Customizing JumpStart Framework for Installation and Recovery

John S. Howard, Enterprise Engineering

Alex Noordergraaf, Enterprise Server Products

Sun BluePrints™ OnLine—August 2002

http://www.sun.com/blueprints

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 U.S.A.
650 960-1300

Part No. 816-7587-10
Revision 1.0, 10/3/01
Edition: August 2002

Please
Recycle

Customizing JumpStart Framework for Installation and Recovery

Editor's Note – This article is the complete ninth chapter of the Sun BluePrints™ book, "JumpStart Technology: Effective Use in the Solaris™ Operating Environment", by John S. Howard and Alex Noordergraaf (ISBN 0-13-062154-4), which is available through www.sun.com/books, amazon.com, and Barnes and Noble bookstores.

The JumpStart system is useful for much more than installing the Solaris OE. This chapter examines the more powerful, yet often overlooked, aspects of the JumpStart system. In several ways, the JumpStart system is like a scripting language, the JumpStart framework provides a toolkit of operators that can be used individually or combined. These operators function well individually, but their true power is realized when they are combined.

Note – This chapter provides techniques that can produce configurations that would not be supported by Sun Enterprise Services. However, that lack of support should not detract from the value of the techniques presented.

This chapter examines the boot and installation processes, demonstrating how to adapt these processes for custom system installation and system recovery. This chapter discusses the following topics:

- Building and testing a bootable installation CD-ROM
- Recovering a failed system with JumpStart
- Altering the boot process
- Adding utilities and manual pages
- Meeting challenges unique to the miniroot

Building a Bootable Installation from CD-ROM

There may be some situations when it is not possible to use a JumpStart server, yet it is necessary to perform an automated (hands-free) installation of the Solaris OE. This section details a procedure to create a bootable installation CD-ROM, which is essentially putting a JumpStart server onto a CD. This CD can then be used to effect a standardized, automated Solaris OE installation from the CD. This technique is especially useful in environments where disk space limitations or networking constraints do not allow for a JumpStart server.

This section examines the structure of a bootable Solaris 8 OE (for a SPARC machine) CD and discusses the appropriate modifications to the default installation scripts that allow a JumpStart installation to be done from CD. Further, this section describes how to create a bootable Solaris 8 OE installation CD for the SPARC platform. Additionally, a Solaris 8 OE system with the CD Read/Write (CDRW) utilities installed is used to write the Solaris 8 OE bootable installation CD. Although several different approaches and software applications are available for writing CDs, this section uses commands available only in the standard Solaris 8 OE to write the bootable installation CD.

The structure of the bootable installation CD can vary with different versions of the Solaris OE, partly because of changes required for the support of additional hardware architectures. Additionally, changes to the Solaris OE from version to version may necessitate changes in the CD or the number of CDs required to install the Solaris OE.

Versions of the Solaris OE can vary structurally, but the concepts and procedures presented here can be adapted or extended to create a bootable installation CD for any of the current versions of the Solaris OE.

Bootable CD Structure

A bootable Solaris OE CD has several components in common with any other hard disk. The boot CD is divided into several partitions (or slices), and a Volume Table Of Contents (VTOC) provides the location and sizes of these slices. In addition to the VTOC, a typical installation CD has six slices. Although the Solaris OE imposes the partitioning of the CD into six slices, it is important to note that the CD is written as one *session*—this fact is important when the CD is written.

Following is an examination of the VTOC and the six slices of the Solaris 8 OE installation CD.

Volume Table of Contents

The VTOC is located at cylinder 0, sector 0 on the CD. You can examine the VTOC of any disk device with the `prtvtoc` command. The VTOC of the Solaris 8 OE Software CD (the installation CD) is as follows:

```
server01# /etc/init.d/volmgt stop
server01# prtvtoc /dev/dsk/c0t6d0s0
* /dev/dsk/c0t6d0s0 partition map
*
* Dimensions:
*     512 bytes/sector
*     640 sectors/track
*       1 tracks/cylinder
*     640 sectors/cylinder
*    2048 cylinders
*    2048 accessible cylinders
*
* Flags:
*   1: unmountable
*  10: read-only
*
* Unallocated space:
*       First     Sector    Last
*       Sector     Count    Sector
*     1301760      2560   1304319
*
*                          First     Sector    Last
* Partition  Tag  Flags   Sector    Count   Sector  Mount Directory
       0      4    10          0   1128960   1128959
       1      2    10    1128960    172800   1301759
       2      0    00    1301760      2560   1304319
       3      0    00    1304320      2560   1306879
       4      0    00    1306880      2560   1309439
       5      0    00    1309440      2560   1311999
server01# /etc/init.d/volmgt start
```

Note – You must stop the CD and diskette volume management in order to execute the `prtvtoc` command on a CD. Restart volume management after executing `prtvtoc`. All filesystems mounted from the CD will be unmounted and become inaccessible while volume management is stopped.

In contrast to a hard disk, the disk geometry that the Solaris OE uses for a CD provides no distinction between a cylinder and a track. As the `prtvtoc` output illustrates, the disk label used for a CD defines a cylinder as being composed of one track. Further, the `prtvtoc` output verifies that each track is defined as having 640 sectors and that one sector is equal to 512 bytes.

Note that the Solaris OE requires that all UFS filesystems align on a cylinder boundary. For a CD, this means that all UFS filesystems on the CD must begin on a sector that is a multiple of 640.

Slices

By reading the VTOC, the Solaris OE sees the CD as having six slices. The contents of those six slices are as follows:

- Slice 0 contains the Solaris OE packages to be installed and is the High Sierra File System (HSFS) partition of the CD.
- Slice 1 contains the generic kernel and the directory that becomes the system's `/` (root) directory after boot.
- Slice 2 contains the boot block for the sun4c architecture.
- Slice 3 contains the boot block for the sun4m architecture.
- Slice 4 contains the boot block for the sun4d architecture.
- Slice 5 contains the boot block for the sun4u architecture.

Slices 2 through 5 are there only to provide hardware-architecture-specific boot blocks. As new hardware architectures are added and old architectures reach their end-of-life, the uses of these slices may change. The file `.slicemapfile` in the top-level directory of slice 0 contains the mapping of a slice to the architecture supported.

As noted earlier, slice 0 is on the HSFS partition and all other slices are on the UFS partitions. Slice 0 is also the largest of the slices and can incorporate any unused space on the CD. The procedures detailed in this section augment the installation procedures in slice 0. However, there is a fixed upper limit in available space for slice 0 that limits our modifications. The total space available on a standard CD is 640 Mbytes. The distribution media for Solaris 8 OE supports four architectures. If the bootable installation CD being created needs to support only one architecture, the space (slices) used by the unneeded architectures can be incorporated into slice 0, enlarging slice 0 but losing the ability to boot other architectures from that CD.

It is also interesting to note that, other than the boot block, the only content of slices 2 through 5 is the file `.SUNW-boot-redirect` in the top-level directory of each of those partitions. This file contains the character 1, which redirects the OpenBoot PROM (OBP) boot loader to load the kernel from partition 1. This mechanism was added with Solaris 2.5 OE as a means of taking advantage of the hardware-independent nature of the kernel to optimize the utilization of space on the CD.

Procedure Overview

Generally, this procedure extracts the contents of slice 0, then splices the desired installation behaviors into the contents of slice 0. The modifications made to slice 0 are to configure the bootable installation CD to partition `c0t0d0` as the boot device. The modifications then enable a fully automated installation of the Solaris 8 OE. The profile specifies that a full Solaris OE is installed (the `SUNWCall` package cluster) with the exception of the Power Management facility.

At a high level, the procedure to create a bootable CD is as follows:

1. Create and populate a work area.
2. Modify the installation behaviors of slice 0.
3. Assemble the individual slices into one CD session and write them to the bootable installation CD.
4. Test the bootable installation CD.

You can also use this procedure to create a bootable CD without the JumpStart software installation behaviors by omitting step 2.

Procedure Specifics

For this example, `server01` is an Ultra Enterprise 420R server running the Solaris 8 OE with the Solaris 8 OE CD creation utilities installed and configured as a JumpStart server. `server01` has a CD-ROM writer connected at `c3t2d0` (identified as `cdrom1` by the `cdrw -l` command).

Creating and Populating a Work Area

Verify the presence of the Solaris OE CD creation utilities. The Solaris 8 OE installation media is already mounted, and `/bicd8` is used as the work area. `/bicd8` is a 2-Gbyte UFS filesystem.

1. **Create `/bicd8` in the following manner:**

```
server01# pkginfo SUNWmkcd SUNWcdrw
system      SUNWcdrw       CD read and write utility for Solaris
system      SUNWmkcd       CD creation utilities
server01# newfs -m 1 /dev/rdsk/c0t1d0s0
newfs: construct a new filesystem /dev/rdsk/c0t1d0s0: (y/n)? y
/dev/rdsk/c0t1d0s0: 4194828 sectors in 1452 cylinders of 27
tracks, 107 sectors
        2048.3MB in 46 cyl groups (32 c/g, 45.14MB/g, 7488 i/g)
super-block backups (for fsck -F ufs -o b=#) at:
 32, 92592, 185152, 277712, 370272, 462832, 555392, 647952, 740512,
833072, 925632, 1018192, 1110752, 1203312, 1295872, 1388432,
1480992, 1573552, 1666112, 1758672, 1851232, 1943792, 2036352,
2128912, 2221472, 2314032, 2406592, 2499152, 2591712, 2684272,
2776832, 2869392, 2958368, 3050928, 3143488, 3236048, 3328608,
3421168, 3513728, 3606288, 3698848, 3791408, 3883968, 3976528,
4069088, 4161648,
server01# mkdir /bicd8
server01# mount /dev/dsk/c0t1d0s0 /bicd8
```

2. **Populate the work area by extracting the partitions from the Solaris 8 OE software CD.**
   a. Since the contents of slice 0 will be manipulated, use `cpio` to copy out partition 0.
   b. Since no changes are made to the contents of slices 1 through 5, use `dd` to take those slices *off* the CD.
   c. Before extracting slices 1 through 5, stop CD and diskette volume management.

      Note – All filesystems mounted from the CD will be unmounted while volume management is stopped.

```
server01# cd /cdrom/sol_8_401_sparc/s0
server01# mkdir /bicd8/s0
server01# find . -print |cpio -pudm /bicd8/s0
server01# cd /bicd8
server01# /etc/init.d/volmgt stop
server01# for i in 1 2 3 4 5
> do
> dd if=/dev/dsk/c0t6d0s${i} of=sol8.s${i} bs=512
> done
172800+0 records in
172800+0 records out
2560+0 records in
2560+0 records out
2560+0 records in
2560+0 records out
2560+0 records in
2560+0 records out
2560+0 records in
2560+0 records out
```

Additionally, since the slice layout of the bootable installation CD being created will not vary from the slice layout of the Solaris 8 OE Software CD, the VTOC from the Software CD can be used later for the bootable installation CD.

3. **Use `dd` to take the VTOC from the CD, and at this point, restart volume management.**

```
server01# dd if=/dev/dsk/c0t6d0s0 of=/bicd8/sol8.cdrom.vtoc \
> bs=512 count=1
1+0 records in
1+0 records out
server01# /etc/init.d/volmgt start
```

Several choices are available if the slice layout of the CD being created needs to vary from that of the Software CD (for example, if the VTOC needs to be changed). Use CD creation software—such as the toolkit for building bootable CDs, available from Sun Professional Services, Gear Pro for UNIX, or Young Minds—to generate a correct and valid VTOC. Or create a new VTOC and disk label programmatically by creating and writing the `dkl_vtoc` and `dk_label` structures, respectively. See the Solaris system file `/usr/include/sys/dklabel.h` for more information on these structures.

Modifying Installation Behaviors of Slice 0

Modify the default installation behaviors in slice 0 by deleting the contents of the `.install_config` directory and adding the desired JumpStart rules and profile to this directory. Note that the parsed `rules.ok` file (the output from the `check` script), not the `rules` file, must be placed in the `.install_config` directory. If any begin or finish scripts are being used, place them in the `.install_config` directory as well.

1. **Modify slice 0 as follows.**

```
server01# cd /jumpstart
server01# rm /bicd8/s0/.install_config/*
server01# cat /jumpstart/Profiles/S8-Server.profile
install_type    initial_install
system_type     standalone
partitioning    explicit
root_device     c0t0d0s0
#
# 1.5GB / and 512MB swap on a 2GB disk
#
filesys         rootdisk.s0    691:2040       /
filesys         rootdisk.s1    1:690          swap
cluster         SUNWCall
package         SUNWpmowm       delete
package         SUNWpmowr       delete
package         SUNWpmowu       delete
package         SUNWpmr         delete
package         SUNWpmu         delete
package         SUNWpmux        delete
server01# cp /jumpstart/Profiles/S8-server.profile \
> /bicd8/s0/.install_config
server01# cat rules
any - - S8-server.profile -
server01# ./check
Validating rules...
Validating profile S8-server.profile...
The custom JumpStart configuration is ok.
server01# cp rules.ok /bicd8/s0/.install_config
```

The setup of the installation profile directory is controlled by the `profind` script. You must modify this script to redirect the configuration directory environment variable (`${SI_CONFIG_DIR}`) used by the JumpStart software to the `.install_config` directory on the bootable installation CD.

2. **Edit the** `bicd8/s0/Solaris_8/Tools/Boot/usr/sbin/install.d/profind` **shell script and replace the** `cdrom()` **function with the following function:**

```
cdrom()
{
    #
    # stub images, indicated by the file /tmp/.preinstall
    #
    if [ -f /tmp/.preinstall ]; then
        mount -o ro -F lofs ${CD_CONFIG_DIR} ${SI_CONFIG_DIR} >/
dev/null 2>&1

        if [ $? -eq 0 ]; then
            verify_config "defaults" "CDROM"
        fi
    fi
    gettext " <<< using CD default >>>"; echo     # added bicd8
    rmdir ${SI_CONFIG_DIR}                        # added bicd8
    ln -s /cdrom/.install_config ${SI_CONFIG_DIR} # added bicd8
    exit 0                                        # added bicd8
}
```

This modification instructs the installation process to use the `.install_config` directory that was populated with the desired JumpStart software profiles and `rules` file.

Assembling and Writing Slices to Bootable Installation CD

At this point, the VTOC, the modified slice 0, and the unmodified slices 1 through 5 are written to the bootable installation CD being created. The individual slices are combined into one image to be written to a blank CD.

It is important to keep in mind that slice 0 of the Solaris 8 OE CD is at almost 100 percent utilization of the total available space of slice 0. Further, the Solaris 8 OE product is on two CDs because all of the software package will not fit on one CD. If the modified slice 0 exceeds the size of the original slice 0, you must either create a new VTOC or remove unneeded files from slice 0. Also keep in mind that the iso9660 filesystem has some overhead, which increases the image (created by `mkisofs`) as well.

1. **Create an automated install CD (without having to swap CDs during the installation), by removing from slice 0 those software packages that will not be installed or are not needed by the installation client.**

Additionally, removing unneeded files from slice 0 is much simpler than handcrafting a VTOC. A good place to start removing unneeded files is the `Product` subdirectory. Rarely does a Solaris OE installation require all the packages from the `Product` directory. For example, most servers do not (and should not) have the power management packages installed. Removing the power management packages before executing the `mkisofs` command helps minimize the size of the created iso9660 HSFS image.

Remember that the profile you are using should reflect these changes to the `Product` directory; that is, don't try to install the removed packages. The removed packages should also be removed from the software package cluster definition file, `/bicd8/s0/Solaris_8/Product/.clustertoc`.

2. **Before combining and writing the CD, execute the `mkisofs` command to convert the modified slice 0 in the `/bicd8/s0` work area into an HSFS filesystem.**

3. **Since no changes to the miniroot or supported architectures are required, extract slices 1 through 5 from the Solaris 8 OE software CD and write them, unchanged, to the bootable installation CD being created.**

4. It is important to note that `mkisofs` creates a VTOC at offset 0 within this image.

   Use `dd` to remove this invalid VTOC from the HSFS image by skipping the first 512-byte block. For this example, the unneeded power management packages are removed from the `Product` directory before the iso9660 filesystem is created from `/bicd/s0`.

```
server01# cd /bicd8/s0/Solaris_8/Product
server01# rm -rf SUNWpmowr/* SUNWpmowu/* SUNWpmr/* SUNWpmux/*
server01# cd /bicd8
server01# mkisofs -R -d -L -l -o /bicd8/sol8.S0 /bicd8/s0
     .
     .
     .
Total extents actually written = 282170
Total translation table size: 0
Total rockridge attributes bytes: 4246465
Total directory bytes: 24463360
Path table size(bytes): 175770
Max brk space used 167a000
282170 extents written (551 Mb)
server01# dd if=/bicd8/sol8.S0 of=/bicd8/new.sol8.s0 bs=512 skip=1
1128679+0 records in
1128679+0 records out
server01# rm /bicd8/sol8.S0
```

5. The VTOC specifies a size for slice 0, so slice 0 must be padded to maintain the validity of the VTOC and maintain the correct cylinder boundaries. The size of the pad is computed by adding 1 to the number of sectors in the HSFS slice 0 image (this accounts for the VTOC) then subtracting that sum from the number of sectors (reported by `prtvtoc`) in the unmodified slice 0 on the CD.

   Create the pad by using `dd` to read the appropriate number of zeros from `/dev/zero`.

```
server01# bc
1128960-(1128679+1)
280
server01# dd if=/dev/zero of=pad.s0 bs=512 count=280
280+0 records in
280+0 records out
```

6. As with any automated installation, `sysidtool` needs all installation client identification information such as host name, IP address, time zone, etc. The location of this information depends on whether the installation client is connected to a network or off-network during the installation. If the installation client is connected to a network during installation, this information must be available from a name service such as NIS+ or NIS, or provided from the `/etc/bootparams`, `/etc/ethers`, and `sysidcfg` files from a host on the network. The minimum entries required in the `/etc/bootparams` file are as follows:

```
server01# cat /etc/bootparams
client06 sysid_config=server01:/jumpstart/Sysidcfg/Solaris_8
```

   The `sysidcfg` file specified by `/etc/bootparams` contains the following:

```
server01# cat /jumpstart/Sysidcfg/Solaris_8/sysidcfg
system_locale=en_US
timezone=US/Pacific
network_interface=primary   {netmask=255.255.255.0
                        protocol_ipv6=no}
terminal=vt100
security_policy=NONE
root_password=Q7jsh1m6IztTU
name_service=NONE
timeserver=localhost
```

   To perform an automated installation without network connectivity, you must have placed a `sysidcfg` file in the `/etc` directory of the filesystem image taken from slice 1 of the Solaris 8 OE CD.

Mount the filesystem image file by using the Solaris 8 OE loopback file driver administration commands.

7. **After mounting the filesystem image, use standard Solaris OE commands to remove the symbolic link for the default `sysidcfg` file and to copy a complete `sysidcfg` file to the filesystem image.**

```
server01# cat /bicd8/sysidcfg
system_locale=en_US
timezone=US/Pacific
network_interface=primary {hostname=client06
                           ip_address=10.1.1.9
                           netmask=255.255.255.0
                           protocol_ipv6=no}
terminal=vt100
security_policy=NONE
root_password=Q7jsh1m6IztTU
name_service=NONE
timeserver=localhost
server01# lofiadm -a /bicd8/sol8.s1
/dev/lofi/1
server01# mount /dev/lofi/1 /mnt
server01# ls -al /mnt/etc/sysidcfg
lrwxrwxrwx   1 root     other         24 Nov 28 16:38 /mnt/etc/
sysidcfg -> ../tmp/root/etc/sysidcfg
server01# rm /mnt/etc/sysidcfg
server01# cp /bicd8/sysidcfg /mnt/etc/sysidcfg
server01# umount /mnt
server01# lofiadm -d /dev/lofi/1
```

Note – For the off-network automated installation, the host name, IP address, netmask, and IPv6 specification *must* be in the `sysidcfg` file.

See Chapter 11, ”System Cloning,” for a fully automated technique for a JumpStart software installation with no network connectivity, using the WebStart Flash extensions.

8. **Concatenate the VTOC, HSFS image, padding, and unmodified images of slices 1 through 5 into one image and write it to the CD writer on device `c3t2d0` with the `cdrw` command:**

```
server01# cat sol8.cdrom.vtoc new.sol8.s0 pad.s0 \
sol8.s1 sol8.s2 sol8.s3 sol8.s4 sol8.s5 >bicd8.image
server01# cdrw -d cdrom1 -i bicd8.image
Initializing device...done.
Writing track 1...done.
done.
Finalizing (Can take up to 4 minutes)...done.
```

Testing the Bootable Installation CD

To validate the newly created bootable installation CD, place it in the CD drive of the installation client, `client06`. For this example, the client is off-network while the installation occurs and the `sysidcfg` file in the `/etc` directory of slice 1 of the CD was modified, as shown in step 3 of "Procedure Specifics" on page 5. After the OBP `boot cdrom` command is issued, `client06` boots from the CD and performs an automated installation of the Solaris 8 OE.

Issue the `boot cdrom` command with the `- install` options to initiate the automated installation:

```
screen not found.
{0} ok boot cdrom - install
Boot device: /pci@1f,4000/scsi@3/disk@6,0:f  File and args: -
install

SunOS Release 5.8 Version Generic_108528-05 64-bit
Copyright 1983-2000 Sun Microsystems, Inc.  All rights reserved.
Configuring /dev and /devices
Using RPC Bootparams for network configuration information.
SUNW,hme0 : No response from Ethernet network : Link down -- cable
problem?
Skipping interface hme0
SUNW,hme0 : No response from Ethernet network : Link down -- cable
problem?

The system is coming up.  Please wait.
SUNW,hme0 : No response from Ethernet network : Link down -- cable
problem?
```

(continued on next page)

(continued from previous page)

```
Starting remote procedure call (RPC) services: sysidns done.
Starting Solaris installation program...
Searching for JumpStart directory...
SUNW,hme0 : No response from Ethernet network : Link down -- cable
problem?
<<< using CD default >>>
Checking rules.ok file...
Using profile: S8-server.profile
Executing JumpStart preinstall phase...
Searching for SolStart directory...
Checking rules.ok file...
Using begin script: install_begin
Using finish script: patch_finish
Executing SolStart preinstall phase...
Executing begin script "install_begin"...
Begin script install_begin execution completed.

Processing default locales
    - Specifying default locale (en_US)
Processing profile
    - Selecting cluster (SUNWCall)
    - Deselecting package (SUNWpmowm)
    - Deselecting package (SUNWpmowr)
    - Deselecting package (SUNWpmowu)
    - Deselecting package (SUNWpmr)
    - Deselecting package (SUNWpmu)
    - Deselecting package (SUNWpmux)
    - Selecting locale (en_US)

Installing 64 bit Solaris packages
    - Selecting all disks
    - Configuring boot device
    - Using disk (c0t0d0) for "rootdisk"
    - Configuring / (c0t0d0s0)
    - Configuring swap (c0t0d0s1)
    - Deselecting unmodified disk (c0t1d0)
    - Deselecting unmodified disk (c1t8d0)
    - Deselecting unmodified disk (c1t9d0)
    - Deselecting unmodified disk (c1t10d0)
    - Deselecting unmodified disk (c1t11d0)
    - Deselecting unmodified disk (c1t12d0)
    - Deselecting unmodified disk (c1t13d0)
    - Deselecting unmodified disk (c2t0d0)
    - Deselecting unmodified disk (c2t1d0)
```

(continued on next page)

(continued from previous page)

```
    - Deselecting unmodified disk (c2t2d0)
    - Deselecting unmodified disk (c2t3d0)
    - Deselecting unmodified disk (c2t4d0)
    - Deselecting unmodified disk (c2t5d0)

Verifying disk configuration
    - WARNING: Unused disk space (c0t0d0)
    - WARNING: Changing the system's default boot device in the
EEPROM
Verifying space allocation

    - Total software size:  737.00 Mbytes

Preparing system for Solaris install

Configuring disk (c0t0d0)
    - Creating Solaris disk label (VTOC)
Creating and checking UFS filesystems
    - Creating / (c0t0d0s0)

Beginning Solaris software installation
Starting software installation
    SUNWxwrtx...done.  736.96 Mbytes remaining.
    SUNWxwrtl...done.  736.91 Mbytes remaining.
    SUNWwbapi...done.  736.40 Mbytes remaining.
                .
                . (package listing deleted for brevity)
                .

    SUNWnamos...done.  257.17 Mbytes remaining.
    SUNWnamow...done.  257.09 Mbytes remaining.
    SUNWnamox...done.  256.90 Mbytes remaining.

Completed software installation

Customizing system files
    - Mount points table (/etc/vfstab)
    - Unselected disk mount points (/var/sadm/system/data/
vfstab.unselected)
    - Network host addresses (/etc/hosts)

Customizing system devices
    - Physical devices (/devices)
    - Logical devices (/dev)
```

(continued on next page)

(continued from previous page)

```
Installing boot information
    - Installing boot blocks (c0t0d0s0)
    - Updating system firmware for automatic rebooting

Installation log location
    - /a/var/sadm/system/logs/install_log (before reboot)
    - /var/sadm/system/logs/install_log (after reboot)
Installation complete
Executing SolStart postinstall phase...
Executing finish script "patch_finish"...

Finish script patch_finish execution completed.
Executing JumpStart postinstall phase...
The begin script log 'begin.log'
is located in /var/sadm/system/logs after reboot.


The finish script log 'finish.log'
is located in /var/sadm/system/logs after reboot.

syncing filesystems... done
rebooting...
Resetting ...

screen not found.
Can't open input device.
Keyboard not present.  Using ttya for input and output.

Sun Ultra 60 UPA/PCI (2 X UltraSPARC-II 450MHz), No Keyboard
OpenBoot 3.27, 2048 MB memory installed, Serial #13100146.
Ethernet address 8:0:20:c8:ff:fa, Host ID: 80c8fffa.

Initializing Memory
Rebooting with command: boot
Boot device: disk:a  File and args:
SunOS Release 5.8 Version Generic_108528-05 64-bit
Copyright 1983-2000 Sun Microsystems, Inc.  All rights reserved.
configuring IPv4 interfaces: hme0.
Hostname: client06
Configuring /dev and /devices
Configuring the /dev directory (compatibility devices)
The system is coming up.  Please wait.
Configuring network interface addresses: hme0
```

(continued on next page)

(continued from previous page)

```
SUNW,hme0 : No response from Ethernet network : Link down -- cable
problem?
SUNW,hme0 : No response from Ethernet network : Link down -- cable
problem?
.
SUNW,hme0 : No response from Ethernet network : Link down -- cable
problem?
SUNW,hme0 : No response from Ethernet network : Link down -- cable
problem?
RPC: Timed out
SUNW,hme0 : No response from Ethernet network : Link down -- cable
problem?
SUNW,hme0 : No response from Ethernet network : Link down -- cable
problem?
Starting IPv4 routing daemon.
starting rpc services: rpcbind done.
System identification is completed.
Setting netmask of hme0 to 255.255.255.0
SUNW,hme0 : No response from Ethernet network : Link down -- cable
problem?
Setting default IPv4 interface for multicast: add net 224.0/4:
gateway client06
syslog service starting.
Print services started.
volume management starting.
The system is ready.

client06 console login:
```

Note – The repeated warnings about the lack of network response and the RPC time-out error during the postinstallation boot are due to the installation client being disconnected from the network.

Recovering a Failed System with JumpStart

The typical mechanism for attempting recovery on a failed system is to boot the host from the Solaris OE installation CD. This approach is the most direct method to bring the to-be-recovered host to a point where some form of corrective action can be initiated. However, this method does have a major drawback: the unalterable nature of the CD restricts the available tools and thereby restricts the recovery procedures or adversely affects the recovery time. Tools commonly used in the datacenter, such as Veritas Volume Manager (VxVM), Veritas NetBackup, or Solstice Backup™ software are either completely unavailable or very cumbersome to use when booted from the CD.

Booting from the network with JumpStart technology is virtually identical to booting from the CD-ROM. Although the mechanics of access to the filesystem and data files may differ between a network boot and a CD boot, the operating environment image loaded when booting from the network is identical to the one loaded when booting from the CD-ROM. In addition, booting from the network restricts the system administrator in the same fashion as booting from the CD itself; the root filesystem provided to the client is mounted read-only. However, since the JumpStart server's boot image originates on writable media, it can be modified at the server side before the client is booted.

JumpStart Recovery Techniques

An example of this concept is patching the JumpStart boot image for the Solaris OE. The boot image is patched, on the server, with the `patchadd` command. When a client does a network boot, it is served the patched kernel. Similarly to this straightforward patching procedure, the boot image can also be changed to allow a different startup processing or can even be augmented with other tools that might prove useful during a service event or a system recovery.

Additionally, by modifying the client's boot image, you can install software tools commonly used in the datacenter into the JumpStart boot image and configure them for use by any system which uses that boot image.

Further, by installing Solaris for IA on a laptop and modifying the JumpStart miniroot on the laptop, you can create a mobile recovery server. This mobile recovery server can then be deployed onto whatever subnet is necessary at the time of system recovery.

$ROOTDIR Directory

To understand how to correctly modify or augment the client's boot image, you must understand how the JumpStart server maps the filesystem that contains the boot image to the client's view of the same filesystem. For example, to modify the network services available on the client, you may need to modify the client's `/etc/inetd.conf` file. If this is the case, you must locate and modify the client's `/etc/inetd.conf` file on the JumpStart server.

This information is controlled, in part, by the manner in which the JumpStart server was installed. `/jumpstart` is the filesystem that is used as the location for all JumpStart configuration information, profiles, boot images, and software installation packages.

1. **Use commands `setup_install_server` and `add_to_install_server` to copy the boot image and install packages from the Solaris OE Software CD-ROMs to the specified directory (`/jumpstart/OS/Solaris_8_2001-04`) on the JumpStart server.**
2. **Execute the `add_install_client` command to configure the sample client (`client06`) to boot from this server image.**
3. **Use the `add_install_client` command to update (or create if necessary) the `/etc/bootparams` file containing the information for the specified client.**
4. **Specify the `-e` and `-i` options and instruct `add_install_client` to update `/etc/ethers` and `/etc/hosts`, respectively, if necessary.**

   `/jumpstart` is shared through the NFS system with read-only access rights and the effective UID of unknown users mapped to `root`. For the `anon=0` option, see the `share_nfs`(1M) man page for additional details.

The following command sample illustrates a typical installation and configuration of a JumpStart server.

```
server01# share -F nfs -o ro,anon=0 /jumpstart
server01# mkdir /jumpstart/OS/Solaris_8_2001-04
server01# cd /cdrom/sol_8_401_sparc/s0/Solaris_8/Tools
server01# ./setup_install_server /jumpstart/OS/Solaris_8_2001-
04
Verifying target directory...
Calculating the required disk space for the Solaris_8 product
Copying the CD image to disk...
Install Server setup complete
[ insert Solaris 8 Software cd 2 of 2 ]
server01# cd /cdrom/sol_8_401_sparc_2/Solaris_8/Tools
server01# ./add_to_install_server /jumpstart/OS/Solaris_8_2001-
04

The following Products will be copied to /jumpstart/OS/
Solaris_8_2001-04/Solaris_8/Product:

Solaris_2_of_2

If only a subset of products is needed enter Control-C
and invoke ./add_to_install_server with the -s option.

Checking required disk space...

Copying the Early Access products...
213481 blocks


Processing completed successfully.
server01# cd /jumpstart/OS/Solaris_8_2001-04/Solaris_8/Tools
server01#./add_install_client \
> -i 129.153.47.6 -e 8:0:20:7c:ff:d0 \
> -p server01:/jumpstart/Sysidcfg/Solaris_8 \
> -c server01:/jumpstart \
> client06 sun4u
```

Examination of `/etc/bootparams` shows how the JumpStart server, `server01`, view of the client's root filesystem is mapped to the client server, `client06`, view.

```
client06  root=server01:/jumpstart/OS/Solaris_8_2001-04/
Solaris_8/Tools/Boot install=server01:/jumpstart/OS/
Solaris_8_2001-04 boottype=:in sysid_config=server01:/jumpstart/
Sysidcfg/Solaris_8 install_config=server01:/jumpstart
rootopts=:rsize=32768
```

The parameter `root` is set to `server01:/jumpstart/OS/Solaris_8_2001-04/Solaris_8/Tools/Boot`. This is what the NFS filesystem `client06` mounts as its root filesystem when booting over the network. Since the JumpStart environment variable, `$ROOTDIR`, is set to this value, the JumpStart server's view of this directory is commonly referred to as `$ROOTDIR`. In this example, `$ROOTDIR` is set to `server01:/jumpstart/OS/Solaris_8_2001-04/Solaris_8/Tools/Boot`. Locating or placing a file into the client's filesystem on the JumpStart server simply becomes a matter of prefixing `$ROOTDIR` to the file. For example, if the client needs to have a file placed in its `/etc` directory, that file must be placed in `$ROOTDIR/etc` on the JumpStart server.

Altering the Boot Process

This section demonstrates how to augment the client's boot image to make it suitable for use as a platform for system recovery operations:

- Modify the option and argument processing during boot
- Provide services and daemons
- Provide an interactive shell

It is important to note that the following modifications to the client's boot image offer additional functionality. The default functionality of the client's boot image remains unchanged, and you can still use the client's boot image to install the Solaris OE.

The first challenge in transforming the install boot image into a boot image suitable for recovery operations is to change the boot process, or startup process, of the miniroot. The client should come up to multiuser mode, yet not enter the default action of beginning the installation process. To take control away from the default boot process, you must modify the scripts that run at startup time on the client.

Processing Options and Arguments During Boot

The first task in converting the install boot image into a recovery boot image is to augment the boot process with a “recover” mode. You do that by altering the boot parameter processing logic in the startup scripts. For example, the OBP boot command, `boot net - install`, is normally used to perform a JumpStart (network) boot and installation. The `install` argument is passed through the kernel and to `init` as the system startup scripts are executed. We want to find where that argument is parsed and add logic to allow other keywords (such as `recover`).

The OBP passes options to the kernel, such as the `-s` (boot to single-user mode) and `-r` (initiate a device tree rebuild and kernel reconfiguration) options. In this instance the kernel is `genunix`, the miniroot. The OBP passes arguments along to the kernel as well. Since the kernel does not process command-line arguments (it only processes the options it recognizes), the arguments are ignored by the kernel and passed along to `init`. To prevent confusion, kernel switches and arguments are separated by a lone dash, which is why the space following the minus character (`-`) is crucial in the command `boot net - install`.

In turn, `init` passes all arguments on to the scripts that it calls. FIGURE 0-1 shows an overview of the startup process.

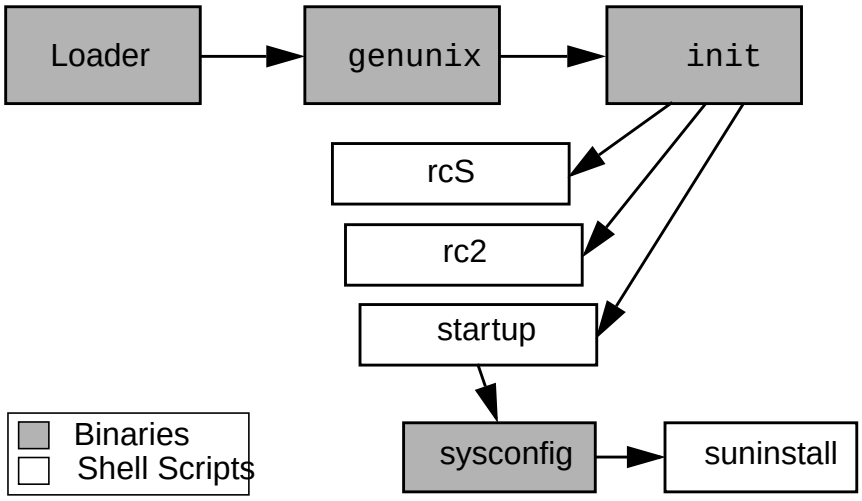

FIGURE 0-1 Overview of the Startup Process

By using the processing control table used by `init`, `$ROOTDIR/etc/inittab`, as a road map to the startup processing, you can determine that `$ROOTDIR/sbin/rcS` is where the argument processing takes place. The portion of `$ROOTDIR/sbin/rcS` relevant to the argument processing is as follows:

```
set -- ""
set -- '/sbin/getbootargs 2>/dev/null'
if [ $# -gt 0 ] ; then
        while [ $# -gt 0 ] ; do
                case $1 in
             FD=*)
# at end of script, save root dev in /tmp/.preinstall
# this is an unambiguous indication of stub boot
                        FJS_BOOT="yes"
                     From='(IFS="="; set -- $1; echo "$2 $3 $4 $5" )'
                        break
                        ;;
                browser)
                        cat < /dev/null > /tmp/.install_boot
                        cat < /dev/null > /tmp/.smi_boot
                        shift
                        ;;

                install)
                        INSTALL_BOOT="yes"
                        cat < /dev/null > /tmp/.install_boot
                        shift
                dhcp)
                        TRY_DHCP="yes"
                        shift
                        ;;
                tape*)
                        echo "$1" >/tmp/.cjfiles_method
                        shift
                        ;;

                mansysid)
                        cat < /dev/null > /tmp/.manual-sysid
                        shift
                        ;;

                w)
                        cat < /dev/null > /tmp/.nowin
                        shift
                        ;;
                *)
                        shift
                        ;;
                esac
        done
fi
```

It is now relatively straightforward to add recognition and processing for the `recover` argument by adding another “word” to the case statement. The modified section of `$ROOTDIR/sbin/rcS` is as follows:

```
set -- ""
set -- ‘/sbin/getbootargs 2>/dev/null‘
if [ $# -gt 0 ] ; then
        while [ $# -gt 0 ] ; do
                case $1 in
             FD=*)
# at end of script, save root dev in /tmp/.preinstall
# this is an unambiguous indication of stub boot
                        FJS_BOOT="yes"
                      From=‘(IFS="="; set -- $1; echo "$2 $3 $4 $5" )‘
                        break
                        ;;
                browser)
                        cat < /dev/null > /tmp/.install_boot
                        cat < /dev/null > /tmp/.smi_boot
                        shift
                        ;;
                recover)
                        cat < /dev/null > /tmp/._recover_startup
                        shift
                        ;;
                install)
                        INSTALL_BOOT="yes"
                        cat < /dev/null > /tmp/.install_boot
                        shift
                dhcp)
                        TRY_DHCP="yes"
                        shift
                        ;;
                tape*)
                        echo “$1” >/tmp/.cjfiles_method
                        shift
                        ;;
                mansysid)
                        cat < /dev/null > /tmp/.manual-sysid
                        shift
                        ;;

                w)
                        cat < /dev/null > /tmp/.nowin
                        shift
                        ;;
                *)
                        shift
                        ;;
                esac
        done
fi
```

It is important to note that almost no action is performed within the case statement or `$ROOTDIR/sbin/rcS`. As in the case of the `install` argument, the processing of the `recover` argument consists *only* of creating a state file (or flag file). Using this mechanism eases the decision process in scripts executed later, without the need to parse the arguments directly. If recovery scripts or tools must start automatically on a recovery boot or if scripts must determine the state of the system and take appropriate action, these scripts *only* need to check for the existence of the file `/tmp/._recover_startup` to determine the system's state. Because this method of modifying `$ROOTDIR/sbin/rcS` reduces complexity and minimizes the opportunity for human error in modifying the crucial `$ROOTDIR/sbin/rcS` script, it is strongly recommended that you use the same or a similar approach whenever adding additional startup processing.

Providing Services for Recovery

The default boot process does not start any service daemons that are not required by the Solaris OE installation process. However, some of these daemons may facilitate or are needed during a system recovery operation. For example, the Internet service daemon, `inetd`, is not needed for installation. However, a recovery operation is greatly facilitated by having `inetd` start automatically on a recovery boot.

Unfortunately, daemons can be started from many scripts and there is no reference or mapping of where a particular daemon may be started. However, most of the standard or common Solaris OE daemons are present, but commented out (referred to as *stubbed*), in the startup scripts executed by `init`. The only recourse is to follow the execution flow of the scripts launched by `init` to locate what services are started from which scripts, uncommenting the required daemons. For example, `inetd` is present but commented out in `$ROOTDIR/sbin/sysconfig`. To start `inetd` during a recovery boot, modify `$ROOTDIR/sbin/sysconfig` (at the point where `inetd` startup is commented out), as follows:

```
if [ -f /tmp/._recover_startup ] ; then
    /usr/sbin/inetd -s
fi
```

Starting `inetd` is sufficient for the most common network services. "Adding a Recovery Tool" on page 26 gives an example of adding additional services. If a service or daemon is required at startup but is not stubbed in one of the startup scripts, you must add the invocation to the startup processing. If it is necessary to add a call to a script or the execution of a command, it is recommended that the addition be done as late as possible in the startup sequence to avoid impacting dependencies or setup work that services may have on other services or daemons. For example, many network services require that `sysidnet` has been run, which implies the presence of a name service or `sysidcfg` file.

Providing an Interactive Shell

The default installation boot image enters the installation utility (`suninstall`) at the end of its startup processing. To have the default installation boot image act as a recovery platform, you will find it advantageous to provide an interactive shell rather than to execute `suninstall`. To provide the interactive shell, add the following lines to the end of `$ROOTDIR/sbin/sysconfig`.

```
if [ -f /tmp/._recover_startup ] ; then
    echo “Starting interactive Korn shell...“
    exec /bin/ksh -o vi
fi
```

Adding Utilities and Manual Pages

The methodology of the previous section can also be applied to the task of augmenting the client’s boot image with tools, utilities, and documentation. Most of the tools to be added to the client’s boot image do not require special configuration. In most cases, the utility or tool only needs to be copied into its expected location under `$ROOTDIR` to be accessible and usable by the client after a `boot net -recover`. However, it is important to note that device drivers, kernel extensions, and utilities expecting a writable `/var/tmp` are a special case that is examined in “Files in `/var`” on page 29.

Adding a Recovery Tool

Let’s take Solstice Backup software as an example of adding a recovery tool to a client’s boot image. Solstice Backup software installs its client-side utilities in `/usr/bin/nsr` and `/usr/lib/nsr`. To provide the client portions of Solstice Backup software to the client’s recovery boot image, you need only replicate or relocate the `/usr/bin/nsr` and `/usr/lib/nsr` directory hierarchies on the JumpStart server in `$ROOTDIR/usr/bin/nsr` and `$ROOTDIR/usr/lib/nsr`, respectively. You accomplish this relocation by using the root-path option (`-R`) of `pkgadd` or simply by copying the directory hierarchies from an installed Solstice Backup Client to `$ROOTDIR/usr/bin/nsr` and `$ROOTDIR/usr/lib/nsr`.

Adding Device Drivers

As with system services and daemons, the only device drivers provided in the default client's boot image are those device drivers that may be necessary for installing the Solaris OE. Providing the client's recovery boot image with additional device drivers, such as FDDI, ATM, or some other high-bandwidth network driver, can be useful or required in system recovery situations.

Consider the FDDI network interface. The default client's boot image does not provide this device driver. However, many datacenters use a dedicated FDDI network for backups and restores. During a recovery operation, the system administrator would benefit from having the recovery boot image available to access this high-bandwidth network to recover data from the backup server.

In principle, device drivers are added in much the same way that standard files are added to the JumpStart server. Device drivers or kernel extensions are merely copied to `$ROOTDIR/kernel/drv`. Unfortunately, a device driver may need to modify several other configuration files in order to function properly. The Solaris `add_drv` command is the mechanism by which the required files can be updated in a controlled fashion. For example, to add the FDDI driver to the recovery boot image, execute the following commands:

```
server01# ROOTDIR=/jumpstart/OS/Solaris_8_2001-04/Solaris_8/
Tools/Boot ; export ROOTDIR
server01# cd /kernel/drv
server01# cp fddi fddi.conf $ROOTDIR/kernel/drv
server01# add_drv -b $ROOTDIR /kernel/drv/fddi
```

Note – Before installation, consult the manufacturer or vendor instructions for all device drivers.

For most drivers, these commands are sufficient for installation into the recovery boot image. However, some device drivers, such as those that create or require entries in `/dev` upon boot, may require further configuration by hand. Some device drivers may require modifications to files such as these:

- `$ROOTDIR/etc/devlink.tab`
- `$ROOTDIR/etc/driver_aliases`
- `$ROOTDIR/etc/system`

After you configure the drivers by hand, you must then appropriately modify the respective files under `$ROOTDIR` in order to affect the client's boot image. Consult the installation documentation for the device driver for any specific modifications that may be necessary.

For example, the Veritas Volume Manager (VxVM) product includes the device drivers `vxspec`, `vxio`, and `vxdmp`. In addition to copying these drivers into `$ROOTDIR/kernel/drv`, you must load these device drivers at boot by adding the following directives to `$ROOTDIR/etc/system`.

```
forceload drv/vxdmp
forceload drv/vxspec
forceload drv/vxio
```

Additionally, these VxVM device drivers require device entries to be created in `/dev` according to the specific template in `/etc/devlink.tab`.

Meeting Challenges Unique to the Miniroot

As previously discussed, the miniroot is a subset Solaris OE kernel and filesystem. The miniroot must fit in a prescribed amount of physical memory and so does not contain all of the software packages that an installed system would contain. In addition to the size limitations, there are challenges in installation that are unique to the miniroot. This section covers these specific challenges:

- Read-only media
- Files in `/var`
- `path_to_inst` file

Read-Only Media

While it is obvious that the boot image on a Solaris OE Software CD is read-only, it may not be obvious that the client's boot image served by the JumpStart server is also read-only. Because of this, all systems booted from this image for recovery or installation mount their filesystems read-only. Whether booted from a CD or over the network from a JumpStart server, the `/tmp` memory-mapped `tmpfs` filesystem is the only writable space for any utility or service that requires writable media.

The Solaris OE `tmpfs` filesystem is fashioned from the virtual memory in the client and as such, it loses all its contents when the client reboots. Any file or directory on the remaining filesystems that must behave as if it is writable must be redirected to `/tmp` by means of a symbolic link. For example, the `/etc/vfstab` file must be writable, but `/etc` is on a read-only filesystem. The file `/etc/vfstab` is actually a symbolic link, created at the client's boot, to a writable `vfstab` file in `/tmp`. All such

writable file links are created on the read-only CD image or on the client's boot image on the JumpStart server. The targets of all such symbolic links are then created in `/tmp` when the miniroot is booted.

The same mechanism can be used to provide writable space for tools added to the recovery boot image. The following commands provide a writable log space for NetBackup in `$ROOTDIR/usr/openv/logs`.

```
server01# cd /jumpstart/OS/Solaris_8_2001-04/Solaris_8/Tools/
Boot/usr/openv
server01# ln -s /tmp/_openv_logs ./logs
```

However, this approach addresses only half the problem. The target of the symbolic link, in `/tmp`, must still be populated at boot. Any required link targets in `/tmp` can be populated by one of the following methods:

1. Create a start script that executes `mkdir /tmp/_openv_logs`.
2. Utilize the built-in script for populating `/tmp`.

Method one is a brute force approach, but effective on a small or limited scale. The second method takes advantage of the existing method that the miniroot uses at boot to populate `/tmp`.

A prototype (or template for) `/tmp` is maintained on the miniroot and is copied into `/tmp` very early in the boot cycle. This prototype is the directory `$ROOTDIR/.tmp_proto`. Early in the startup sequence (in `$ROOTDIR/sbin/rcS`), `cpio` is used to populate `/tmp` from this prototype. These prototypes become the targets referenced by symbolic links throughout the miniroot image.

For our example above, creating the directory `$ROOTDIR/.tmp_proto/_openv_logs` ensures that `/tmp/_openv_logs` are created when the client miniroot is started.

Files in `/var`

The `/var` directory tree is an extension of the writable directory problem outlined in "Adding Utilities and Manual Pages" on page 26. Most facilities in the Solaris OE utilize `/var/tmp` or write temporary files into some subdirectory of `/var`. Because so many utilities expect `/var` to be writable, the entire `/var` filesystem is made writable by means of a symbolic link from `/var` to `/tmp/var`. This is accomplished by the same mechanism as described in "Adding Utilities and Manual Pages" on page 26. The prototype for the `/var` hierarchy is in `$ROOTDIR/.tmp_proto/var`. For a file or directory to be relocated into the client's `/var`, the file or directory must be the `$ROOTDIR/.tmp_proto/var` prototype.

path_to_inst File

The Solaris OE device instance number file, path_to_inst, is a special case. It is special because this file must be created from scratch each time a system boots from the miniroot. The path_to_inst file must be writable; however, it cannot simply link to /tmp.

The miniroot cannot possibly have a valid device tree and instance number file for each client that might boot from it. The path_to_inst file and the /dev and /devices directory trees must be created by the miniroot each time a system boots from the CD or JumpStart boot image. The OBP is responsible for probing the bus and inventorying all of the devices in the system. The OBP hands off the result of this inventory (the OBP device tree) to the kernel when it begins loading. The OBP device tree is used as the basis for the kernel to map the device location (path) to the device driver instance numbers. This mapping is what is stored in the path_to_inst file. For a boot from read-only media (for example, from CD or from a JumpStart server) the newly created path_to_inst file is generated and written to /tmp/root/etc/path_to_inst early in the miniroot boot process.

Because a valid /etc/path_to_inst is required by the kernel very early in the boot cycle, using a writable surrogate in /tmp is not possible, since /tmp is not yet populated when the instance number file is needed.

For the kernel to be redirected to the *real* path_to_inst file that was written in /tmp/root/etc/path_to_inst, *two* path_to_inst files are used. The *real* path_to_inst, which contains valid device paths mapped to the correct instance names, is created in /tmp/root/etc/path_to_inst. The bootstrap version, required by the kernel but invalidated after the kernel loads, is in /etc. This bootstrap version of $ROOTDIR/etc/path_to_inst consists only of the line:

```
#path_to_inst_bootstrap_1
```

This bootstrap path_to_inst can cause problems for utilities that are hardcoded to reference /etc/path_to_inst for the device instance number file.

System utilities and diagnostic tools, like SunVTS™ software or older versions of STORtools, that must build internal tables of the devices attached to the system typically read /etc/path_to_inst directly, rather than obtaining the device instance information from the kernel.

Unfortunately, if these utilities hardcode the path name for /etc/path_to_inst into their object modules, libraries, or binaries, the path name is not easily changed. This situation prevents the use of such utilities while the client is booted from the JumpStart server or CD.

Summary

This chapter provided techniques to augment a CD-ROM-based installation with the services and behaviors provided by a JumpStart server. The techniques provided here are suitable to situations when a hands-free Solaris OE installation is necessary but when a JumpStart server cannot be used.

This chapter also detailed a procedure to create a bootable installation CD, examined the structure of a bootable Solaris OE CD, and provided specifics on the modification of the installation behaviors.

Additionally, this chapter examined the startup processing performed by the miniroot and the use of the miniroot as a platform for recovery operations. The chapter described methods and techniques to augment the startup processing to provide a framework for recovering a system. Also described were techniques for adding tools to the client's boot image.

Finally, the chapter examined the constraints of the miniroot and techniques for working within these constraints.