NFS over TCP/IP

NFS across TCP/IP Transport in HP-UX Release 11.0

Thomas McNeal and Lisa Liu
Hewlett-Packard
19420 Homestead Road
Cupertino, CA 95014
(408) 447-2300
FAX: (408) 447-3660
tom_mcneal@hp.com, lisa_liu@hp.com

When NFS was first developed in the 1980's, the User Datagram Protocol, or UDP, was chosen as the simplest, most efficient, and most reliable transport protocol for client/server file system transactions in a local area network. At the time, the remote procedure call protocol (RFC 1050, later updated by RFC 1057) implemented for NFS specifically assumed that the transport protocol would be UDP. In the early 1990's, when developing the newer NFS protocol know as Protocol Version 3 (RFC 1813), a "transport independent" version of RPC was developed, in order to support the eventual use of connection oriented protocols, such as TCP/IP.

The TCP protocol is now supported by many implementations of NFS, and is considered a desirable component of the increasingly distributed applications now available on Solaris, AIX, and other UNIX platforms, as well as HP-UX. Since development environments are expanding into widely dispersed locations, particularly with 3rd party applications, the guaranteed connectivity offered by the TCP protocol is becoming a virtually mandatory component of NFS.

This paper will discuss the modifications required to support NFS over the TCP/IP transport protocol, as well as our implementation of "NFS/TCP" within the context of HP-UX Release 11.0. We will show that this implementation allows us to maintain binary compatibility, even in the face of kernel-intrusive applications, while preserving the excellent UDP performance results released on 11.0 in December, 1998. We will show the modifications necessary to a few commands and daemons in order to support TCP, and will discuss the internal system behavior imposed upon NFS and RPC by the differences between connectionless and connection oriented protocols such as UDP and TCP.

Finally, we will review the additional HP-UX user space modifications in release 11.0 which are necessary in order to support the delivery of NFS/ TCP via a general release HP-UX patch without affecting those customers who have no desire alter their transport environment utilized by NFS.

Impact of UDP on NFS behavior

The connectionless behavior of UDP has required RPC to manage individual messages, requiring confirmation of message receipt, in addition to error recovery from delivery problems. On the client, this is particularly evident in the complex timeout and message retransmission algorithms necessary to handle network traffic across a connectionless protocol. An equivalent situation exists on the server, with its requirements to keep track of and reply to incoming messages - it

must reassemble messages correctly if message fragmentation occurs across the wire, and must manage duplicate requests which inevitably occur when the client is unaware that the server has successfully received its messages.

The timeout and retransmission mechanism on the NFS client is particularly complex, involving timeout and error handling loops in both the NFS and RPC layers. The general behavior is that a specific timeout length may be specified during mount time, with a default that may differ from one vendor to another. (In the case of HP-UX release 11.0, and conforming to the Solaris releases, the timout defaults are given in table 1-1). In addition, the number of attempts made to reach the server can be defined at mount time (again, in conformance with SUN, the default number of retransmissions after the first attempt is 5). If the first attempt fails, the length of the timeout doubles, up to a maximum of 20 seconds, until the defined number of retransmissions has been made. If all of these attempts have been unsuccessful, the client will mark the kernel data structures to show that the server is "down", and a message ("NFS failed for server xyz", on HP-UX release 11.0, or the more familiar "NFS Server not responding" on earlier releases) is displayed at the console and sent to the system log.

Call Type	Minimum timeout in seconds	Calls
Lookup	.75	null, getattr, lookup, access, readlink, fsstat, fsinfo, pathconf
Read	.85	setattr, read, readdir
Write	1.25	write, create, mkdir, symlink, mknod, remove, rmdir, rename, link, readdirplus, and commit

Table 1.1 – Minimum UDP Timeout Values

With soft mounts, once a server has been marked "down", any further NFS requests to that server will be attempted only once, if at all, and a failure will generate another error message at the console. Hard mounts will continue to try the defined number of times before posting the error message. If the maximum timeout length had not previously been reached, the length of the timeout will continue to double with each attempt until it hits maximum value of 20 seconds. Once the client succeeds in reaching the server, the "down" flag is removed, and the "NFS Server OK" message is displayed at the console. Starting from that point, all timeout parameters float back to their original value, using a Van Jacobsen algorithm to continually adjust the operational timeout values down until they once again reach the minimum value given above.

On the server, the fundamental problem with a connectionless protocol is that the client may never receive the reply from the server that a request was successfully received and acted upon. The client will then continue to send the same request, as long as no reply is received. In this case, the server handles these duplicate requests differently depending upon whether or not they are "idempotent". An idempotent request, such as the "getattr" attribute query, may be executed any number of times without changing the state of the server; in that case, depending upon the implementation, the server may simply execute the request again. For the requests which are not idempotent, or which are more expensive to execute than to throw away, a duplicate request cache is consulted in order to determine if the request had been received and executed previously.

In contrast to UDP, the guaranteed connectivity of TCP removes the need to manage individual messages within the client, but the server continues to utilize the duplicate request cache, in order to allow requests encountering permission denials to be reattempted once the client's credential caches are updated. This is highly dependent upon the security flavors implemented on the platform in question, and is not generally used in HP-UX.

Managing TCP connections

Instead of managing messages in the UDP environment, NFS must be able to manage TCP connections. In an abstract way, this provides some stateful knowledge to the server, since the server then knows which clients are attached to it, but this is invisible at the NFS layer, and so NFS retains its stateless nature regardless of the transport type.

Within the STREAMS environment, a new module, RPCMOD, will provide much of the connection management. In both the client and the server environment, this module is pushed onto the Streams stack, and will, on the server, communicate directly with kernel components of the NFS Daemon. Since this is a configured STREAMS module, you will see its name (rpcmod) - along with the STREAMS UDP module used by the NFS Server Daemon (nfsm) – in the system configuration file used to build the HP-UX kernel (***/stand/system***).

On the client, an RPC request to the server will result in a connection request to the server, using a client structure to manage the client's requests and responses through the STREAMS stack. Each client can have maximum CLNT_MAX_CONNS number of connections to each server. The default value of CLNT_MAX_CONNS is 1. The total maximum number of connections on the client is the number of NFS servers multiplied by CLNT_MAX_CONNS. For example, if CLNT_MAX_CONNS is 5 and there are 10 NFS mount across 3 servers, this client may have up to 15 connections.

Figure 1.1 shows the sequence of events occuring when a connection is established. All types of events noted are defined in the transport header file, ***/usr/include/sys/tihdr.h***, and simply consist of external and internal STREAMS requests, along with the acknowledgements of each.

A connection remains in the client until it becomes inactive. "Inactive" is defined as either "idle"or "disconnected". By default, the idle time is defined to be 5 minutes, meaning that if there is no outbound request on this channel for more than 5 minutes, the system tears down this connection by sending a close request to the stream head. The stream subsystem performs all the close functions defined for each downstream module, and eventually tears down the whole stream stack.

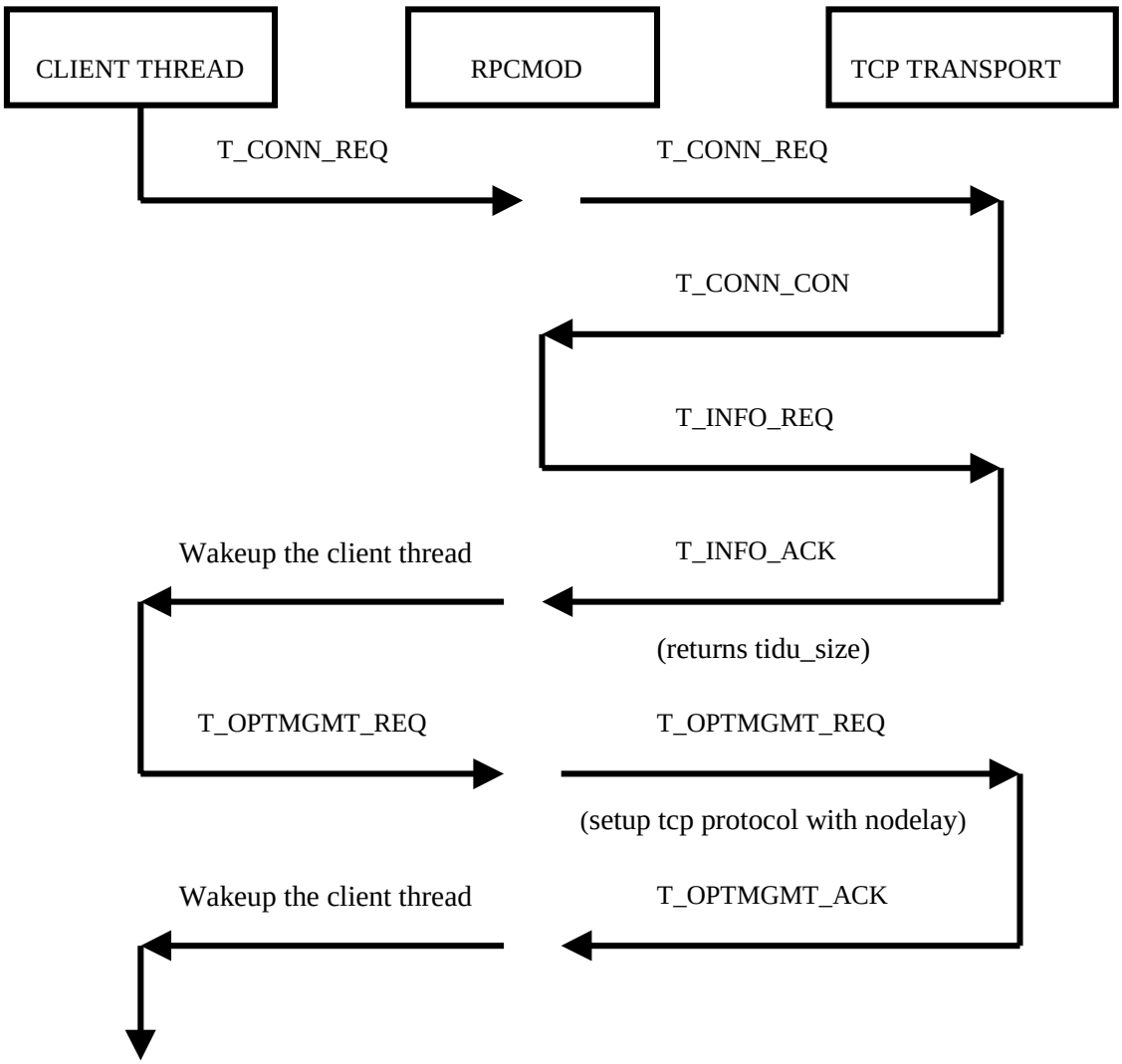

Figure 1.1 – Establishing a TCP connection on a NFS client

On the server, a request for a TCP connection will be identified correctly if the service has been advertised in the ***/etc/services*** file, with the entry "nfsd 2049/tcp". When a request for a TCP connection comes from a client, the NFS Daemon listening for TCP requests (by polling the entries in a poll array) will open a connection to the client, using the STREAM stack devoted to TCP.

The NFS TCP server can support up to the maximum number of connections specified in the nfsd command line with the -c option. The default number of connections is unlimited. Each of the connections has an entry in a poll array. The nfsd daemon polls on the entires in the poll array for events and errors.

The poll array has an initial size of 64 entries. When the total number of connection grows, the system re-allocates a larger array to accommodate more connections. When the nfsd user thread receives a T_LISTEN event, it opens a new connection. If the system has already opened the maximum number of connections specified on the command line, it drops the oldest connection before accepting the new connection

A Server's connection is torn down by the NFS daemon, nfsd, when one of the following events happens:

- When the RPCMOD detects an idle connection, it sends a signal to the NFS daemon. The NFS daemon closes this connection accordingly.
- The NFS daemon receives a connection request after it has reached the maximum number of connections allowed. It finds the oldest one from the polling list and closes this connection.
- The NFS daemon receives a disconnecting event or unrecoverable error event. Its response to such events is to close the connection.

NFS Performance modifications in HP-UX 11.0

Preserving the SPECsfs93 (also known as SPECsfs 1.1) benchmark results was more difficult than preserving compatibility, since the NFS layer UDP pathways needed to continue using the current RPC pathways which contained the benchmark modifications. This was important on the server, where the benchmark results are measured, but not so important on the client, which had no real modifications related to SPECsfs93.

In the HP-UX 11.0 server (as patched with the performance modifications available in December 1998), the kernel structures handling incoming traffic on port 2049, the standard NFS server port, were cloned and distributed per processor, as seen in figure 1.2. This called for replicating the STREAMS stack which handled UDP transport traffic, in addition to assuring that enough NFS daemons existed to service each stack.

Several other enhancements were important in achieving the outstanding benchmark numbers first published in June 1998, just prior to the closure of the benchmark by the SPEC consortium, in favor of the SPECsfs97 benchmark. These enhancements included:

- Task stealing by the kernel component of the NFS daemons, in order to prevent contention at a single processor if all daemons assigned to that processor are already busy.
- Avoidance of some buffer copies in the NFS Server's read pathways by taking advantage of the ability to modify a buffer created at the driver level into a STREAMS transport buffer.
- Enhanced hashing functions for several internal tables.
- Enhanced vnode synchronization within the underlying file systems (both HFS and VxFS).
- Efficient performance based optimization, allowing the HP-UX compilers to create more efficient code paths by providing previous code path analysis at compile time.

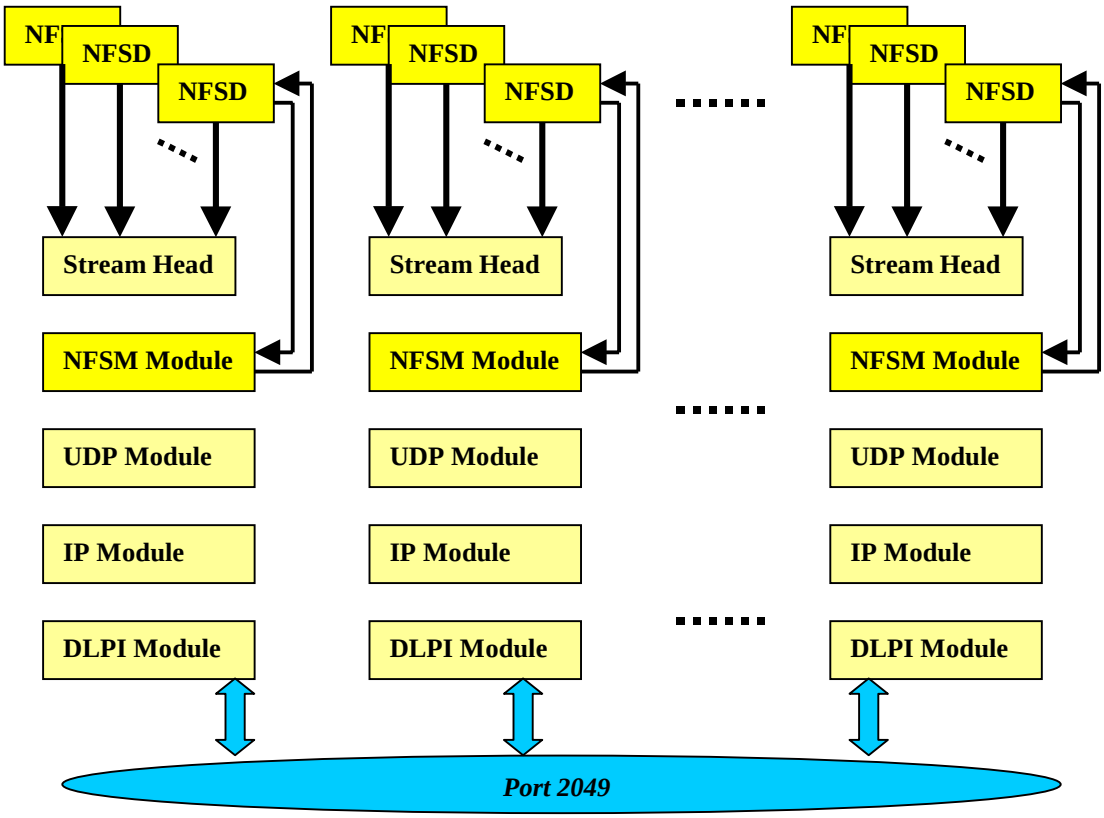

Figure 1.2 – NFS Server UDP Transport Stream Stack

HP has now begun analyzing and measuring the latest NFS benchmark, SPECsfs97. This benchmark exercises the Protocol Version 3 (PV3) of NFS, as well as the older PV2 protocol used in SPECsfs93. The newer benchmark will also change the mix of operations executed by the load generating clients, so the absolute numbers of operations per second (IOPS) and the response time of the server should only be compared with other SPECsfs97 results, and not with the SPECsfs93 results. In addition, the mix of operations will differ between the PV2 component and the PV3 component of SPECsfs97, so the IOPS of the two protocols should also not be compared with one another. The only valid comparisons should be between various NFS vendor's platforms using the same protocol - PV2 or PV3. HP's SPECsfs97 benchmark values on the L-class and the N-class platforms, running PV2 and PV3 across both UDP and TCP transport will be available at conference time (April 2000).

With the additional support of NFS over the TCP/IP protocol, an additional STREAMS stack will be created for each TCP connection created, as shown in figure 1.3. The number of connections that can be supported by the NFS daemon is specified by the "-c" option of the nfsd command. By default, the number of connections is unlimited.

Each connection may have maximum of 10 kernel threads serving requests in the kernel space. The actual number of kernel threads is dynamically adjusted by the system according to the system load.

Each connection is associated with a stream stack headed by the RPCMOD module. The nfsd program communicates to the stream through the stream head, but the kernel nfsd thread, known as KRPC, communicates to the stream by passing packages between the RPCMOD module and KRPC directly, bypass the stream head.

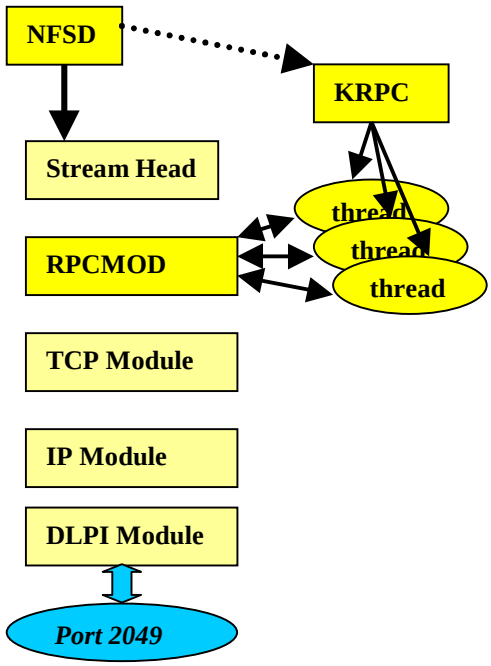

Figure 1.3 – NFS TCP Transport Stream Stack

Delivering Support for NFS/TCP

In order to deliver NFS support for TCP transport connections in HP-UX Release 11.0, a method for limiting the impact of the transport upon system management and system performance was required. It was determined that since NFS/TCP could only be delivered in a general release patch on 11.0 (it will be supported in the first release of HP-UX 11.11), it should be delivered so that it had no impact upon the unsuspecting user and/or system administrator.

This capability will therefore be delivered but not activated in a general release patch form as of March, 2000. It will remain inactive – that is, neither a server nor a client will attempt to establish TCP connections – unless activated by a command, ***/sbin/setoncenv***, with the specific parameter "NFS_TCP". This will allow the system administrator to configure the NFS client and/or server to attempt and complete TCP connections.

The system configuration will follow the methodology used in HP_UX release 10.20 (starting with the May 1999 HP-UX Networking ACE releases), as well as HP-UX 11.0, where the NFS configuration file, ***/usr/rc.config.d/nfsconf***, contains environment variables which determine what protocols and products are supported by the system. As with the AUTOFS variable, for example, which tells the system whether or not AutoFS is a supported product, the NFS_TCP variable will tell the system whether or not to allow NFS to use TCP transport.

As delivered, the HP-UX 11.0 general release patches for NFS will not alter the value of the NFS_TCP environment variable, if it exists at all. If the system administrator wishes to activate the usage of NFS across TCP/IP transport, she must set the NFS_TCP variable to 1 – either by

means of the ***/sbin/setoncenv*** activation command, or manually – and then stop and restart the NFS client and server, either with ***/sbin/init.d/nfs.server*** and ***/sbin/init.d/nfs.client*** command, or by rebooting the platform. Depending on the value of the NFS_TCP environment variable, the client mount command and the server NFS daemon will determine, at startup time, whether or not to allow any communication across TCP transport. Table 1-2 gives the name and the location of the environment variables which can be modified by ***/sbin/setoncenv***

Variable	Value	Location
AUTOFS	0 | 1	/etc/rc.init.d/nfsconf
AUTOMOUNT	0 | 1	/etc/rc.init.d/nfsconf
NFS_SERVER	0 | 1	/etc/rc.init.d/nfsconf
NFS_CLIENT	0 | 1	/etc/rc.init.d/nfsconf
NFS_TCP	0 | 1	/etc/rc.init.d/nfsconf
NIS_CLIENT	0 | 1	/etc/rc.init.d/namesvrs
NISPLUS_SERVER	0 | 1	/etc/rc.init.d/namesvrs
NISPLUS_CLIENT	0 | 1	/etc/rc.init.d/namesvrs
NIS_MASTER_SERVER	0 | 1	/etc/rc.init.d/namesvrs
NIS_SLAVE_SERVER	0 | 1	/etc/rc.init.d/namesvrs
PCNFS_SERVER	0 | 1	/etc/rc.init.d/nfsconf
START_MOUNTD	0 | 1	/etc/rc.init.d/nfsconf

Table 1.2 – ONC+ Environment Variables

Changes to the NFS Server daemon command

Aside from the product configuration mentioned above, very few changes are necessary to support NFS transactions with the TCP/IP protocol. As mentioned, the NFS Server daemon, ***/usr/sbin/nfsd***, has been modified with additional parameters which, among other transport oriented features, allows the designation of the maximum number of connections. The new man page, which will be delivered with the NFS daemon,is attached in the appendix.

As part of the support requirements for NFS across TCP, the fact that port 2049 supports TCP traffic must be advertised in the service file, ***/etc/services***. This advertisement will be added to the service file whether or not TCP support is activated. With this advertisement, the NFS Server will recognize and accept incoming requests for TCP connections; otherwise, even if the NFS configuration has been set to activate TCP, the request will be denied because the service is ostensibly unavailable.

As with the current performance enhancements, the NFS Server daemon will replicate itself at least as many times as there are processors on the Server, regardless of the number of server daemons specified in the NFS configuration file. If support for NFS over TCP has not been activated, the total number of NFS daemons managing UDP traffic will be some multiple of the

number of processors. If TCP support has been activated, then one more NFS Server daemon process will be initiated. This daemon, as seen in figure 1.3, will launch a kernel component used to manage kernel threads used to handle a particular connection's traffic; The kernel component will wait for incoming TCP traffic on port 2049, and will launch kernel threads to handle that traffic.

Changes to the NFS Mount command

The ***mount*** command will contain an additional NFS specific option which specifies the transport protocol which is desired. As with SUN, use of the TCP/IP protocol will be the default, with UDP being the alternative if the server does not support TCP. If the command specifically calls for the UDP protocol, only that protocol will be requested, and no other request will be made regardless of the server's response.

If the client has not been configured to use TCP, then UDP will be the default (and only) protocol, and attempts to use TCP transport by specifying it in the mount options will generate an error.

The actual variable which must be used with the protocol option (***-o proto=tcp***, or ***–o proto=udp***) will be the "netid" specified in the protocol configuration file used by transport independent RPC, ***/etc/netconfig***.

Conclusion

When activated, the underlying support within NFS required by the TCP protocol should be unseen by the application user, and should only be visible to the system administrator through minor changes to commands and daemons, and through entries in supporting files required to advertise and support the TCP protocol. Its presence has been anticipated by the implementation of the Transport Independent Remote Procedure Call components of ONC+ in the initial release of HP-UX 11.0, and the architectural design within the HP-UX kernel has made every effort to provide a smooth transition to general usage of NFS across the TCP/IP transport protocol.

References

"*The Mystery of NFS Timeouts*", Van Co, August 1999

"*NFS/TCP Connection Management in HP-UX*", Lisa Liu, August, 1999

"*Implementing NFS over TCP/IP Transport in HP-UX*", Tom McNeal & Lisa Liu, HPWorld, 1999

Appendix 1

`mount_nfs(1M)`

NAME

mount, umount - mount and unmount an NFS file systems

SYNOPSIS

/usr/sbin/mount [-l] [-p|-v]

/usr/sbin/mount -a [-F nfs] [-eQ]

/usr/sbin/mount [-F nfs] [-eQrV] [-o specific_options]
{host:path|directory}

/usr/sbin/mount [-F nfs] [-eQrV] [-o specific_options]
host:path directory

/usr/sbin/umount -a [-F nfs] [-h host] [-v]

/usr/sbin/umount [-v] [-V] {host:path|directory}

DESCRIPTION

The mount command mounts file systems. Only a superuser can mount file systems. Other users can use mount to list mounted file systems.

The mount command attaches host:path to directory. host is a remote system, path is a directory on this remote system and directory is a directory on the local file tree. directory must already exist, be given as an absolute path name and will become the name of the root of the newly mounted file system. If either host:path or directory is omitted, mount attempts to determine the missing value from an entry in the /etc/fstab file. mount can be invoked on any removable file system, except /.

If mount is invoked without any arguments, it lists all of the mounted file systems from the file system mount table, /etc/mnttab. The umount command unmounts mounted file systems. Only a superuser can unmount file systems.

OPTIONS

-r Mount the specified file system read-only.

-o specific_options
Set file system specific options according to a comma-separated list chosen from words below.

rw|ro resource is mounted read-write or read-only. The default is rw.

suid|nosuid Setuid execution allowed or disallowed. The default is suid.

remount If a file system is mounted read-only, remounts the file system read-write.

bg|fg If the first attempt fails, retry in the background, or, in the foreground. The default is fg.

quota Enables quota(1M) to check whether the user is over quota on this file system; if the file system has quotas enabled on the server, quotas will still be checked for operations on this file system. The default is quota.

noquota Prevent quota(1M) from checking whether the user exceeded the quota on this file system; if the file system has quotas enabled on the server, quotas will still be checked for operations on this file system.

retry=n The number of times to retry the mount operation. The default is 1.

vers=<NFS version number>
By default, the version of NFS protocol used between the client and the server is the highest one available on both systems. If the NFS server does not support NFS Version 3, then the NFS mount will use NFS Version 2 .

port=n Set server UDP port number to n (the default is the port customarily used for NFS servers).

proto=<transp> Use the transport protocol <transp> for this mount. Valid values for <transp> are tcp (connection-oriented), if support for tcp has been activated by the setoncenv command, and udp (connectionless). If activated, the default behavior is to attempt a tcp connection. If tcp is not activated, or if the tcp connection attempt fails when defaulting to tcp, a udp connection will be attempted.

grpid By default, the GID associated with a newly created file will obey the System V semantics; that is, the GID is set to the effective GID of the calling process. This behavior may be overridden on a per-directory basis by setting the set-GID bit of the parent directory; in this case, the GID of a newly created file is set to the GID of the parent directory (see open(2) and mkdir(2)). Files created on file systems that are mounted with the grpid option will obey BSD semantics independent of whether the set-GID bit of the parent directory is set; that is, the GID is unconditionally inherited from that of the

parent directory.

rsize=n Set the read buffer size to n bytes. The default value is set by kernel.

wsize=n Set the write buffer size to n bytes. The default value is set by kernel.

timeo=n Set the NFS timeout to n tenths of a second. The default value is set by kernel.

retrans=n Set the number of NFS retransmissions to n. The default value is 5.

soft|hard Return an error if the server does not respond, or continue the retry request until the server responds. The default value is hard.

intr|nointr Allow (do not allow) keyboard interrupts to kill a process that is hung while waiting for a response on a hard-mounted file system. The default is intr.

noac Suppress attribute caching.

nocto Suppress fresh attributes when opening a file.

devs|nodevs Allow (do not allow) access to local devices. The default is devs.

acdirmax=n Hold cached attributes for no more than n seconds after directory update. The default value is 60.

acdirmin=n Hold cached attributes for at least n seconds after directory update. The default value is 30.

acregmax=n Hold cached attributes for no more than n seconds after file modification. The default value is 60.

acregmin=n Hold cached attributes for at least n seconds after file modification. The default value is 3.

actimeo=n Set min and max times for regular files and directories to n seconds. actimeo has no default; it sets acregmin, acregmax, acdirmin, and acdirmax to the value specified.

-O Overlay mount. Allow the file system to be mounted over an existing mount point, making the underlying file system inaccessible. If a mount is attempted on a pre-existing mount point without setting this flag, the mount will fail, producing the error device busy.

Options (umount)
umount recognizes the following options:

-a Attempt to unmount all file systems described in /etc/mnttab. All optional fields in /etc/mnttab must be included and supported. If -F nfs option is specified, all NFS file systems in /etc/mnttab are unmounted. File systems are not necessarily unmounted in the order listed in /etc/mnttab.

-F nfs Specify the NFS file system type (see fstyp(1M)).

-h host Unmount only those file systems listed in /etc/mnttab that are remote-mounted from host.

-v Verbose mode. Write a message to standard output indicating which file system is being unmounted.

-V Echo the completed command line, but performs no other action. The command line is generated by incorporating the user-specified options and other information derived from /etc/fstab. This option allows the user to verify the command line.

NFS File Systems

Background vs. Foreground

File systems mounted with the bg option indicate that mount is to retry in the background if the server's mount daemon (mountd(1M)) does not respond. mount retries the request up to the count specified in the retry=n option. Once the file system is mounted, each NFS request made in the kernel waits timeo=n tenths of a second for a response. If no response arrives, the time-out is multiplied by 2 and the request is retransmitted. When the number of retransmissions has reached the number specified in the retrans=n option, a file system mounted with the soft option returns an error on the request; one mounted with the hard option prints a warning message and continues to retry the request.

Hard vs. Soft

File systems that are mounted read-write or that contain executable files should always be mounted with the hard option. Applications using soft mounted file systems may incur unexpected I/O errors.

To improve NFS read performance, files and file attributes are cached. File modification times get updated whenever a write occurs. However, file access times may be temporarily out-of-date until the cache gets refreshed. The attribute cache retains file attributes on the client. Attributes for a file are assigned a time to be flushed. If the file is modified before the flush time, then the flush time is extended by the time since the last modification (under the assumption that files that changed recently are likely to change soon). There is a minimum and maximum flush time extension for regular files and for directories. Setting actimeo=n sets flush time to n seconds for both regular files and directories.

EXAMPLES

To mount an NFS file system:

mount serv:/usr/src /usr/src

To mount an NFS file system readonly with no suid privileges:

mount -r -o nosuid serv:/usr/src /usr/src

To mount an NFS file system over Version 3:

mount -o vers=3 serv:/usr/src /usr/src

To unmount all file systems imported from a given host, enter the following command as root:

umount -h mysystem.home.work.com -a

The hostname must match what is in /etc/mnttab exactly (as shown by the bdf command). For example, if bdf shows:

mysystem:/projects,

the umount command would be

umount -h mysystem -a.

FILES

/etc/mnttab table of mounted file systems.
/etc/fstab list of default parameters for each file system.

SEE ALSO

fsclean(1M), mount(1M), quotaon(1M) setoncenv(1M), mount(2), fstab(4), mnttab(4), fs_wrapper(5), quota(5).

STANDARDS COMPLIANCE

mount: SVID3

umount: SVID3

Appendix 2

`nfsd(1M)`

`NAME`
nfsd, biod - NFS daemons

`SYNOPSIS`
/usr/sbin/nfsd and so forth [nservers]

/usr/sbin/biod [nservers]

`DESCRIPTION`
nfsd starts the NFS server daemons that handle client file system requests (see nfs(7)). nservers is the suggested number of file system request daemons that will start. The minimum number of daemons will be equal to the number of active processors, or to nservers, whichever is greater. The actual number of daemons started will either be a multiple of the number of active processors, or, if tcp is activated, to that multiple plus 1.

The server daemon will support NFS over connection-oriented (tcp) transport, if support for tcp has been activated by the setoncenv command. To obtain the best performance in most cases, set nservers to at least eight.

biod starts nservers asynchronous block I/O daemons. This command is used on an NFS client to buffer cache handle read-ahead and write-behind. nservers is a number greater than zero. For best performance, set nservers to at least sixteen.

`OPTIONS`
nfsd recognizes the following options:

-a Start a NFS daemon over all supported connectionless and connection-oriented transports, including udp and tcp. The NFS_TCP environment variable in /etc/rc.config.d/nfsconf configuration file, must have been set to 1 in order to support connect-oriented (tcp) transport.

-c conns This sets the maximum number of connections allowed to the NFS server over connection-oriented transports, if tcp transport is activated. By default, the number of connections is unlimited.

-p protocol
Start a NFS daemon over the specified protocol.

-t device Start a NFS daemon for the transport specified by the given device.

nservers nservers is the suggested number of file system request daemons that will start. The actual number

of daemons started will be one tcp daemon (to support kernel tcp threads) plus the number of udp daemons. The minimum number of udp daemons will be equal to a multiple of the active processors, or to nservers, whichever is greater. To obtain the best performance in most cases, set nservers to at least eight.

AUTHOR

nfsd was developed by Sun Microsystems, Inc.

SEE ALSO

mountd(1M), exports(4), setoncenv(1M).

Appendix 3

setoncenv(1M)

NAME
setoncenv - NFS environment configuration command

SYNOPSIS
/sbin/setoncenv variable [0|1]

DESCRIPTION
setoncenv initializes the value of NFS configuration environment variables, found either in /etc/rc.config.d/nfsconf or in /etc/rc.config.d/namesvrs (See rc.config(4)). The values may either be 0 (off, or inactive) or 1 (on, or active).

OPTIONS
setoncenv recognizes the following environmental variables:

NFS_TCP NFS connections across TCP/IP transport are supported.

AUTOFS The AutoFS product is supported, if AUTOMOUNT=1.

AUTOMOUNT One of the two Automount products is supported.

NFS_CLIENT The NFS client daemons will be launched.

NFS_SERVER The NFS server daemons will be launched

NISPLUS_CLIENT This platform can act as an NIS+ client.

NISPLUS_SERVER This platform can act an NIS+ server.

NIS_CLIENT This platform can act as an NIS client.

NIS_MASTER_SERVER This platform will be an NIS master server.

NIS_SLAVE_SERVER This platform will be an NIS slave server.

PCNFS_SERVER PCNFS client requests will be answered.

START_MOUNTD The Mount daemon will be launched at system startup.

AUTHOR

setoncenv was developed by the Hewlett-Packard Company.

SEE ALSO

nfsd(1M) mountd(1M), mount_nfs(1M) rc.config(4).

Appendix 4

Included is a representative sample of the output from the ***nfsstat*** command. Please note the division of data into the differing transport types - Connection oriented, as with TCP, and Connectionless, as with UDP.

```
# nfsstat

Server rpc:
Connection oriented:
calls     badcalls   nullrecv   badlen     xdrcall    dupchecks  dupreqs
7         0          0          0          0          0          0
Connectionless oriented:
calls     badcalls   nullrecv   badlen     xdrcall    dupchecks  dupreqs
0         0          0          0          0          0          0


Server nfs:
calls     badcalls
6         0
Version 2: (0 calls)
null      getattr    setattr    root       lookup     readlink   read
0 0%      0 0%       0 0%       0 0%       0 0%       0 0%       0 0%
wrcache   write      create     remove     rename     link       symlink
0 0%      0 0%       0 0%       0 0%       0 0%       0 0%       0 0%
mkdir     rmdir      readdir    statfs
0 0%      0 0%       0 0%       0 0%
Version 3: (6 calls)
null      getattr    setattr    lookup     access     readlink   read
1 16%     1 16%      0 0%       0 0%       1 16%      0 0%       0 0%
write     create     mkdir      symlink    mknod      remove     rmdir
0 0%      0 0%       0 0%       0 0%       0 0%       0 0%       0 0%
rename    link       readdir    readdir+   fsstat     fsinfo     pathconf
0 0%      0 0%       0 0%       2 33%      0 0%       1 16%      0 0%
commit
0 0%


Client rpc:
Connection oriented:
calls     badcalls   badxids    timeouts   newcreds   badverfs   timers
32        0          0          0          0          0          0
cantconn  nomem      interrupts
0         0          0
Connectionless oriented:
calls     badcalls   retrans    badxids    timeouts   waits      newcreds
0         0          0          0          0          0          0
badverfs  timers     toobig     nomem      cantsend   bufulocks
0         0          0          0          0          0
```

Client nfs:
calls badcalls clgets cltoomany
32 0 32 0
Version 2: (0 calls)
null getattr setattr root lookup readlink read
0 0% 0 0% 0 0% 0 0% 0 0% 0 0% 0 0%
wrcache write create remove rename link symlink
0 0% 0 0% 0 0% 0 0% 0 0% 0 0% 0 0%
mkdir rmdir readdir statfs
0 0% 0 0% 0 0% 0 0%
Version 3: (32 calls)
null getattr setattr lookup access readlink read
0 0% 8 25% 2 6% 2 6% 10 31% 0 0% 1 3%
write create mkdir symlink mknod remove rmdir
1 3% 1 3% 0 0% 0 0% 0 0% 0 0% 0 0%
rename link readdir readdir+ fsstat fsinfo pathconf
0 0% 0 0% 0 0% 4 12% 1 3% 1 3% 0 0%
commit
1 3%