UnixWorld Online: Tutorial Article No. 007

File Sharing for a Laboratory Network

The Design and Implementation of File Sharing for a Laboratory Network, including the propagation of password files

By James P. Schwar

Questions regarding this article should be sent to the author at *schwarjp@lafcol.lafayette.edu*.

File sharing is a convenient method for centralizing the administration and storage of commonly used information on a computer network. Sun Microsystems' Solaris operating system implements file sharing through their Network File System or NFS. This technique allows the user's directories and associated files to reside on a central workstation or server. The server, through NFS, shares these files and makes them available for mounting on remote (client) workstations. This paper discusses Sun/Solaris NFS as it is used in a teaching/laboratory environment. In addition, the detailed implementation of a method for propagating the server password files to each of the client machines is presented. With NFS and password file sharing, a user can log on to any workstation and access his or her home directory as if it part of the local file system even though this directory is physically located on the server.

Introduction

The Solaris 2.x operating system is based on UNIX System V Release 4. As this article is written, the latest version of Solaris is SunOS 5.4. It is a multitasking operating system where multiple users can work on the same system, each user taking advantage of the processing power of the machine or workstation.

The networking environment of Solaris includes NFS, which allows users to share files and directories among computers on a network. NFS requires one or more servers that provide as their service shared file systems and several clients that mount these shared file systems as if they were local to the clients. Machines that function as servers are the producers of shared file systems while machines that function as clients are the consumers. A Sun workstation, running Solaris 2.x, can act as a server, a client, or both. Each server must be running two programs: `nfsd(1M)` (network file system daemon) and `mountd(1M)` (mount daemon). NFS makes the actual location of the file system irrelevant to the user. The singular advantage of NFS is that it eliminates the need to place copies of commonly used files on each machine by centralizing these files on a server.

Commands for File Sharing and More

The `nfsd` daemon handles client file system requests. A daemon is a program that runs continuously, often asleep, waiting for some event to occur. It then awakens to perform some action such as servicing an NFS request. For example:

```
/usr/lib/nfs/nfsd -a 20
```

starts the daemon over all available connectionless transports for up to a maximum of 20 concurrent requests on the server machine.

The `mountd` daemon is a Remote Procedure Call (RPC) server that answers file system mount requests and reads the file `/etc/dfs/sharetab` to find which file systems are available for mounting by which machines. It runs as a daemon on the server machine. For example, `/usr/lib/nfs/mountd` starts the daemon.

The `share_nfs(1M)` program makes local NFS file systems on the server machine available for mounting by remote (client) machines. For example:

`share -F nfs -o rw /usr2`

makes the `/usr2` directory and all of its files available for sharing with read/write access. Each time the `share` command executes successfully, an entry is made in `/etc/dfs/sharetab`.

The `unshare(1M)` program makes local resources unavailable for mounting by remote (client) machines. For example:

`unshare /usr2`

removes the `/etc/dfs/sharetab` entry for the `/usr2` directory.

The `mount_nfs(1M)` program mounts remote NFS resources by attaching the named resources to the client file system at the location specified by an already existing mount point. For example:

`mount -F nfs -o rw,hard` ***`server`*** **`:/usr2 /usr2`**

will mount the shared `usr2` file system on the server at the local (client) mount point directory, `/usr2`. The server directory (`/usr2`) will appear as if it were a local file system. The server name must be a valid IP address or a symbolic name equated to the IP address as found in `/etc/inet/hosts`. The IP address is a unique internet-wide number used to identify each computer in a network.

This example specifies a ‘‘hard mount,’’ which is the default. If one attempts a hard mount, the client continues to retry the mount operation until the server responds, whereas a soft mount would return an error if the server does not respond within the time-out period. When it is essential that the shared file system is mounted before continuing with the execution of commands, use a hard mount. A successful mount makes an entry in `/etc/mnttab`.

The `umount(1M)` program unmounts file systems and remote resources. For example, `umount /usr2` unmounts the `usr2` file system and removes its entry from `/etc/mnttab`.

The `su(1M)` program allows a user to substitute another user identify, such as the superuser (root) or another ordinary user. In general, one must be logged in as superuser to execute any of the previous commands that implement NFS on the server and client machines.

Some files related to NFS include:

`/etc/dfs/sharetab`

This file lists the resources that can be shared.

`/etc/mnttab`
: This file lists the file systems mounted currently.

`/etc/vfstab`
: This virtual file system table lists all disk slices and file systems available, including a list of file systems that are mounted automatically when the system is booted into multiuser operation.

`/etc/dfs/dfstab`
: The distributed file system table contains commands for sharing resources across a network, where each line in this file is a `share` command line. These commands are run automatically when the system enters run-level three, a multiuser state.

To initiate file sharing under NFS, the server must launch the `nfsd` and `mountd` daemons and share the appropriate file systems. The client must then mount the remote (shared) file systems to complete the operation. A recommended procedure for this would be to install a specifically designed startup script, one for the server and another for the clients. These scripts can be installed in `/etc/rc3.d` so that they are automatically executed at boot or reboot time when the system enters run level three. Example scripts might be as follows:

```
#server script for NFS startup
#
/usr/lib/nfs/nfsd -a 20
/usr/lib/nfs/mountd
#
share -F nfs -o rw /usr2
share -F nfs -o ro /usr/local
#end of script

#client script for NFS startup
#
mount -F nfs -o rw,hard server:/usr2 /usr2
mount -F nfs -o ro,hard server:/usr/local /usr/local
#end of script
```

File Sharing at Lafayette

The Computer Science laboratory at Lafayette College consists of seventeen Sun workstations interconnected using a 10-megabit Ethernet. Sixteen of these workstations act as clients and one workstation is the server. The workstations are currently running Solaris 2.4. The principal shared file systems are: (1) `/usr2`, which is a 963-megabyte slice that holds all the faculty and student (home) directories. This slice resides on the server and is shared read/write for mounting on each of the sixteen clients; and (2) `/usr/local`, which is a 963-megabyte slice that holds the application software (and more). This slice resides on the server and is shared read only for mounting on each of the sixteen clients. User files and application software exists only on the server so that maintaining and backing up these files is a centralized operation. For clients with small hard drives, additional subdirectories--such as `/usr/openwin` and `/usr/share/man`--should be shared read only. The appropriate `share` commands for the server are:

```
share -F nfs -o rw /usr2
share -F nfs -o ro /usr/local

share -F nfs -o ro /usr/openwin
```

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

Each client machine must mount at least the following:

```
mount -F nfs -o rw,hard server:/usr2 /usr2
mount -F nfs -o ro,hard server:/usr/local /usr/local
```

Some of the application software that resides in `/usr/local` includes:

```
gcc         GNU C compiler
g++         GNU C++ compiler
gdb         GNU symbolic debugger
libg++      GNU C++ class libraries
basic       BASIC interpreter
make        GNU make
perl        Practical Extraction and Report Language
smalltalk   GNU implementation of this Object Oriented Language
emacs       GNU implementation of this text editor
TeX         document preparation system
LaTeX       package of TeX scripts
bash        GNU Bourne Again SHell
gnuplot     GNU interactive plotting program
pico        simple text editor
fileutils   GNU file utilities
shellutils  GNU shell utilities
textutils   GNU test utilities
```

The Password Files

The `usr2` shared file system (directory) holds the user subdirectories for faculty and students. Each subdirectory is created when a new user is added to the server. One way of adding a new user is by invoking the `useradd(1M)` command, which creates a new user account and home directory. For example:

```
useradd -d /usr2/student -m -g 60001 -u 6001 student
```

creates the new user account named ‘‘student’’ whose home directory is `/usr2/student`, and whose user id is 6001. The user id or uid is a unique value used by the system at log on. Each new user added to the system creates an entry in the `/etc/passwd` file and an entry in the `/etc/shadow` file. Only one entry per user-ID is allowed in each file. The password file essentially contains the information entered in the `useradd` command. The shadow file contains the encrypted version of the plain-text password that must be entered at login for the user to gain access to the system and home directory. Write access to both of these files and read access for the ‘‘shadow’’ file is restricted to the root (superuser) account on the machine on which the files are created.

In order for the user to log on to a client machine and have access to the home directory in `/usr2`, there must be a scheme for propagating the user portion of the password and shadow files from the server to each of the client machines. Once an account is created on the server this key information needs to appear on all the clients. Thus a user can log on to any (laboratory) workstation using the same account name and password. Because the user home directory is already shared and mounted, the user is logged on as if he or she were on the server (where the password files exist), but use the local resources of the client machine.

Several C programs and shell scripts were developed that extract the user portion of `/etc/passwd` and `/etc/shadow` and copy them to `/usr2` with the file extension ‘‘.net’’. On each client, the original password files are copied to `/etc/passwd.org` and `/etc/shadow.org`.

When a new user is added to the server, a program is run that creates the ‘‘.net’’ files. These changes are propagated to each client by copying the ‘‘.org’’ files to `/etc/passwd` and `/etc/shadow` and then appending the respective ‘‘.net’’ file. This process has been automated through the use of the `crontab(1)` command and `cron(1M)`-table files.

The account named ‘‘sysop’’ is special. This account is added to all machines and becomes part of the original password file. With a user-ID of 6000, user ‘‘sysop’’ can access the ‘‘control software’’ found in `/usr2/sysop` from any workstation. This account is also the owner of this software and is, in effect, the system administrator. For security reasons, user ‘‘sysop’’ must become superuser in order to execute most of these programs. Additional users are added starting with a user-ID of 6001, then 6002, and so forth.

The subdirectory `/usr2/sysop` contains the ‘‘control software’’ that implements password file sharing. Some of the more important items found in this subdirectory are: (1) a file that is invoked by `cron` and, in turn, executes scripts to create and purge the network password files, (2) a file that is invoked by `cron` and, in turn, executes a script to update the client (local) password files, (3) scripts that automate adding and deleting users, (4) the script that updates the client (local) password files, (5) the script that purges the network password files, and (6) a compiled C program that creates the network password files.

The script that updates the client password files must be run on each client machine and be executed if and only if the network password files exist. The script that purges the network password files must be run on the server. The compiled C program that creates the network password files should be run on the server every time the server password files are changed.

The server password files are the ‘‘global’’ password files propagated to all client machines on the network. Only changes to the server password files will find their way to each of the clients. The net result is that a user must log on to the server and execute the `passwd(1)` command. The shared directory `/usr2`, which resides on the server and is mounted on each client, holds the network password files and the user home directories.

Certain commands should be executed immediately after loading the operating system on a client machine. This need be done but once. Namely:

- Make the mount-point directories, `/usr2` and `/usr/local`
- Add the user account named ‘‘sysop’’, with user-ID of 6000 and home directory `/usr2/sysop`
- Assign a password to the new account
- Copy `/etc/passwd`, naming it `/etc/passwd.org` and similarly for `/etc/shadow`
- Place the client script for NFS startup in `/etc/rc3.d`
- Set up the script that updates the client password files
- Use `crontab` to add a `cron`-table entry to automate the execution of the script that updates the client password files

Certain commands should be executed immediately after loading the operating system on the server. This need be done but once. Namely:

- Create the ‘‘sysop’’ account, with user-ID 6000 and home directory `/usr2/sysop`
- Assign a password to ‘‘sysop’’
- Add the ‘‘control software’’ to `/usr2/sysop`
- Place the server script for NFS startup in `/etc/rc3.d`
- Set up the script that purges the network password files
- Use `crontab` to add a `cron`-table entry to automate the execution of the compiled C program that creates the network password files and the execution of the script that purges the network password files

The startup or initialization scripts stored in `/etc/rc3.d` must be properly named. A suggested file name is `S32local.startup`. See the `README` file in `/etc/init.d` for the rules associated with naming startup files. As an alternative to user-written scripts for sharing file systems on the server and mounting file systems on a client, one can add the appropriate entries to `dfstab` and `vfstab` respectively. This technique is discussed, in detail, in Reference 1, *Solaris System Administrator's Guide*.

Sample `cron`-table files and utility scripts for adding and deleting users can be found under Listing 1. The scripts to update and purge password files are shown under Listing 2.

System administrators who want a general-purpose interface to add and delete users without the need to modify the sample scripts should invoke admintool which offers a graphical user interface to the useradd and userdel commands.

The program `upass` is a compiled C program that creates the network password files by copying all user information whose user-ID is greater than 6000 from `/etc/passwd` and `/etc/shadow` to `/usr2/passwd.net` and `/usr2/shadow.net`. See Listing 3 for the source code.

The appropriate commands that invoke `crontab` and schedule execution by the `cron` daemon are:

`crontab` ***`server-cron-table-file`***

and:

`crontab` ***`client-cron-table-file`***

Security

In general, a superuser on a client machine is not allowed access to file systems shared across a network. Unless the server specifically grants superuser privileges, a user who is logged on as root on a client cannot gain root access to the files on the server. NFS implements this security feature by changing the user-ID of the requester to that used for the ‘‘nobody’’ account, generally 60001, whose access rights are the same as those given to the public--or ‘‘other’’ user category-- for a particular file. Earlier versions of Solaris and standard BSD assign 65534 to the ‘‘nobody’’ account.

However, an NFS server can grant superuser privileges on a shared file system on a per-host basis, using the `root=`*`hostname`* option of the `share` command. For example:

```
share -F nfs -o rw,root=hostname /usr2
```

allows the superuser of client `hostname` superuser access to the shared directory `/usr2`.

If a user exists on the client machine and has the same user-ID as a user on the server then the root account on the client can gain access to the NFS-shared files for that user on the client machine, as shown:

```
su
[enter the client root password]
useradd -g 60001 -u uid anyname
su anyname
```

This technique works with any name on the client machine because only the user-ID must match the value found in the server password file. Access to the user's shared directory is then granted. This works with or without automounting.

Access to shared files can be restricted to a selected number of clients by using the `rw=client[:client]...` or the `ro=client[:client]...` option of the `share` command. The path name of the shared directories is available only to the listed clients. When the number of restricted clients becomes large, the single line for the `share` command may exceed 256 characters, which would allow exporting to any host. Under these conditions a netgroup can be used. The details for implementing a network-wide group of hosts and users is discussed in the netgroup(4) Solaris online manual pages.

The user portion of the password file and shadow file are present on the shared directory `/usr2` for a brief period of time. A sophisticated user could access these files and possibly compromise user file security.

In general, this method for propagating password files is best suited to standalone networks and small subnetworks where the system administrator can restrict access to ‘‘root’’ and ‘‘sysop.’’ Those individuals who have root access to a client should be ‘‘trusted users.’’

Summary

File sharing under Solaris NFS consists of invoking the appropriate programs (system commands) and populating certain system tables. The important commands are `nfsd` and `mountd`, which launch the NFS daemons (programs) that implement network file sharing. The `share` and `unshare` commands enable file systems or directories--located on the server--to be available or unavailable for mounting by remote (client) machines. The `mount` and `umount` commands either mount or unmount the shared files systems on the client machines. The `sharetab` file holds a table of file systems shared on the server and available for remote mounting. The `mnttab` file contains a table of all file systems currently mounted on a particular machine. If a client reboots without unmounting the contents of `mnttab` may be in error.

The `hosts` file is a table whose entries consist of the IP addresses that map to one or more symbolic names. There is one entry in this file for each machine.

The propagation of the password files requires additional software and user-generated files. The key commands are `upass`--executed on the server--which creates the shared network password files `passwd.net` and `shadow.net`. The `upass.cli` command--executed on each client--accesses these shared network files for the update. There is one entry in each of the shared network password files for each user. These user-written commands and user-generated files are located in a shared directory available to the server and to all the clients.

An alternative for propagating the network password files would be to use the remote shell (`rsh`) and remote copy (`rcp`) commands, where each client would permit root `rsh` from the server.

Conclusions

This simple, yet relatively effective method of sharing file systems and password files is easy to maintain and implement. Once the appropriate support files are installed on the clients and servers, little or no additional maintenance is required. The `share, mount, nfsd` and `mountd` commands can be incorporated into startup scripts.

The commands needed to propagate the password and shadow files can be run by `cron` using `crontab` to schedule their execution. These important password files will then be automatically updated on the client hosts at regular intervals. Security can be enhanced by restricting physical access and root access to the servers and by limiting file sharing to selected client machines.

The purpose of this technique is to minimize system load and to simplify administrative tasks. It should be particularly effective in a teaching/laboratory environment where a professional network administrator is not available. After over a year of operation at Lafayette College in the Computer Science laboratory, no down time has been experienced as a result of this technique. It should be noted that there are more sophisticated and vastly more complicated methods for sharing files, such as NIS+.

It must be emphasized that this method for propagating password files stresses simplicity at the expense of security. There is a small window in time during which the password files exist in a shared directory. This technique is best suited to standalone networks and small subnetworks where user access to the root and sysop accounts are controlled.

References

1. *Solaris System Administrator's Guide* by Janice Winsor, Ziff-Davis (1993), ISBN 1-56276-080-1.
2. *Solaris Advanced System Administrator's Guide* by Janice Winsor, Ziff-Davis (1993), ISBN 1-56276-131-5.
3. *Guide to Solaris* by John A. Pew, Ziff-Davis (1993), ISBN 1-56276-087-4.
4. *UNIX System V Release 4: An Introduction* by Rosen, Rosinski, and Farber, Osborne/McGraw-Hill (1990), ISBN 0-07-881552-5.
5. *NFS Administration Guide, Solaris 2.4 System Administrator Answerbook*, Sun Microsystems, Inc. (1994).
6. *SunOS Reference Manual, Solaris 2.4 Reference Manual Answerbook*, Sun Microsystems, Inc. (1994).

Edited by Becca Thomas / Online Editor / UnixWorld Online / beccat@wcmh.com
Software tested by John Skinner and Jesse Pollard.

Last Modified: Sunday, 12-Nov-95 19:32:54 PST