Network Application Configuration

Student Guide

Sun

Sun Microsystems Computer Corporation
Technical Education Services
MS UMIL07-14
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part Number SA-285
Revision B, July 1993

Contents

Administering Remote Access

Objectives

Upon completion of this module, you will be able to:

- Describe the terms: Ethernet interface and address, broadcast bus, and carrier sense multiple-access with collision detect.
- Distinguish between different internetwork terms.
- Describe the function of each of the following files:
  - `/etc/inet/hosts`
  - `/etc/defaultdomain`
  - `/etc/nodename`
  - `/etc/hostname`.*xxy*
- Describe the function of the `/etc/hosts.equiv` and `.rhosts` files in relation to network sercurity.
- Send `ping` and `spray` requests to a remote host to test its response.
- Execute the `netstat -i` command and interpret the output.

References

SunOS 5.1 Administering TCP/IP and UUCP, Chapters 1-3

Introduction

The designers of the UNIX® operating system built in the concepts of users and groups before networks became popular. The UNIX mechanisms for controlling which resources are shared and which are private over the network are a natural extension of its standalone mechanisms for user authentication (login security) and discretionary access control (data security).

Thus, you already know much of what is required to set up and maintain a network that permits users to log in and execute commands across the network. There are, however, several new concepts and procedures associated with setting up, maintaining, and troubleshooting a secure network that permits users to login and execute commands across the network. You will learn these concepts and procedures in this lesson.

In addition to learning how to set up and maintain a secure network, you will learn many of the concepts and procedures that form the foundation of setting up and maintaining the NFS® distributed file system and Network Information Services Plus (NIS+) environments.

Network Hardware Terminology

[illegible]

- **Ethernet Interface:** `le0` or `ie0`

  All Sun™ workstations have an Ethernet® interface built into the CPU board. The `le0` interface is for the AMD Lance interface. The `ie0` interface is for the Intel™ interface.

- **Broadcast Bus**

  Ethernet is a local area network (LAN) technology. It is a *broadcast* network, meaning that all workstations on the network hear all of the traffic. It employs a *bus* topology, which means all of the workstations are connected to one single wire.

- **Carrier Sense, Multiple-Access with Collision Detect**

  Ethernet uses an access method called *Carrier Sense, Multiple-Access* with *Collision Detect* (CSMA/CD) in order to avoid chaos on the network. Each workstation listens for a signal (carrier) on the wire. The lack of carrier indicates the network is idle. Any workstation can transmit onto an idle network. It listens as it writes to be certain what is on the wire is what it wrote. If not, a collision takes place. The workstation stops its transmission and waits for the network to become idle, at which point it tries again.

  In 1985, the IEEE[1] published the 802.3 standard for CSMA/CD LANs. The Ethernet built into Sun workstations adheres to this standard.

- **Ethernet Address:** `8:0:20:9:4e:cc`

  The Ethernet address is a 48-bit number. It is represented by hexadecimal digits and is subdivided into six 2-digit fields separated by colons.

  The Ethernet address is also called the *hardware* address. It is guaranteed by the manufacturer to be unique, and forms the basis of a network addressing scheme. All Sun workstations have an Ethernet address burned into the system ID PROM.

1. Institute of Electrical and Electronic Engineers, an important standards body.

What Is an Internet Network?

There are several different terms associated with the term "internet" which can usually distinguished by context.

An *internetwork* is a linked group of networks such as LANs connected to a wider area network (WAN).

The Internet Protocol Suite

Sun networking software is based on a set of specifications (also called protocols) known as the Internet protocol suite (these protocols are also called the Transmission Control Protocol/Internet Protocol (TCP/IP) protocols, after two of the main protocols that make up the suite). Local networks that use the Internet protocol suite are often referred to as internet networks.

The Internet Network

The Internet is an international wide area network of government agencies, commercial companies, and educational institutions (and several other types of organizations) using TCP/IP that is administered by the Network Information Center (NIC) at Government Systems, Inc.

```
Government Systems, Inc.
Attn: Network Information Center
14200 Park Meadow Drive
Suite 200
Chantilly, VA   22021
U.S.A.
1-800-365-3642 (1-800-365-DNIC)
1-703-802-4535
```

Running out of numbers within the next 2 yrs
for TCP/IP addresses

What Is an Internet Network?

Internet Addressing

For sites to communicate using telephones, each site must have a phone number that is known by the other sites. Similarly, for a network of computers to communicate, they must each have a unique address that is known to the other computers on the network.

Internet Address Representation

Internet addresses are 32 bits, divided into four 8-bit fields. Each 8-bit field is represented by a decimal number between 0 and 255.

```
[ 0 - 255 . 0 - 255 . 0 - 255 . 0 - 255 ]
```

The four decimal numbers are separated by periods. For example, 127.0.0.1. Each Internet address is divided into a *network number* and a *host number.*

Network Number

The network number identifies your network to the outside world. It is recommended that the network number be obtained from the NIC, especially if you intend to connect your network to the Internet. The information you need to provide is detailed in Appendix A of the *SunOS 5.1 Administering TCP/IP and UUCP* guide.

Host Number

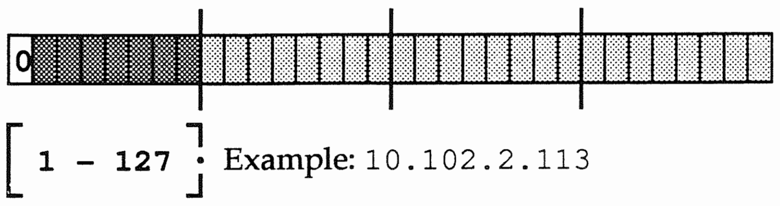
You assign the host number that uniquely identifies your workstation on your network. Do not use 0 or 255 for your host number. The Solaris® 2.*x* operating system uses a host number of all ones (decimal 255 = binary 11111111) for the *broadcast address.*

Internet Network Classes

Network numbers are divided into classes, depending on the size of the network.

Class A - Very Large Networks (up to 16 Million Hosts)

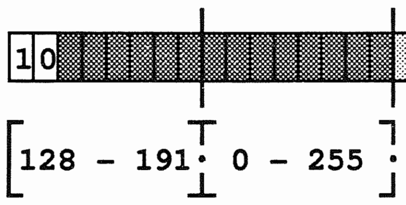
If the first bit is 0, then the next seven bits are the network number. This allows up to 127 Class A networks.



Example: `10.102.2.113`

Class B - Large Networks (up to 65,000 Hosts)

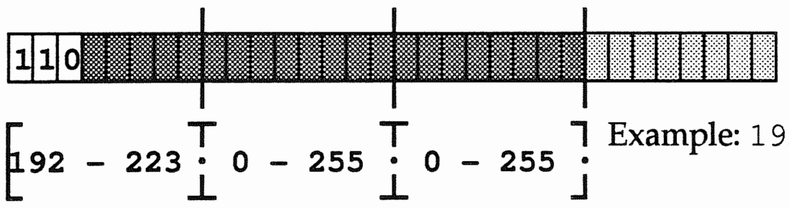
If the first 2 bits are 10, then the next 14 bits are the network number. This allows 16,384 Class B networks.



Example: `129.150.254.2`

Class C - Small and Mid-Sized Networks (up to 254 Hosts)

If the first 3 bits are 110, then the next 21 bits are network number. This allows up to 2,097,152 Class C networks.



Example: `192.9.227.13`

Configuring Network-Related Information

This section describes the files that are used to define a system's characteristics in terms of network configuration. These files are created automatically during the system configuration phase of installing a system.

The /etc/inet/hosts File

Each Internet address has a corresponding host name. The /etc/inet/hosts file associates IP addresses with host names. This means users and administrators can refer to systems by their host names rather than their IP addresses when using networking software. The /etc/hosts file is a symbolic link to this file.

The following hosts file defines the IP address and host name for the local system (venus) and two systems on the network.

Loop Back Network

```
# Internet host table
127.0.0.1 localhost loghost
129.150.212.16 venus
129.150.212.11 mars
129.150.212.2 jupiter
```

If you edit this file manually, note that trailing spaces after the host name renders that entry ineffective.

The Local Host

Network 127 is reserved for the *local host* network number. The local host address is available so that the local system can run network software without a network, if necessary.

Aliases

The host name can be followed by one or more *aliases*. Aliases permit the host to be known on the network by additional names. Sometimes, an alias identifies the host to be the provider of a special network wide service (for example, the main mail provider is aliased as mailhost).

Configuring Network-Related Information

The /etc/defaultdomain File

This file sets a system's name service domain name and is used in another module.

```
$ cat /etc/defaultdomain
solar.com.
$
```

A *name service* provides centralized information that users, workstations, and applications use to communicate on a network.

The /etc/nodename File

This file sets a system's host name.

```
$ cat /etc/nodename
venus
$
```

The /etc/hostname.*xxy* File

This file name identifies the Ethernet interface, such as le0, and contains the host name or the host's IP address.

```
$ cat /etc/hostname.le0
venus
$
```

Changing a System's Host Name

A system's host name is specified in several locations.

Follow these steps if you want to change a system's host name.

1. Modify these files to identify the new host name:

   - `/etc/nodename`
   - `/etc/hostname.`*xxy*
   - `/etc/inet/hosts`
   - `/etc/net/ticlts/hosts`
   - `/etc/net/ticots/hosts`
   - `/etc/net/ticotsord/hosts`

   The `/etc/net` directory is used for selecting network services and resolving network addresses.

2. Reboot the system to activate the new system name.

Remote Access Commands

These remote access commands may have been covered in a previous module or course. The `rlogin`, `rsh`, and `rcp` commands are used to discuss remote access security, and are summarized below.

The `rlogin` Command

The `rlogin` command allows a log in session on a remote system, if you have an account set up on the remote system. It may be possible to log in without specifying a password, if the remote system allows this type of access.

```
$ rlogin venus
Last login: Thu Jun 3 16:01:13 from mars
Sun Microsystems Inc. SunOS 5.2 Generic March 1993
$
```

The `rsh` Command

The `rsh` command is used to execute a program on a remote system.

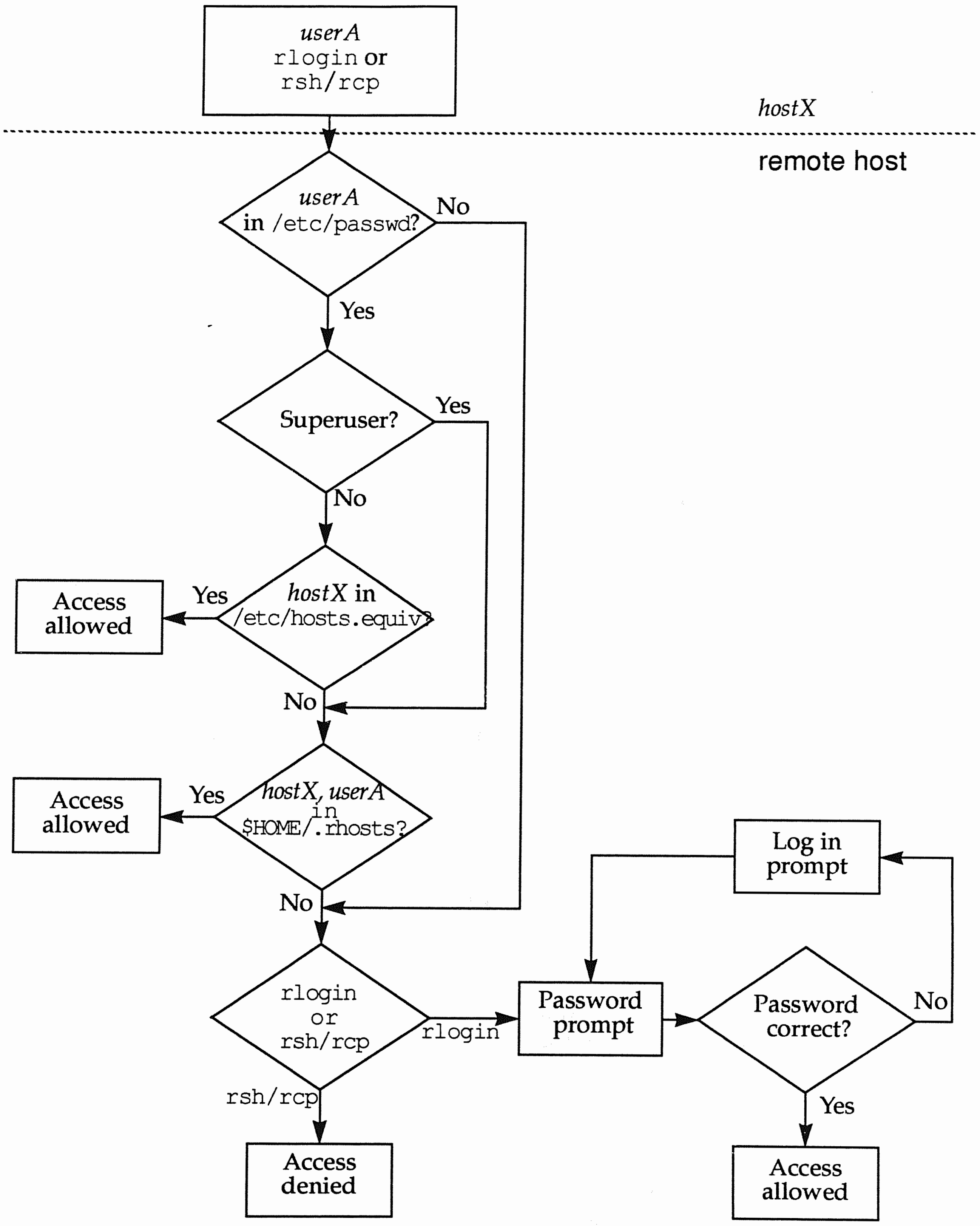
```
# rsh venus showrev
Hostname: venus
Hostid: 72303700
Release: 5.2
Kernel architecture: sun4m
Application architecture: sparc
Hardware provider: Sun_Microsystems
Kernel version: SunOS 5.2 Generic March 1993
```

The `rcp` Command

The `rcp` command allows you to copy files between remote systems.

```
$ rcp jupiter:/etc/inet/hosts /tmp/hosts
```

Remote Access Authentication



Remote Access Authentication

Related Files

There are several files that are important to network security when using the `rlogin`, `rsh`, and `rcp` commands.

The `/etc/passwd` File

An entry for a user (*userA* from *hostX*) in the remote system's password file allows that user to log in remotely. If the password entry also includes a corresponding entry in the `/etc/shadow` file, the remote user (*userA*) must supply a password upon login.

The `/etc/hosts.equiv` File

The next file is `/etc/hosts.equiv`, which identifies the remote machine (*hostX*) as a *trusted host*. This means if a host (*hostX*) is listed in this file, all users from that system that are listed in the remote password file can `rlogin` to the remote machine without supplying a password.

This is useful for sites where it is common for users to have accounts on different machines and eliminates the security risk of sending ASCII passwords across the network. If the remote command is trying to execute as the superuser, the `/etc/hosts.equiv` test is skipped.

You must create this file; it doesn't exist by default.

The User's `.rhosts` File

The `rlogin` process checks for the `.rhosts` file in the user's home directory on the remote system. In the case of the superuser, this is the `/.rhosts` file. If the host (*hostX*) is listed in this file, it is considered a trusted host for this user. By default, this file does not exist in any users' home directories.

The `rlogin` program defaults to the standard password-based login procedure if the remote authentication fails. If the user (*userA*) is unknown to the remote system, the password test always fails and the user is prompted to enter a login name.

Remote Access Authentication

Related Files (continued)

The `/etc/hosts.equiv` and `.rhosts` files

While both files have the same format, the same entries in each file have very different effects. The general format is presented here. Explanations and examples of the meanings of each type of entry are presented on the following pages.

(Neither file exists by default; they must be created.)

- Both files are formatted as a list of one-line entries.

  hostname
  hostname username

- If only the *hostname* is used, then users from the named host are trusted. That is, every user from the named host is a trusted user, provided they are known to the remote system.

- If both *hostname* and *username* are used, then only the named user from the named host can access the system.

  The special character '+' can be used in place of either *hostname* or *username* to match any known host or user. For example, the + entry by itself allows a user from any host to access the remote system as a user with the same *username*.

- The host names in `/etc/hosts.equiv` and `.rhosts` files must be the official name of the host, not one of its nicknames.

- For more information on the format of both of these files see the `hosts.equiv` manual page.

Related Files (continued)

Using the /etc/hosts.equiv File

Typically, this file contains a list of host names, one per line, or a single plus sign (+), indicating that all hosts are trusted hosts[2] on the remote system.

- All hosts are trusted hosts in this /etc/hosts.equiv example.

```
+
```

- This is an example where two hosts are trusted hosts.

```
neptune
pluto
```

- This is an example of a specific user (rimmer) from a host (pluto). This user can log in as *any* user on the remote system.

```
pluto rimmer
```

This specifies that the user rimmer can access the remote host from the host pluto as any local user. Under most circumstances, *this is very unsecure.*

For example, the above entry would permit rimmer to use any of the following commands to access venus without a password:

```
$ rlogin venus

$ rlogin venus -l lister

$ rlogin venus -l kryten
```

2. This gets extended by NIS and NIS+ to all machines in the hosts database.

Related Files (continued)

Using the `$HOME/.rhosts` File

If the `/etc/hosts.equiv` test fails, the next step is to check for a `.rhosts` file in the user's home directory on the remote host.

- The default is no `.rhosts` file.
- The most conventional usage is to explicitly name a host. This allows the user access to the remote machine from the named remote host. It assumes that the login name is the same on both hosts.

```
pluto
```

- The following is an example of a user's `.rhosts` file, which allows any user with the same login name as the local user to `rlogin` from any host.

```
+
```

- As with the `/etc/hosts.equiv` file, any host name in a user's `.rhosts` file can be followed by a login name.

```
pluto rimmer
```

If this `.rhosts` file is in the user's (`lister`'s) home directory on the remote system, the user `rimmer` can access the remote machine from the host `pluto`, as the user `lister`.

Additional Remote Access Commands

The telnet Command

The telnet command is an industry standard application that uses a server process to connect to the operating system. The telnet server (telnetd) simulates a terminal to allow a user to log into a remote host system and work in that environment.

Using the telnet command, you can log into remote systems including non-UNIX systems. The telnet command is an integral part of the PC-NFS® software.

```
$ telnet venus
Trying 129.150.212.16 ...
Connected to 129.150.212.16.
Escape character is '^]'.

UNIX(r) System V Release 4.0 (venus)

login: lister
Password:
Last login: Sat Jun 4 10:31:47 on console
Sun Microsystems Inc. SunOS 5.2 Generic March 1993
$
```

The telnet command relies only on password information to determine if a user can log onto a remote system. A user always has to enter a password that is transmitted across the network as ASCII characters. This is a security risk.

Additional Remote Access Commands

The `ftp` Command

The `ftp` command is used to send files to a remote system, get files from a remote system, transfer files using ASCII, binary, DOS, or UNIX format, and perform actions on multiple files and directories, including the use of wildcards.

Archive servers sometimes provides an *anonymous* `ftp` account so users can pull files off the server. Setting up an anonymous `ftp` account is covered in the `ftpd` manual page.

```
$ ftp venus
Connected to 129.150.212.16.
220 venus FTP server (UNIX(r) System V Release 4.0) ready.
Name (129.150.212.16:lister): Return
331 Password required for lister.
Password: xxx
230 User lister logged in.
ftp> cd /etc
250 CWD command successful.
ftp> get vfstab /tmp/vfstab
200 PORT command successful.
150 ASCII data connection for vfstab (129.150.182.30,34681)
(665 bytes).
226 ASCII Transfer complete.
local: /tmp/vfstab remote: vfstab
681 bytes received in 0.0071 seconds (94 Kbytes/s)
ftp> quit
```

Network Monitoring

The rusers Command

The rusers command is used to identify who is logged in on the network.

```
# rusers -la
Sending broadcast for rusersd protocol version 3...
root        localhost:console       May 21 09:27 195:08
root        mars:console            May 21 09:27 195:08
root        pluto:console           Jun 3 08:29 2:43
Sending broadcast for rusersd protocol version 2...
starbug     starbug:console         May 17 12:26 1:48
#
```

Options:

a	Gives a report for the system, even if no users are logged in.
l	Displays the long listing (such as the who command format).

Network Monitoring

The ping Command

The /usr/sbin/ping command is used to determine if the named host is up and running. It sends an echo request to the named host and reports when a reply is received. The default timeout is 20 seconds, which is a long time on a LAN.

Sending a ping request to a remote system does not tell you the state of the machine, only that its network interface is configured.

Examples:

1. Send a ping request to a running host.

```
$ ping pluto
pluto is alive
$
```

2. Send a ping request to an unreachable host.

```
$ ping saturn
no answer from saturn
$
```

3. Send a ping request to a system not found in the host table.

```
$ ping krypton
ping: unknown host krypton
$
```

Network Monitoring

The spray Command

Because the ping command uses a very low-level protocol, it is still possible for a host to respond to a request even though it is overloaded. Unlike the ping command, the /usr/bin/spray command uses the higher-level protocols. Its success, partial success, or failure can give some hints about what is happening on your network or on the remote host. The spray command is typically used to test the response of a machine on the network over a period of time.

Examples:

1. The spray command is successful.

```
$ spray pluto
sending 1162 packets of length 86 to pluto ...
        127 packets (10.929%) dropped by pluto
        1326 packets/sec, 114043 bytes/sec
$
```

2. This example shows a remote host having trouble responding to the spray command.

```
$ spray mars
sending 1162 packets of length 86 to mars
...SPRAYPROC_CLEAR : RPC: Timed out
$
```

3. This example shows the result of trying to spray a host not found in the hosts file.

```
$ spray earth
spray: cannot clnt_create earth:netpath: RPC:
Name to address translation failed - n2a:
hostname not found
$
```

Network Monitoring

The `netstat` Command

The `netstat` command displays the status of various network-related data structures. The form shown in the example (using the `-i` option) shows the usage of the network interfaces on your machine. The most useful information is the ratio of `Collis/Opkts` multiplied by 100.

```
# netstat -i
Name  Mtu   Net/Dest Address   Ipkts   Ierrs Opkts   Oerrs Collis Queue
lo0   8232  loopback localhost 769     0     769     0     0      0
le0   1500  neptune  venus     211053  0     18405   0     676    0
```

The fields are:

`Name`	The network interface. `lo0` is the loopback interface, used to test the network protocols in software.
`MTU`	The maximum transmission unit (MTU) is the size, in bytes, of the data of the largest packet this interface supports. Ethernet has a MTU of 1,500 bytes, and Fiber optic Data Distribution Interface (FDDI) is 4,428 bytes. The loopback network has a MTU of 8,232.
`Net/Dest`	The name of the network destination. This is derived from the network number. A name can be assigned in the `/etc/inet/networks` file.
`Address`	The host name. If you use the `-n` option, you also see the Internet address.
`Ipkts/Ierrs`	Shows the number of input packets and errors since the interface was configured.
`Opkts/Oerrs`	Shows the number of output packets and errors since the interface was configured.
`Collis`	The number of collisions on this interface. To calculate a percentage rate, use (`Collis/Opkts`) * 100 = *collision rate*. An excellent collision rate is 0–2%, fair is 3–5%, and poor is over 5%.
`Queue`	The number of packets awaiting transmission at the interface. This is almost always zero.

Summary

In this lesson, you learned that:

- All Sun workstations have a unique Ethernet address that is set by the manufacturer.
- The `/etc/hostname.`*xxy* file reflects a system's Ethernet interface.
- An IP address uniquely identifies a machine on the network.
- A system's host name is set in the `/etc/nodename` file.
- The `/etc/inet/hosts` file contains a system's host name and IP address, and host names and IP addresses of other systems on the network.
- The remote access commands are used to log in, copy files, and execute commands on remote systems.
- Several layers of authentication are used to enable/disable remote access.
- The `/etc/hosts.equiv` and `.rhosts` files are used to establish trusted users and hosts for remotely accessing systems.
- The `ping`, `spray`, and `netstat` commands are used to perform basic network monitoring.

Exercise 1-1

The purpose of this lab is to set up your `hosts` file, monitor the network, and use some network access commands to verify remote access authentication.

Part I—Using Remote Monitoring Commands

Procedure

1. Become superuser.
2. Edit the `/etc/inet/hosts` file to include all of the machines on the lab network. *Check with your instructor for a listing of the IP addresses and host names for your lab environment.*

   Example:

   ```
   192.9.200.11          mars
   ```

   Ensure that the *network number* is correct.
3. Execute the `ping` and `spray` commands to test being able to reach the network and other hosts on the network. See the Network Monitoring section for examples of the `ping` and `spray` commands.
4. Execute the `rusers` command to identify which remote users are logged in.
5. Completely log out of the workstation.

Administering Remote Access Lab

Part II—Network Security

Refer to the previous security module if you need more `/etc/passwd` file information. This part of the lab requires that you work with a partner. Assign one machine to be the local machine and one to be the remote machine.

Check with the instructor to make sure there is a common user name (`student`*) with UID (*`999`*), a password of* `cangetin`*, and* `/export/home/student` *as the home directory on all systems before doing this part of the lab.*

1. Log into the *local machine* as the user `student`. Use the `rcp` command on the *local machine* to get a copy of the `/etc/inet/hosts` file from the *remote machine.*

   Example:

   ```
   $ rcp remote_machine:/etc/inet/hosts .
   ```

   Did you get the file? You shouldn't have.

2. Log out from the *local machine.*

3. On the *remote machine,* create the `/etc/hosts.equiv` file, adding the host name of the *local machine.*

4. Log into the *local machine* as the `student` user. Try the `rcp` command you executed in step 1. Did you get the file? Why were you successful this time?

5. Log out from the *local machine.*

6. Log into the *local machine* as the regular user of that machine. Again try to `rcp` a file from the *remote machine* to your user's home directory. This time get the `/etc/hosts.equiv` file. Did you get the file? Why was your attempt unsuccessful?

Configuring the NFS Environment 2

Objectives

Upon completion of this lesson, you will be able to:

- Describe the functions of an NFS server and an NFS client.
- List the three conditions that must be met in order to share files in the NFS environment.
- Make a resource available for mounting.
- Make a resource unavailable for mounting.
- On an NFS server system, edit the `/etc/dfs/dfstab` file to enable automatic sharing of resources.
- Display a server's available resources for mounting.
- Mount a resource from another system.
- On an NFS client system, edit `/etc/vfstab` to automatically mount resources.
- Describe the function of these commands: `mountall`, `umountall`, `shareall`, and `unshareall`.

References

SunOS 5.1 Administering NFS and RFS, Chapter 1, "How to Use DFS," and Chapter 2, "How to Set Up NFS"

Introduction

Users working on different workstations can share files using the `ftp` and `rcp` commands to copy files over the network. This results in using disk space on multiple machines and, in the case of files that may be edited on multiple machines, is a synchronization nightmare.

Making files stored on one machine directly accessible to other machines so that remote users can access those files as though they were local is a far better solution.

The Solaris 2.*x* operating system supports sharing remote file resources as if they were local files and directories. The sharing of file resources is accomplished through *distributed file systems*—file system types that provide the architectural support for mounting over networks.

The Solaris 2.*x* operating system supports two distributed file system products—Sun's NFS product and AT&T's Remote File Sharing (RFS) product. The Solaris 2.*x* system also provides a Distributed File System (DFS) Administration package that contains files and commands used to administer both the NFS and RFS products.

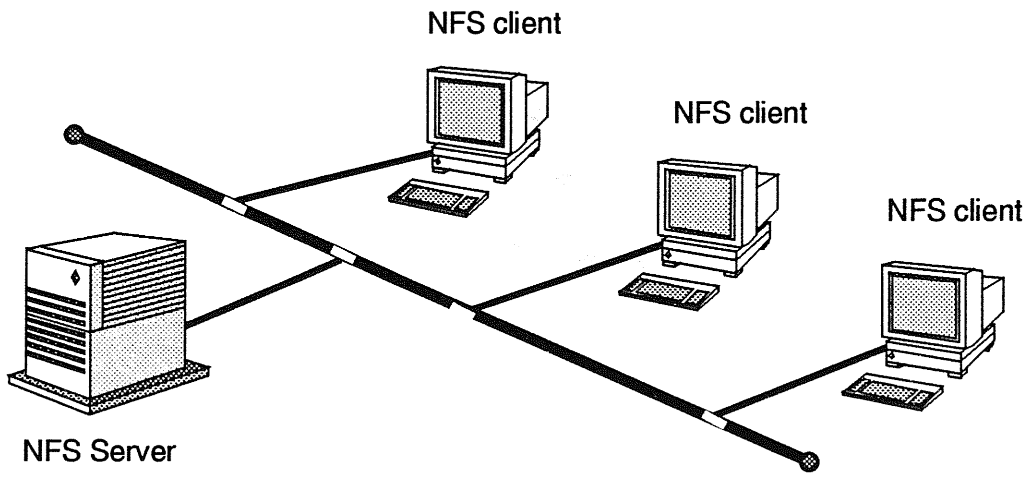
The NFS filesystem is a Sun product built on top of TCP/IP and Sun's Remote Procedure Call (RPC) and eXternal Data Representation (XDR) specifications that have been licensed to many vendors. The NFS environment provides file sharing in a heterogeneous environment, potentially containing many different operating systems, including UNIX, MS-DOS®, and VMS® systems. While both the NFS and RFS products are supported by the Solaris 2.*x* environment, NFS has become the industry standard.

This lesson presents the concepts and procedures involved in configuring the NFS environment using the DFS Administration commands.

How the NFS Environment Works

Three conditions must be met in order to share files in the NFS environment:

- The machines on the network must know each other's host names and internet addresses.
- A server must share a file system or directory (referred to as a file resource) in order to make it available over the network.
- The client must remotely mount that file resource.



NFS Servers

An NFS file server designates local file resources to be shared with other machines on the network. The server reads or writes files in response to client requests. Any machine with a local file system can be an NFS file server.

NFS Clients

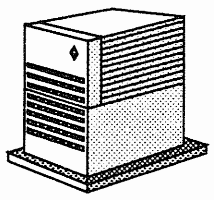
An NFS client machine mounts file resources that are shared over the network and treats them as if they were local file systems. Any machine can be an NFS client. All dataless and diskless clients are NFS clients. An NFS file server can also be an NFS client. The server, however, cannot serve directories that it has mounted as a client.

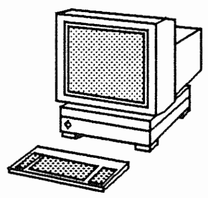
Sharing and Mounting File Resources

The Distributed File System (DFS) Administration command set contains commands for sharing and mounting file resources. The superuser can use these commands to share or mount a remote file resource by typing them on the command line.

This package also provides several files used to automatically share and mount remote file resources. The superuser can set up a system to automatically share and mount file resources by adding entries into these files.

The main commands and files used on both the server and client systems are summarized below.





Sharing Resources	**Mounting Resources**
`share` and `unshare`	`mount` and `umount`
`shareall` and `unshareall`	`mountall` and `umountall`
`/etc/dfs/dfstab`	`/etc/vfstab`

Sharing File Resources Overview

The following pages describe how to setup the NFS client/server environment. Details on the specific command syntax are covered later.

The NFS Server

1. Use the `share` command to make the resource available.

   ```
   # share /usr/share/man
   ```

   This command is not effective unless the NFS server daemons are already running.

2. Edit the `/etc/dfs/dfstab` file to enable automatic sharing of resources whenever the system enters run level 3.

   ```
   share -F nfs /usr/share/man
   ```

   └ file system type

   The `-F nfs` option must be used in `share` command entries in this file for automatic NFS sharing to work properly.

   You should set up automatic sharing if you need to share the same set of file resources on a regular basis.

3. Start the NFS server daemons by executing the following script.

   ```
   # /etc/init.d/nfs.server start
   ```

4. Use the `dfshares` command to verify that the resource is available.

   ```
   # dfshares
   RESOURCE                          SERVER ACCESS    TRANSPORT
       venus:/usr/share/man              venus   -            -
   #
   ```

The server keeps a table of clients mounting its resources in the `/etc/rmtab` file.

Mounting File Resources Overview

The NFS Client

1. Use the `dfshares` command to display a server's available resources.

```
# dfshares
RESOURCE                               SERVER ACCESS   TRANSPORT
     venus:/usr/share/man                    venus    -           -
#
```

2. Use the `mount` command to access the remote file resource.

```
# mount venus:/usr/share/man /usr/share/man
```

   The `/usr/share/man` directory on the client is the mount point in the local system's file hierarchy. This directory should be empty.

3. Edit the `/etc/vfstab` file to add an entry for the remote resource that will be automatically mounted whenever the system enters run level 2.

```
venus:/usr/share/man - /usr/share/man nfs - yes -
```

4. Remote file resources can be unmounted from the client by using the `umount` command.

```
# umount /usr/share/man
nfs mount: /usr/share/man: is busy
```

   This common error message usually means a user is using the resource if it is an application, or someone is accessing the directory using the `cd` command.

Mounting and Unmounting Remote File Resources

The NFS Client

The `mountall` Command

```
mountall -r [ -F nfs ]
```

Without any arguments, the `mountall` command mounts all file resources listed in the `/etc/vfstab` file with a `mount-at-boot` value of `yes`. To limit the action of this command to remote file resources, use the `-r` option.

The `-F nfs` option is only required to restrict the action of this command to NFS resources if other remote file resource types (such as `rfs`) are listed in the `/etc/vfstab` file along with the `nfs` resources.

```
# mountall -r
#
```

The `umountall` Command

```
umountall -r [ -F nfs ]
```

Without any arguments, the `umountall` command unmounts all currently mounted file resources. To limit the action of this command to remote file resources, use the `-r` option.

The `-F nfs` option is only required to restrict the action of this command to NFS file resources if other remote file resource types are currently mounted along with the `nfs` resources.

```
# umountall -r
#
```

Required NFS Daemons

On the NFS Server

NFS operation requires two types of daemons running on the NFS server.[1] If a system has entries in its `/etc/dfs/dfstab` file, these daemons are started automatically when the system enters run level 3.

The Mount Daemon

When a system issues an NFS mount request, the mount process contacts the server's mount daemon (`/usr/lib/nfs/mountd`) to get a *file handle* for the to-be-mounted file resource. The local `mount` process then writes the file handle (along with other information about the mounted resource) to the `/etc/mnttab` file. The file resource is mounted, and the local kernel uses the file handle during all attempts to access the file resource.

The NFS Server Daemon

When a process on the client attempts to access a remote file resource, one of eight NFS server daemons (`/usr/lib/nfs/nfsd`) running on the server gets the request (along with the resource's file handle) and performs the file operation. It then returns any data to the requesting process on the client.

These server daemons are started from the `/etc/init.d/nfs.server` script.

1. This is in addition to the `rpc.bind` daemon that is required for all operations involving remote procedure calls including both NFS server and client operations.

Required NFS Daemons

On the NFS Client

Several daemons run on NFS clients. These client daemons are started automatically when a system enters run level 2 if the system has any NFS entries in its `/etc/vfstab` file.

The Locking Daemons

Two daemons, `/usr/lib/nfs/lockd` and `/usr/lib/nfs/statd`, run on NFS clients. These daemons work together to provide both crash/recovery functions and locking services in the NFS environment. These daemons typically do not require administrative intervention.

The `automount` Daemon

The `automount` daemon is used to mount file resources automatically (without adding entries to the `/etc/vfstab` file). (The automounter is discussed in another lesson in this module.)

These daemons are started from the `/etc/init.d/nfs.client` script.

The share Command

The share command makes file resources available for mounting by remote systems. If no argument is specified, then share displays all file resources currently shared.

Command format:

```
share [ -F nfs ] [ -o options ] [ -d description ] pathname
```

Options:

-F nfs	This option must be used with share command entries in the /etc/dfs/dfstab file for the shareall script to work properly. This option is not required at the command line because nfs is the default remote file system type (that is, is listed as the first file system type in the /etc/dfs/fstypes file).
-o *options*	Specify a comma separated list of file system specific options. These options include rw to allow read-write for all clients, ro to limit access to read-only for all clients, rw=*client*[:*client*]... to permit read-write access for the listed clients, ro=*client*[:*client*]... to permit read-only access for the listed clients, and root=*host*[:*host*]... to allow the client *host* to have root access.
-d *description*	Provide a comment that describes the file resource to be shared. This comment is displayed by the share command with no arguments.
pathname	Specify the path name of the file resource to be shared.

The following share command provides the manual pages with read-only access:

```
# share -F nfs -o ro /usr/share/man
```

The share command writes information about all shared file resources into the /etc/dfs/sharetab file.

The `unshare` Command

The `unshare` command makes file resources unavailable for mounting by remote systems.

Command format:

unshare [-F nfs] *pathname*

Options:

`-F nfs`	Specify `nfs` as the file system type. This option is not typically required because `nfs` is the default remote file system type.
pathname	Specify the path name of the file resource to be unshared.

The following example makes the resource unavailable for mounting:

```
# unshare /usr/share/man
#
```

The /etc/dfs/dfstab File

The /etc/dfs/dfstab file contains share commands. The shareall and unshareall commands are used to explicitly execute the commands in this file.

```
# cat /etc/dfs/dfstab
#  place share(1M) commands here for automatic execution
#  on entering init state 3.
#
#  share [-F fstype] [ -o options] [-d "<text>"] <pathname> [resource]
#  .e.g,
#  share  -F nfs  -o rw=engineering  -d "home dirs"  /export/home2
share -F nfs -o ro /usr/share/man
```

The /etc/dfs/dfstab File

The commands in the /etc/dfs/dfstab file are executed when:

- The system enters run level 3.
- The shareall command is executed by the superuser.
- The /etc/init.d/nfs.server script (which contains a shareall command) is executed by the superuser with the start argument.

The shareall Command

The `shareall` and `unshareall` commands are used to share and unshare multiple resources such as all `nfs` resources.

Command format:

```
shareall [ -F nfs ]
```

Without any arguments, the `shareall` command shares all file resources listed in the `/etc/dfs/dfstab` file. The `-F nfs` option is only required if other file resource types are listed in the `/etc/dfs/dfstab` file along with `nfs` resources.

The unshareall Command

Command format:

```
unshareall [ -F nfs ]
```

Without any arguments, the `unshareall` command unshares currently shared file resources. The `-F nfs` option is only required if other remote file resource types (for example, `rfs`) are currently shared along with `nfs` resources.

The `mount` Command

The `mount` command is used to attach either a local or remote file resource to the file system hierarchy.

Command format:

mount [-F nfs] [-o *options*] *server:pathname mount-point*

If the file resource is listed in the `/etc/vfstab` file, you can specify either *server:pathname* or *mount-point* on the command line because the `mount` command consults the `/etc/vfstab` file for more information.

```
# mount /usr/share/man
```

If both *server:pathname and mount-point* are specified, and the `-F` option is omitted, the `mount` command takes the file system type from the `/etc/vfstab` file.

Options:

`-F nfs`	Specify `nfs` as the file system type. This option is not required since `nfs` is the default remote file system type.
`-o` *options*	Specify a comma-separated list of file-system specific options such as `rw` to mount the file resource read-write, `ro` to mount the file resource read-only (the default is `rw`). See the following page for additional options.
server:pathname	Specify the name of the server separated by a colon (`:`) and the path name of the remote file resource.
mount-point	Specify the path name of the mount point on the local system (which must already exist).

See the `mount` and `mount_nfs` manual pages for additional options.

The mount Command

Command format:

```
mount [ -F nfs ] [ -o options ] server:pathname mount-point
```

Options:

defaults

Specific parameters for NFS file systems include the following options:

good for ro filesystem

default

use with hard mounts

`rw|ro`	The resource is mounted read-write or read-only. The default is read-write.
`suid|nosuid`	Allow or disallow `setuid` execution. The default allows `setuid` execution.
`bg|fg`	If the first mount attempt fails, retry in the background or foreground. The default is retry in the foreground.
`soft|hard`	Returns an error if server does not respond or continues to retry the mount until the server responds. The default is a `hard` mount.
`timeout=`*n*	Sets timeout to `n` tenths of a second. The default timeout is 1.1 seconds.
`retry=`*n*	Sets the number of times to retry the mount operation. The default is 10,000 times.
`intr|nointr`	Allow or disallow keyboard interrupts to kill a process that is hung waiting for a response on a hard-mounted file system. (The default is all mounts are interruptable.)

The `nosuid` option provides additional network security because the `setuid` permissions on `nfs` resources are ignored.

Using a `hard` mount is recommended for all file systems that are mounted read-write such as users' home directories. This means the client continues to retry the mount request until the server responds.

The umount Command

The `umount` command is used to detach either a local or remote file resource from the file system hierarchy.

Command format:

`umount [ -F nfs ]` *server:pathname mount-point*

The command line can specify either *server:pathname* or *mount-point*.

```
# umount /usr/share/man
#
```

Options:

`-F nfs`	Specify `nfs` as the file system type. This option is not required, since `nfs` is the default remote file system type.

The /etc/vfstab File Review

↳ System 5

The following section describes the syntax of the /etc/vfstab file with a focus on NFS specifics.

This an excerpt from an /etc/vfstab file showing the header comment, which describes the fields, and a single NFS entry.

```
#device                 device    mount           FS    fsck  mount    mount
#to mount               to fsck   point           type  pass  at boot  options
#      1                   2        3              4     5      6        7
venus:/usr/share/man    -         /usr/share/man  nfs    -     yes       -
```

The fields are:

device to mount	The name of the server separated by a colon (:) and the path name of the remote file resource.
device to fsck	NFS resources are never checked from the client so this field will always be null for NFS resources.
mount point	The default mount point for the file resource.
FS type	Always nfs for NFS resources.
fsck pass	NFS resources are never checked from the client so this field will always be null for NFS resources.
mount at boot	Either yes or no indicating whether the file resource should be mounted when the system enters run level 2, or when the mountall command is issued.
mount options	A comma-separated list of mount options.

Common NFS Error Messages and Possible Solutions

Most NFS problems are discovered through console messages or symptoms on a client.

Error Message:

```
nfs mount: mers:: RPC: Name to address translation
failed - n2a: hostname not found
```

This message can appear during boot or in response to an explicit mount request and indicates an unknown server.

Solution:

Check that the host name in the hosts database is spelled correctly.

Error Message:

```
NFS server mars not responding, still trying
```

This message can appear during boot or in response to an explicit mount request and indicates a known server that is unreachable.

Solution:

1. Check to see if the server is down.
2. Check to see if the network between your machine and the server is down by pinging the server (`ping` *server*).

Common NFS Error Messages and Possible Solutions

Error Message:

```
nfs mount: mars:: RPC: Program not registered
```

This message can appear during boot or in response to an explicit mount request and indicates a server that is reachable but is not running one or more of the server daemons.

Solution:

1. Use `who -r` on the server to check whether it is at run level 3. If it is not, change the run level to 3 with the `init 3` command.
2. Use `ps -e` on the server to check whether the mount daemon and NFS server daemons are running. If they are not, start them with the `/etc/init.d/nfs.server` script and `start` keyword.

Error Message:

```
nfs mount: mars:/opr: No such file or directory
```

This message can appear during boot or in response to an explicit mount request and indicates an unknown file resource name on the server.

Solution:

Check that the directory exists on the server and is spelled correctly on the command line or in the `/etc/vfstab` file.

Common NFS Error Messages and Possible Solutions

Error Message:

```
mount: mount-point /DS9 does not exist.
```

This message can appear during boot or in response to an explicit mount request and indicates a non-existent mount point.

Solution:

Check that the mount point exists on the client and is spelled correctly on the command line or in the `/etc/vfstab` file.

Error Message:

```
le0: No carrier - transceiver cable problem?
```

This message can appear during boot or in response to an explicit mount request and indicates a network problem.

Solution:

Check the physical network connections between your machine and the server including terminators.

Common NFS Error Messages and Possible Solutions

Error Message:

```
stale NFS file handle
```

This message can appear when a process attempts to access a remote file resource and the file handle is out of date.

Solution:

The file resource may have been moved on the server. Unmount and mount the resource again on the client. You may get the message "`nfs mount: mars:/usr/share/man: No such file or directory`." In this case, contact the administrator of the server and ask about the lost file resource.

NFS Command and File Summary

	Server	Client

Commands	`share` *directory*
	`unshare` *directory*
	`shareall`
	`unshareall`
	`dfmounts`
	`/etc/init.d/nfs.server`
		`mount` *server:directory mount-point*
		`mountall -r`
		
		`umountall -r`
		`dfshares` *server*
		`/etc/init.d/nfs.client`

Files	`/etc/dfs/fstypes`
	`/etc/dfs/dfstab`
	`/etc/dfs/sharetab`
	`/etc/rmtab`
		`/etc/dfs/fstypes`
		`/etc/vfstab`
		`/etc/mnttab`

Daemons	`/usr/lib/nfs/nfsd` `/usr/lib/nfs/mountd`	`/usr/lib/nfs/statd` `/usr/lib/nfs/lockd`

Summary

In this lesson, you learned that:

- The NFS environment provides a convenient way to share file resources among a network community of systems.
- The `share` and `shareall` commands are used to make remote resources available for mounting.
- The `unshare` and `unshareall` commands are used to make resources unavailable.
- The `mount` and `mountall` commands are used to access remote resources.
- The `umount` and `umountall` commands are used to unmount remote resources.
- The `/etc/dfs/dfstab` file contains `share` commands that are shared automatically when the system enters run level 3.
- The `/etc/vfstab` file contains mount entries that are mounted automatically when the system enters run level 2.

Exercise 2-1

The purpose of this exercise is to use the DFS administration commands to mount and share resources.

Procedure

Select a partner to complete this exercise. Setup one system as an NFS server to share the `/usr/share/man` directory. Set up the other system as an NFS client to mount this resource. Follow the steps listed below.

1. Become superuser on both systems and add an entry for each host in the `/etc/inet/hosts` file, if necessary.
2. On the server, modify the `/etc/dfs/dfstab` file to share the `/usr/share/man` directory.
3. Execute the `/etc/init.d/nfs.server` script to start the appropriate NFS server daemons.
4. Use the `dfshares` command to prove you were successful.
5. On the client, mount the `/usr/share/man` directory from your partner's machine.
6. Use the `mount` command to identify the newly mounted resource.
7. Use the `umountall` command to unmount all NFS file systems. Use the `mount` command to verify that is it unmounted.
8. Modify the `/etc/vfstab` file to mount the `/usr/share/man` directory from your partner's machine.
9. Use the `mountall` command to mount all NFS resources in the `/etc/vfstab` file. Use the `dfmounts` command to verify that the resource is mounted.
10. Use the `umountall` command to unmount the `/usr/share/man` directory.

Exercise 2-1

11. On the NFS server, remove the `share` command from the server's `/etc/dfs/dfstab` file.

12. On the NFS client, remove the remote resource from the client's `/etc/vfstab` file.

mkdir /man

mount nodename:/usr/share/man /man

To get it functioning properly follow steps in the lab

optional

Using the Automounter 3

Objectives

Upon completion of this lesson, you will be able to:

- Describe three benefits of using the automounter.
- Describe the purpose of each of the three types of automounter maps.
- Set up automounter to read a direct map.
- Describe when the automount daemon needs to be restarted.

References

SunOS 5.1 Administering NFS and RFS, Chapter 6, "How to Set Up the Automounter," and Chapter 7, "Troubleshooting the Automounter"

Introduction

The automounter automatically and transparently mounts file resources, as needed. This means an NFS client system only mounts a remote file resource when a local process attempts to access a file from that remote resource. This provides flexibility and saves considerable time during system startup.

For example, users' home directories for NFS client systems can be made available on an as-needed basis.

In addition, automounter information, unlike the mount information stored in the `/etc/vfstab` file, can be shared across the network using the NIS+ product. This allows for increased consistency across client systems, which can ease an administrator's work load.

Another benefit of the automounter is that it provides the administrator flexibility in designating different servers to provide the same file resources, so that if one server is unavailable, the client can automatically mount the resource from another server.

In this lesson, you learn the concepts and procedures required to set up, maintain, and troubleshoot the automounter. This lesson does not include information on how to use the automounter with the NIS+ environment.

How the Automounter Works

The `/usr/lib/nfs/automount` program is a client daemon that automatically and transparently mounts and unmounts remote directories. The `/etc/init.d/nfs.client` script starts this daemon by default when the system enters run level 2.

File Resources Are Mounted on Demand

When a user program on an NFS client needs access to a remote file or directory that is controlled by the automounter, the local automounter daemon contacts the server's mount daemon to get a file handle for the to-be-mounted file resource. The local automounter daemon writes the file handle (along with other information about the mounted resource) to the `/etc/mnttab` file, the file resource is mounted, and the local kernel uses the file handle during all attempts to access the remote file resource.

File Resources Are Unmounted Automatically

The remote file resource remains mounted for as long as it is being used. If none of the files or directories in the hierarchy are accessed within a specific time-out period, the automounter automatically unmounts the resource.

Automounting Does Not Impact the Server Side of NFS

An NFS server neither knows nor cares whether its shared directories are being accessed through the `mount` command or the automounter. It is therefore unnecessary to do anything on the server to prepare for the automounter beyond sharing the file resources.

Automounter Maps

The automounter uses a series of *maps* to define the file resources to be mounted.

- Automounter maps are implemented as ASCII data files or NIS+ database files.
- Together, these maps describe information very similar to the information specified in the `/etc/vfstab` file for remote file resources.

Setting up the automounter consists of defining the maps and starting the `automount` program.

Basic Map Types

There are three basic map types:

- Master maps
- Direct maps
- Indirect maps

Automounter Map Overview

Master Map Example

```
# cat /etc/auto_master
# Master map for automounter
#
/net            -hosts          -nosuid
/-              /etc/auto_direct
/home           /etc/auto_home
```

Direct Map Example

```
# cat /etc/auto_direct
#
/usr/frame     -ro,soft  mars:/export/framemaker,v3.1a
/usr/local     -ro,soft  jupiter:/export/unbundled
/usr/share/man -ro,soft  earth:/usr/share/man \
               -ro,soft  saturn:/usr/share/man \
               -ro,soft  mars:/usr/share/man
```

Indirect Map Example

```
# cat /etc/auto_home
# Home directory map for automounter
#
lister                  mars:/export/home/lister
kryten                  reddwarf:/export/home/kryten
rimmer                  starbug:/export/home/rimmer
```

The `/etc/auto_master` and the `/etc/auto_home` maps are created by default and include entries for using maps stored in the NIS+ database. (These entries were removed in the above examples.)

Master Maps

The automounter daemon reads the master map, named `/etc/auto_master` by default. This map lists other direct and indirect maps to be read by the automounter as well as general mount options for each listed map.

File format:

mount_point *map_name* [*options*]

The fields are:

mount_point	This can be a placeholder for the mount point's full path name, or the start of a partial path name.
map_name	The name of the direct or indirect map.
options	The general options for the map.

Master Maps

Examples:

- The following master map entry defines a direct map, named `/etc/auto_direct`, with one general option specifying read-only mounting.

  ```
  /-              /etc/auto_direct    -ro
  ```

  The mount point `/-` is a filler that informs the automounter that the full path names of mount points are defined in the direct map.

- The following master map entry defines an indirect map, named `/etc/auto_home` with no general options.

  ```
  /home           /etc/auto_home
  ```

  The mount point `/home` specifies that all mounts listed in the `/etc/auto_home`, map are mounted below this directory.

- The following master map entry defines a mount using the special built-in map named `-hosts`.

  ```
  /net            -hosts
  ```

  This entry specifies that all shared resources from each host listed in the `hosts` database (`/etc/inet/hosts` for machines not using NIS+) are made available under the directory `/net`/*host*.

  For example, if the host *mars* is in the `hosts` database and a user types:

  ```
  $ cd /net/mars
  ```

  all of the file resources shared by `mars` are mounted under the `/net/mars` directory.

Direct Maps

Direct maps specify additional parameters for each file resource to be mounted. These parameters include the full path name of the mount point, the specific options for this mount (any options in this map take precedence over all options in the master map), and the name of the server sharing the resource, along with the path name of the shared resource.

File format:

The fields for direct and indirect maps are in a different order than the fields of master maps.

key [*options*] *location*

The fields are:

key	The full path name of the mount point.
options	The specific options for a given entry.
location	The location of the file resource specified in *server:pathname* notation.

Example:

The following direct map entry specifies that the client mounts the `/usr/share/man` directory read-only from the servers `mars`, `jupiter` or `saturn`.

```
/usr/share/man   -ro mars,jupiter,saturn:/usr/share/man
```

This entry uses a special notation, a comma-separated list of servers, to specify a powerful automounter feature—multiple locations for a file resource.

Indirect Maps

Indirect maps specify the same parameters as direct maps in the same general format. The only difference is that for indirect maps, the first part of the path name of the mount point is specified in the master map. Indirect maps are useful when you want to mount many remote file resources below a common directory. For example, if you wanted to mount the `/export/home/`*username* directories from several servers below the local `/home` directory on an NFS client system.

File Format:

key   [*options*]   *location*

The fields are:

key	The path name of the mount point relative to the beginning of the path name specified in the `/etc/auto_master` map.
options	The specific options for a given entry.
location	The location of the file resource specified in *server:pathname* notation.

Example:

The following indirect map entry, used with the master map entry for `auto_home` described previously, specifies that for every user, the client mounts the `/export/home/`*username* directory from the server `mars` onto the mount point `/home/`*username*.

```
*                    mars:/export/home/&
```

This entry uses the wildcard character (`*`) to match any key, and the substitution character (`&`) to substitute for the current key, at the end of the location specification.

A Common `auto_home` Setup

The `auto_home` indirect map is used to provide a consistent view of home directories across the network, regardless of which machine a user is currently logged into. This supports the possibility of users accessing each other's home directories and logging into their own accounts on each other's machines.

Example Scenario

With the `auto_home` map it is possible to set up a group of machines, so that each machine's home directories are stored under that machine's local `/export/home` directory and all home directories, whether they are local or remote, are available under each machine's `/home` directory.

```
# Home directory map for automounter
#
lister                  mars:/export/home/lister
kryten                  reddwarf:/export/home/kryten
rimmer                  starbug:/export/home/rimmer
#
```

Requirements

All of the machines sharing home directories need to share a common `auto_home` map and `/etc/passwd` entries. The password entries must list all home directories as */home/username* rather than `/export/home/`*username.*

How It Works

This `auto_home` scheme works by mounting remote `/export/home/`*username* directories as `/home/`*username* and providing symbolic links from local `/export/home/`*username* directories to `/home/`*username.*

Restarting the Automounter

When to Restart the Automounter

The most common symptom of an automounter problem occurs when you try to change into a directory that should be mounted by the automounter and receive a message telling you that the directory does not exist. For example:

```
$ cd /export/home/kryten
/export/home/kryten: does not exist
$
```

Before killing the automounter, you should first check the map entries and ensure that the server and resource names are spelled correctly.

How to Restart the Automounter

1. Type the following to stop the automounter:

   ```
   # /etc/init.d/nfs.client stop
   ```

2. Type the following to start the automounter:

   ```
   # /etc/init.d/nfs.client start
   ```

Summary

In this lesson, you learned that:

- The automounter is used to automatically and transparently mount remote file resources.
- The `auto_master` map is used to specify which direct, indirect, and special maps get read by the automount daemon.
- The `auto_direct` map is used to specify the full pathnames and mount options for automatically mounting remote file resources.
- The `auto_home` map is an example of an indirect map, and is used to provide a consistent view of home directories across the network, regardless of where the user is logged in.
- The automount daemon needs to be restarted when making changes to the `auto_master` map and direct maps.

Exercise 3-1

This exercise describes the steps for automounting the AnswerBook™ on-line documentation product from an available NFS server.

The following steps describe how to create a direct map called `/etc/auto_direct`.

(This exercise assumes that the AnswerBook product is stored in the /opt/Solaris_2.2_AB directory on the server. Check with your instructor to see if it is stored in another location.)

1. Become the superuser. Make sure the resource to be automounted is not already mounted. Also remove any entry in your `/etc/vfstab` file for the resource to be mounted.

2. Edit the `/etc/auto_master` file by placing comment symbols (#) in front of the `+auto_master` entry.

   The file should now look like this:

   ```
   # Master map for automounter
   #
   #+auto_master
   /net            -hosts          -nosuid
   /home           auto_home
   ```

3. Now, add a direct map entry so that the file appears as follows.

   ```
   # Master map for automounter
   #
   #+auto_master
   /net            -hosts          -nosuid
   /home           auto_home
   /-              /etc/auto_direct
   ```

   This entry instructs the automounter to look in the `/etc/auto_direct` file for direct map entries.

Exercise 3-1

4. Create a new file called `/etc/auto_direct` and add the following entry for the Answerbook directory you want to automount. Replace *server* with the host name of the server.

```
/opt/Solaris_2.2_AB  -ro   server:/opt/Solaris_2.2_AB
```

5. Since changes to the master map and new entries to a direct map are only read when the automounter is started, you need to restart the automounter for these changes to take effect.

   Restart the automounter by performing the following steps.

```
# /etc/init.d/nfs.client stop

# /etc/init.d/nfs.client start
```

6. Create an `answerbook_setup` script in the `$OPENWINHOME/bin` directory to identify the location of the AnswerBook product.

   a. The format is:

```
AB_CARDCATALOG=${AB_CARDCATALOG}: \
/opt/Solaris_2.2_AB/ab_cardcatalog
export AB_CARDCATALOG
```

   b. Make sure the script is executable by all.

```
# chmod +rx /usr/openwin/bin/answerbook_setup
```

7. Edit your `.profile` file to make sure the `OPENWINHOME` variable is part of your path:

```
PATH=$OPENWINHOME/bin:/usr/sbin:/sbin:/usr/bin:
/usr/ucb:/etc:.
```

   Execute the `.profile`, if necessary.

8. Invoke the application (and the OpenWindows™ environment, if necessary).

```
$ answerbook
```

Exercise 3-1

9. Use the following `ls` command.

```
$ ls -ld /opt/Solaris_2.2_AB
```

Your output should look similar to the following.

```
lrwxrwxrwx 1 root root 22 Jan 3 16:42 /opt/Solaris_2.2_AB -> /tmp_mnt/opt/Solaris_2.2_AB
```

The automounter mounts everything under the `/tmp_mnt` directory (which it creates) and provides a symbolic link from the requested mount point to the actual mount point under the `/tmp_mnt` directory.

Adding a Diskless Client 4

Objectives

Upon completion of this lesson, you will be able to:

- List the two requirements a server must meet in order to support diskless clients.
- Use Administration Tool to add support for a diskless client.
- Name at least three files in the `/etc` directory that Host Manager edits when adding support for a diskless client.

References

Solaris 2.1 System Configuration and Installation Guide,
Chapter 12, "Introducing a Machine to a Network"

Introduction

This lesson describes how to use Administration Tool, specifically the Host Manager, to add diskless client information.

Diskless client support can only be added to a system that was configured as a server using a custom install. This system must have the `/export` and `/export/swap` file systems already created.

Adding a Diskless Client

The following information is needed to add support for a diskless client:

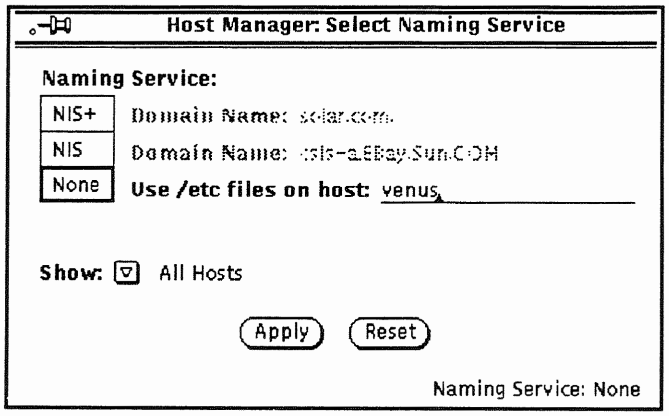
- The client system's host name.
- The client system's Ethernet address.
- The client system's Internet address.
- The client system's region and local time zone.
- The client system's kernel architecture release.

Using Administration Tool

The Host Manager

Follow these steps to add a diskless client:

1. Start Administration Tool and click on the Host Manager icon.
2. Select the None setting to choose the local /etc files instead of a name service, and click on Apply.



Host Manager: Select Naming Service

Naming Service:

NIS+ Domain Name: [illegible]

NIS Domain Name: [illegible]

[✓ None] Use /etc files on host: venus

Show: All Hosts

Naming Service: None

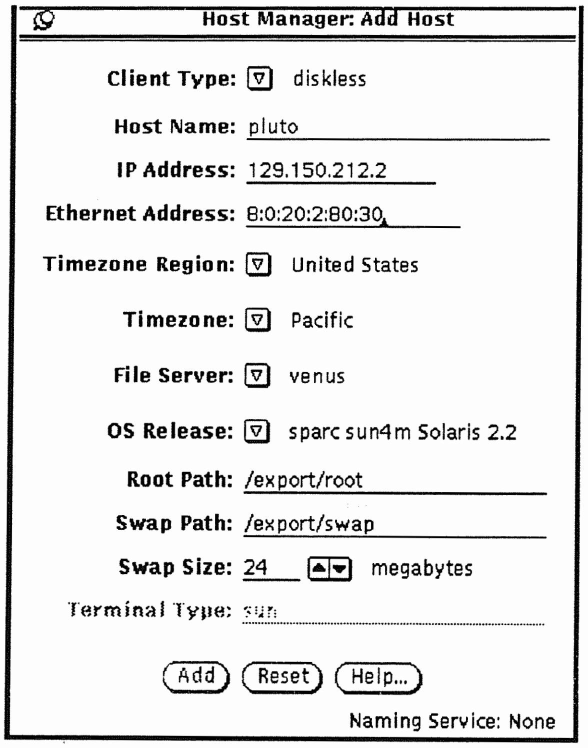
3. Choose Add Host from the Edit menu.

Make sure that the /etc/nsswitch.conf file, that determines name service sources, is consistent with the name service chosen above. (This file is discussed in another module.)

Using Administration Tool

The Host Manager (continued)

4. Choose diskless from the Client Type menu.


Host Manager: Add Host

Client Type:	diskless
Host Name:	pluto
IP Address:	129.150.212.2
Ethernet Address:	8:0:20:2:80:30
Timezone Region:	United States
Timezone:	Pacific
File Server:	venus
OS Release:	sparc sun4m Solaris 2.2
Root Path:	/export/root
Swap Path:	/export/swap
Swap Size:	24 megabytes
Terminal Type:	sun

Naming Service: None

5. Fill out the diskless client form with the information identified on page 4-3.

   The OS Release prompt refers to the client's kernel architecture.

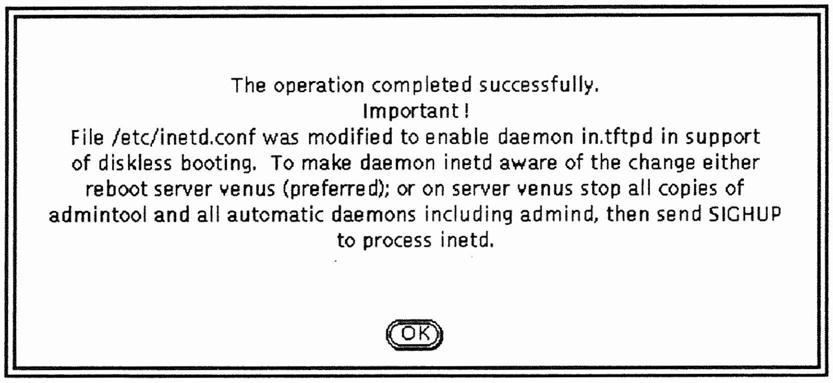
   Leave the default path names alone for the diskless client root and swap areas. Set the size of the swap file by clicking on the arrow keys, if necessary.

6. Click on Add.

Using Administration Tool

The Host Manager (continued)

7. After a few minutes, the diskless client information is added and the following screen appears.



The operation completed successfully.
Important!
File /etc/inetd.conf was modified to enable daemon in.tftpd in support of diskless booting. To make daemon inetd aware of the change either reboot server venus (preferred); or on server venus stop all copies of admintool and all automatic daemons including admind, then send SIGHUP to process inetd.

Click on OK and reboot the system to start the appropriate services.

8. Boot the diskless client system with the following command:

```
ok boot net
```

9. Fill in the following information for the diskless client machine.

   - Geographic Region
   - Time Zone
   - Date and Time
   - Root Password

   Once this information is added for the diskless client system, you will not be prompted again.

Viewing Important Files

The Host Manager adds the information specified on the Add Host form to the appropriate /etc files on the server. There is no need to edit these files manually.

- The /etc/bootparams file contains the name of the server providing the client's root and swap areas.

```
# cat /etc/bootparams
pluto root=venus:/export/root/pluto
swap=venus:/export/swap/pluto dump=:
#
```

- The /tftpboot directory contains information needed for the diskless client to boot.

```
# ls -l tftpboot
total 340
lrwxrwxrwx   1 root     staff          26 Jun  6 16:51
8196D403.SUN4M -> inetboot.sun4m.Solaris.2.2
-rw-r--r--   1 root     staff      161180 Jun  6 16:50
inetboot.sun4m.Solaris.2.2
lrwxrwxrwx   1 root     staff           1 Jun  6 16:50
tftpboot -> .
#
```

The first entry in this listing displays the hexadecimal equivalent of the client's IP address with the kernel architecture as a suffix.

- The /etc/ethers file contains the client's Ethernet address.

```
# cat /etc/ethers
8:0:20:2:80:30 pluto
#
```

- The /etc/timezone file contains the client's time zone information.

```
# cat /etc/timezone
US/Pacific pluto
#
```

Viewing Important Files

An entry for the client's host name and IP address is listed in the server's `/etc/inet/hosts` file.

```
129.150.212.2   pluto
```

The Host Manager also updates the server's `dfstab` file to share the client's file systems at boot time.

```
# cat /etc/dfs/dfstab
#   place share(1M) commands here for automatic execution
#   on entering init state 3.
share -F nfs -o rw=pluto,root=pluto /export/root/pluto
share -F nfs -o rw=pluto,root=pluto /export/swap/pluto
share -F nfs -o ro /export/exec/Solaris_2.2_sparc.all/usr
share -F nfs -o ro
/export/exec/kvm/Solaris_2.2_sparc.sun4m/usr/kvm
share -F nfs -o ro /export/share/Solaris_2.2
#
```

The server's `/export/root` and `/export/swap` file systems now contain root and swap areas for the diskless client.

The client's `vfstab` file in the server's `/export/root` file system is set up to mount the file systems at boot time.

```
# cat /export/root/pluto/etc/vfstab
venus:/export/root/pluto    -    /        nfs    -     -    rw
venus:/export/swap/pluto    -    /dev/swap nfs    -     -    -
/dev/swap   -   -  swap    -   -     -
venus:/export/exec/Solaris_2.2_sparc.all/usr - /usr nfs -  - ro
venus:/export/exec/kvm/Solaris_2.2_sparc.sun4m/usr/kvm  -
/usr/kvm  nfs   -  - ro
venus:/export/share/Solaris_2.2 - /usr/share nfs  -  yes  ro
/proc    -       /proc   proc    -       no      -
swap     -       /tmp    tmpfs   -       yes     -
#
```

You may want to add an additional entry to the file for mounting the user's home directory.

Summary

In this lesson, you learned that:

- Diskless client support is added to the NFS server via the Administration Tool's Host Manager.
- The server must have been installed using a custom installation, with the appropriate kernel architecture selected at that time, in order to define diskless clients using Administration Tool.
- During the custom installation, the server must have created partitions for the `/export` and `/export/swap` file systems, and their sizes must accommodate the number of diskless clients to be added.
- The Host Manager updates the appropriate files on the server to provide services and file systems to the diskless client.

Exercise 4-1

The purpose of this exercise is to practice using Administration Tool to set up diskless client support.

Work with a partner and identify one system as the NFS server and one system as the diskless client.

NFS Server

1. Remove any `vfstab` entry that refers to a remote resource from your partner's system. (*This may exist from a previous exercise.*)
2. Start Administration Tool and click on the Host Manager icon. Select the local `/etc` files as the naming service and click on Apply.
3. Choose Add Host from the Edit Menu to add your partner's host as a diskless client. Use a different host name from your partner's current host name. Add 20 to your partner's current IP address to create the client's IP address.
4. Select diskless client as the client type.
5. Fill out the Add Host form using the appropriate information for the system that will be the diskless client. Click on Add.
6. Reboot the server to start the appropriate services.

Diskless Client

1. Become superuser.
2. After the server system has added the diskless client information, bring your system to the PROM level.

Exercise 4-1

3. Boot your system from the NFS server by using the following command:

```
ok boot net
```

4. Once the diskless client is booted successfully, complete the installation by identifying the client's system information and applying a root password.

5. Log in as the superuser and reboot the system back to your original system configuration.

ok boot le() 1.X Prom Rev
ok boot net 2.X Prom Rev

To get ethernet address on a machine without rebooting:

dmesg

For exercise:

Oldstyle Ether Add: 8:0:20:d:c0:61

New name

Topaz