Specifying the Order of Files on a DVD – Without the Knife

Jim Brown, BSDCG
jimbyg@gmail.com
Sept. 15, 2016

Every once in a while, you run across a problem that seems intractable – a real ball buster. You study it, google about it, ask friends and colleagues, rip out your hair, curse deities on multiple planets, and still you find yourself no closer to a solution. Occasionally, you'll hit on a workaround for one of these zingers, but you know it's not the solution you really need. As it happens, I just found the perfect solution for one of mine.

For the past several years, I have been producing the BSD Certification Group's study DVD. These DVDs include four operating systems on one disc and include an 'El Torito' boot loader, allowing any of the four to be booted to their installation program. Every time I begin my round up of the latest releases of DragonFly BSD, FreeBSD, NetBSD and OpenBSD, I dread the usual fight with each operating system. Producing these DVDs requires code changes to the boot and installer programs for each BSD. I've gotten them to work by modifying and recompiling the code, and staging them for use with the mkisofs(8) pre-mastering program. Usually, the easiest to change is OpenBSD – and it's also the most frustrating to get to work reliably. I sometimes spend many hours in a kungfu-voodoo-knife fight getting it to work.

Here's why. Figure 1 has a a simplified linear layout of the most recent DVD. The ELTORO[1] boot loader is at the beginning of the disc, with various ISO 9660 components after that. The first operating system starts at around 22M into the DVD. I've shown a "lexicographic order" (lexi order) of the operating systems on the disc, and this was the desired original layout. It turns out that this didn't work and hasn't worked for quite some time.

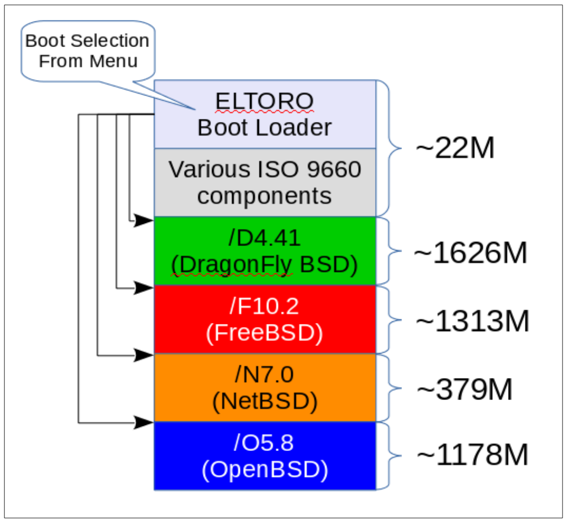

Fig. 1 - Simplified Layout on DVD

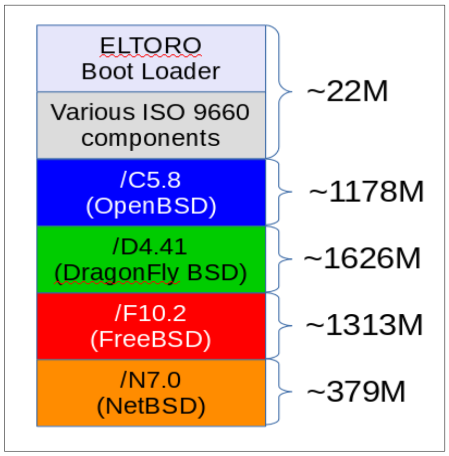

Fig. 2 - Reordering OpenBSD by Renaming the Directory

OpenBSD's cdboot(8) loader has a signed variable somewhere causing it to not be able to read the boot.conf configuration file or load the kernel if the start of the OpenBSD code is past the ~2.1G (2,147,483,647 bytes to be exact) signed 32-bit integer boundary. Thus, when the order is that shown

1 The "ElToro" CD/DVD boot manager. Copyright © 2006 by Oliver Fromme. Used by permission.

in Figure 1 and OpenBSD is selected, cdboot can't find the kernel and it won't load.

Thinking myself clever, I renamed the directory for the OpenBSD distribution to “C5.8”, thus aiming to have it written earlier to disc as shown in Figure 2. I have used this same technique for the last four or five DVDs – always renaming the OpenBSD directory to be C5.x and (eventually) getting the DVD created. Once I win the fight and get all four BSDs to boot and install, I quickly get the DVD burned and copied. By this time, I'm so agitated at having OpenBSD start to work then fail to work then start to work again, that I just want to stop, so I stop.

But why is it always such a pain? What is causing OpenBSD to work sometimes, and – even with my “clever” reordering – not work other times, often just an hour or two later fighting with all the other BSDs?

The answer is not in OpenBSD at all. It's in the implementation of mkisofs(8) – the .iso pre-mastering program.

mkisofs(8), part of the cdrtools[2] bundle, is a terrific program for making .iso files. You just point it at a directory with a few options, and bam! it creates a .iso file ready for mastering. mkisofs(8) has a large number of options allowing fine grained control over including/excluding files and directories, Joliet and Rock Ridge file name mangling, publishing details, boot files, and many other items.

When you ask mkisofs(8) to create a .iso file, it scans through the files you selected and sets the data blocks for these files in order to be written to the output .iso file. Since we see files and directories and list their contents all the time, we intuitively think that mkisofs(8) will put them in lexi order as ls(1) does by default. But – it does not.

mkisofs(8) uses an internal implementation of find(1) to select the files and data blocks, and it orders them in a depth-first traversal. It's equivalent to running find(1) on a directory with the parameters “-type d” (which appears to be the default on many systems). And the important detail here is that mkisofs(8) takes sub-directories **in order as they appear in the current directory**, which can vary if the filesystem code reorders the entries in a directory.[3]

Consider the following list of sub-directories:

```
# ls -al
total 11
drwxr-xr-x  7 root  jpb   7 Sep 10 20:31 .
drwxrwxrwx  5 root  jpb  14 Sep 10 20:31 ..
drwxr-xr-x  2 root  jpb   3 Sep 10 15:17 A
drwxr-xr-x  3 root  jpb   3 Sep 10 15:10 B
drwxr-xr-x  3 root  jpb   3 Sep 10 15:10 C
drwxr-xr-x  3 root  jpb   3 Sep 10 15:10 D
drwxr-xr-x  3 root  jpb   3 Sep 10 15:10 E
#
```

The ls(1) command has, by default, output them in lexi order. The '-f' option will cancel that ordering

2 Homepage at http://cdrtools.sourceforge.net/private/cdrecord.html, currently maintained by Jörg Schilling.

3 Jörg notes: “The ISO-9660 directories are all sorted as required by the standard. Your text seems to create a different impression. The order of data blocks on the medium is what you really refer to, so the example with the directory more or less confuses the reader. You may like to modify your text to make it obvious that you are not talking about directories.”

resulting in:

```
# ls -alf
total 11
drwxr-xr-x  7 root  jpb   7 Sep 10 20:31 .
drwxrwxrwx  5 root  jpb  14 Sep 10 20:31 ..
drwxr-xr-x  3 root  jpb   3 Sep 10 15:10 B
drwxr-xr-x  3 root  jpb   3 Sep 10 15:10 E
drwxr-xr-x  2 root  jpb   3 Sep 10 15:17 A
drwxr-xr-x  3 root  jpb   3 Sep 10 15:10 D
drwxr-xr-x  3 root  jpb   3 Sep 10 15:10 C
#
```

This is the actual order of these sub-directories within the current directory.

The find(1) command using the “-type d” option (and “-depth 1” to see just the current directory) results in an unordered listing similar to “ls -alf”

```
# find . -type d  -depth 1
./B
./E
./A
./D
./C
#
```

It turns out that mkisofs(8) uses this ordering as its default when it gathers data blocks to write them to the .iso file.

Let's look at some real examples. I've created some sub-directories and some files in these sub-directories as follows (shown in lexi order):

```
# find . -print | sort
.
./A
./A/1.bin
./B
./B/two
./B/two/2.bin
./B/two/three
./B/two/three/3.bin
./C
./C/four
./C/four/4.bin
./C/four/five
./C/four/five/5.bin
./C/four/five/six
./C/four/five/six/6.bin
./D
./D/seven
./D/seven/7.bin
./E
./E/eight
./E/eight/8.bin
./E/eight/nine
./E/eight/nine/9.bin
#
```

Each .bin file has 1000 bytes of the character of its base name (2, 3, 4, etc), created with jot(1) as follows:

```
# for i in 1 2 3 4 5 6 7 8 9
> do
> echo ${i}
> jot -b ${i} -s "" 1000 > ${i}.bin
> done
1
2
3
4
5
6
7
8
9
#
# cat 4.bin
44444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444
44444444444444444444444444444444444444444444444444444444444444444444444444444
4444444444444444444444444444444444444444444444444444444444444444444444444444
#
```

I then manually moved them into the directory named after the number of the .bin file – 2.bin was moved to the directory ./B/two/, 6.bin was moved into ./C/four/five/six/ and so on.

Side by side, the listed files and directories look much different with the sorted find(1) output and the (annotated) listing produced by find(1) with the "-type d" parameter.

find . -print | sort	**find . -type d** [annotated]
.	.
./A	./B
./A/1.bin	./B/two [2.bin is here]
./B	./B/two/three [3.bin is here]
./B/two	./E
./B/two/2.bin	./E/eight [8.bin is here]
./B/two/three	./E/eight/nine [9.bin is here]
./B/two/three/3.bin	./A [1.bin is here]
./C	./D
./C/four	./D/seven [7.bin is here]
./C/four/4.bin	./C

./C/four/five	./C/four [4.bin is here]
./C/four/five/5.bin	./C/four/five [5.bin is here]
./C/four/five/six	./C/four/five/six [6.bin is here]
./C/four/five/six/6.bin	
./D	
./D/seven	
./D/seven/7.bin	
./E	
./E/eight	
./E/eight/8.bin	
./E/eight/nine	
./E/eight/nine/9.bin	

Running mkisofs(8) on the current directory creates a .iso file as follows:

```
      mkisofs -iso-level 3 -gui -v -R -l -J -o test_mkisofs.iso .
```

To see the contents in order, use the hd(1) command on the test_mkisofs.iso file:

```
      # hd test_mkisofs.iso
      00000000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
      *
      00008000  01 43 44 30 30 31 01 00  46 72 65 65 42 53 44 20  |.CD001..FreeBSD |
      00008010  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  |                |
      00008020  20 20 20 20 20 20 20 20  43 44 52 4f 4d 20 20 20  |        CDROM   |
      00008030  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  |                |
      ...
            lots of ISO 9660  component content
      …
      0001c0b0  4e 54 49 46 49 45 52 20  49 4e 20 50 52 49 4d 41  |NTIFIER IN PRIMA|
      0001c0c0  52 59 20 56 4f 4c 55 4d  45 20 44 45 53 43 52 49  |RY VOLUME DESCRI|
      0001c0d0  50 54 4f 52 20 46 4f 52  20 43 4f 4e 54 41 43 54  |PTOR FOR CONTACT|
      0001c0e0  20 49 4e 46 4f 52 4d 41  54 49 4f 4e 2e 00 00 00  | INFORMATION....|
      0001c0f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
      *
      0001c800  23 21 2f 62 69 6e 2f 73  68 0a 73 65 74 20 2d 78  |#!/bin/sh.set -x|
      0001c810  0a 0a 6d 6b 69 73 6f 66  73 20 2d 69 73 6f 2d 6c  |..mkisofs -iso-l|
      0001c820  65 76 65 6c 20 33 20 2d  67 75 69 20 2d 76 20 2d  |evel 3 -gui -v -|
      0001c830  52 20 2d 6c 20 2d 4a 20  2d 6f 20 74 65 73 74 5f  |R -l -J -o test_|
      0001c840  6d 6b 69 73 6f 66 73 2e  69 73 6f 20 2e 20 0a 0a  |mkisofs.iso . ..|
      0001c850  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
      *
      0001d000  32 32 32 32 32 32 32 32  32 32 32 32 32 32 32 32  |2222222222222222|
      *
      0001d3e0  32 32 32 32 32 32 32 32  0a 00 00 00 00 00 00 00  |22222222........|
      0001d3f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
      *
      0001d800  33 33 33 33 33 33 33 33  33 33 33 33 33 33 33 33  |3333333333333333|
      *
      0001dbe0  33 33 33 33 33 33 33 33  0a 00 00 00 00 00 00 00  |33333333........|
      0001dbf0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
      *
      0001e000  38 38 38 38 38 38 38 38  38 38 38 38 38 38 38 38  |8888888888888888|
      *
      0001e3e0  38 38 38 38 38 38 38 38  0a 00 00 00 00 00 00 00  |88888888........|
      0001e3f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
      *
```

```
0001e800  39 39 39 39 39 39 39 39  39 39 39 39 39 39 39 39  |9999999999999999|
*
0001ebe0  39 39 39 39 39 39 39 39  0a 00 00 00 00 00 00 00  |99999999........|
0001ebf0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
0001f000  31 31 31 31 31 31 31 31  31 31 31 31 31 31 31 31  |1111111111111111|
*
0001f3e0  31 31 31 31 31 31 31 31  0a 00 00 00 00 00 00 00  |11111111........|
0001f3f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
0001f800  37 37 37 37 37 37 37 37  37 37 37 37 37 37 37 37  |7777777777777777|
*
0001fbe0  37 37 37 37 37 37 37 37  0a 00 00 00 00 00 00 00  |77777777........|
0001fbf0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00020000  34 34 34 34 34 34 34 34  34 34 34 34 34 34 34 34  |4444444444444444|
*
000203e0  34 34 34 34 34 34 34 34  0a 00 00 00 00 00 00 00  |44444444........|
000203f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00020800  35 35 35 35 35 35 35 35  35 35 35 35 35 35 35 35  |5555555555555555|
*
00020be0  35 35 35 35 35 35 35 35  0a 00 00 00 00 00 00 00  |55555555........|
00020bf0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00021000  36 36 36 36 36 36 36 36  36 36 36 36 36 36 36 36  |6666666666666666|
*
000213e0  36 36 36 36 36 36 36 36  0a 00 00 00 00 00 00 00  |66666666........|
000213f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
0006c800
#
```

The files are shown in “find . -type d” order within the .iso file. *Thus, the times when my OpenBSD directory “C5.x” was the first BSD on the .iso was not due to lexicographic ordering based on the filename– it was the chance ordering that “C5.x” was actually the first entry in the directory holding all four BSDs.* That's why it would work sometimes and not work other times.

Ok, not quite as clever as I thought I was. Now – how to fix?

In November, 2001, James Pearson contributed an enhancement to mkisofs(8), the “-sort” option. In the accompanying README.sort file he explains how directories and files can be ordered as desired by assigning numerical weights to the directories and files by means of an external sort file. The explanation in README.sort is enough to determine how to sort the files in the .iso file in whatever order is desired.

Here's an example of sorting the files in numerical ascending order. Higher weights are output earlier in the .iso file:

```
# cat ../sort_asc.txt
./A 10000
./B/two 9000
./B/two/three 8000
./C/four 7000
./C/four/five 6000
./C/four/five/six 5000
```

```
./D/seven 4000
./E/eight 3000
./E/eight/nine 2000
#
```

Running the mkisofs(8) command with this ascending sort file:

```
mkisofs -sort ../sort_asc.txt -iso-level 3 -gui -v -R -l -J -o test_mkisofs.iso .
```

Results in the output:

```
0001c0d0  50 54 4f 52 20 46 4f 52  20 43 4f 4e 54 41 43 54  |PTOR FOR CONTACT|
0001c0e0  20 49 4e 46 4f 52 4d 41  54 49 4f 4e 2e 00 00 00  | INFORMATION....|
0001c0f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
0001c800  31 31 31 31 31 31 31 31  31 31 31 31 31 31 31 31  |1111111111111111|
*
0001cbe0  31 31 31 31 31 31 31 31  0a 00 00 00 00 00 00 00  |11111111........|
0001cbf0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
0001d000  32 32 32 32 32 32 32 32  32 32 32 32 32 32 32 32  |2222222222222222|
*
0001d3e0  32 32 32 32 32 32 32 32  0a 00 00 00 00 00 00 00  |22222222........|
0001d3f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
0001d800  33 33 33 33 33 33 33 33  33 33 33 33 33 33 33 33  |3333333333333333|
*
0001dbe0  33 33 33 33 33 33 33 33  0a 00 00 00 00 00 00 00  |33333333........|
0001dbf0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
0001e000  34 34 34 34 34 34 34 34  34 34 34 34 34 34 34 34  |4444444444444444|
*
0001e3e0  34 34 34 34 34 34 34 34  0a 00 00 00 00 00 00 00  |44444444........|
0001e3f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
0001e800  35 35 35 35 35 35 35 35  35 35 35 35 35 35 35 35  |5555555555555555|
*
0001ebe0  35 35 35 35 35 35 35 35  0a 00 00 00 00 00 00 00  |55555555........|
0001ebf0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
0001f000  36 36 36 36 36 36 36 36  36 36 36 36 36 36 36 36  |6666666666666666|
*
0001f3e0  36 36 36 36 36 36 36 36  0a 00 00 00 00 00 00 00  |66666666........|
0001f3f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
0001f800  37 37 37 37 37 37 37 37  37 37 37 37 37 37 37 37  |7777777777777777|
*
0001fbe0  37 37 37 37 37 37 37 37  0a 00 00 00 00 00 00 00  |77777777........|
0001fbf0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00020000  38 38 38 38 38 38 38 38  38 38 38 38 38 38 38 38  |8888888888888888|
*
000203e0  38 38 38 38 38 38 38 38  0a 00 00 00 00 00 00 00  |88888888........|
000203f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00020800  39 39 39 39 39 39 39 39  39 39 39 39 39 39 39 39  |9999999999999999|
*
00020be0  39 39 39 39 39 39 39 39  0a 00 00 00 00 00 00 00  |99999999........|
00020bf0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
```

```
00021000  23 21 2f 62 69 6e 2f 73  68 0a 73 65 74 20 2d 78  |#!/bin/sh.set -x|
00021010  0a 0a 6d 6b 69 73 6f 66  73 20 2d 73 6f 72 74 20  |..mkisofs -sort |
00021020  2e 2e 2f 73 6f 72 74 5f  61 73 63 2e 74 78 74 20  |../sort_asc.txt |
00021030  2d 69 73 6f 2d 6c 65 76  65 6c 20 33 20 2d 67 75  |-iso-level 3 -gu|
00021040  69 20 2d 76 20 2d 52 20  2d 6c 20 2d 4a 20 2d 6f  |i -v -R -l -J -o|
00021050  20 74 65 73 74 5f 6d 6b  69 73 6f 66 73 2e 69 73  | test_mkisofs.is|
00021060  6f 20 2e 20 0a 0a 00 00  00 00 00 00 00 00 00 00  |o . ............|
00021070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
0006c800
```

Clearly, the .iso is now sorted as desired. Note also that the script file, which showed up at the beginning in the earlier output, is now at the end. It was not assigned a weight in the sort_asc.txt file, and by default its weight is zero, less than all the other specified files. All files in a directory inherit the weight of that directory and its parent(s). It is therefore possible to sort every single file on the .iso in whatever order is required.

The answer then, for my semi-annual knife fight is to order the directories the way I *thought* I already was. Here is the sort file for the next DVD:

```
./ELTORO 30000
./etc 20000
./C6.0 10000
./D4.6 5000
./dloader.rc 5000
./boot.cfg 4000
./N70.1 4000
./F11.0 3000
```

Final note - I've stopped cursing deities on other planets, put my knives and voodoo dolls away, and can now answer James Pearson's rhetorical comment at the end of the README.sort file:

> “I have no idea if this is really useful ...”
> James Pearson 22-Nov-2001

Yes, James. Yes it really, really is. Thank you!

-jpb