Solaris Jumpstart Basics

Hal Pomeranz
Deer Run Associates

Hal Pomeranz * Founder/CEO * *hal@deer-run.com*
Deer Run Associates * PO Box 20370 * Oakland, CA 94620-0370
+1 510-339-7740 * *http://www.deer-run.com/*

Wouldn't It Be Great If..?

- Adding a new machine were as simple as setting up the hardware?
- Machines automatically customized themselves to their environment?
- Broken systems could be swapped out quickly and with low admin overhead?
- You could upgrade your network (or do patch installs) simply by rebooting?

If you run a network of more than a dozen or so Sun workstations, you're probably spending an inordinate amount of time installing systems, upgrading systems, applying patches, etc. It may seem like you don't even get done with one round of upgrades before you need to start thinking about the next one. You might be in a situation where all of your systems have slightly different configurations depending on when they were installed and who set them up. None of these situations is desirable.

Wouldn't it be great if you could create a single system image for all of your machines and upgrade that image across your entire network just by rebooting? Well, you can…

What is Jumpstart?

- Mechanism for "one-button installs" from central server
- Simultaneously supports multiple system configurations and OS versions
- Extensible to allow automatic local customizations

The Jumpstart mechanism was developed by Sun in order to simplify installations on large networks of mostly similar hardware. The basic idea is that system configuration information is stored on a central server. When new clients are added to the network, the client boots from the central configuration server and then runs an automated install program which partitions the client's disk(s), installs the operating system, and makes appropriate local configuration changes (setting network parameters, hostname, etc.). Creating the configuration server requires a fair amount of System Administration expertise, but adding clients can then be accomplished by completely untrained technicians– this is called "leveraging key employees".

The Jumpstart process also allows the local administrator to create custom scripts that are run either before ("pre-install") or after ("post-install") the Jumpstart process (or both). This allows sites to create even more finely customized install routines for their particular site. More on pre- and post-install scripts in the last section of this talk.

Note that Jumpstart works for both Sparc and Intel-based systems. However, Intel-based machines don't have the appropriate boot ROM code to do a network boot, so the administrator must create a "boot floppy" to be used during the initial install. A single Jumpstart server may support booting clients from multiple different hardware platforms and/or operating system revisions. For example, the author has a single Jumpstart server at home which is capable of booting machines on any OS release from 2.5.1 through Solaris 8.

References for Further Reading

- Solaris *Advanced Installation Guide* (Chapters 6 through 11)
- Hal's jumpstart info page:
  `www.deer-run.com/~hal/jumpstart/`

Sun's primary reference for Jumpstart configuration is the *Solaris Advanced Installation Guide* which may be found on the Web at

```
http://docs.sun.com/ab2/coll.214.7/
    SPARCINSTALL/@Ab2PageView/6302
```

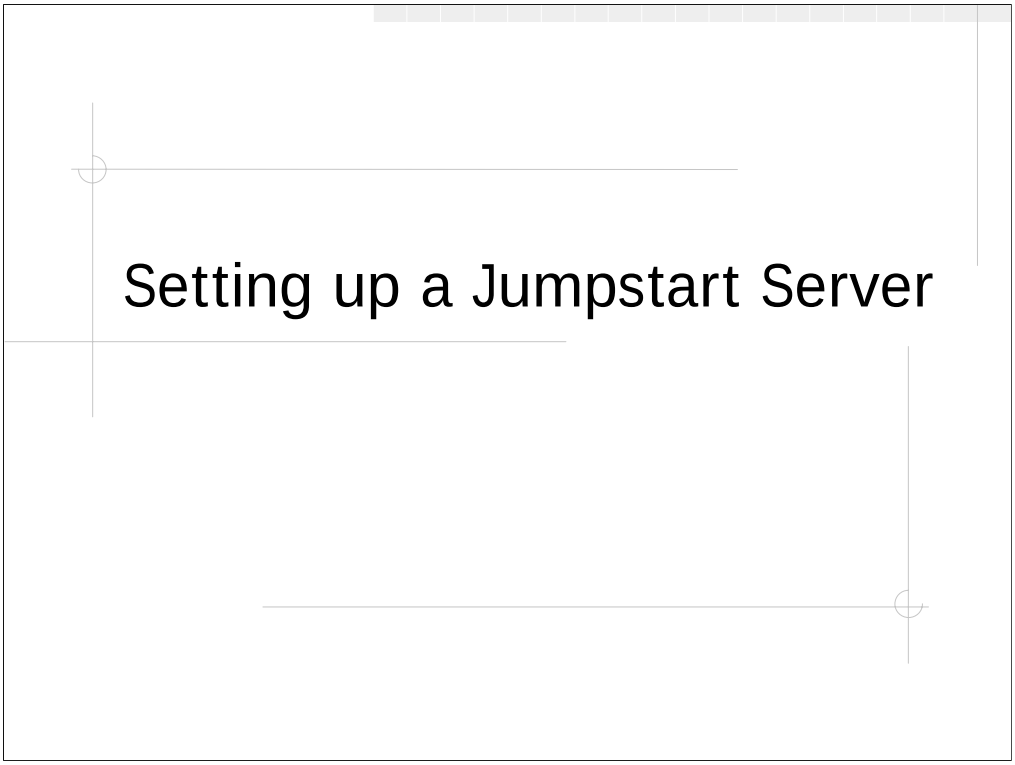
Note that the above URL has been broken in the middle for readability– feed it to your Web browser as a single long line.

The *Advanced Installation Guide* spends about 125 pages talking about Jumpstart, while this presentation covers much of the same material in about 30 slides. Needless to say, some detail is lost. It's a good idea to print out the relevant chapters from the *Advanced Installation Guide* and keep them around as a reference.

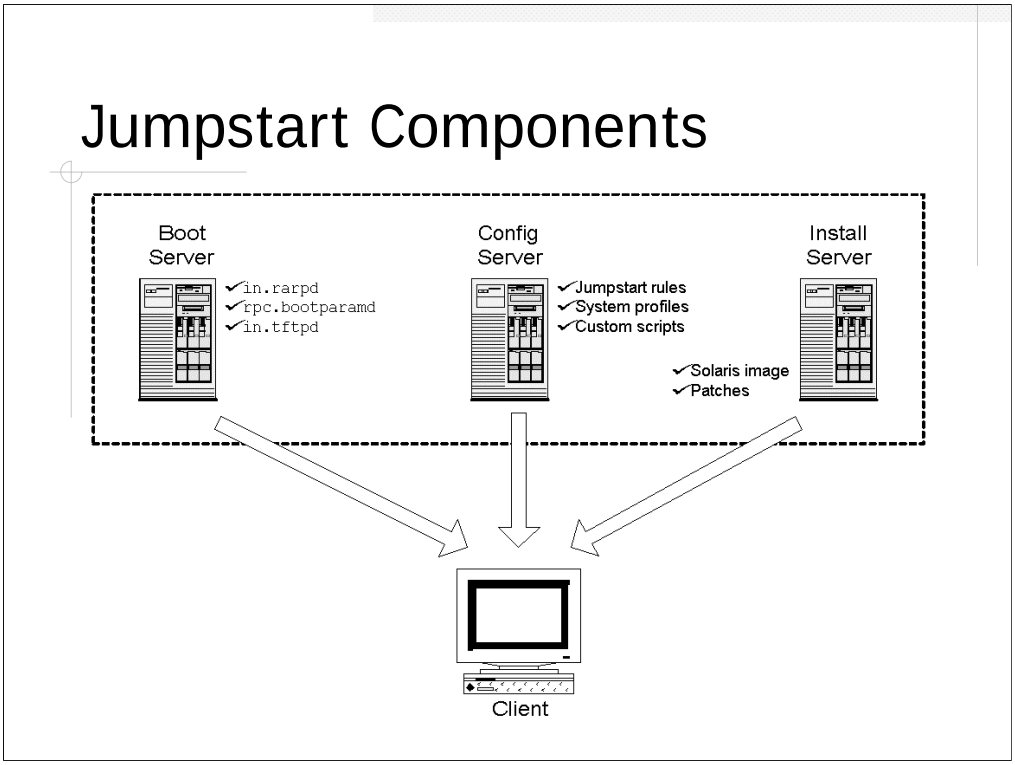
Electronic versions of this presentation, plus helpful scripts and other Jumpstart-related tools and information can be found at

```
http://www.deer-run.com/~hal/jumpstart/
```

Hopefully this information will be updated on a regular basis, so you may want to check back periodically.



This section covers the "quick and dirty" procedure for getting your first Jumpstart server on the network. Preparing for individual client installs will be covered in the next section.



When we talk about a "Jumpstart Server", we are really talking about three different processes. When a client is booting under Jumpstart, it first needs to contact a *Boot Server* so that the client can set its basic network parameters (via RARP and `bootp`) and download the network boot code (via TFTP). The client actually boots off a Solaris image stored on the *Install Server*– the boot image is mounted on the client via NFS. Once the client is booted, it has to look up configuration information from the *Configuration Server*. The client then proceeds to run the Jumpstart install program and load the OS from another directory on the Install Server.

For the rest of this talk, we'll be assuming that we have a single machine which is to be the Boot Server, the Install Server, and the Configuration Server. However, on large networks it may be advisable to split these functions across multiple machines. For example, the `bootp` protocol is LAN-based and you generally need a Boot Server on each of your networks (though you can finesse this by using proper router configuration). If you are planning on Jumpstarting a large number of machines simultaneously, you may want to deploy multiple install servers because they can quickly become saturated if multiple clients are being built in parallel.

Overview of Steps

1. Create install and configuration dirs
2. Copy OS media to install directory
3. Copy scripts to configuration directory
4. Create `sysidcfg` file in install dir
5. Create `/tftpboot`
6. Start system daemons

In order to build your Jumpstart server, you will need a copy of the OS media for each of the OS platforms you wish to support. The install server will require about 500MB of space (750MB for Solaris 8) per OS image.

Note that the Boot Server must run `in.rarpd`, `rpc.bootparamd`, and `in.tftpd` (via `inetd`). The Install and Configuration servers will be needing to share file systems via NFS with the client machines.

Step 1: Create Install/Config Dirs

```
# mkdir –m 755 /export/jumpstart /export/jump_5.8
# chown root:root /export/jumpstart /export/jump_5.8
# cat >>/etc/dfs/dfstab
share –F nfs –o ro,anon=0 /export/jumpstart
share –F nfs –o ro,anon=0 /export/jump_5.8
^D
# shareall
```

First the administrator must create the install directories (one per supported OS version) and the configuration directory (only one of these no matter how many OS versions you plan to support). If you're planning to split the Configuration Server and the Install Server onto separate machines, then the install directories belong on the Install Server and the configuration directory lives on the Configuration Server.

The directories may be located anywhere in the file system and may be given any name. However, the author recommends that install directories be named something like `jump_<osvers>` in order to make automatic scripting easier. This will save you lots of time when creating customized pre- and post-install scripts. For the rest of this talk, we will assume the naming conventions used above.

Once the directories are created, they must be shared via NFS to the rest of the network. Note that the file systems are being shared read-only (that's the "`ro`" option above) but that all machines on the network are being given anonymous root access to the file systems (any machine which can access the Jumpstart server can mount these file systems and remotely read any file with root privilege). The administrator may choose to restrict root privilege to only the clients being installed (see the `root=` option on the `share_nfs` manual page), but managing access on a per-machine basis can be difficult across a large network.

Step 2: Copy OS Media

- Install OS:

```
# mount –r –F hsfs /dev/dsk/c0t2d0s0 /mnt
# cd /mnt/Solaris_8/Tools
# ./setup_install_server /export/jump_5.8
```

- Solaris 8 has a second OS disk:

```
# cd /
# eject cdrom
# mount –r –F hsfs /dev/dsk/c0t2d0s0 /mnt
# cd /mnt/Solaris_8/Tools
# ./add_to_install_server /export/jump_5.8
```

Once the install directory has been created, the OS must be read off CD-ROM and placed in the directory. If the volume manager is running on your Jumpstart server, then the OS media will be mounted automatically. Otherwise, the OS CD-ROM must be mounted manually using commands similar to those shown above (although the actual disk device corresponding to your CD-ROM drive may vary from system to system).

Once the CD-ROM is mounted, navigate over to the `Tools` directory and run the `setup_install_server` script, specifying the install directory name you created and shared in the previous step. Note that this script takes an inordinate amount of time to run, so go get coffee while the install directory is being created. This process must be repeated for every OS version you want to support on your Jumpstart server.

Solaris 8 (and probably future versions of Solaris) now comes on two CD-ROMs. After the first CD-ROM has been installed, mount the second CD-ROM and run the `add_to_install_server` script to complete the Solaris 8 install directory.

Digression: Know Your Install Dir

- Interesting stuff is in **`/export/jump_5.8/Solaris_8`**
- **`Misc/jumpstart_sample`** contains sample configs and scripts
- **`Tools`** contains scripts for adding clients
- **`Tools/Boot`** is boot image for clients
- **`Product`** directory contains Solaris packages which will be installed

Before we continue setting up our Jumpstart server, it's worthwhile to review the contents of the install directory that was created in the last step. Underneath your install directory will be a directory named `Solaris_<vers>` (e.g., `Solaris_2.6` or `Solaris_8`). This directory is where all of the interesting components live.

The `Misc/jumpstart_sample` directory contains a sample configuration directory which you can use to base your own client configurations on. The Sun documentation recommends just copying the entire contents of the `jumpstart_sample` directory to your configuration directory, but we are going to be more selective. The `Tools` subdirectory contains the `add_install_client` script which is necessary when adding client configuration information to a Boot Server (more on this later).

`Tools/Boot` is a complete copy of the Solaris OS, and is the directory which the clients use for NFS booting when they are first being booted off the network. Note that this directory is the unpatched Solaris image off the CD-ROM, so you may want to patch this version of the OS for security (use `patchadd –C <dir>` to patch the install directory image). The `Product` directory contains all of the Solaris OS packages from the CD-ROM (thus, the install directory really contains two distinct copies of the Solaris OS)– these are the packages which will be installed onto the client machine by the Jumpstart process. You could add your own local packages to the `Product` directory if desired (more on how to specify installed packages in the next section).

Digression (cont.)

- **`Patches`** subdirectory contains patches to install during jumpstart
- An **`MU`** directory may also exist which contains maintenance update patches
- Patches are installed in order based on when they were added to directory
- This is almost never what you want...

The install directory also contains a `Patches` subdirectory– patches stored here will be automatically installed by the Jumpstart process. However, the Jumpstart process doesn't use the same `patch_order` file functionality that the Sun Recommended Patch Clusters use to ensure patches get installed in the proper dependency order. Instead, the Jumpstart process just installs patches based on the timestamp on the patch subdirectory in the `Patches` directory (i.e, when the patch was added to the `Patches` directory). This is just terrible behavior because it means that patches are often installed in the incorrect order, which can actually cause the patching process to abort.

Note that the install directory may (or may not) contain an `MU` directory. `MU` stands for "Maintenance Update", and various releases of an operating system may include Maintenance Updates which either add support for new hardware, add functionality, and/or fix bugs in the original release (generally called the "First Customer Ship" or "FCS release"). The `MU` directory contains lots of files and some `README`s about the contents of the update, but ultimately the update is really just another collection of Sun patches which will be installed by the Jumpstart process out of one of the subdirectories of the `MU` directory.

Patches vs. Jumpstart

- Lack of **`patch_order`** file really hurts jumpstart package install functionality
- Installing lots of patches slows down jumpstart significantly
- *Recommendation:*
  - Remove contents of **`Patches`** directory (and totally remove **`MU`** directory, if any)
  - Install Sun recommended patch cluster as part of local post-install process

If you've ever installed the Sun Recommended Patch Cluster on a machine, you know that it can take longer to install patches than to install the basic OS. It's also probably the case that you will be installing the Sun Recommended Patch Cluster as part of your local custom post-install process, because the Recommended Patch Cluster is a superset of the patches that come off the OS CD-ROM. Aside from performance issues, the fact that the Jumpstart process doesn't obey any sort of `patch_order` file, makes using Jumpstart to install patches not the way to go. So, once the install directory has been created, simply go ahead and remove all patches from the `Patches` directory in the install area.

It's your call whether or not to keep the Maintenance Update directory. On the one hand, the update may add useful functionality or fix critical bugs for your platform (on the other hand, the update may have no impact at all on your platform). However, installing the update will make the Jumpstart take longer on each client, even if the update doesn't apply to that client platform. Look at the documentation which comes with the update and decide for yourself whether or not you want to install it.

Exception to the Recommendation

- There's a bug in the Solaris 7 autoconf routines from CD-ROM
- As a fix, patch 106978 must be installed by the Jumpstart
- Make sure a recent version of this patch exists in your **`Patches`** directory

If you are creating a Solaris 7 install directory, however, it is critical that the Patches directory contains a recent copy of Sun Patch ID 106978. This patch fixes bugs in the auto-configuration routines which are required for the client machines to boot fully unattended.

Step 3: Copy Scripts to Config Dir

- We definitely need the **check** script:

```
# cd /export/jump_5.8/Solaris_8
# cd Misc/jumpstart_sample
# mkdir –p –m 755 /export/jumpstart/bin
# cp check /export/jumpstart/bin
# chmod 755 /export/jumpstart/bin/check
# chown –R root:root /export/jumpstart/bin
```

- You may want to look at sample configuration files in this directory...

With the install directory created and properly configured, we now want to set up our configuration directory. We need a copy of the `check` script from the `Misc/jumpstart_sample` directory. Note that if your Jumpstart server supports multiple OS revisions, make sure to use the `check` script from the latest supported OS release. Thus, if your system boots both Solaris 7 and Solaris 8 clients, grab the `Solaris_8/Misc/jumpstart_sample/check` script.

The `jumpstart_sample` directory also contains some sample client configuration files. It may be useful to review these sample files after hearing the information in the next section of this talk.

Step 4: The `sysidcfg` File

```
system_locale=en_US
timezone=US/Pacific
timeserver=localhost
terminal=xterms
network_interface=PRIMARY \
   {netmask=255.255.255.0 protocol_ipv6=no}
name_service=DNS \
   {domain_name=deer-run.com name_server=192.168.1.2}
security_policy=NONE
root_password=papAq5PwY/QQM
```

The information in the `sysidcfg` file is used by clients to set various system parameters during the Jumpstart process and when the client reboots for the first time. The format of this file is (slightly) OS version-dependent but shouldn't vary from client to client, so it's probably easiest to locate the file at the top of the install directory (`/export/jump_5.8/sysidcfg` in our example).

It is important to note that the `sysidcfg` file contains a copy of the client's encrypted root password entry for `/etc/shadow`, so the file should certainly be mode `400` and owned by `root`. Recall, however, that the install directory was exported with anonymous root access, so anybody on another machine could mount the install directory and read the file. Make sure that the client systems have a different root password from all of your other machines! Note also that a copy of the `sysidcfg` file is retained in the client's `/etc` directory– you probably want to delete this file after the first reboot.

Other parameters in the file include the system's default locale and time zone (consult the *Advanced Installation Guide* for more info). The netmask of the primary network interface can be specified as well as name service parameters (NIS is supported as well– see the *Advanced Installation Guide*). Note that the `protocol_ipv6` and `security_policy` options are supported only under Solaris 8– delete these options and the above file is appropriate for Solaris 7 (see next slide for note on Solaris 2.6). The `security_policy` option is not documented in the manual pages or the *Advanced Installation Guide*– see instead

http://www.sun.com/software/solutions/blueprints/0300/sysidcfg.pdf

Solaris 5.6 `sysidcfg` File

```
timezone=US/Pacific
timeserver=localhost
terminal=xterms
network_interface=hme0 {netmask=255.255.255.0}
name_service=NONE
root_password=papAq5PwY/QQM
```

The Solaris 2.6 `sysidcfg` file is considerably more primitive than the Solaris 7 and 8 versions (`sysidcfg` is not supported prior to Solaris 2.6, meaning Jumpstarts for Solaris 2.5.1 and earlier require some administrator intervention during the boot process).

In particular, note that DNS is *not* supported for the `name_service` parameter– you will have to manually configure DNS after the system boots or during the post-install phase. Also note that the primary interface for the system must be explicitly specified. This means you can't use the same `sysidcfg` file for systems that have `le0` interfaces (older microSparc-based machines).

Step 5: Create `/tftpboot`

- Create the directory

```
# mkdir –m 711 /tftpboot
# chown root:root /tftpboot
```

- You'll also need to uncomment the right line in **`/etc/inet/inetd.conf`**

The Boot Server must have a `/tftpboot` directory. Aside from being the location for the network boot code for the client machines, the presence of the `/tftpboot` directory is what triggers the Boot Server to start the `in.rarpd` and `rpc.bootparamd` processes at boot time.

However, in order for the system to service TFTP requests, the administrator must also uncomment the appropriate line in `/etc/inet/inetd.conf` and send a `HUP` signal to the running `inetd` process (or reboot the system). The line you're looking for in `inetd.conf` is

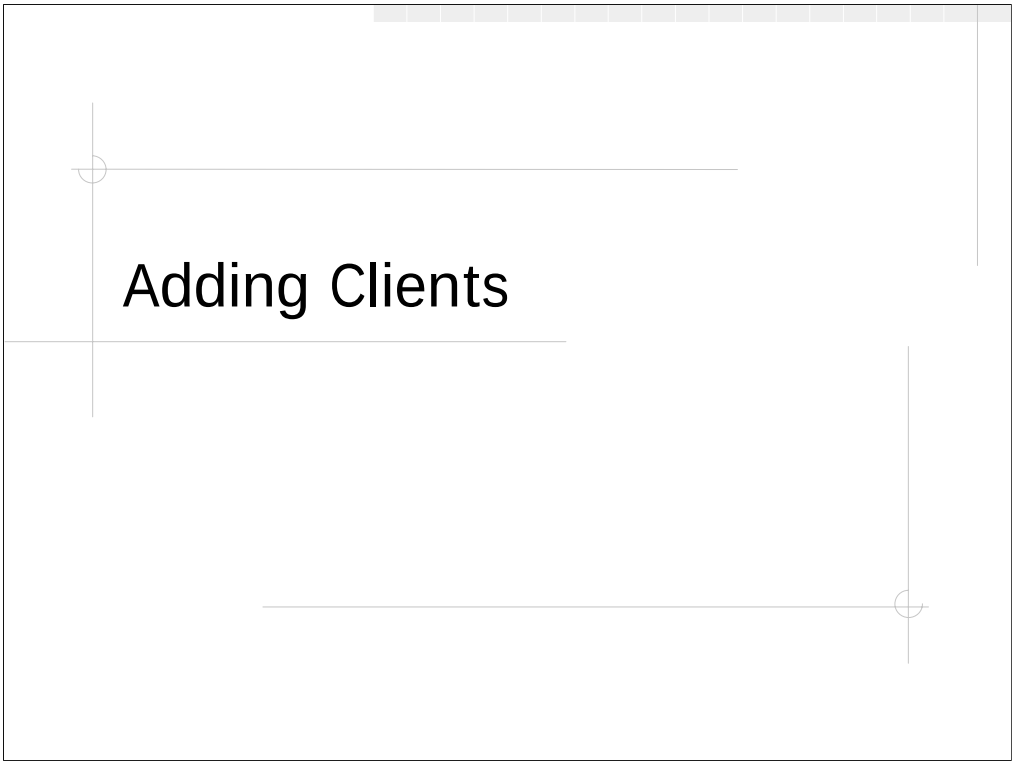
```
#tftp   …  /usr/sbin/in.tftpd   in.tftpd -s /tftpboot
```

Step 6: Start System Daemons

- A bunch of daemons to (re)start:
  - NFS daemons
  - **`in.rarpd`**
  - **`rpc.bootparamd`**
  - **`inetd`**
- Rebooting the jumpstart server is probably easiest...

Once all directories are created and configured, the administrator needs to make sure that all of the appropriate daemons are running. The Installation and Configuration Servers require that the NFS server processes (`mountd`, `nfsd`, `statd`, and `lockd`) all be running. The Boot Server must be running `in.rarpd` and `rpc.bootparamd` and have TFTP properly configured in `inetd.conf` and have the `inetd` process running.

Frankly, assuming that our Jumpstart server has no other particular duties, the easiest thing is to just reboot the system. Assuming all of the directories and configuration files are in good order, all of the necessary daemons should be started automatically by the boot process.



With the server properly configured, it's time to turn our attention to creating individual client configurations and updating our Jumpstart server to allow particular clients to boot.

Steps for Adding a Client

1. *Create client profile*
2. *Create pre- and post-install scripts*
3. *Update* ***/export/jumpstart/rules***
4. *Run* ***check*** *script*
5. Add **`ethers`** and **`hosts`** information
6. Run **`add_install_client`** script
7. Reboot client machine

The administrator must complete the seven steps listed above before a client can be successfully Jumpstarted. However, the first four steps generally do not need to be performed for every client– the administrator can usually create a small number of client profiles and pre- and post-install scripts which will suffice for a large number of machines. More on this in the next few slides.

Step 1: The Client Profile

```
install_type    initial_install
system_type     standalone

cluster         SUNWCprog
package         SUNWaccr        add
package         SUNWaccu        add

partitioning    explicit
filesys         c0t3d0s0         512    /
filesys         c0t3d0s1        2048    /var
filesys         c0t3d0s2         all    overlap
filesys         c0t3d0s3        2048    swap
filesys         c0t3d0s4        1024    /usr
filesys         c0t3d0s5        free    /local
```

The client profile file is used to describe how individual machines should be configured. Generally speaking, the client profile describes how the system's disk(s) should be partitioned and which OS software packages should be loaded on the machine– machines with similar disk partitioning and OS configurations can use the same profile file (even if those machines are running different OS revisions).

Each profile file must begin with the `install_type` directive– `initial_install` means blow away everything on the disks and start from scratch, but `upgrade` is another possibility (see the *Advanced Installation Guide* for more info). Various `system_type` choices exist– `standalone` means a machine with a full OS install on the system's local disks (probably the most common configuration in these days of large disk drives).

Next the administrator specifies which OS cluster should be installed. Cluster choices are `SUNWCreq` (aka the *Core System Support* cluster), `SUNWCuser` (*End-User* cluster), `SUNWCprog` (*Developer* cluster), and `SUNWCall` (Every OS package). Packages may then be added *or deleted* from the cluster by using `package` directives.

Administrators may specify the exact disk partitioning using `partitioning explicit` (as opposed to having the Jumpstart do an automatic partitioning which is usually sub-optimal). Partition sizes are in megabytes. Note that the size of the last partition is listed as `free` which means that this partition consumes any remaining unallocated space. When configured carefully, the same partition table can work even on disks of unequal sizes!

Step 2: Pre- and Post-Install Scripts

- Always strictly optional
- Careful! New system's disks are mounted on **`/a`** during jumpstart
- Script output automatically saved to **`/a/var/sadm/system/logs`**
- More on all of this in a later section...

The administrator may optionally create pre- and post-install scripts. The pre-install script runs before the system profile file is read and executed (i.e., before the system's disk drives are repartitioned and the new OS image loaded). This means that pre-install scripts are an excellent place to back up various files from the original system (e.g., configuration files under `/etc`, log files, SSH host keys, et al). Note that the pre-install script will have to explicitly mount (and unmount) the file systems from the system's local drives, because the local file systems won't be mounted at the time the pre-install script runs.

By the time the post-install script runs, the new file local file systems will be created and the OS will have been loaded. Note, however, that the new file system created on the system's local drives will be mounted with the local root file system at `/a`, so make sure the post-install script follows the proper indirection. Post-install script are a good place to do local system customization and restore files that were backed up by the pre-install script.

The output of the pre- and post-install scripts can be found in the `/var/sadm/system/{begin,finish}.log` files on the new system once the Jumpstart is completed and the new system has rebooted.

Step 3: The `rules` File

- Format of entries are:

  <match rule> <pre-inst> <profile> <post-inst>

- Sample File:

```
hostname srvr1.deer-run.com \
        -     srvr1.prof          bin/make-serv.sh
network 192.168.10.0  -  eng.prof  bin/enghost.sh
network 192.168.128.0 && karch sun4m \
        -     old-sup.prof        bin/sup-tools.sh
network 192.168.128.0 \
        -     sup.prof            bin/sup-tools.sh
any     -     generic.prof        bin/do-patch.sh
```

The purpose of the `rules` file is to associate profile file and pre- and post-install scripts with a particular machine or group of machines. Entries are searched in order until the match criteria in the first column fits the client being booted; that rule is then executed ("first match and exit" behavior).

Each rule is a single line, but lines may be continued using "\" as shown above. Pre- and post-install scripts may be omitted by putting a "-" in the appropriate column (actually, as we'll see in the next section, even the profile can be omitted in some cases). Comments are allowed if prefixed with "#". Script names and profile file names are relative to the top of the Jumpstart configuration directory.

Match criteria cover a wide variety of different system parameters, and not just the simple criteria shown above. For a complete list, see the *Advanced Installation Guide*. Note that logical operations (and, or, not) are supported.

The first line above shows an example of a rule for a particular machine. Generally, however, rules apply to a group of machines (a network or particular hardware type which should all be configured identically) as we see in later rules. The third and fourth lines above take advantage of the "first match and exit" behavior to configure older microSparc machines using one profile and newer (probably UltraSparc) machines using another. However, both classes of machines use the same post-install script. The last line is a catch-all or default entry for machines which don't match any of the previous rules. It may be dangerous to allow any random machine which connects to your network to Jumpstart from your server– you may not wish to include a default rule in your file.

Step 4: Run `check` Script

- Script should be run each time the **`rules`** file is updated
- Script checks the syntax of profiles and verifies that scripts exist
- As a side-effect, creates the **`rules.ok`** file for jumpstart process

The `check` script that we copied from the `jumpstart_sample` directory is used to validate and pre-process the `rules` file. The `check` script verifies the syntax of the profile files that are listed in all rules, and checks that the listed pre- and post-install scripts exist (but doesn't check script syntax). More importantly perhaps, the `check` script creates the `rules.ok` file which is the file that is actually consulted during the Jumpstart (the `rules` file itself is only used by the administrator and the `check` script).

Again, if your Jumpstart server is providing configurations for several different OS revisions, make sure to use the `check` script from the most recent Solaris version (some profile entries in newer versions of Solaris are not backwards compatible with older `check` scripts).

Step 5: Update Host Info

- **`/etc/ethers`** should contain the MAC address and FQDN of the host
- Also make sure the jumpstart server can resolve the name of the host
- If you're using hosts files or NIS/NIS+, list the FQDN first

Each time a new host is added to the Jumpstart network, the Boot Server needs to be updated. The ethernet (MAC) address and hostname of the machine need to be added to the Boot Server's `/etc/ethers` file. The machine's ethernet address is displayed in the Sun banner when the system boots, and is also available on running systems by running `ifconfig` (as root) and/or from the packing slip which comes with each new machine.

The Boot Server also needs to be able to resolve the machine's IP address, either from its own `hosts` file or from NIS/NIS+ or DNS, depending on how the Boot Server is configured. This will mean updates on either the Boot Server machine itself or on your name server.

Generally, it's good policy to use the fully qualified domain name (FQDN) form for all entries in the `/etc/ethers`, `/etc/inet/hosts`, and even in the `rules` file in the Jumpstart configuration directory (for `/etc/inet/hosts` list the FQDN first followed by the unqualified form). Being consistent throughout will save a lot of headaches down the road.

Step 6: **add_install_client**

```
# cd /export/jump_5.8/Solaris_8/Tools
# ./add_install_client \
     -c jumpsrvr:/export/jumpstart \
     -p jumpsrvr:/export/jump_5.8 \
     -s jumpsrvr:/export/jump_5.8 \
     sun01.deer-run.com sun4u
```

Once all of the client information has been updated on the Boot Server, the administrator needs to run the `add_install_client` script. This script is found in the `Tools` directory in the appropriate install directory for the version of Solaris that you want the client to run. Make sure you use the correct `add_install_client` script! The `add_install_client` script is responsible for placing the appropriate boot code and symbolic links in `/tftpboot` to allow the client machine to boot. `add_install_client` also updates the `/etc/bootparams` file used by `rpc.bootparamd`.

The `-c` flag specifies the server name and path for the Jumpstart configuration directory (the server name here would be your Configuration Server). The `-s` flag specifies the Install Server and pathname to the install directory. `-p` is the location of the `sysidcfg` file (don't use this option for Solaris releases prior to 2.6), which we've stored at the top of our install directory. You also need to specify the name of the machine (again use the FQDN here) and the kernel architecture of the client (this is the output `uname -m` on the client host).

A Better Way

- Too much typing!
- Lots of redundant information
- OS version dependent
- How about this instead:

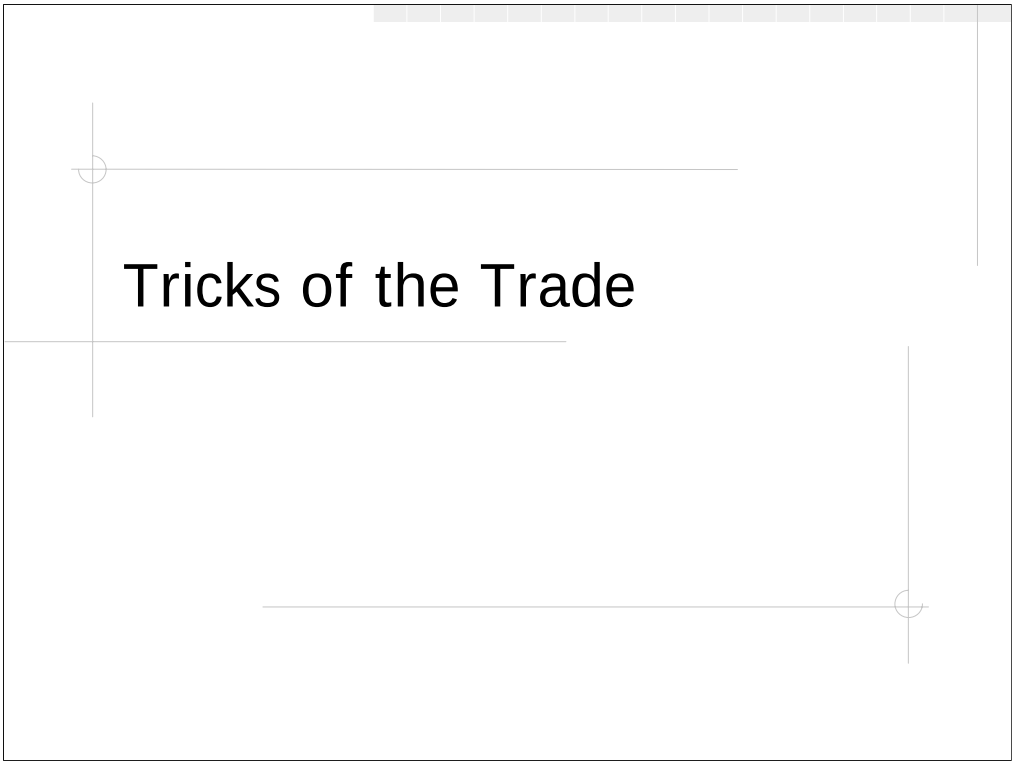
```
# cd /export/jumpstart/bin
# ./add_client sun01.deer-run.com sun4u 5.8
```

Frankly, the `add_install_client` requires way too much (redundant) typing, and forces the administrator to get to the correct install directory to run the right version of the script. You'll find a simpler `add_client` script at the `http://www.deer-run.com/~hal/jumpstart/` site. Once you've downloaded the script, edit the file and make sure the `CONF_SERVER`, `CONF_DIR`, `INST_SERVER`, and `INST_ROOT` variables are set appropriately for your server. Note that the `add_client` script assumes that the install directories are `$INST_ROOT/jump_<osvers>` (i.e., the conventions we've been using in this talk).

Step 7: Reboot Client

```
ok boot net - install
```

Once the Boot Server setup is completed, boot the client system as shown above. Note that the boot line is "`boot` <*space*> `net` <*space*> `-` <*space*> `install`". The most common error is to type "`-install`" as a single final argument, but then the Jumpstart won't proceed.



With the basic Jumpstart configuration procedure out of the way, it's time to look at some tips and tricks for writing pre- and post-install scripts. We'll also discuss why and how to bypass the "normal" Jumpstart installation procedure in order to make system installs more efficient.

Testing Pre-/Post-Install Scripts

- Things don't always work as expected in the jumpstart environment
- **`boot net`** (with no additional args) brings up interactive install
- Exit the **`suninstall`** program and you can do your script testing

One of the problems with writing pre- and post-install scripts is that it can be difficult to simulate the Jumpstart environment for testing purposes. The good news is that you don't have to.

The trick is to configure the Jumpstart server as if you were preparing to boot a new Jumpstart client (set up `/etc/ethers` and `/etc/inet/hosts`, run the `add_client` script, etc.). However, when you boot the client just use "`boot net`" without the "`- install`" flags. This will cause the client to boot over the network and start the interactive `suninstall` program. You may, however, quit out of this program on the first screen (hit `<F5>` and then `<F2>`) and end up at a shell prompt in the Jumpstart environment. You can then mount your pre- and post-install scripts via NFS from the Configuration Server and test to your heart's contentment.

Discretion, Valor, et al...

```
ROOT=/a
BOOTSCRIPT=/etc/rc2.d/S74Patch
SERVER=192.168.1.1
FILESYS=/export/patches
MOUNTPT=/mnt

cp /dev/null ${ROOT}/${BOOTSCRIPT}
chmod 744 ${ROOT}/${BOOTSCRIPT}

echo '#!/sbin/sh'                   >> ${ROOT}/${BOOTSCRIPT}
echo "mount ${SERVER}:${FILESYS} ${MOUNTPT}" \
                                    >> ${ROOT}/${BOOTSCRIPT}
echo "cd ${MOUNTPT}/\`uname -r\`"  >> ${ROOT}/${BOOTSCRIPT}
echo "./install_cluster -q -nosave">> ${ROOT}/${BOOTSCRIPT}
echo "rm -f ${BOOTSCRIPT}"         >> ${ROOT}/${BOOTSCRIPT}
echo "reboot -- -r"                >> ${ROOT}/${BOOTSCRIPT}
```

Sometimes the most appropriate time to run a post-install script is not during the Jumpstart process at all, but rather immediately after the client system boots for the first time.

This slide shows a post-install script whose only job is to create a boot script on the client system which will be triggered when the client boots for the first time. This generated script actually installs the Sun Recommended Patch Cluster which it obtains over the network via NFS from a central server. Installing the patch cluster in the Jumpstart environment is more difficult because all of the client's local file systems are mounted on `/a`.

It may be difficult to read the code above, but the generated script ends up in `/etc/rc2.d/S74Patch` and reads as follows:

```
#!sbin/sh
mount 192.168.1.1:/export/patches /mnt
cd /mnt/`uname –r`
./install_cluster –q –nosave
rm –f /etc/rc2.d/S74Patch
reboot -- -r
```

Note that we're using the IP address of the patch server since we can't be guaranteed that name service is working properly at this point. Also note that the script removes itself before it calls reboot. Nothing wrong with this behavior– the script won't actually be removed completely until all processes which have the file open are terminated.

The Bad News

- Jumpstart installs software in a very inefficient manner
- Patches can also take a long time to install, depending on time since FCS
- Full install + recommended patch cluster can take 1.5 hours

The truly unfortunate aspect of the Jumpstart install process is that each OS package is added onto the system one at a time via the `pkgadd` process. There's a lot of overhead to `pkgadd`, not to mention the fact that the file system containing the packages has to be read via NFS from the central Install Server host. Then you're probably going to want to install at least the Recommended Patch Cluster. Total install time can be as much as 90 minutes, which may not seem like a long time unless (a) you've got 500 machines to build in one evening, or (b) you're trying to replace a user's desktop so that they can get back to work.

However, Virginia, there is a Santa Claus…

The Good News

- You don't have to use the standard jumpstart install process
- Custom pre- and post-install scripts can be used instead
- Sample **`rules`** file entry:

```
network 192.168.128.0 \
        bin/clone   -     bin/clone-postinst
```

The good news is that if you're clever at writing your pre- and post-install scripts then you can completely bypass the normal Jumpstart install process. If the administrator does not specify a profile file in the `rules` entry for a given machine, then Jumpstart expects that the pre- and post-install scripts are completely responsible for installing the OS on the local client machine.

Frankly, the really difficult part about installing systems using custom pre- and post-install scripts is partitioning the local disk properly, though there are some other cute hacks that need to be reviewed as well. The rest of this section will focus on a specific example of a custom pre-install script which rapidly copies a default OS image onto a new client.

The `clone` Script

- Build one machine (by hand or with standard jumpstart)
- Make a level 0 backup of all file systems (and copy of partition table)
- **`clone`** script operates on "blank" host:
  - Copies partition table and builds file systems
  - Restores dumps from "standard" host
  - Tweaks **`hosts`** files to change identity

The `clone` script operates by restoring another system's image onto a new machine's local disk drives via `ufsrestore`. The administrator needs to somehow create a "gold standard" version of a particular platform (either via a standard Jumpstart or manually). Once satisfied with the system configuration, the admin makes level 0 backups of that machine's file systems and copies the dump files (`compress`ing or `gzip`ing the files is a good idea) to a central server.

The `clone` script will simply mount the dump files from the central server and `ufsrestore` them onto the client's disks. Of course, the `clone` script has to first partition the client's local disk appropriately and create file systems in the new partitions (more on this coming up), so part of the prep work before running the `clone` script is copying the partition table from the "gold" machine to the central server where the dump files reside. Once the `clone` script restores the dump images onto the new machine, it needs to tweak half a dozen files so that the new system comes up with a different hostname and IP address from the "gold" machine.

The `clone` script runs in less than half the time of the standard "package-by-package" Jumpstart install process (install speed is essentially limited only by your network bandwidth and disk speed on the target host). You can find copies of the `clone` script and related files at the usual `http://www.deer-run.com/~hal/jumpstart/` site.

Some Defaults to Get Started

- What's my host name?

```
HOSTNAME=`uname -n`
```

- What OS version is this?

```
OSVERS=`uname -r`
```

- What kind of machine am I?

```
PLATFORM=`prtconf | awk '/^SUNW,/ { print }'`
```

The `clone` script needs to set a bunch of defaults before getting underway. Much information about the client machine can be derived from the `uname` command once the client has booted up in the Jumpstart environment.

In particular, `uname -i` usually returns the system's hardware type– this is a string like `SUNW,Ultra-5_10`. However, on some non-Sun hardware `uname -i` is not always completely reliable. A more cumbersome (but also more portable) method is to pull this information out of `prtconf` as shown above. To get an idea of what's going on, it's helpful to look at the output of `prtconf`

```
% prtconf
System Configuration:  Sun Microsystems  sun4u
Memory size: 384 Megabytes
System Peripherals (Software Nodes):

SUNW,Ultra-5_10
```

[…additional lines deleted …]

The `awk` line simply matches the line which starts with "`SUNW,`" and prints it.

More Defaults

- What's my disk device?

```
PRIM_DISK=`ls /dev/rdsk | \
  head -1 | sed 's/..$//'`
```

- What kind of disk is it?

```
DISK_NAME=`format -d $PRIM_DISK \
  -f $CROOT/lib/format.cmd | \
  awk '/^</ { print $1 }' | sed 's/<//'`
```

On single-disk systems, determining the system's primary disk device is straightforward. Our script gets a listing of `/dev/rdsk` and simply snatches off the first entry– usually something like `c0t3d0s0`. If you want the disk device and not a disk slice, then you need to drop the last two characters (`c0t3d0`). On multi-disk systems, the disk which is sorted first by `ls` (usually the disk with the lowest SCSI target ID) is not guaranteed to be the system's boot disk, so proceed with caution!

Finding out the manufacturer's name for this disk turns out to be tricky. The only place this information is available is from the `format` command:

```
# format -d c0t0d0
[… lines deleted …]
format> current
Current Disk = c0t0d0
<ST39120A cyl 17660 alt 2 hd 16 sec 63>
/pci@1f,0/pci@1,1/ide@3/dad@0,0

format> quit
#
```

The string we're trying to get at is "`ST39120A`", but `format` likes to be run interactively rather than in a script. The work-around is to create a "command file" and feed it to `format` with the `–f` option. The command file contains the `current` and `quit` commands we would normally enter in an interactive session. We feed the output to `awk` to pull out the string we need.

What They Didn't Teach You...

```
echo "Writing partition table (VTOC) to disk:"
if [ -f $CROOT/disks/$PART_FILE ]; then
        fmthard -s $CROOT/disks/$PART_FILE \
            /dev/rdsk/${PRIM_DISK}s2
else
        echo "No $CROOT/disks/$PART_FILE"
        exit 255
fi
echo ""
```

If you've been administering Solaris machines for a long time, you may think that the way you write partition tables to drives is with the `format` command. However, as we discussed on the last slide, `format` is a pain to run from inside a non-interactive script.

It turns out that Solaris also supplies the `fmthard` command for non-interactively writing partition tables (formally speaking, that's the disk's VTOC or *volume table of contents*) based on a data file. The data file format used by `fmthard` is tricky, but Solaris also supplies the `prtvtoc` command which can dump out the VTOC from an existing disk drive in the format used by `fmthard`.

So, as far as the `clone` script goes, the administrator needs to partition the "gold" system or some other machine with the same type of disk drive as the target platform and then run `prtvtoc` to dump that partition table into a file. The partition file should then be stored on the same server that the dump images are kept on. The name for the partition file for the `clone` script is (by default) the manufacturer's disk name which we extracted via `format` on the previous slide.

Note that disk geometry (cylinders, tracks, heads, etc.) varies widely from manufacturer to manufacturer and from disk to disk. You almost certainly can't use the same VTOC on a 9GB disk from two different manufacturers, so make sure you run the `prtvtoc` command on a system which has a matching disk as compared to your target machine. Note that Sun regularly changes disk drive vendors, so three Ultra5s bought at three different times may have three completely different disks.

Building the Root File System

```
echo "Building root file system:"
newfs /dev/dsk/${PRIM_DISK}s0 </dev/null
mount /dev/dsk/${PRIM_DISK}s0 /a
cd /a
zcat $CROOT/images/$SYS_IMAGE/root.dump.Z | \
      ufsrestore -rf -
rm -f restoresymtable
echo ""
```

Once the VTOC has been written, the `clone` script needs to start building file systems with `newfs` and then doing the restores (you can't do a restore into a raw disk partition, so you need to `newfs` first). `newfs` will run interactively unless its input is not a tty, so we redirect its input to come from `/dev/null`. Technically we should run `fsck` on the file system between the `newfs` and the `mount`, but frankly it's never been a problem for your author.

The `clone` script restores the root file system first so that it can get at the `/etc/vfstab` file and find out about the other file systems that need to be configured for the "standard" system image. Note that we are maintaining the Jumpstart convention of building the file systems on the local disk by rooting them on `/a` in the Jumpstart environment. The `restoresymtable` file is an artifact of the `ufsrestore` process and can be safely deleted.

Reading the vfstab File

```
set -- `awk
      '(!/^#/ && $4 == "ufs" && $3 != "/") \
       { printf("%s %s\n", $1, $3) }'      \
      /a/etc/vfstab`

while [ $# -ge 2 ]; do
      DEV=$1
      FS=$2
      shift 2

      # … rest of code on next slide …
done
```

Next we need to find all of the other UFS file systems which need to be created on the local client. The `awk` script looks at the `/a/etc/vfstab` file we just restored and pulls out all non-comment lines (`!/^#/`) which refer to UFS file systems (the fourth column of the `vfstab` file is equal to "`ufs`") which are not the root file system (the third column not equal to "`/`") that was already restored. The `awk` script prints out the first (disk device) and third (mount point) columns of any matching lines and "`set --`" makes the output of the `awk` script the current argument list for the script (which means we can manipulate the output fields as `$1`, `$2`, etc. and with the `shift` operator).

We then fire off a `while` loop which will pull off pairs of arguments from our new argument list and operate on them. The `while` loop continues until all arguments are exhausted.

Isn't shell scripting fun?

Building Each File System

```
echo "Creating $FS on device $DEV"
newfs $DEV </dev/null
mount $DEV /a$FS

FROOT=`echo $FS | sed 's/^\///' | sed 's/\//-/g'`
if [ -f $CROOT/images/$SYS_IMAGE/$FROOT.dump.Z ]
then
   cd /a$FS
   zcat $CROOT/images/$SYS_IMAGE/$FROOT.dump.Z | \
      ufsrestore -rf -
   rm -f restoresymtable
fi
echo ""
```

What we do inside of the while loop is essentially the same steps we used to restore the root file system earlier: run `newfs`, mount the file system under `/a`, and then use `ufsrestore` to pull back the "gold" image of the file system. Note that not all file systems will have dump files associated with them (you might choose to populate non-system directories like `/usr/local` or `/home` through some other mechanism).

Clean Up Hosts Files

- Files where the old host name appears:

  `/etc/nodename`
  `/etc/hostname.*`
  `/etc/inet/hosts`
  `/etc/net/*/hosts`

- See notes for other potentially "interesting" files to change...

A system's hostname appears in all of the files listed above (that's six total files because there are three hosts files in directories under `/etc/net`). The system's IP address appears in `/etc/inet/hosts`. The `clone` script needs to tweak all of these files so that the new machine boots up with a different identity from the "gold" image.

As part of the post-install process, you may also want to think about modifying the `/etc/defaultrouter`, `/etc/resolv.conf`, `/etc/inet/ntp.conf`, `/etc/ssh_host*_key`, and other similar files.

Partial Code Listing

```
set -- `netstat -in | awk \
   '/^[a-z]*e0/ { printf("%s %s\n", $1, $4) }'`
PRIM_INT=$1
IPADDR=$2

echo "127.0.0.1 localhost"   >  /a/etc/inet/hosts
echo "$IPADDR   $HOSTNAME loghost" \
                              >> /a/etc/inet/hosts

rm -f /a/etc/hostname.*
echo $HOSTNAME >/a/etc/hostname.$PRIM_INT
```

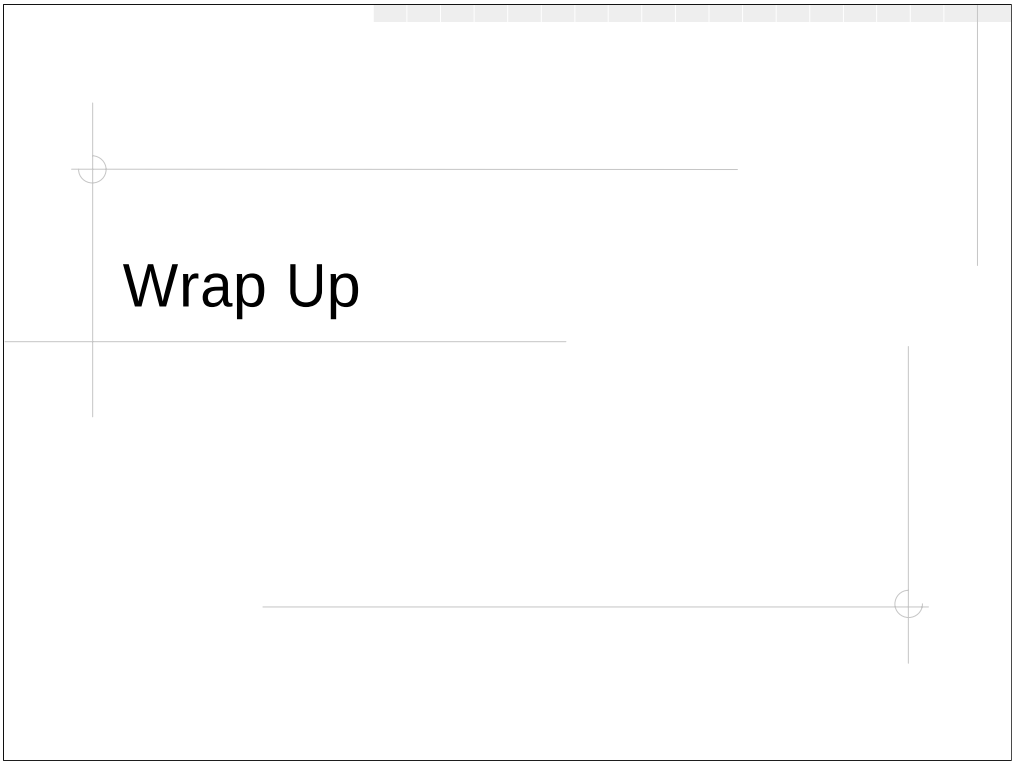
It's worthwhile to look at how the `clone` script goes about deducing and setting the new system's network parameters.

The output of `netstat -in` looks like this:

```
# netstat -in
Name  Mtu  Net/Dest      Address     ...
lo0   8232 127.0.0.0     127.0.0.1   ...
hme0  1500 10.66.0.0     10.66.2.6   ...
```

The `awk` script matches any lines where the interface name ends in "`e0`" and extracts the interface name (column 1) and the IP address (column 4).

If there were more than one ethernet interface on the system, we might have a problem, but the Jumpstart generally only activates the system's primary network interface (the interface on the primary CPU board on the system). If the system *does* have multiple interfaces, you'll have to configure the extra devices as part of the post-install process (or manually when the system reboots).



Time to ask any final questions and review the URLs where you can find additional information.

That URL for the *Advanced Installation Guide* again is:

```
http://docs.sun.com:80/ab2/coll.214.7/
    SPARCINSTALL/@Ab2PageView/6302
```

(that's a single long line as far as your browser is concerned).