare these ancient media for “thermal recycling”.

SIMON RUTISHAUSER



In time, most Linux users will acquire a heap of boot floppies, I am speaking from experience here, as I recently discovered no less than 16 somewhat dusty floppies on my own shelves. Previous attempts to do away with them were doomed to failure, considering the fact that every one of them could have saved me hours of work if the worst came to the worst.

Help is at hand – instead of 16 floppies, I am now the proud owner of one do-it-yourself CD – a multiboot CD, to be more exact. Put simply, this is an El Torito boot CD (see the “El Torito: A little known standard”) that contains images of my boot floppies. When you boot from the CD, a menu pops up, allowing you to choose the floppy image of your choice to launch.

Compiling a CD of this kind is not as easy as you might think, but thanks to Linux and a helpful tool called Boot Scriptor, the obstacles on the road to a multiboot CD have become easier to manage.

Boot Scriptor [1] is, at the same time, a programmable and powerful front-end for diskemu [2] and isolinux/memdisk [3], as well as an interpreter for a simple scripting language that allows you to

display convenient and complex menus on screen.

### Licensing Quirks

Boot Scriptor is not licensed under the GPL, but has its own license that permits non-profit projects to use it provided they credit the author. Boot Scriptor can also be used for proprietary programs, although an undefined charge is levied in this case.

The program’s author reserves the right to refuse a license, with the inclusion of clauses like “And in extreme cases (although I can’t immediately think of a reason we might actually want to do this) we may refuse to let you do it at all.”

### Boot Scriptor Controls How Your PC Starts Up

Boot Scriptor allows you to boot from CD as shown in Figure 1. If your BIOS is capable of booting from CD (see “El Torito: A little known standard”), it will fire up automatically. ATAPI CD drives can also choose to detour via Smart Boot [4] before booting from CD.

The BIOS first loads a program from the CD boot sector – in Boot Scriptor’s case this is the *loader.bin* program. This

tool copies itself to an address other than 0x07c0 and runs Boot Scriptor. This is similar to an MBR without a boot manager that automatically runs specific code, typically the Linux kernel.

Boot Scriptor then searches for the *bscript.ini* script in */bscript* and launching it when found. Failing this, Boot Scriptor displays a command prompt. Finally, Boot Scriptor launches an external module – as specified in a script or on the command line – normally with the goal of simulating a floppy or hard disk boot.

Currently, Boot Scriptor provides the following five modules:

- Memdisk [3] helps to solve the issues Isolinux has with some BIOS types. Memdisk writes the complete image to a ramdisk before booting, thus allowing the running operating system to write to the virtual disk. Of course these changes are not persistent.

#### THE AUTHOR

Simon Rutishauser still attends the MNG Kirchenfeld highschool in Bern, Switzerland and dabbles in topics such as (multi)boot CDs, Java programming with SWT, Linux (Debian Potato, Kernel 2.2.17 and later), and recently FreeBSD.

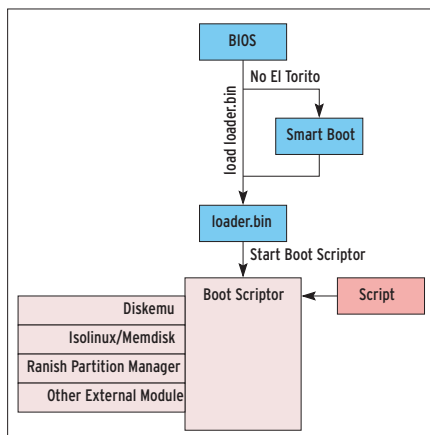


Figure 1: Booting from a multiboot CD with Boot Scriptor

- Isolinux [3] provides similar functionality to Boot Scriptor and also includes an emulator for booting images. Boot Scriptor provides a flexible scripting language and a front-end/backend instead.
- Diskemu [2]: The predecessor of Boot Scriptor was originally designed as a stand alone program; its development has been discontinued in favor of Boot

Scriptor. Programs launched from a floppy disk cannot write to disk when Diskemu is active.

In contrast to Memdisk, Diskemu does not store images in RAM. The tool is still useful in some scenarios where Isolinux/Memdisk will not run.

- The well-known Ranish Partition Manager tool also provides direct support for Boot Scriptor.
- Boot Scriptor is open for customization and enhancement; which will need to be written in Assembler at present, although support for C is planned.

An emulator for Boot CD ISO images is at the top of most people's list, as both Diskemu and Isolinux/Memdisk only support floppy and hard disk images at present. Volunteers with expert knowledge are welcome, says the author!

## Creating a Directory Structure

To create your own multiboot CD, your first need a directory tree as shown in

Figure 2. The easiest way to create a directory tree is to use the current version of Build Multiboot (refer to the section on burning later on in this article) and the Boot Scriptor plugin (downloadable from [20]):

```
unzip buildmultiboot-0.5.zip
unzip bootscriptor-1.2.14.zip
```

Only the contents of the *lib/bscript* and *cds/bscript/disk1/bscript* directories in this directory tree will actually be placed on the boot CD. The former contains the current Boot Scriptor [1] version, or to be more precise the *bscript* folder from the binary archive.

Alternatively, you can type the following to update to the latest Boot Scriptor version after downloading (currently 1.2.18; 1.1 is no longer supported):

```
cd /tmp
unzip >
Boot\ Scriptor\ (1.2.18).zip
cp -R bscript ~/BuiltMultiboot>
/lib/bscript
```

## El Torito: A little known standard

IBM and BIOS manufacturer Phoenix released the El Torito standard in 1994, and its principles are accepted world-wide. El Torito envisages three ways of booting from CD:

- No Emulation: The BIOS loads a fixed number of sectors from the CD (up to 640 Kbytes, the real mode memory limit for a PC) and runs the code.
- Floppy emulation: The BIOS handles the image of a floppy on CD (1.2 Mbytes, 1.44 Mbytes or 2.88 Mbytes) like a genuine floppy.
- Hard disk emulation: The BIOS handles the image of a hard disk – with a maximum size of 700 Mbytes like a real hard disk. Of course, read access is slower than for a genuine hard disk, and many operating systems have difficulty coping with the read-only medium.

Practical implementations of the specification are tricky. Floppy and hard disk emulation is hardly supported, and BIOS support for multiboot CDs, as originally envisaged, is totally obscure. All BIOSs support no emulation mode, and that means that running Boot Scriptor is no trouble at all.

## Booting from IDE Drives

The BIOS on the motherboard has the responsibility for booting ATAPI drives. The El Torito specification is an issue with many older boards, and even some new ones. The reason for this is that the mechanism is a lot more complicated than booting a hard disk or floppy.

Help: The Smart Boot bootmanager from [4] is capable of booting from ATAPI CD drives, no matter whether the BIOS supports El Torito. Stored on a floppy it provides invaluable support for users of Boot Scriptor or Knoppix CDs [5].

## Booting from SCSI Drives

SCSI drives boot differently; typically the Motherboard BIOS has nothing to do with this process as the adapter will provide a BIOS of its own. It is the adapter BIOS that is responsible for implementing the El Torito standard. SCSI adapters without a BIOS are either incapable of booting – this applies equally to hard disks – or they place this responsibility firmly in the hands of the motherboard BIOS. A Smart Boot extension to provide native support for SCSI drives is not planned, and highly unlikely due to the lack of space in the MBR.

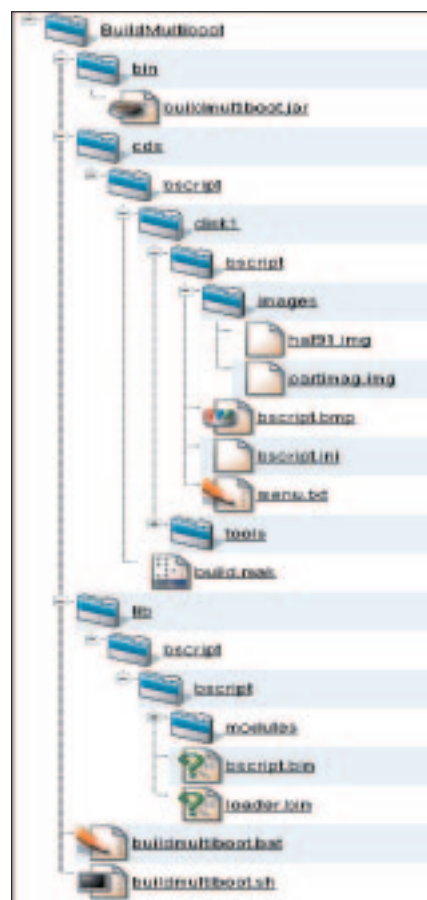


Figure 2: A directory tree like the one shown is needed for creating multiboot CDs



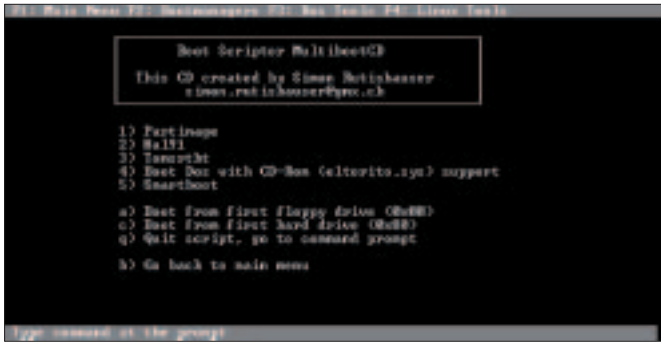


Figure 3: A simple menu for a multiboot CD, as defined by *bscript.ini*. This example shows Listing 1 and 2 in action



Figure 4: A graphical menu that is quite complicated to program, although it does not actually offer more functionality than the menu in Figure 3

presses a key, *keys* first performs a few global queries; for example, [Q] and [Esc] quit the script. If the user does not press any of these keys, Boot Scriptor moves down to the *return* keyword in line 34 and then to line 8 where the method is called, and then goes on to parse the next line.

Keypresses are used to select an image for launching. If the user presses the [1] key, the instructions in *keys* are omitted and the command *memdisk partimag.img* at line 10 is launched. *memdisk* or *diskemu* will terminate Boot Scriptor and launch the corresponding image.

When the user shuts down a Linux system launched by *partimag.img* [7], the PC will not return to Boot Scriptor, but go down normally. A list of script commands is available from [1]. If you feel ambitious, you can use [19] to build

more complex menus that deserve to be called graphic, as shown in Figure 4, for example. Is it really worth the effort?

The menu in Figure 3 is just as easy to use as the one in Figure 4. You can even integrate a bitmap, the Windows equivalent of an XPM, as a splash screen that hides the console while it displays. The bitmap file is called *~/BuildMultiboot/cds/bscript/disk1/bscript/bscript.bmp*, and the image can be 320 by 240 pixels with 8 bit depth. The file cannot be larger than 126 Kbytes and must be RLE compressed.

Alternatively, you can use the *tolls/convert* tool from the standard Boot Scriptor distribution to convert a 24 bit, 640 by 480 pixel image, and display it instead of the bitmap. The path will be *~/BuildMultiboot/cds/bscript/disk1/bscript/bscript.bsi* in this case. The *show*

*image file.bmp* command displays the bitmap in the script, and *show console* restores the console.

The *bscriptw.com* from the Boot Scriptor package can be used to test your script. It should allow you to test more complex menus and create screenshots without wasting any blank CDs. *Bscriptw.com* must be in the same folder as the *bscript.ini* file you want to test; the only way to do this at present – due to restrictions in *Bscriptw.com* – is to create a temporary copy of *bscriptw.com* and the *modules* directory. Unfortunately, the program will not run on Wine.

### Burning: Traditional Approach or Swing Front-end

All that remains now is to store the results of that hard work on a CD, a task

Table 1: Practical Boot Disks

Name	Description
Eltorito.sys	A boot disk with DOS and the Eltorito.sys CD driver is a must. It allows you to access the CD drive you booted from no matter whether it is attached to an IDE, SCSI, USB or Firewire bus. See the "DOS Boot Disk with Eltorito.sys" for further details. Download from [2].
Grub	The boot manager runs Linux from almost any file system. Grub searches the path to find the kernel in the filesystem and only needs to re-write the MBR in case of massive changes. Cannot boot from CD! Download from [11].
Smart Boot	The mini boot manager may not be able to hold a candle to Grub for normal operations, but it can boot from IDE CD ROM drives, independently of BIOS support for this feature. Download from [4].
GNU Parted	The partitioning program moves, deletes, resizes, and creates FAT, Ext-2/3 and ReiserFS partitions. Mandrakes Diskdrake is based on GNU Parted. Download from [6].
Partimage	Can create an image of any partition (FAT, NTFS, ReiserFS, BSD-FFS, Ext-2/-3, JFS, XFS), either on another partition or across a network. Partimage uses <i>gzip</i> or <i>bzip2</i> to compress images. Download from [7].
Toms Root	Tomsrtbts Motto "The most Linux on one Disk" is programmatic. Available either on an overlength 1.44 Mbyte or a 2.88 Mbyte floppy, where the former uses the Lilo bootloader, which can cause problems. Download from [8].
Hal91	Not as many programs as Tomsrtbt, but Hal91 requires only a normal 1.44 Mbyte format floppy. Download from [9].
Fli4l	An all purpose Mini-Linux from print server, through firewall to wireless router. Download and extensive documentation from [10].
Do-it-Yourself	A specialized Linux version (based on [3]) with kernel 2.4, USB and Firewire.

Table 2: Boot Scriptor's Color Codes

Hexcode	Color
0	Black
1	Blue
2	Green
3	Turquoise
4	Red
5	Pink
6	Orange
7	Light gray
8	Dark gray
9	Light blue
a	Light green
b	Light turquoise
c	Light red
d	Light pink
e	Yellow
f	White

typically performed by `mkisofs` and `cdrecord`:

```
mkisofs -no-emul-boot -eltorito
-boot BSCRIPT/loader.bin -hide
BSCRIPT/loader.bin -c BSCRIPT
/boot.catalog -hide BSCRIPT/
boot.catalog -boot-load-
size 4 -boot-
-load-seg 0x07C0
```

```
~/BuildMultiboot/lib/bscript
~/BuildMultiboot/cds/bscript/
disk1 | cdrecord -dev=0,0,0
-speed=4 -eject -
```

Command-line challenged user might also like to look into the Swing Build Multiboot program (see Figure 5). It requires a current Java 2 JRE. The front-end is launched by `java -jar build`

`multiboot.jar` in the `~/BuildMultiboot/bin` directory or by `~/BuildMultiboot/buildmultiboot.sh`. The dialog prompts you for a SCSI device, the CD you want to burn and a few other options.

You might prefer to start with a CDRW, as first attempts very rarely work perfectly, but some trial and error should quickly lead to useable results. And that brings us right back to “thermal recycling” for floppies – now where’s that lighter?

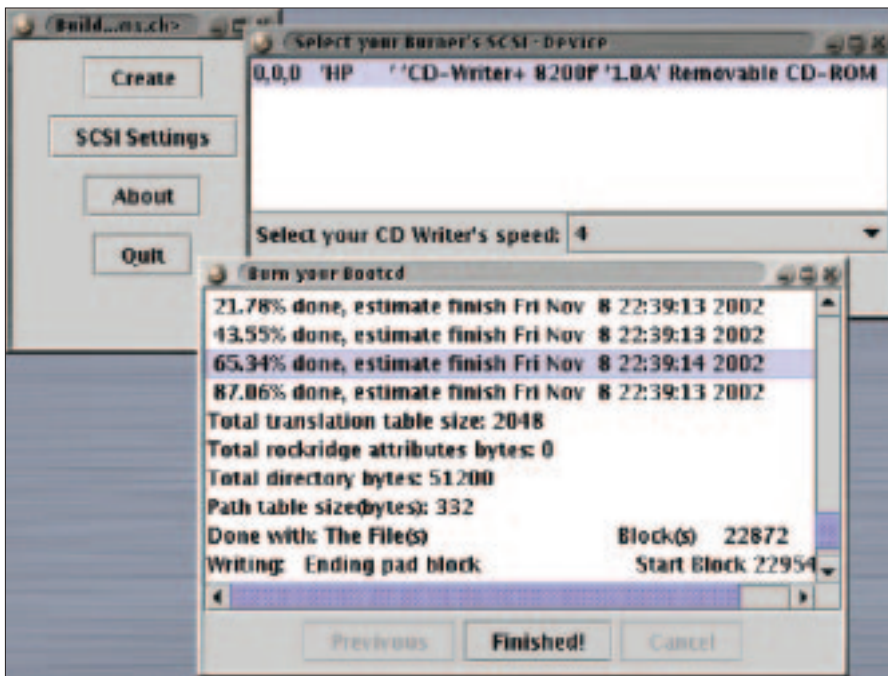


Figure 5: Build Multiboot is a comfortable Java Swing front-end for burning multiboot CDs. It is ideal for users who are allergic to endless chains of commands

## A DOS Boot Disk with Eltorito.sys

The Eltorito.sys CD ROM driver may be a piece of DOS software but interesting all the same. It enables DOS to access the CD ROM drive it booted from, no matter whether the drive is attached to a SCSI controller, a USB, IDE or parallel port. Thanks to Bart Lagerweij's documentation, it is quite simple to create a suitable disk (Bart's El Torito CD-ROM Boot Disk [2]):

- Create a DOS boot floppy using MS DOS 6.x or later: `format a: /u /s`.
- If the `drvspace.bin` is on the disk: `attrib -s -h -r a:\drvspace.bin` and `del a:\drvspace.bin`.
- Now create a folder called `\bin` on the disk and add the DOS files `himem.sys`, `emm386.exe`, and `smartdrv.exe` to the folder.
- Download the `modboot21.zip` archive from [12] and extract it on the disk under Linux: `mount /floppy; cd /floppy; unzip ~/modboot21.zip`.
- Download archives [13] through [18] and copy them to the corresponding directories on the disk (do not extract them!): [13] to `\level3`; [14] to `\lib`; [15] and [16] to `\level0`; [17] and [18] to `\level1`.
- Optionally create a text file, `\diskid.txt`, on the disk.
- Create an image of the floppy on Linux using `cat /dev/fdo > eltorito.img`.
- Following the instructions in this article, add the image to your multiboot CD.

If you do not own an MS-DOS license, or are a staunch Open Source supporter, you can of course use FreeDOS [21]. The boot CD available on the website also uses Eltorito.sys; thus you can either use the image directly, or as a template for your own disk.

## INFO

- [1] Boot Scriptor: <http://www.bootscriptor.org>
- [2] Diskemu and Eltorito boot disk: <http://www.nuz.nu>
- [3] Isolinux and Syslinux: <http://syslinux.zytor.com>
- [4] Smart Boot: <http://btmgr.sourceforge.net>
- [5] Knoppix: <http://www.knopper.net/knoppix>
- [6] GNU Parted: <http://www.gnu.org/software/parted/parted.html>
- [7] Partimage: <http://www.partimage.org>
- [8] Tomsrtbt: <http://www.toms.net/rb/>
- [9] Hal91: <http://www.itm.tu-clausthal.de/~perle/hal91/>
- [10] Fli4l: <http://www.fli4l.de>
- [11] Grub: <http://www.gnu.org/software/grub>
- [12] Modboot 2.1 files: <http://download.nuz.nu/nuzfiles/modboot21.zip>
- [13] Eltorito.sys CD driver: <http://download.nuz.nu/nuzfiles/elboot.cab>
- [14] CD ROM extensions: <http://download.nuz.nu/nuzfiles/mscdex.cab>
- [15] Batch utilities: <http://download.nuz.nu/nuzfiles/utills.cab>
- [16] CD Autorun: <http://download.nuz.nu/nuzfiles/cdauntrun.cab>
- [17] Help modules: <http://download.nuz.nu/nuzfiles/help.cab>
- [18] Mouse support: <http://download.nuz.nu/nuzfiles/mouse.cab>
- [19] Complex boot menu: <http://91cd.tripod.com>
- [20] Build Multiboot: <http://buildmultiboot.sourceforge.net>
- [21] FreeDOS: <http://www.freedos.org>