

Embedded Solaris

(by: Noel Milton Vega)
Rensselaer Technology Group, LTD.
V3.0

Creating appliance versions of Sun Solaris for embedded applications

Introduction

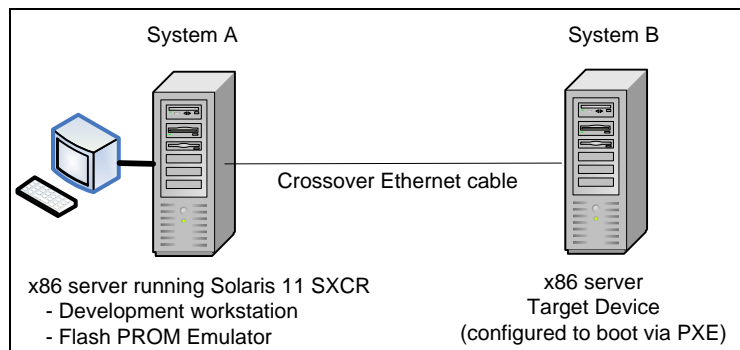
The Solaris operating system, because of the many years of R&D invested in it by Sun Microsystems engineers, and also by contributors to the Open Solaris community since 2005, is arguably the most advanced O/S today. By sharing its source code with the community¹ starting in 2005, Sun and Solaris took another step in that direction: by allowing it to be examined, modified, and compiled from source code, Solaris can now be adapted to suit purposes other than the traditional ones it served prior to that (namely to provide the operating environment for enterprise computers).

One such adaptation of Solaris is the creation of customized small footprint versions of it that can be embedded into devices/systems that perform specific functions. Such a device, when supplied with power, boots a custom version of Solaris and begins to run as designed. This is what this we will cover in is this, sure to grow, document.

x86 Target Device

Embedded systems development environments have two parts to it: the development environment; and the target, or embedded system. (explain each here).

We begin our discussion with an example that uses two x86 servers², configured as shown in the following diagram:



The x86 server on the left (System A), which runs the “Entire Distribution” meta-cluster of Solaris 10 SXCR, will serve two distinct purposes: (1) it will serve as our embedded application development and build environment, and (2) through its crossover Ethernet cable connection, it will emulate the FLASH PROM of the x86 based target device shown on the right (System B). The x86 server on the right (System B), is our target device.

This initial example is useful because it demonstrates basic embedded systems concepts. But, as we will see, it is also obliquely interesting since it demonstrates how to build and PXE boot a tiny version of the Solaris operating environment that can customize itself at boot-time to function as whatever appliance we like; so administrators will also find it to be a somewhat relevant example.

¹ Under the terms outlined in the “Common Development and Distribution License” (CDDL) license.

² If you want to follow along, but don’t have two servers, you can emulate this environment using Solaris Containers (Zones) or with VMWare Server running on a PC running Linux or Windows.

Creating appliance versions of Sun Solaris for embedded applications

Setting up the FLASH PROM device

In this example, the setup of our FLASH PROM device consists of the following steps on System A:

- (a) Extracting the pxegrub, multiboot, and x86.miniroot components from the Solaris distribution media (usually CDR #1).
- (b) Setting up a DHCP daemon to act as a PXE boot server. We'll use ISC's (isc.org) DHCP server.
- (c) Setting up a TFTP daemon to serve up the initial multiboot kernel, and then our custom Solaris ramdisk based O/S (x86.miniroot).
- (d) Open up and modifying x86.miniroot ramdisk image to customize our Solaris O/S. Particularly we'll reduce its size, and add wget(1) along with a boot script to create the appliance we want.

<<Describe the function x86.miniroot, multiboot, pxegrub here>>.

Creating and populating /tftpboot

Create the /tftpboot directory. Then, within /tftpboot, create a subdirectory that will hold the files "x86.miniroot", "multiboot", and "pxegrub", which we copy off the boot (CD#1) of the Solaris distribution media. The name of this subdirectory should reflect the version of Solaris from which these files were obtained. We'll assume version "Solaris 10 SXCR snv40" throughout, so:

```
root# mkdir -p /tftpboot/SOLARIS10SXCR_snv27.d
root# mkdir -p /tftpboot/SOLARIS10SXCR_snv40.d
root# mkdir -p /tftpboot/SOLARIS10SXCR_snv42.d
(and so on)
```

Next, mount the Solaris boot CD and copy the 3 files mentioned previously to the destination directory corresponding to the version of Solaris on that CD. So, for a boot CD containing Solaris Express 10 Community Release version 40, the files would be copied to /tftpboot/SOLARIS10SXCR_snv40.d:

```
root# mount -F hsfs -o ro /dev/dsk/c0t0d0s0 /mnt
root# cp /mnt/boot/x86.miniroot /tftpboot/SOLARIS10SXCR_snv40.d
root# cp /mnt/boot/multiboot /tftpboot/SOLARIS10SXCR_snv40.d
root# cp /mnt/boot/grub/pxegrub /tftpboot/SOLARIS10SXCR_snv40.d
root# cd /; umount /mnt
```

Note: Your cdrom device name may differ.

At this point our /tftpboot directory recursively looks like this:

```
root# ls -lR /tftpboot (output reformatted for brevity).

/tftpboot/SOLARIS10SXCR_snv27.d: multiboot, pxegrub, x86.miniroot
/tftpboot/SOLARIS10SXCR_snv40.d: multiboot, pxegrub, x86.miniroot
/tftpboot/SOLARIS10SXCR_snv42.d: multiboot, pxegrub, x86.miniroot
```

Creating appliance versions of Sun Solaris for embedded applications

With the required files in place to network-boot a mini version of Solaris (x86.miniroot), we turn our attention to creating the per-client specific entries in /tftpboot that will steer incoming pxeboot request from those clients towards the particular mini version of Solaris they should boot. For each client, these entries will consist of simply two files: (1) a symbolic link called "pxegrub", which will point to the specific Solaris version of pxegrub it should use, and (2) a file named "menu.lst", which specifies the ramdisk file containing the kernel and filesystem for the mini version of Solaris that it will boot.

Continuing with the scalable approach, these entries are client specific, and as such will be placed in client specific sub-directories whose name will be of the form "hostname.d", where hostname is the host name of the booting client. For two clients whose host names are "galaxy11", and "galaxy12", here is how their respective directories would be created:

```
root# mkdir /tftpboot/galaxy11.d
root# mkdir /tftpboot/galaxy12.d
```

Next we create the "pxegrub" and "menu.lst" entries in the directories of each client. In the example that follows, we are going to have the client identified by the host name, galaxy11, boot "Solaris 10 SXCR snv40", so:

```
root# cd /tftpboot/galaxy11.d
root# ln -s ../SOLARIS10SXCR_snv40.d/pxegrub pxegrub
root# vi menu.lst (and insert only the following lines)
```

```
default=0
timeout=30
title Solaris10SXCR_snv40 RAM O/S
        kernel SOLARIS10SXCR_snv40.d/multiboot kernel/unix
        module SOLARIS10SXCR_snv40.d/x86.miniroot
```

/tftpboot/galaxy11.d/menu.lst contents

```
root# ln -s /tftpboot/galaxy11.d/menu.lst /tftpboot/menu.lst.01000C294C5C64
```

(Note: Any letter(s) appearing in the MAC address, **must be** specified as CAPITAL letters. pxegrub will not find the "menu.lst.01MAC" file otherwise. I was able to verify this using `tcpdump -i <iface> -n -nn port 69` on the dhcp server (and saw the filename pxegrub was looking for).

The previous step (in blue) requires some explanation. As of this writing, the Solaris pxegrub binary (the one we copied off the installation CD above) does not seem to support DHCP option '150', which is the option that allows you to specify where to find the menu file in the dhcpd.conf file. pxegrub, in its current form, assumes the menu file for the booting client to be: "/tftpboot/menu.lst.01MAC", where MAC is the MAC address of the interface over which the client is booting (this too, was verified using tcpdump: `tcpdump -i <iface> -n -nn host <clientIP>`). Should this deficiency ever be corrected, the last step above will no longer be necessary, and instead we would augment the dhcpd.conf file to include the following (show here for host galaxy11):

```
option grubmenu code 150 = text; # this line exists outside any host/client specific stanza.
```

```
host galaxy11
{
    [ ... other options now shown ... ]
    option grubmenu "galaxy11.d/menu.lst";
}
```

Note: In the "menu.lst" file, all paths for the "kernel" and "module" directives, whether or not they are prefixed with a slash ("/"), assume the relative base directory of "/tftpboot"; since, by default, in.tftpd is started with the '-s' secure switch, causing that daemon to chroot into "/tftpboot" directory.

Creating appliance versions of Sun Solaris for embedded applications

In the future, inspect `/etc/dhcp/inittab` to see if option "150" has been added. Though it may exist, there is no guarantee that pxegrub has been modified to handle that option. But it may provide a clue.

In the future if we want this client to boot "Solaris 10 SXCR snv27" instead, we'd simply make adjustments to the pxegrub symlink, as well as to the paths specified for the "kernel" and "module" directives in the "menu.lst" file. Pretty simple.

Setting up the TFTP, DHCP, FTP server daemons

With the `/tftpboot` directory populated with our per-Solaris specific kernels, pxegrub, multiboot images, as well as with our per-client boot configurations, we now turn our attention to the next step of setting up the network services that will serve up these configurations to network (PXE) booting clients. For this we need only three server daemons: TFTP, DHCP, and FTP. Strictly speaking, FTP is not necessary to boot clients to the default x86.miniroot environment; however, we will need it later on when we start customizing the run-time personality of our clients at boot time, so we might as well take a moment to configure it now.

Solaris ships with implementations for each of these services and, with an exception for DHCP, we will use them. For DHCP we'll use ISC's implementation³ because of its simplicity relative to the one shipped with Solaris. It should be noted that replacements for TFTP and FTP are also possible, but not covered here. Replacing the Solaris' `in.tftpd` and/or `in.ftpd` with, for example, `atftpd` and `proftpd` respectively, might appeal to minimal and/or security conscious environments not wanting to run the `inetd` service daemon.

Configuring `in.tftpd`

It's very likely that the `in.tftpd` service is already running, and configured to use `/tftpboot` as its operating directory. But let's go through the steps of configuring the service, just in case:

```
# Ensure that the root/base directory for in.tftpd is "/tftpboot". Either or both of the
# following commands will display this setting. You should see something like
# "/usr/sbin/in.tftpd -s /tftpboot" in the output. If the directory does not indicate
# "/tftpboot", use the svccfg(1M) command to modify the setting.
#
root# svccfg -s svc:/network/tftp/udp6 listprop
root# inetadm -l svc:/network/tftp/udp6:default

# Check that the TFTP service is enabled, and if not enable it.
# Check (either of these will show the status):
#
root# svcs -l svc:/network/tftp/udp6:default
root# svcs -a | grep tftp

# Enable (if necessary). Either of these commands will attempt to enable the service, but the
# first method and the collections of commands associated with it (i.e. svcadm; svcs;
# svccfg; svcprop), will help debug service dependency issues, should not start:
#
root# svcadm enable -rs svc:/network/tftp/udp6:default
root# inetadm -e svc:/network/tftp/udp6:default

# Finally, check via netsat.
root# netstat -a (or) netstat -an
```

³ Freely obtained at <http://www.isc.org>

Configuring in.ftpd

Similarly, it's very likely that the in.ftpd service is already running and configured to use "/etc/ftpd/ftppaces" as its start-up configuration file. But let's go through the steps of configuring the service just in case:

```
# Create an anonymous ftp user with a chrooted root directory of /JS.d (note that the in.ftpd
# service does not need to be running to run this. <<explain why /JS.d here>> .
root# ftpconfig /JS.d

# Check that in.ftpd is started as "/usr/sbin/in.ftpd -a", which informs the daemon to consult
# the file "/etc/ftpd/ftpaccess" for its start-up configuration. You should see something like
# "/usr/sbin/in.ftpd -a" in the output. If you see something different, use the svccfg(1M)
# command to modify the "exec" string.
#
inetadm -l svc:/network/ftp:default | grep exec

# Check that the FTP service is enabled, and if not enable it.
# Check (either of these will show the status):
#
root# svcs -l svc:/network/ftp:default
root# svcs -a | grep tftp

# Enable (if necessary). Either of these commands will attempt to enable the service, but the
# first method and the collections of commands associated with it (i.e. svcadm; svcs;
# svccfg; svcprop), will help debug service dependency issues, should not start:
#
root# svcadm enable -rs svc:/network/ftp:default
root# inetadm -e svc:/network/ftp:default

# Finally, check via netsat.
root# netstat -a (or) netstat -an
```

Creating appliance versions of Sun Solaris for embedded applications

Configuring ISC's DHCP server

Solaris ships with an implementation of DHCP, but we'll be using the freely available implementation developed by the Internet Systems Consortium (ISC) because of its flexibility and simplicity. We'll be compiling and installing this version with GNU gcc and make, and assume that you have reasonably up-to-date versions of these installed on your development system (System A).

To begin, download the latest copy of the ISC DHCP source code from <http://www.isc.org>. As of this writing, the latest version is 3.0.4, and comes packaged in the gzipped-tarfile: "dhcp-3.0.4.tar.gz". Create the directory "/var/tmp/DHCP-3.0.4.d" and copy the downloaded file to that directory, then unpack it as shown here:

```
root# mkdir -p /var/tmp/DHCP-3.0.4.d
root# cd /var/tmp/DHCP-3.0.4.d

[ here is where you copy the "dhcp-3.0.4.tar.gz" file into this directory ]

root# gzip -dc ./dhcp-3.0.4.tar.gz | tar -xf /dev/fd0

root# ls -la
root@solaris10sxcrcr# ls -la ./dhcp-3.0.4
total 294
drwxr-xr-x  15 10200   100           512 Jul 24 12:57 .
drwxr-xr-x   4 nmvega  staff         512 Jul 22 20:58 ..
drwxr-xr-x   3 10200   100           512 Apr 27 17:46 client
drwxr-xr-x   2 10200   100          1024 Apr 27 17:46 common
-rwxr-xr-x   1 10200   100          6429 Sep 10 2004 configure
drwxr-xr-x   3 10200   100           512 Apr 27 17:46 contrib
drwxr-xr-x   2 10200   100           512 Apr 27 17:46 dhcpctl
drwxr-xr-x   3 10200   100           512 Apr 27 17:46 doc
drwxr-xr-x   2 10200   100           512 Apr 27 17:46 dst
drwxr-xr-x   8 10200   100           512 Apr 27 17:46 includes
-rw-r--r--   1 10200   100           966 Feb 22 17:43 LICENSE
-rw-r--r--   1 10200   100          2824 Jun 10 2004 Makefile
-rw-r--r--   1 10200   100          3145 Jul 22 13:55 Makefile.conf
-rw-r--r--   1 10200   100          1935 Jun 10 2004 Makefile.dist
drwxr-xr-x   2 10200   100           512 Apr 27 17:46 minires
drwxr-xr-x   2 10200   100           512 Apr 27 17:46 omapip
-rw-r--r--   1 10200   100         31787 Apr 27 17:46 README
drwxr-xr-x   2 10200   100           512 Apr 27 17:46 relay
-rw-r--r--   1 10200   100         66235 Apr 27 17:38 RELNOTES
drwxr-xr-x   2 10200   100           512 Apr 27 17:46 server
-rw-r--r--   1 10200   100           94 Jul 7 1999 site.conf
drwxr-xr-x   3 10200   100           512 Apr 27 17:46 tests
drwxr-xr-x  10 root    root          512 Jul 22 13:56 work.sunos5
```

Creating appliance versions of Sun Solaris for embedded applications

Next, replace the entire contents of "Makefile.conf" with the following (you may want to create a backup copy of Makefile.conf to "Makefile.conf.FCS" beforehand).

```
# Makefile.conf for Solaris 2.x using gcc for compiling.
SCRIPT = none
USERBINDIR = /opt/local/dhcpd/bin
BINDIR = /opt/local/dhcpd/sbin
CLIENTBINDIR=/opt/local/dhcpd/sbin
ADMMANDIR = /opt/local/dhcpd/man/man1m
ADMMANEXT = .0
FFMANDIR = /opt/local/dhcpd/man/man4
FFMANEXT = .0
LIBMANDIR = /opt/local/dhcpd/man/man3
LIBMANEXT = .0
USRMANDIR = /opt/local/dhcpd/man/man1
USRMANEXT = .0
MANCAT = cat
INSTALL = install -c -m 444
MANINSTALL = install -c
CHMOD = chmod
CATMANPAGES =
ETC = /etc
VARRUN = /var/run
VARDB = /var/db
LIBDIR=/opt/local/dhcpd/lib
INCDIR=/opt/local/dhcpd/include
LIBS =
COPTS = $(BINDEF) $(CC_OPTIONS)
DEBUG = -g
RANLIB = ranlib
MKDEP = mkdep
CLIENT_PATH = "PATH=/opt/local/dhcp/sbin:/usr/ucb:/usr/bin:/usr/sbin:/bin:/sbin"

BINDLIB = ../minires/libres.a
BINDINC =
MINIRES = minires

MAJORVERSION=MajorVersion
MINORVERSION=MinorVersion

INSTALL=/usr/ucb/install
MANINSTALL=/usr/ucb/install
LIBS = -lresolv -lsocket -lnsl -lgen
CC=gcc
COPTS = $(BINDEF) -Wall -Wno-unused -Wno-implicit -Wno-comment \
        -Wno-uninitialized -Wno-char-subscripts -Wno-switch $(WARNERR) \
        -DSOLARIS_MAJOR=$(MAJORVERSION) -DSOLARIS_MINOR=$(MINORVERSION) \
        $(CC_OPTIONS)
CF = cf/sunos5-5.h
ADMMANDIR = /opt/local/dhcpd/man/man1m
ADMMANEXT = .1m
FFMANDIR = /opt/local/dhcpd/man/man4
FFMANEXT = .4
LIBMANDIR = /opt/local/dhcpd/man/man3
LIBMANEXT = .3
USRMANDIR = /opt/local/dhcpd/man/man1
USRMANEXT = .1
MANCAT = man
VARRUN = /var/run
VARDB = /var/run
SCRIPT=solaris
```

Note: this configuration file will result in the software being installed in "/opt/local/dhcpd". If you require a different location, replace "/opt/local/dhcpd" where it appears, with the your location.

Creating appliance versions of Sun Solaris for embedded applications

Next, compile and install it. The commands to run are as follows:

```
root# mkdir -p /opt/local/dhcpd
root# ln -s /opt/local /usr/local

root# pwd
/var/tmp/DHCP-3.0.4.d
root# ./configure          [ ... output omitted ... ]
root# /usr/ccs/bin/make    [ ... output omitted ... ]
root# /usr/ccs/bin/make install [ ... output omitted ... ]
root# cd /opt/local/dhcpd
root# mkdir ./etc ./var
root# touch ./etc/dhcpd.conf ./etc/dhclient.leases ./etc/isc_dhcpd
root# ln -s /opt/local/dhcpd/etc/dhcpd.conf /etc/dhcpd.conf [For convenience.]
```

After running the commands above, integrate a startup/shutdown script into the /etc/init.d and /etc/rc2.d structure so that the dhcpd daemon is started each time Solaris boots. To do this, we first insert the contents of the following into the file "/opt/local/dhcpd/etc/isc_dhcpd". This is where master copy of this start script will be archived.

```
#!/bin/sh
#
# /etc/rc2.d/S73isc_dhcpd -> /etc/init.d/isc_dhcpd
#####
# IFACE=<networkInterface> #
#####
IFACE=pcn0
#####
#
case "$1" in
'start')
    if [ -f /opt/local/dhcpd/etc/dhcpd.conf -a \
        -f /opt/local/dhcpd/sbin/dhcpd ]
    then
        {
            /usr/bin/echo 'Starting ISC dhcpd...'

            /usr/bin/rm -f /opt/local/dhcpd/var/dhcpd.pid
            /opt/local/dhcpd/sbin/dhcpd -p 67 \
                -cf /usr/local/dhcpd/etc/dhcpd.conf \
                -lf /usr/local/dhcpd/etc/dhclient.leases \
                -pf /usr/local/dhcpd/var/dhcpd.pid ${IFACE}
        }
    fi
    ;;

'stop')
    /usr/bin/echo 'Stopping ISC dhcpd...'
    /usr/bin/pkill -9 dhcpd
    ;;

*)
    /usr/bin/echo "Usage: $0 { start | stop }"
    exit 1
    ;;

esac
exit 0
```

Creating appliance versions of Sun Solaris for embedded applications

Now, execute the following commands to integrate this start into the /etc/init.d structure:

```
root# ln -s /opt/local/dhcpd/etc/isd_dhcpd /etc/init.d/isc_dhcpd
root# ln /etc/init.d/isc_dhcpd /etc/rc2.d/S73isc_dhcpd
```

Note: Strictly speaking, we should use the `svccfg(1M)` command to properly integrate this service into the Solaris Service Management Facility (`smf(5)`), but the author has chosen to use the classic `init/rc` method to maintain topic focus.

Next, insert the contents of the following into the ISC DHCP startup config file, `"/opt/local/dhcpd/etc/dhcpd.conf"`:

```
ddns-update-style none;
authoritative;

option searchlist code 119 = string;
option grubmenu code 150 = text;

#Static IP Allocations
host galaxy11
{
    option host-name "galaxy11";
    hardware ethernet 00:0C:29:4C:5C:64;
    fixed-address          192.168.1.8;
    option routers         192.168.1.1;
    option subnet-mask     255.255.255.0;
    option broadcast-address 192.168.1.255;
    default-lease-time 51000;
    max-lease-time 51000;
    next-server 192.168.1.9;

    filename "galaxy11.d/pxegrub";

    # =====
    # in /tftpboot/galaxy11.d (for Solaris)
    # =====
    #option grubmenu "tftpboot/menu.lst.01000C294C5C64"; <--- required.
    #option grubmenu "galaxy11.d/menu.lst.solaris"; <--- wish.

    # =====
    # in /tftpboot/galaxy11.d (for Linux)
    # =====
    #option grubmenu "galaxy11.d/pxelinux.cfg/default"; <--- required.
    #option grubmenu "galaxy11.d/menu.lst.linux"; <--- wish.
}

# Dynamic IP Allocations
subnet 192.168.1.0 netmask 255.255.255.0 {
    option routers 192.168.1.1;
    option subnet-mask 255.255.255.0;
    option broadcast-address 192.168.1.255;
    range 192.168.1.200 192.168.1.210;
    default-lease-time 51000;
    max-lease-time 51000;
}
```

In a moment, we'll briefly cover the contents of this fairly basic and intuitive `dhcpd.conf` file. Refer to the manual pages in `/opt/local/dhcpd` for complete coverage on the format of and options available to `dhcpd.conf`; and, in general, on the operation of ISC's DHCP implementation.

Creating appliance versions of Sun Solaris for embedded applications

Finally, start the dhcpd server:

```
root# /etc/rc2.d/S73isc_dhcpd start [ ... output omitted ... ]
root# pgrep -lf dhcpd
3534 /opt/local/dhcpd/sbin/dhcpd -p 67 -cf /usr/local/dhcpd/etc/dhcpd.conf -lf ...
```

The dhcpd.conf file

At this point we have prepared the /tftpboot directory contents, as well as configure the network services that will allow us to boot the x86 based embedded client (System B). We'll talk about the boot sequence in a moment, but first let's briefly review the parameters of the dhcpd.conf file, as specific to our setup.

Each client (System B) that will load Solaris via PXE boot, will have a dedicated dhcpd.conf stanza similar to the one shown for host 'galaxy11' (highlighted in blue). The only difference between each client host's stanza will be reflected in attributes/options that specify its *hostname*; its statically assigned *IP address*; and the *MAC address* of the interface over which it PXE boots. It is in such a client stanza that the triple of hostname, IP, and MAC are fused together and essentially become synonymous. You'll notice that the hostname appears twice in a stanza: (1) as the name of the stanza itself, and (2) as the value of the 'host-name' DHCP option within that stanza. While the name assigned to a stanza is somewhat arbitrary and used to differentiate it from other stanzas, it is nevertheless given a name that reflects the host to which it applies. On the other hand, the value assigned to the 'host-name' option within the stanza is not arbitrary; it needs to be the hostname you wish to assign to a booting client, since that is the name that will be returned to DHCP query clients (e.g. pxegrub, pxelinux.0, dhcpinfo(1M), dhcpagent(1M), netbootinfo(1), udhcpc(1M), and others). Some of these clients will be is launched during a secondary phase of the boot sequence (more on that later).

The MAC address specified as the value for the 'hardware ethernet' option within the stanza, of course has to be the MAC address of the interface over which the client will boot. You will notice that the lines beginning with "option grubmenu ..." are commented out. As chronicled earlier, this is because of a current deficiency in the Solaris pxegrub program whereby it does not utilize DHCP option 150 (the option that allows us to specify a path to the 'menu.lst' file in dhcpd.conf). This unfortunately requires us to create a symlink, "/tftpboot/hostname.01MAC", for each client (which prevents us from keeping everything neatly packaged within its own client specific subdirectory).

The Boot Sequence

- talk about pxe/dhcp conversation between client and server.
- Talk about pxegrub fetching over menu.lst.01MAC.
- Talk about what the "kernel" and "module" entries in the menu.lst.01MAC mean, and how they are loaded by pxegrub.

Creating appliance versions of Sun Solaris for embedded applications

To be continued ... (cover dhcpd.conf/boot sequence/menu.lst.01MAC in tftpboot).

```
GNU GRUB  version 0.95  (586K lower / 522176K upper memory)

Solaris10SXCR_snv40 RAM 0/S

Use the ↑ and ↓ keys to select which entry is highlighted.
Press enter to boot the selected OS, 'e' to edit the
commands before booting, or 'c' for a command-line.

The highlighted entry will be booted automatically in 28 seconds.
```

Network version of GRUB (Solaris calls this binary pxegrub, while in Linux it is called pxelinux.0). Per our menu.lst contents above, this output was expected.

```
SunOS Release 5.11 Version snv_40 32-bit
Copyright 1983-2006 Sun Microsystems, Inc. All rights reserved.
Use is subject to license terms.
WARNING: CPU 0 has 0 MCA banks; expected 5: disabling MCA on this CPU
WARNING: /pci@0,0/pci15ad,1976@7,2 (uhci0): No SOF interrupts have been received
, this USB UHCI host controller is unusable
NOTICE: MPO disabled because memory is interleaved

WARNING: BIOS microcode patch for AMD Athlon(tm) 64/Opteron(tm) processor
erratum 109 was not detected; updating your system's BIOS to a version
containing this microcode patch is HIGHLY recommended or erroneous system
operation may occur.

Configuring devices.
WARNING: /pci@0,0/pci15ad,1976@7,2 (uhci0): No SOF interrupts have been received
, this USB UHCI host controller is unusable
WARNING: /pci@0,0/pci15ad,1976@7,2 (uhci0): No SOF interrupts have been received
, this USB UHCI host controller is unusable
Searching for installed OS instances...
No installed OS instance found.

Starting shell.
# -
```

Solaris booting into memory.

Creating appliance versions of Sun Solaris for embedded applications

```
# df -k
Filesystem          kbytes    used    avail capacity  Mounted on
/ramdisk:a          134441   130289      0    100%    /
/devices            0         0         0     0%    /devices
ctfs                0         0         0     0%    /system/contract
proc               0         0         0     0%    /proc
mnttab             0         0         0     0%    /etc/mnttab
swap              469160    188   468972     1%    /etc/svc/volatile
objfs              0         0         0     0%    /system/object
swap              469284    312   468972     1%    /tmp
/tmp/dev           469284    312   468972     1%    /dev
fd                 0         0         0     0%    /dev/fd
# _
```

Filesystems: Solaris EMBEDDED entirely in memory of this x86 based client.

```
# ps -ef
  UID    PID  PPID  C   STIME TTY      TIME  CMD
  root     0     0   0  23:15:24 ?        0:22  sched
  root     1     0   0  23:15:28 ?        0:00  /sbin/init
  root     2     0   0  23:15:28 ?        0:00  pageout
  root     3     0   0  23:15:28 ?        0:00  fsflush
  root    49    10   0  23:15:36 console 0:00  /sbin/sh /sbin/install-recove
ry
  root     7     1   0  23:15:28 ?        0:00  /lib/svc/bin/svc.startd
  root    10     7   0  23:15:29 console 0:00  /sbin/sh /sbin/install-discov
ery
  root     9     1   0  23:15:28 ?        0:01  /lib/svc/bin/svc.configd
  root   196    49   0  23:15:37 console 0:00  /bin/sh /sbin/sulogin
  root    26     1   0  23:15:29 ?        0:00  /usr/lib/sysevent/syseventd -
r /tmp
  root    28     1   0  23:15:32 ?        0:00  /usr/lib/devfsadm/devfsadmd -
r /tmp -p /tmp/root/etc/path_to_inst
  root   201   197   0  23:17:59 console 0:00  ps -ef
  root   197   196   0  23:15:37 console 0:00  /sbin/sh
# uname -a
SunOS 5.11 snv_40 i86pc i386 i86pc
# _
```

Solaris Process Table and version.

Modifying x86.miniroot: Creating a custom operating environment / appliance

```

root#
root# cd /tftpboot/SOLARIS10SXCR_snv49.d
root# cp x86.miniroot x86.miniroot.FCS
root# gzip -dc < x86.miniroot > x86.miniroot.Uncompressed
root# lofiadm -a /tftpboot/SOLARIS10SXCR_snv49.d/x86.miniroot.Uncompressed
root# mount -F ufs -o rw /dev/lofi/1 /mnt
root#
root# cp /mnt/etc/svc/repository.db /mnt/etc/svc/repository.db.FCS
root# svccfg << XxX
repository /mnt/etc/svc/repository.db
list
delete system/install-setup
list
end
XxX
root#
root# cp /mnt/sbin/install-discovery /mnt/sbin/install-discovery.FCS
    
```

Next, replace the contents of the "/sbin/install-discovery" file inside x86.miniroot (the original FCS contents of which we just saved above), with the contents of the following table:

/sbin/install-discovery (replacement contents)

```

#!/sbin/sh
#

PATH=/sbin:${PATH}
PLATFORM=`/sbin/uname -p`
I386="i386"
SPARC="sparc"
ROOT=/tmp/root
LOGS=${ROOT}/var/sadm/system/logs
NETBOOTINFO_CMD=/usr/lib/inet/wanboot/netbootinfo
INTERFACE=`${NETBOOTINFO_CMD} interface-name`
export PATH PLATFORM

init_keyboard()
{
    [ -x /usr/lib/set_keyboard_layout ] && /usr/lib/set_keyboard_layout
    /usr/bin/loadkeys
    [ -h /dev/kbd -a -x /usr/bin/kbd ] && /usr/bin/kbd -i > /dev/null 2>&1
}

#####
# mount tmpfs. For now it can't swap so we can't fill it too full.
#####
/sbin/mount -F tmpfs swap /tmp
if [ ${?} -ne 0 ]
then
{
    echo "tmpfs mount failed."
    /sbin/sh
}
fi

#####
# Unpack writeable initialized files into /tmp.
# NOTE: send output to /tmp (not /dev/null) to
# avoid nfs bug on remote ro mounted fs
#####
    
```

Creating appliance versions of Sun Solaris for embedded applications

```
( cd /tmp/proto; find . -print -depth | cpio -pdm /tmp 2>/tmp/cpio.out )
mkdir -p /tmp/root/var/sadm/system/logs
echo "swap - /tmp tmpfs - no -" >> /etc/vfstab
echo "/proc - /proc proc - no -" >> /etc/vfstab

if [ -f /kernel/fs/dev -o \
    -f /kernel/fs/sparcv9/dev -o \
    -f /kernel/fs/amd64/dev ]
then
{
# Configure /dev filesystem with /tmp/dev as the writable attribute store

echo Configuring /dev

# First populate /tmp/dev
find dev -depth -print | cpio -pdum /tmp >/dev/null 2>&1
/usr/sbin/devfsadm -r /tmp -p /tmp/root/etc/path_to_inst

# Now remount /dev filesystem with /tmp/dev as the attribute store
/sbin/mount -F dev -o remount,attrdir=/tmp/dev /dev /dev

#
# syseventd and devfsadmd need to be started so that lofi can run.
# Startup devfsadmd now that /dev is writable. We manually start devfsadmd
# rather than depend on syseventd since we want to start it up with
# special options.
#
mkdir -p /tmp/etc/sysevent
/usr/lib/sysevent/syseventd -r /tmp
/usr/lib/devfsadm/devfsadmd -a /tmp -p /tmp/root/etc/path_to_inst
}
else
{
# configure devfs in /tmp/dev
#
find dev -depth -print | cpio -pdum /tmp >/dev/null 2>&1
ln -sf /devices /tmp/devices
/sbin/mount -F lofs -O /tmp/dev /dev

_INIT_RECONFIG=set; export _INIT_RECONFIG
mkdir -p /tmp/etc/sysevent
/usr/lib/sysevent/syseventd -r /tmp

# devfsadm needs to be run to cause everything to be linked. This
# causes devfsadmd to be started
/usr/lib/devfsadm/devfsadmd -r /tmp -p /tmp/root/etc/path_to_inst
}
fi
#
echo "fd - /dev/fd fd - no -" >> /etc/vfstab
/sbin/mount -F fd /dev/fd

#####
# load keyboard mappings on x86
# Included from init.d/keymap
#####
if [ ${PLATFORM} = ${I386} ]
then
{
kbdtype=`prtconf -v /devices | sed -n '/kbd-type/{;n;p;}' | cut -f 2 -d \ `
grep -v "setprop kbd-type" /boot/solaris/bootenv.rc > /tmp/bootenv$$
echo "setprop kbd-type '${kbdtype}'" >> /tmp/bootenv$$
cat /tmp/bootenv$$ > /boot/solaris/bootenv.rc # Note: you cannot do a simple mv(1) here.
rm /tmp/bootenv$$

init_keyboard

eval ` /sbin/get_root -t Roottype -b Rootfs /`
```

Creating appliance versions of Sun Solaris for embedded applications

```
echo "${Rootfs} - / ${Roottype} - no ro" >> /etc/vfstab

#
# We are done with devices, create disk to BIOS disk mapping.
# This will be consumed late by bootadm in grub menu creation.
# We do this early because Install/Upgrade will likely break
# disk content comparison.
/boot/solaris/bin/create_diskmap
}
fi

#####
# Configure the logical and network interfaces.
# Through dhcpagent and ifconfig, we will
# configure the interface that was used to
# pxegrub to this initial environment.
#####
/sbin/ifconfig lo0 plumb
/sbin/ifconfig lo0 127.0.0.1 up
/sbin/ifconfig -a plumb > /dev/null 2>&1
/sbin/ifconfig -a up > /dev/null 2>&1
/sbin/dhcpagent -a
#/sbin/ifconfig ${INTERFACE} dhcp primary start wait 120
#####

[ -f /etc/netboot/etc/inet/hosts ] && cat etc/netboot/etc/inet/hosts >> /etc/inet/hosts

#####
# Since sparc still uses the dvd stub miniroot which doesn't contain
# loadkeys, this can't be run when x86 runs it because /usr would not
# have been mounted yet.
#####
[ "${PLATFORM}" = "${SPARC}" ] && init_keyboard

[ "`/sbin/getbootargs`" = "- embedSolaris" ] && exec /sbin/embedSolaris.sh

#####
# now we exit SMF rcS and let it proceed the legacy rcS scripts, and the
# rc files bring the system further on up (ie networking comes up...)
#####
exit 0
```

Next, create the file `"/sbin/embedSolaris.sh"` and populate it with the following contents:

```
#!/usr/bin/sh
#
NETBOOTINFO_CMD=/usr/lib/inet/wanboot/netbootinfo
#
# Note that the "netbootinfo" command does not rely on any interface being up or even configured. It extracts the DHCP
# information it provides (about the DHCP parameters involved during last DHCP boot) from a location within the system
# (perhaps in a file or in memory) that stores such information for later retrieval. On the contrary, "dhcpinfo" probes the
# network DHCP server for such information, thus requiring the loopback and primary interfaces to be configured and up.
#
INTERFACE=` ${NETBOOTINFO_CMD} interface-name`
BOOTSVRV=` /sbin/dhcpinfo -i ${INTERFACE} bootsrva`
HOSTNAME=` /sbin/dhcpinfo -i ${INTERFACE} hostname`
/usr/bin/uname -S ${HOSTNAME}
PS1="root@${HOSTNAME}#"
#
/usr/bin/tftp << eofXXeof > /dev/null 2>&1
connect ${BOOTSVRV}
get /${HOSTNAME}.d/cfgCmds /tmp/cfgCmds
quit
eofXXeof

/sbin/sh /tmp/cfgCmds
exit 0
```

Note: `chmod 554 /sbin/embedSolaris.sh`

Creating appliance versions of Sun Solaris for embedded applications

```
#!/bin/sh
#
RAMDISK_SIZE=
RAMDISK_FILE=
RAMDISK_FILE_MNTPT=(use one of the 'volatile' directories

mkfile ${RAMDISK_SIZE} "${RAMDISK_FILE}"
LOFIDEV=`lofiadm -a "${RAMDISK_FILE}"`

newfs -m 1 ${LOFIDEV} < /dev/null 2> /dev/null
mkdir "${RAMDISK_FILE_MNTPT}"
mount -F mntfs mnttab /etc/mnttab > /dev/null 2>&1
mount -o nologging ${LOFIDEV} "${RAMDISK_FILE_MNTPT}"

cd "${RAMDISK_STAGE_DIR}"
find . -print 2> /dev/null | \
    cpio -pdm "${RAMDISK_FILE_MNTPT}" 2> /dev/null

umount "${RAMDISK_FILE_MNTPT}"
lofiadm -d "${RAMDISK_FILE}"
rm -rf "${RAMDISK_FILE_MNTPT}"
gzip -c "${RAMDISK_FILE}" > "${RAMDISK_STATE_DIR}/x86.miniroot.$$"
```

Things I need to incorporate:

- "ifconfig ifName start dhcpd" (which starts dhcpagent -a).
- Once the interface is configured, do a dhcpinfo to get the IP of the dhcpserver (which is also the initialization server), and tftp or wget over the commands file that will customize this machine.
- The commands file will be specific to the server by MAC/HOSTNAME/IP, and will bring over the minimalist filesystem and app that defines this appliance; and finalize the boot.
- The boot and complete configuration of this machine (including format/fdisk and newfs of the drives should take no longer that 80 seconds (i.e. bare metal to running app).

To Be Continued ...

Creating appliance versions of Sun Solaris for embedded applications

`/usr/lib/inet/wanboot/netbootinfo param [param ...]`

(where param is any of the strings in "quotes" to the right below).

(extracted from bootinfo.h)

```
#define BI_NET_CONFIG_STRATEGY          "net-config-strategy"
42 #define BI_HOST_IP                    "host-ip"
43 #define BI_SUBNET_MASK                 "subnet-mask"
44 #define BI_ROUTER_IP                  "router-ip"
45 #define BI_HOSTNAME                   "hostname"
46 #define BI_HTTP_PROXY                 "http-proxy"
47 #define BI_CLIENT_ID                  "client-id"
48
49 #if defined(_BOOT)
50 #define BI_NETWORK_BOOT_FILE          "network-boot-file"
51 #define BI_BOOTFILE                   "bootfile"
52 #define BI_BOOTP_RESPONSE             "bootp-response"
53 #define BI_BOOTSERVER                 "bootserver"
54 #define BI_AES_KEY                    "aes"
55 #define BI_3DES_KEY                   "3des"
56 #define BI_SHA1_KEY                   "sha1"
57 #else
58 #define BI_SYSIDCFG                    "sysidcfg"
59 #define BI_JUMPSCFG                    "jumpscfg"
60 #define BI_ROOTFS_TYPE                 "rootfs-type"
61 #define BI_INTERFACE_NAME             "interface-name"
62 #endif /* defined(_BOOT) */
```

```
/sbin/getbootargs
```

```
# Set up local loopback interface. (dhcpageant(1M) which we will run
# requires this to internally communicate with ifconfig(1M) and
# dhcpinfo(1)).
#
/sbin/ifconfig lo0 plumb
/sbin/ifconfig lo0 127.0.0.1 up
/sbin/ifconfig -a plumb > /dev/null 2>&1
/sbin/ifconfig -a up
/sbin/dhcpageant -a
```

To get information from the DHCP server regarding this nascent tiny server which we are dynamically building, we can query it via the `/usr/lib/inet/wanboot/netbootinfo` command as shown below, or via the `dhcpinfo` command, also demonstrated below:

```
/usr/lib/inet/wanboot/netbootinfo \  
    interface-name \  
    hostname \  
    host-ip \  
    subnet-mask \  
    router-ip \  
    net-config-strategy
```

```
/sbin/dhcpinfo -i e1000g0 BootSrvA (see dhcp_inittab(4) and /etc/dhcp/inittab for a
list of parameters you can query for with dhcpinfo(1). BootSrvA, specified above, is
and example of one of them).
```

Creating appliance versions of Sun Solaris for embedded applications

```
/sbin/ifconfig ${IFACE} dhcp primary start wait 120
if [ $? -eq 0 ] ; then
    echo Success
    break
else
    # No luck, give up on this interface
    /sbin/ifconfig ${IFACE} dhcp drop
Fi
```

Creating appliance versions of Sun Solaris for embedded applications

```
root# mount -o remount,rw /
```

```
#!/bin/sh
#
NETBOOTINFO_CMD=/usr/lib/inet/wanboot/netbootinfo
#
# Note that the "netbootinfo" command does not rely on any
# interface being up or even configured. It extracts the
# DHCP information it provides (about the DHCP parameters
# involved during last last DHCP boot) from a location
# within the system (perhaps in a file or in memory) that
# stores such information for later retrieval.
#
# On the contrary, "dhcpinfo" probes the network DHCP
# server for such information, thus requiring the
# loopback and primary interfaces to be configured and
# up.
#
INTERFACE=`${NETBOOTINFO_CMD} interface-name`
HOSTIP=`${NETBOOTINFO_CMD} host-ip`
#
/sbin/ifconfig lo0 plumb
/sbin/ifconfig lo0 127.0.0.1 up
/sbin/ifconfig -a up
/sbin/dhcpagent -a; sleep 5
/sbin/ifconfig ${INTERFACE} dhcp primary start wait 120
#
BOOTSRVR=`/sbin/dhcpinfo -i ${INTERFACE} bootsrva`
HOSTNAME=`/sbin/dhcpinfo -i ${INTERFACE} hostname`

/usr/bin/tftp << eofXXXeof
connect ${BOOTSRVR}
get /${HOSTNAME}.d/cfgCmds /tmp/cfgCmds
quit
eofXXXeof

/sbin/sh /tmp/cfgCmds
```

This script will be embedded into x86.miniroot and made to run during boot. It will most likely be pre-pended to /sbin/install-discovery so as to avoid having to tweak the default smf services (keeping things portable).

The client specific "cfgCmds" bourne/korn shell script will contain customization instructions (and also create zpool/zfs filesystems to transfer binaries and configurations to).

Creating appliance versions of Sun Solaris for embedded applications

```
#!/usr/bin/sh

PATH=/usr/bin:/usr/sbin:/usr/ucb

# Define ${VDISK}. This will be dynamically determined as this script is
# enhanced.
#
VDISK=c1t0d0

mkdir -p /tmp/root/mntpt
zpool create -f -R /tmp/root -m /mntpt mypool ${VDISK}
zfs create mypool/FS2.d
zfs create mypool/FS1.d
```