Creating a Customized Boot CD/DVD for the Solaris™ Operating System for x86 Platforms

John Cecere, Sun Services
Dana Fagerstrom, Sun Services

Sun BluePrints™ OnLine — December 2005

Part No 819-3731-10
Revision 1.0, 11/29/05
Edition: December 2005

Please
Recycle

Creating a Customized Boot CD/DVD for the Solaris™ Operating System for x86 Platforms 1
Introduction 1
A New Boot Architecture 2
Disk Layout (Hard Disk) 3
Partition Structures 3
Solaris Disk Device Paths 4
fdisk Master Boot Record 5
mboot, pboot, and bootblk 6
Solaris System VTOC 6
Disk Layout (CDROM) 7
El Torito 8
fdisk on CDROM 8
Solaris System VTOC on CDROM 9
Steps for Creating a Customized Solaris on x86 Boot CD/DVD 9
Step 1: Create an ISO9660 File System 9
Step 2: Create a Solaris Miniroot 10
Step 3: Create the Boot Files 12
Step 4: Create the Composite ISO CD Image 13
Conclusion 13
References and Related Sources 14
About the Authors 14
John Cecere 14
Dana Fagerstrom 14
Ordering Sun Documents 15
Accessing Sun Documentation Online 15

Creating a Customized Boot CD/DVD for the Solaris™ Operating System for x86 Platforms

This Sun BluePrints™ Online article provides all the information necessary to create a customized boot CD/DVD for the Solaris™ Operating System for x86 Platforms (Solaris on x86). This article consists of the following sections:

- Introduction
- A New Boot Architecture
- Disk Layout (Hard Disk)
- Disk Layout (CDROM)
- Steps for Creating a Customized Solaris on x86 Boot CD/DVD
- Conclusion
- References and Related Sources
- About the Authors
- Ordering Sun Documents
- Accessing Sun Documentation Online

Note – Unless otherwise noted, this article makes no distinction between a CD or DVD. Because the file systems and layout used are the same, a DVD in the context of this article is just a CD with a lot more space (4.7GB versus 650MB) available. Because of this, the terms CD and CDROM can be used interchangeably with DVD in this article.

Introduction

This purpose of this article is to explain the mechanics of the Solaris on x86 boot process so that you understand what is needed to create a customized CD. It explains both the hard disk and CD boot processes, and points out the differences between the two.

There are a number of practical applications for this topic:

- **Solaris JumpStart™ software**—The feature in Solaris that allows access to Solaris installation media and configuration rules over a network.
- **Diagnostics**—The ability to create a bootable CD for the purpose of diagnosing system problems without accessing or modifying the copy of the operating system that is installed on the target system.
- **Restoration**—The ability to create a bootable CD with tools that aid in the repair and restoration of a down system.
- **Diskless clients that cannot do PXE booting**—PXE is a DHCP-based network-based installation technology similar to Solaris Jumpstart software. Some older x86-based systems are incapable of using PXE.
- **Canned Firewall**—The creation of a bootable CD that starts Solaris on a system configured with multiple network interfaces. A preset ipf configuration is then used to establish a network firewall on that system.

The following topics are outside the scope of this article: X86BOOT boot partition (refer to http://www.ata-atapi.com/hiwmbr.htm), floppy disk booting (refer to http://www.ata-atapi.com/hiwdos.htm), PXE booting (refer to http://docs.sun.com/app/docs/doc/817-5504/6mkv4nh8g?a=view), and booting other (non-Solaris) operating systems.

Information on Solaris JumpStart software has also been intentionally left out of this article because a wealth of available information already exists, including several Sun Blueprints documents described in “References and Related Sources” on page 14. For more information about Solaris JumpStart software, refer to: http://docs.sun.com/app/docs/doc/817-5506/6mkv6ki5q?q=jumpstart&a=view

This article begins by examining the layout of a hard disk in the x86 architecture and the components on it that are used for booting. It then describes the pieces that are unique to a CD boot. Finally, this article puts the pieces together and creates an image file that can be burned to CD.

A New Boot Architecture

As described in Jan Setje-Eilers's blog (see http://blogs.sun.com/roller/page/setje?entry=a_new_boot_architecture), there are current plans to provide a new boot architecture for Solaris on x86 that will use the open source GRUB (GRand Unified Bootloader, see http://www.gnu.org/software/grub/) as the bootloader in place of the Device Configuration Assistant (DCA). The planned initial delivery is based on GRUB version 0.95, and would be updated as newer versions come out. Unlike Linux and FreeBSD, the Solaris kernel would be fully compliant with the Multiboot Specification (http://www.gnu.org/software/grub/manual/multiboot/). Hence, Solaris could be booted via any bootloader that implements the Multiboot Specification.

Disk Layout (Hard Disk)

The following figure shows the layout of a Solaris on x86 hard disk.

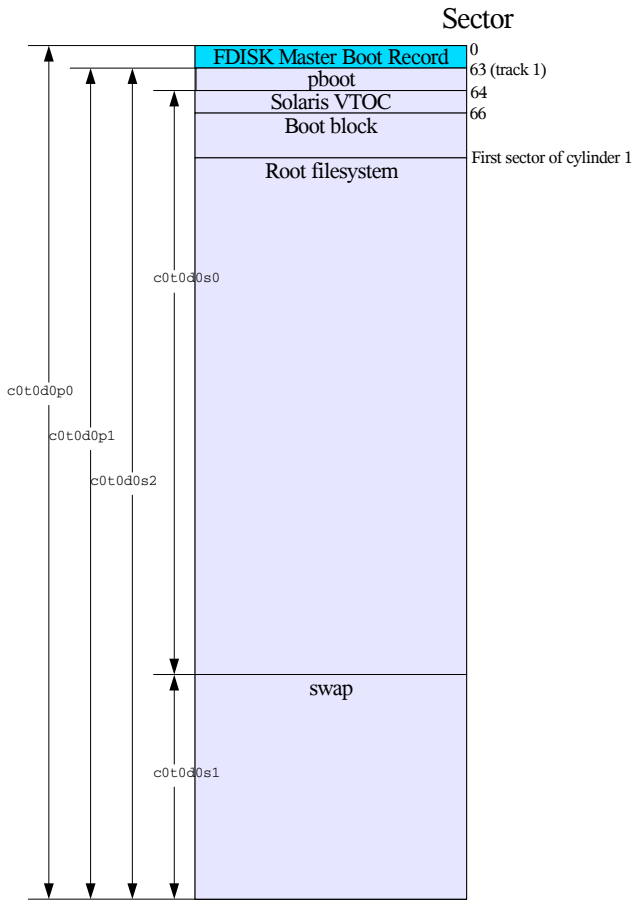

Figure 1. Hard Disk Layout

This figure assumes that the boot drive is `c0t0d0`, that the Solaris OS is the only operating system installed, and that it uses the entire disk. In the example shown in the figure above, Solaris is installed with a single (root) file system and a swap partition.

Partition Structures

The first thing to understand about the layout of a Solaris on x86 boot disk is that there are two different partition structures within it:

- The standard x86 architecture partitioning scheme, commonly referred to as `fdisk`.
- The Solaris Volume Table of Contents (VTOC). The VTOC is actually embedded within an `fdisk` partition. From the perspective of the VTOC, the first `fdisk` partition that has a Solaris system type (`0x82` or `0xbf`) in `fdisk` is seen as the whole disk, which has the effect of creating a sort of virtual disk from an `fdisk` partition. This design allows Solaris on x86 to coexist with other operating systems that use `fdisk`.

The `fdisk` Master Boot Record is a key component. The installation process installs the program known as `mboot` into the first 446 bytes of this sector. The next 64 bytes is the `fdisk` partition table, which is capable of holding information for four partitions. The last 2 bytes contain the signature `0xAA55`. This signature must be set; otherwise, the MBR is considered corrupt by the BIOS and will not be used.

In Figure 1, the one and only partition starts at sector 63 (LBA track boundary). The first sector of the partition contains the Solaris partition boot (`pboot`). The `installboot` program is responsible for placing these two components after the first sector according to the following rules:

- The `/usr/platform/i86pc/lib/fs/ufs/pboot` file is copied first, and is exactly 1 sector long.
- The `pboot` file is followed by 2 sectors that are allocated for the Solaris system VTOC. The `installboot` skips over this.
- The boot block (`bootblk`) follows immediately after the VTOC space. The `/usr/platform/i86pc/lib/fs/ufs/bootblk` file is copied here.

The cylinder following the `bootblk` is typically where the root file system starts.

The following table describes the resulting layout.

Table 1. Boot Components Layout

Component	Start	Length	Contents
`fdisk` Master Boot Record	sector 0 of the disk	1 sector	• **Bytes 0-445**: Boot code (`/usr/lib/fs/ufs/mboot`) • **Bytes 446-509**: partition table • **Bytes 510-511**: Signature 0xaa55
pboot (partition boot)	1st sector of `fdisk` partition	1 sector	`/usr/platform/i86pc/lib/fs/ufs/pboot`
Solaris system VTOC	2nd sector of `fdisk` partition	2 sectors	Disk label, Volume Table of Contents
Boot block	4th sector of `fdisk` partition	Can occupy up to the end of the first cylinder	`/usr/platform/i86pc/lib/fs/ufs/bootblk`
VTOC partition space	Beginning of the first cylinder after the boot block	Up to 2 cylinders before the end of the `fdisk` partition	File systems, swap, raw devices, and so on.

Solaris Disk Device Paths

In Solaris on SPARC, where VTOC is the only partitioning scheme used, the partitions (or slices) are referred to by the "`s`" value in the device path. For example, `/dev/dsk/c0t0d0s0` refers to the first partition on the disk. This does not change for Solaris on x86, but because it has a second partitioning scheme in which the VTOC is embedded, some differences need to be clarified. Normally, slice 2 refers to the entire disk in Solaris on SPARC. In Solaris on x86, however, slice 2 refers to everything on the disk that can be addressed using the Solaris system VTOC structure—the entire `fdisk` partition on which Solaris on x86 resides, minus the last two cylinders.[1]

1.These two cylinders were previously used for alternate cylinders but are no longer used as such in Solaris 10.

Solaris on x86 also has device path names, described in the following table, to handle access to the `fdisk` structure of a disk.

Table 2. `fdisk` *Solaris Device Path Names*

Component	Solaris Device Paths
Entire Physical Disk	`/dev/dsk/c0t0d0p0`, `/dev/rdsk/c0t0d0p0`
`fdisk` partition 1	`/dev/dsk/c0t0d0p1`, `/dev/rdsk/c0t0d0p1`
`fdisk` partition 2	`/dev/dsk/c0t0d0p2`, `/dev/rdsk/c0t0d0p2`
`fdisk` partition 3	`/dev/dsk/c0t0d0p3`, `/dev/rdsk/c0t0d0p3`
`fdisk` partition 4	`/dev/dsk/c0t0d0p4`, `/dev/rdsk/c0t0d0p4`

For both Solaris on x86 and Solaris on SPARC, the device names given to SCSI disks use the `cXtYdZs#` (controller,target,device,slice) convention. However, Solaris on x86 systems typically come with ATA (Advanced Technology Attachment) controllers and disks. ATA does not use the concept of targets, and a controller is capable of addressing only up to four devices. The following table shows the mapping of devices on an ATA controller to Solaris device paths.[2]

Table 3. ATA Device Path Names

Function	Solaris Device Paths
Primary Master	`/dev/rdsk/c0d0p0`, `/dev/dsk/c0d0p0`
Primary Slave	`/dev/rdsk/c0d1p0`, `/dev/dsk/c0d1p0`
Secondary Master	`/dev/rdsk/c1d0p0`, `/dev/dsk/c1d0p0`
Secondary Slave	`/dev/rdsk/c1d1p0`, `/dev/dsk/c1d1p0`

fdisk Master Boot Record

In addition to understanding Solaris OS disk device paths, it is useful to understand the structure of `fdisk`. The first 512 bytes of any disk in the x86 architecture comprise what is commonly called the *Master Boot Record*. This sector contains the initial boot code for booting an operating system. In Solaris on x86, this program is called `mboot`. The function of `mboot` is to locate the active partition, then load and execute a program contained in sector 0 of that partition. In the case of Solaris, this is the program `pboot`.

The MBR also contains the `fdisk` partition table. As will be described later in this article, the `fdisk` partition table needs to manipulated to create a bootable CD/DVD. The partition table contains four 16-byte entries, one for each possible primary partition.[3]

Out of these 16 bytes, the following fields have impact on the boot process: the active flag, system type, relative sector, and the number of sectors.

2.Unlike Solaris on x86, ATA device paths in Solaris on SPARC are given target IDs.
3.Solaris on x86 does not currently support extended partitions.

- The active flag field denotes whether this partition is the boot partition. It can contain the value `0` (not a boot partition) or `0x80` (boot partition). The `mboot` program looks for the first partition on the disk with this flag set to `0x80`, and then attempts to boot from it.
- The system type field contains a 1-byte tag that identifies what the partition is being used for. No standards body has defined values for this tag. As a result, cases exist in which a specific value has more than one meaning, as is true for Solaris on x86. Prior to Solaris 10, Solaris partitions had a type of `0x82`, which also happened to be the type of a Linux swap partition. This conflict caused problems when installing both operating systems on a single disk, because each OS wanted to use the partition tagged with `0x82` for its own purpose. With the release of Solaris 10, Sun has officially changed the partition type code to `0xbf` (decimal 191). For more information about the values used for x86 partition types, refer to the following document:

  http://www.win.tue.nl/~aeb/partitions/partition_types-1.html

- The `fdisk` partition table also contains information about the starting sector and size of the partitions. Later in this article, these values will be calculated, based on the size of input files, to create a bootable CD/DVD.

mboot, pboot, and bootblk

These three components of the disk actually exist as files within the file system structure of Solaris on x86. However, because they cannot exist in any random location, they need to be transferred to specific sectors of the disk (for example, via `dd`) in order for the boot process to function properly. They essentially make up what are called the *stage0* (`mboot` and `pboot`) and *stage1* (`bootblk`) steps of the boot process.

As mentioned previously, the `mboot` (master boot) program is responsible for locating the first `fdisk` partition that contains a bootable/active flag. It then loads the first sector from that partition and executes it. In the case of Solaris on x86, the first sector of this partition contains the `pboot` (partition boot) program. The job of the `pboot` program is to locate the boot block (`bootblk`) within the partition, then load and execute it. As shown in Figure 1, the boot block program is always located directly after the VTOC, which is two sectors long and starts directly after the `pboot`. The `bootblk` will then run the Device Configuration Assistant (DCA), which will in turn load the kernel. As mentioned previously in this article, plans exist for a future version of the Solaris OS to introduce a new boot architecture in which the `bootblk` will load GRUB (GRand Unified Bootloader) instead of the DCA, and will use GRUB to load the kernel and kernel modules instead.

Solaris System VTOC

In Solaris on SPARC, the layout of a hard disk is defined by the VTOC. Disk space is addressed in 512-byte chunks of data called *blocks*. The VTOC is on the first block of the partition. Unlike an `fdisk` defined partition, no boot code exists on the first block, only the VTOC information. The boot code, referred to as the *boot block*, is contained in blocks 1-15.

The structure of the VTOC in Solaris on x86 is slightly different from its SPARC processor counterpart. The VTOC for Solaris on x86 is able to accommodate 16 partitions, not just 8. This might not be immediately evident, as the `format` command does not currently provide the capability to manipulate partitions 8-15,

but `prtvtoc` and `fmthard` do. The other difference in Solaris on x86 is that there are two sectors (instead of just one) allocated for the VTOC.

The VTOC contains information about the size, location, and type of partitions on the disk, along with information about the disk's geometry. When a CD/DVD image is created, this information gets generated based on the file sizes of the miniroot and ISO image.

Disk Layout (CDROM)

The following figure shows the layout of a Solaris on x86 CDROM.

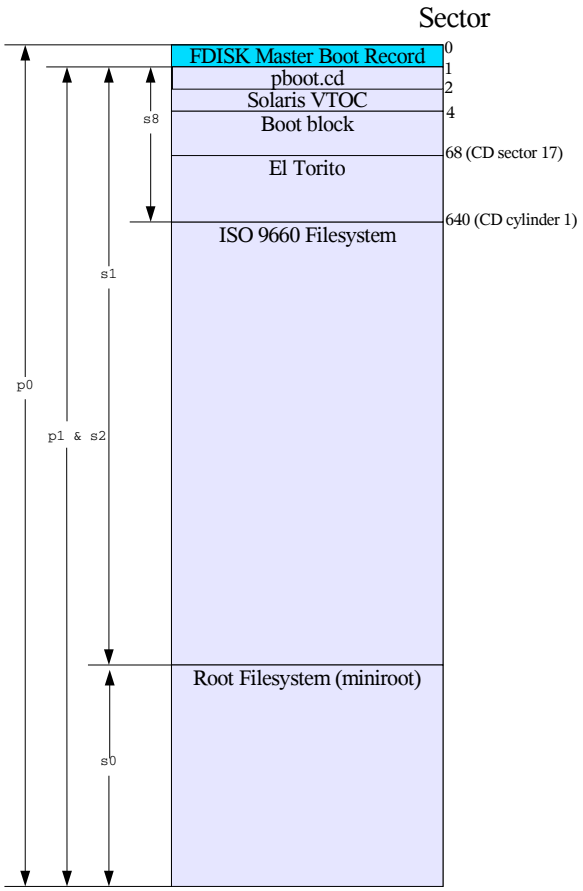

Figure 2. Solaris Disk Layout (CDROM)

The geometry of a CD is different from a hard disk in several ways.

- The sector size on a CDROM is defined as 2048 bytes, not 512. Even so, the tasks in this article work mainly with 512-byte sector sizes, because this is what Solaris uses for the size of a disk block, regardless of the medium.
- There is no concept of a *head*—a CDROM drive has only one head. Because of this, the track size and cylinder size are the same—160 sectors (327680 bytes).

As with a hard disk, the CD uses `mboot`, `pboot`, and `bootblk` to boot. However, the `mboot` and `pboot` programs differ slightly to account for the different sector size. The `mboot` and `pboot` files for a CD (`mboot.cd` and `pboot.cd`) do not exist in the root file system of the CD and must be extracted from the CD itself using the `dd` command.

The placement of `pboot.cd`, the VTOC, and the boot block differs slightly as well. Unlike a disk drive, they are placed contiguously on the CD at the beginning, with each one starting at a 512-byte (disk block) boundary.

El Torito

In addition to `mboot.cd`, `pboot.cd`, and `bootblk`, El Torito is the other boot component to consider with a bootable CDROM. The El Torito specification for CDROM booting was designed by Phoenix Technologies as a way to implement a bootable CDROM without needing to make drastic changes to the BIOS. The El Torito information starts at CD sector 17 (2048*17 = byte 34816), which comes after the `bootblk`, but before cylinder 1 of the CD, which is where an ISO 9660 file system is located.

El Torito specification defines:

- a boot catalog to allow multiple choices to boot from
- a boot record volume descriptor (`brvd`), which is a special volume descriptor on the CD to identify the beginning of the boot catalog
- a validation entry as a means to check whether the boot catalog is a valid one
- an initial/default entry that describes what needs to be loaded

Other components of El Torito are not directly pertinent to the tasks described in this article.
The information needed to set up in El Torito for Solaris focuses on loading and executing the first 1024 bytes of the CD (`mboot.cd` and `pboot.cd`). For more information about El Torito, refer to:

http://www.phoenix.com/NR/rdonlyres/98D3219C-9CC9-4DF5-B496-A286D893E36A/0/specscdrom.pdf

fdisk on CDROM

The CD created using the instructions in “Steps for Creating a Customized Solaris on x86 Boot CD/DVD” on page 9 (and on the standard Solaris on x86 boot CD) will have only one `fdisk` partition. This partition, with the exception of the MBR, will span the entire disk and contain the VTOC structure.

Solaris System VTOC on CDROM

The Solaris system VTOC on a standard Solaris on x86 boot CD defines four slices, as described in the following table.

Table 4. VTOC Slices Defined on a Solaris on x86 Bootable CD

Partition	VTOC tag	Function
slice 0	root	Root file system.
slice 1	unassigned	Placeholder for ISO9660 file system. This will encompass all of the sectors from the beginning of the VTOC addressable space to the sector prior to the start of slice 0.
slice 2	backup	The entire CDROM, minus the MBR on the first sector. s2 and p1 refer to the same area of the CDROM.
slice 8	boot	Pointer to the boot information needed by Solaris.

Steps for Creating a Customized Solaris on x86 Boot CD/DVD

As shown in Figure 2, the CDROM image is really an ISO9660 file system with a Solaris UFS partition placed immediately after it on the disk. Having the UFS appended in this way does not affect the ISO9660 file system itself. The ISO9660 file system ignores the presence of the UFS because its geometry is specified within the ISO9660 volume descriptor table.

Having this structure makes it easy to create a copy of the CD. The `dd` command can be used to copy the entire `p0` device. The resulting image can be burned to CD with no modifications. This differs from Solaris on SPARC, which uses the VTOC to define the geometry of the CD. With Solaris on SPARC, there are six slices that all need to be copied separately, and then the slices need to be concatenated together as one file to create a burnable image. If any modifications were made to any of the slices, padding must be added to ensure that each slice starts on a CDROM cylinder boundary (327680 bytes).

To create a customized Solaris on x86 CD, complete the following steps:

- Step 1: Create an ISO9660 File System
- Step 2: Create a Solaris Miniroot
- Step 3: Create the Boot Files
- Step 4: Create the Composite ISO CD Image

The rest of this section describes how to perform these steps.

Step 1: Create an ISO9660 File System

On the standard Solaris on x86 1 of 2 boot CD, the ISO9660 file system contains (among other things) installation packages, the network boot utilities (such as `add_install_client`), and Solaris JumpStart software examples. Either keep whatever you want in this directory, or create an entirely new one with your own content. The advantage of storing content here is that any operating system can read this file system from the CD. This will not be true of the miniroot that will be created later.

In the simplest case, create a directory (such as `/tmp/isodir`):

```
# mkdir /tmp/isodir
```

You can populate this directory with anything you want. Consider storing files that need to be accessible from any operating system. Remember that only Solaris systems will be able to access files in the miniroot file system.

Step 2: Create a Solaris Miniroot

To boot Solaris on x86, begin by creating a root UFS file system, which you can get from a Solaris on x86 boot CD or DVD. The CD and DVD differ in this regard—the Solaris CD contains a single miniroot. This miniroot is capable of being a root file system for a network boot client. Patches and packages can be added to it. When the CD is inserted into a Solaris on x86 system running `vold`, this miniroot will be accessible as s0.

The DVD contains two miniroots:

- The first miniroot also comes up in `vold` as s0, but is a very stripped down version of a root file system and serves the sole function of being able to perform a Solaris installation (patches and packages *cannot* be added to it).
- The second miniroot resides in the `Solaris_10/Tools/Boot` directory of the ISO file system (what `vold` presents as s2). This is the equivalent of the s0 miniroot from the CDROM (patches and packages can be added to it). For a customized boot CD, use either the second miniroot from the DVD or the miniroot from the CD. *Never* use the s0 miniroot from the DVD!

To create a miniroot for use on a customized boot CD, start by creating an image file using the Solaris `lofiadm` command. Begin by determining the desired size of the root file system, either by mounting the root file system off of the Solaris on x86 CD (CD only), or by using `du -sk` in the DVD miniroot directory.

To see the size of the CD miniroot, insert the Solaris system CD in the drive of a Solaris on x86 system.[4] The following examples assume that `vold` is running and always mounts a CD as `/cdrom/cdrom0`. You should see two partitions from the CD—slice 0 and slice 2. Slice 0 is the slice that contains the root file system.

```
# df -kl
Filesystem            kbytes    used   avail capacity  Mounted on
/dev/dsk/c0d0s0      10867092 8382503 2375919    78%    /
/devices                   0       0       0     0%    /devices

<snip>

/vol/dev/dsk/c1t1d0/sol_10_305_x86/s2
                       219744  219744       0   100%    /cdrom/sol_10_305_x86/s2
/vol/dev/dsk/c1t1d0/sol_10_305_x86/s0
                       363852  353448       0   100%    /cdrom/sol_10_305_x86/s0
```

1. If you are using the DVD, insert it into the DVD drive. Either let `vold` mount the partitions, or mount them manually. In either case, `cd` to the `s2/Solaris_10/Tools/Boot` directory and run `du -sk`:

4.You will not be able to read the root file system off of the CD on a Solaris on SPARC system because the UFS file system on the CD is stored in little endian format and SPARC systems use big endian. This endian difference is the same reason that the last five slices of a Solaris on SPARC CD on a Solaris on x86 system are not visible.

```
# cd /cdrom/cdrom0/s2/Solaris_10/Tools/Boot
# du -sk .
347205  .
```

2. Once you know the size of the miniroot off the CD or DVD, add about 5% to this for file system overhead. If you plan to add patches or packages to the miniroot, considering adding more than 5%. Using this size estimate, create a UFS file system image via `mkfile`, `lofiadm`, and `newfs`, then mount it. In this section, the DVD miniroot is used as an example.
3. Create an empty file of size 364565K (347205+5%).

   ```
   # mkfile 364565k /var/tmp/my.miniroot
   ```

4. Map the file to a block device via the `lofi` driver.

   ```
   # lofiadm -a /var/tmp/my.miniroot
   /dev/lofi/1
   ```

5. Create a file system on the character device (notice the use of `rlofi` instead of `lofi`).

   ```
   # newfs -i 8192 -m 1 /dev/rlofi/1
   newfs: construct a new file system /dev/rlofi/1: (y/n)? y
   /dev/rlofi/1:   729000 sectors in 1215 cylinders of 1 tracks, 600 sectors
           356.0MB in 76 cyl groups (16 c/g, 4.69MB/g, 576 i/g)
   super-block backups (for fsck -F ufs -o b=#) at:
    32, 9632, 19232, 28832, 38432, 48032, 57632, 67232, 76832, 86432,
    633632, 643232, 652832, 662432, 672032, 681632, 691232, 700832, 710432,
    720032,
   ```

6. Mount the block device.

   ```
   # mount /dev/lofi/1 /mnt
   # df -k /mnt
   Filesystem            kbytes    used   avail capacity  Mounted on
   /dev/lofi/1           357794    1041  353176     1%    /mnt
   ```

7. Once the file system is mounted, copy a root image into it. The following example commands assume that `vold` is running and the CD/DVD is mapped as `/cdrom/cdrom0`.

   ```
   # cd /cdrom/cdrom0/s0 #(for CD only)
   ```

   OR

   ```
   # cd /cdrom/cdrom0/s2/Solaris_10/Tools/Boot #(CD or DVD)
   # find . -depth -print | cpio -pdm /mnt
   820112 blocks
   ```

   At this point, you can make changes to the file system by modifying files under the `/mnt` mount point. Some of the more useful changes include installing packages on top of this root file system via `pkgadd -R /mnt`, or updating Solaris patches on it via `patchadd -C /mnt`. Ensure that the updates you want to make will fit in the file system that you have created and mounted. If more space is needed, add it to the `mkfile` size described previously. Because the CD consists of a standard ISO9660 file system plus this UFS file system, make sure that the size of the miniroot plus the size of the ISO image does not exceed what can be put on a CD (~650MB) or DVD (~4.7GB).

8. After completing modifications to the file system, `umount` the file system and delete the lofi device. This file will later be used to append to the ISO file.

```
# umount /mnt
# lofiadm -d /dev/lofi/1
```

The result of these steps is a file representing a valid Solaris on x86 root file system with the changes you have made.

Extracting Boot Files

Before creating the combined ISO and UFS image of the CD, you still need valid copies of the `mboot`, `pboot`, and `bootblk` from the Solaris on x86 CD. To generate these files:

1. Turn off the Solaris Volume Manager:

```
# /etc/init.d/volmgt stop
```

2. Insert the Solaris for x86 CD into the CDROM drive of a system running Solaris on x86.
3. Combine `mboot` and `pboot` into a single file called `mboot+pboot.cd`.

```
# dd if=/dev/dsk/c1t1d0p0 of=mboot+pboot.cd bs=512 count=2
```

4. Save the Solaris on x86 `bootblk` into a file named `bootblk.cd`.

```
dd if=/dev/dsk/c1t1d0p0 of=bootblk.cd bs=512 iseek=4 count=60
```

   The `bootblk` can also be obtained by copying the following file:

```
cdrom/cdrom0/s0/usr/platform/i86pc/lib/fs/ufs/bootblk
```

   from the CD's miniroot. This approach assumes that `vold` is running and is using `/cdrom/cdrom0` as the base for mounting the CD.

   Alternatively, with the Solaris system DVD, you can manually mount the DVD and retrieve the following file:

```
Solaris_10/Tools/Boot/usr/platform/i86pc/lib/fs/ufs/bootblk
```

   The advantage of this approach is that it works in operating systems that are unable to read a UFS off of a DVD. However, this will work only from the DVD on non-Solaris operating systems, because the `Solaris_10/Tools/Boot` directory on a CD is just a symbolic link to the `s0` device (which will exist only when running `vold` in Solaris).

Step 3: Create the Boot Files

The next step is to create the boot files that `mkisofs` will use to build the Solaris on x86 image file. Note that copies of the boot files are used so that they are not corrupted by potential mistakes.

1. Copy the `mboot+pboot.cd` file to a file named `.bootimage` in the root directory for the ISO file system:

```
# cp mboot+pboot.cd /tmp/isodir/.bootimage
```

2. Create the generic boot file for `mkisofs` by copying `mboot+pboot.cd` to a file named `genboot`, place a 1K gap in the file (the `dd` command below adds a gap consisting of 1024 zeros), and then append the Solaris `bootblk`:

```
# cp mboot+pboot.cd genboot
# dd if=/dev/zero bs=1024 count=1 >> genboot
```

```
# cat bootblk.cd >> genboot
```

Step 4: Create the Composite ISO CD Image

Using `mkisofs` with the `-sunx86-boot` option, create the composite ISO image file.

```
# mkisofs -r -no-emul-boot \
   -b .bootimage \
   -c .catalog \
   -G genboot \
   -o composite.iso \
   -sunx86-boot /var/tmp/my.miniroot \
   /tmp/isodir/
```

The `mkisofs` command will build a composite ISO image that contains the `fdisk` partition table and Solaris system VTOC based on the sizes of the ISO directory and the miniroot. It will also create the El Torito boot records required to boot the CD. The result is a combined image file named `composite.iso`, which can be burned as an ISO image to the CD. This CD will be a bootable Solaris on x86 CD and will mount the miniroot partition as its root file system.

Burning the Image

This ISO image can be burned using any software that is capable of taking a straight ISO image and burning it to CD. The following command can be used in Solaris to burn the image, regardless of whether it is a CD or DVD:

```
# cdrw -i -d <device name> composite.iso
Initializing device...done.
Writing track 1...done.
Finalizing (Can take several minutes)...done.
```

where <*device name*> is the device displayed from the output of `cdrw -l`.

Conclusion

There are multiple ways to achieve the same result. Parts of the process in this article can be adapted to suit an individual user's preferences. The real trick is to create the composite image. Before the `sunx86-boot` option was included in `mkisofs`, users needed to write a program to create the composite image from an ISO image and a miniroot.

You might want to use other `mkisofs` options, such as providing a volume name, Joliet support, and so on. Refer to the `mkisofs` manual page for more information.

References and Related Sources

- El Torito Bootable CD-ROM Format Specification Version 1.0. Phoenix Technologies Inc., January 25, 1995

  http://www.phoenix.com/NR/rdonlyres/98D3219C-9CC9-4DF5-B496-A286D893E36A/0/specscdrom.pdf

- *Performing Network Installations Without a Local Boot Server*, Howard, John S. (Sun BluePrints Online—May 2004).

  http://www.sun.com/blueprints/0504/817-7288.pdf

- *Building a Bootable DVD to Deploy a Solaris Flash Archive*, Howard, John S. (Sun BluePrints Online—April 2004).

  http://www.sun.com/blueprints/0404/817-6991.pdf

- *Customizing JumpStart Framework for Installation and Recovery*, Howard, John S., and Noordergraaf, Alex (Sun BluePrints Online—August 2002).

  http://www.sun.com/blueprints/0802/816-7587-10.pdf

- `mkisofs`(1) Manual page
- `lofiadm(1M)` Manual page

  http://docs.sun.com/app/docs/doc/816-0211/6m6nc66ue?a=view

- `fdisk` partition types

  http://www.win.tue.nl/~aeb/partitions/partition_types-1.html

About the Authors

John Cecere

John started his career in computers with the United States Air Force, where he worked on Sperry Univac mainframes. He took this experience to the civilian sector, where he perform similar job functions at Bell Communications Research. There he first encountered Unix, and he went on to be a system administrator for five years. He joined Sun in 1997 as a System Support Engineer and subsequently a Regional SSE. He is currently involved in several projects writing internal tools for Sun Services.

Dana Fagerstrom

Dana Fagerstrom has been on the Service end of Sun's products for the past eight years, where he has been supporting the US Telcos. Dana has a diverse background that spans system administration, software development, and program management. He started at Bell Labs, moved on to Bellcore (now Telecordia), and then to Sun. He has a MS in Computer Science from The Stevens Institute of Technology.

Ordering Sun Documents

The SunDocs℠ program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject at `http://docs.sun.com/`.

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at:
`http://www.sun.com/blueprints/online.html`