Network File System

NFS Servers and Clients - some notes

- The machine which provides the disk files and directories to share is called the **NFS server**.
- The machine which accesses the server is called the **NFS client**.
    * Any machine with a disk is therefore a potential NFS server.
    * Any machine is a potential NFS client.
    * Any number of machines on a network could be accessing one set of files (e.g. manual pages) on one machine.
    * It is possible to restrict access to a defined list of machines.
    * It is possible to encrypt NFS traffic using secure NFS options when mounting.
    * NFS activity can be logged using options to NFS shares.

Network File System

NFS Server Configuration

- The first task is to enable our machines to be NFS servers, then we can proceed to access their files and directories as clients.
- The NFS service is provided, as with many other services, by daemons which can automatically be started at boot time.
- These daemons are:-
  - *nfsd* - this daemon provides the actual reading and writing services. It is written using multi-threading programming technniques, allowing it to handle a specified number of concurrent NFS requests. (16 by default - change this in */etc/default/nfs*)
  - *mountd* - this daemon answers and validates mount requests from remote machines.
- Under Solaris 10, the SMF facility must be employed to enable the NFS services - these are actually enabled by default. (More about this shortly)
- There is a very simple condition which causes these daemons to be started at boot time, and thus automatically configure the machine as an NFS server:-
  - The file */etc/dfs/dfstab* must exist, and must contain valid *share* commands.
- Overleaf is an example *share* command.

Network File System

NFS Server Configuration

- *share* commands allow a server to make it's disk resources available to other machines, for example:-

share -F nfs /export/home

- This means:-
  - * - any known host can mount from */export/home* or anywhere underneath.
- To view resources that are currently being shared:-

share

\- /export/home rw ""

- This shared resource could be made permanently available by placing the share command in the */etc/dfs/dfstab* file.
- If you have placed share entries in */etc/dfs/dfstab*, and want to start NFS without rebooting:-

svcadm enable nfs/server

- You can also use svcadm to restart or disable NFS:-

svcadm restart nfs/server (Will not work if nfs disabled)
svcadm disable nfs/server

- Note that if you omit the *-F nfs* option to the *share* command (or any NFS-related command which supports *-F FSType*) the the first entry in the file */etc/dfs/fstypes* is taken as the default filesystem type.
- The happens to be *nfs* in all Solaris versions.

Network File System

The share command

- The command has several options, for example:-

share -F nfs -o ro,rw=harwell:fa /usr

  * This shares */usr* read-only to all machines except *harwell* and *fa*, to which it is shared read-write.

share -F nfs -o rw=harwell:fa,root=harwell /usr

  * This shares */usr* read-write to only *fa* and *harwell*, with root access granted to the root account on *harwell*.
  * **Sharing as root is dangerous; don't be tempted to use this too frequently**!

share -F nfs -o ro,anon=0 /extra/sol10

  * Shares */extra/sol10* read-only, with the UID for unkown user ID access as 0, and logs accesses.
  * The *anon=0* effectively gives all machines full root access to this share, but with *ro* it's read-only.

- If the machine is already running as an NFS server, we can dispense with the "-F nfs" option when entering *share* commands on the command line:-

share /opt

  * Default access is read-write to all known hosts.

- The *root* option must be given a list of machines, or an error message is generated:- "share_nfs: missing root list"

Network File System

The share command - further options

* To allow mounting at a top-level directory structure only (pre-Solaris 10 clients only):-

share -F nfs -o nosub,ro /software/conf_files

* Sharing with all clients that are part of a DNS (or LDAP) domain call acme.com:-

share -F nfs -o rw=.acme.com /export/home

* Sharing with all clients that are part of a subnet:-

share -F nfs -o rw=@192.168.200.0 /export/home

* Other network specifiers are as follows:-

@eng_net - Network name as in */etc/networks*
@192.168.200.32/27 - Subnet 192.168.200.32 with 27-bit mask (as in CIDR notation)

* Note that zero values are assumed in the low-order bytes, for example 192.168.200 is equivalent to 192.168.200.0, and 130.100 is equivalent to 130.100.0.0.

* Negative values may also be specified:-

share -F nfs -o rw=honda:-yamaha:@support_net /extra

* When evaluating the list, the first match is used, so if *yamaha* is part of *support_net* (a network), it is denied access, as the *-yamaha* entry is seen first.

- Another entity which can be specified is a *netgroup*, which is a group of hosts defined in the NIS or NIS+ name services - more about this soon.

Network File System

The share command - file systems / hard slices

- A couple of points worth noting concerning the way NFS treats file systems/hard slices:-
- 1. The *share* command limits the sharing to the current file system (i.e. hard slice).
  - ❖ Therefore, if you share "/" you are sharing only the root slice.
  - ❖ The fact that "/" on the server has other slices mounted within it does not mean that you will see these mounts at the NFS client end.
  - ❖ You will see the empty mount points only.
  - ❖ Suppose we wished to share all local hard slices; a *df -k* would quickly show which slices we would need to share:-

Filesystem	kbytes	used	avail	capacity	Mounted on
/dev/dsk/c0t3d0s0	67663	23359	44237	35%	/
/dev/dsk/c0t3d0s6	529294	406096	122669	77%	/usr
/proc	0	0	0	0%	/proc
fd	0	0	0	0%	/dev/fd
/dev/dsk/c0t3d0s7	81303	3089	78133	4%	/export/home
/dev/dsk/c0t3d0s5	192423	107988	84243	57%	/opt
swap	95160	204	94956	1%	/tmp

  - ❖ Those directories mounted from hard slices *(/, /usr, /opt, /export/home*) are the local file systems available to be shared

Network File System

The share command - file systems behaviour

- 2. Once a directory is shared, you cannot then share a directory (in the same slice) below it; for example:-
  - If */usr* is shared, */usr/man* cannot then be shared separately. (If it is on the same slice as */usr*, which it probably is). You'll get an error message:-

share_nfs: /usr/man: parent-directory (/usr) already shared

  - You may, for example, wish to do this to apply different share permissions such as read-only or read-write.
  - You will have to share each directory below */usr* on an individual basis if you wish to do this.
  - You can share a subdirectory, if it is on a separate slice; for example, if you have shared "/", you can still share */opt* and */usr*.
- **Remember**: to make the shared file systems available permanently, place the *share* commands in */etc/dfs/dfstab*.
- Details of file systems currently being shared are held in */etc/dfs/sharetab*, but this file is maintained by the system.

Network File System

The share command - common options

- Here is a list of the most common options to *share*:-

option	Meaning	Example
rw	Read-Write access granted according to normal UNIX permissions.	**share -F nfs -o rw=cod:plaice:pike /usr** Shares /usr to cod, plaice and pike in full read-write mode. No other hosts are allowed access, and root write access is not granted.
ro	Read-only access granted.	**share -F nfs -o ro /usr/local** Share /usr/local to all known hosts read-only
root	Allows the root account on specified remote hosts to access the file system with full root permissions.	**share -F nfs -o root=cod:plaice:perch /** Shares the root file system (/) with full root access to cod, plaice and perch. The root option must be given a host list.
rw and **root**	Read-Write access granted according to normal UNIX permissions, and root access to defined hosts.	**share -F nfs -o rw=cod:pike,root=cod /opt** Shares /opt with cod and pike, with cod allowed full root access.
anon	Treat remote accesses from unkown UIDs as though they were the UID specified by anon= Default is 60001 (nobody)	**share -F nfs -o anon=75 /data1** Shares /data1 to all known hosts. Read-Write access granted according to normal UNIX permissions. Unkown UIDs treated as 75. (You can give root access to all machines very simply by specifying anon=0, and prevent access to unknown UIDs with anon=-1)
-d desc	Allows a description of the resource to placed on the command line.	**share -F nfs -d "Home directories" \ /export/home** The description can only be seen at the server end!
network access	Shows the use of network names and numbers in share directives.	**share -F nfs -o \ rw=@10:@192.168.200.32/27:@eng_net \ /var/logdir**

Network File System

The shareall command

- If you have updated */etc/dfs/dfstab* without running the *share* commands manually, you can make them come into force by running:-

shareall

Unsharing

- To take a resource out of the share list, simply *unshare* it:-

unshare /opt

  * Unshares the /*opt* directory

unshareall

  * Unshares all currently shared directories.

- Unsharing, or changing share options, comes into effect immediately.
- For example, if I am accessing an NFS mounted file system, and it is then unshared on the server, I will get the message:-

Stale NFS file handle

- .. if I try to access it.
- If the share option on a writeable directory is changed to read-only:-

mkdir: Failed to make directory "projectx"; Read-only file system

Network File System

Exercise

- Make file systems available for mounting by other machines as follows:-
    * Root slice to any machine read-write.
    * /*usr* available to any machine in read-only mode.
- Now start the NFS server processes by running the */etc/init.d/nfs.server* start-up script.
- Create a new directory under the */export/home* slice (or whatever slice your home directory space occupies) called *root_dir*.
- Now make this single directory available to just the other machines on the network, and with root access to all of them.
- Ensure that the NFS server processes are started when the machine reboots.

Network File System

The mount command

- NFS resources are made available on the client by using the *mount* command.
- (It is common for remote directories to be automatically mounted, and we will look at this later.)
- The *mount* ommand has many options, and can only be executed by the superuser:-

mount -o bg,intr,ro dibble:/usr/man /usr/man

bg	Retry in background later if mount fails
intr	Allow keyboard interrupt on hard mount
ro	Do not allow write access to users, regardless of UNIX file permissions

- and some more....

hard	Keep trying until server responds(default) or the *retry* value is reached.
soft	Give error message if server doesn't respond
retry n	Number of times to retry the mount (def:10000)
nosuid	Setuid execution not allowed.
sec=dh	This provides secure NFS, requiring the use of passwords based on public key encryption using the Diffie-Helman encrytion technique. Other modes are also available using the Kerberos authentication mechanism. The *sec* option is also available under the *share* command. The sec option is best used with a name service, which makes it a lot easier to store password credentials

Network File System

The mount command (cont'd)

- The *mount* command final argument specifies the mount point in the local file system.
- This is a directory, usually empty. (**Though not necessarily**)
- If you have permission to mount */usr*, you can mount from any point under it, for example */usr/man* and */usr/local*.
- You can also mount onto any mount point, including another NFS-mounted file system directory.
- You MUST mount from the server (i.e. the machine with the physical resource attached), not from a client.
- If machine *A* has a local hard partition */usr* with */usr/man* mounted NFS, and you mount */usr* from *A*, you will not see */usr/man*- it will just be empty.

Network File System

The mount command - multiple machines for failover

- It is also possible to mount from a list of machines.
- This is known as file system replication:-

mount -o ro cod:/extra,carp:/extra /extra

  - * Mounts */extra* from *cod*, and automatically fails over to *carp:/extra* if *cod* is unavailable.
  - * File systems must be mounted read-only, and must be identical (see *rsync* for maintaining identical directory structures)

- If the mounts from cod and carp were from identical paths, we could have specified the mount command like this:-

mount -o ro cod,carp:/extra /extra
df -h /extra

Filesystem	size	used	avail	capacity	Mounted on
cod:/extra,carp:/extra	32G	2.0G	30G	7%	/extra

Network File System

The umount command

- To unmount filesystems, use *umount*:-

umount /usr/man

- If you get the message:-

umount: Device Busy

- You probably are *cd*'d into */usr/man* - move out and try again.
- Or there is a process running that is using the path */usr/man.*
- Sometimes the *umount* will **insist** that the device is busy; try the following:-

umount -f /usr/man

- A couple of useful commands are available to check on open files:-
  - * *fuser* - identifies users of open files and devices; run it with a mount point as an argument.
  - * *lsof* - an Open Source command, available for download at *www.sunfreeware.com* Lots of options with very detailed ouput.
- To unmount all remote filesystems:-

umountall -r