Доброго времени, читатели и гости моего блога. Очень большой перерыв между постами был, но я снова в бою). В сегодняшней статье рассмотрю работу протокола NFS, а так же настройку сервера NFS и клиента NFS на Linux.

Введение в NFS

NFS (Network File System - сетевая файловая система) по моему мнению - идеальное решение в локальной сети, где необходим быстрый (более быстрый по сравнению с SAMBA и менее ресурсоемкий по сравнению с удаленными файловыми системами с шифрованием - sshfs, SFTP, etc...) обмен данными и во главе угла не стоит безопасность передаваемой информации. Протокол NFS позволяет монтировать удалённые файловые системы через сеть в локальное дерево каталогов, как если бы это была примонтирована дисковая файловая система. Тем самым локальные приложения могут работать с удаленной файловой системой, как с локальной. Но нужно быть осторожным (!) с настройкой NFS, ибо при определенной конфигурации можно подвесить операционную систему клиента в ожидании бесконечного ввода/вывода. Протокол NFS основан на работе протокола RPC, который пока не поддается моему пониманию)) поэтому материал в статье будет немного расплывчат... Прежде, чем Вы сможете использовать NFS, будь это сервер или клиент, Вы должны удостовериться, что Ваше ядро имеет поддержку файловой системы NFS. Проверить поддерживает ли ядро файловую систему NFS можно, просмотрев наличие соответствующих строк в файле /proc/filesystems:

```
ARCHIV ~ # grep nfs /proc/filesystems
nodev nfs
nodev nfs4
nodev nfsd
```

Если указанных строк в файле /proc/filesystems не окажется, то необходимо установить описанные ниже пакеты. Это скорее всего позволит установить зависимые модули ядра для поддержки нужных файловых систем. Если после установки пакетов, поддержка NFS не будет отображена в указанном файле, то необходимо будетперекомпилировать ядро, с включением данной функции.

История Network File System

Протокол NFS разработан компанией Sun Microsystems и имеет в своей истории 4 версии. NFSv1 была разработана в 1989 и являлась экспериментальной, работала на протоколе UDP. Версия 1 описана в RFC 1094. NFSv2 была выпущена в том же 1989 г., описывалась тем же RFC1094 и так же базировалась на протоколе UDP, при этом позволяла читать не более 2Гб из файла. NFSv3 доработана в 1995 г. и описана в RFC 1813. Основными нововведениями третьей версии стало поддержка файлов большого размера, добавлена поддержка протокола TCP и TCP-пакетов большого размера, что существенно ускорило работоспособность технологии. NFSv4 доработана в 2000 г. и описана в RFC 3010, в 2003 г. пересмотрена и описана в RFC 3530. Четвертая версия включила в себя улучшение производительности, поддержку различных средств аутентификации (в частности, Kerberos и LIPKEY с использованием протокола RPCSEC GSS) и списков контроля доступа (как POSIX, так и Windows-типов). NFS версии v4.1 была одобрена IESG в 2010 г., и получила номер RFC 5661. Важным нововведением версии 4.1, является спецификация pNFS — Parallel NFS, механизма параллельного доступа NFS-клиента к данным множества распределенных NFS-серверов. Наличие такого механизма в стандарте сетевой файловой системы поможет строить распределённые «облачные» («cloud») хранилища и информационные системы.

NFS сервер

Так как у нас NFS - это сетевая файловая система, то необходимо настроить сеть в Linux. (Так же можно почитать статью основные понятия сетей). Далее необходимо установить соответствующий пакет. В Debian это пакет nfs-kernel-server и nfs-common, в RedHat это пакет nfs-utils. А так же, необходимо разрешить запуск демона наобходимых уровнях выполнения ОС (команда в RedHat - /sbin/chkconfig nfs on, в Debian - /usr/sbin/update-rc.d nfs-kernel-server defaults).
Установленные пакеты в Debian запускается в следующем порядке:

```
ARCHIV ~ # ls -la /etc/rc2.d/ | grep nfs
```

lrwxrwxrwx 1 root root 20 Окт 18 15:02 S15nfs-common -> ../init.d/nfs-common

lrwxrwxrwx 1 root root 27 Окт 22 01:23 S16nfs-kernel-server -> ../init.d/nfs-kernel-server

То есть, сначала запускается nfs-common, затем сам сервер nfs-kernel-server. В RedHat ситуация аналогичная, за тем лишь исключением, что первый скрипт называется nfslock, а сервер называется просто nfs. Про nfs-commonнам rel="nofollow" href="http://packages.debian.org/squeeze/nfs-common" target="_blank" style="margin-top: 0px; margin-right: 0px; margin-bottom: 0px; margin-left: 0px; padding-top: 0px; padding-right: 0px; padding-bottom: 0px; padding-left: 0px; color: rgb(0, 94, 172); text-decoration: none; border-bottom-width: 1px; border-bottom-style: dotted; border-bottom-color: rgb(223, 223, 223); "сайт debian дословно говорит следующее: общие файлы для клиента и сервера NFS, этот пакет нужно устанавливать на машину, которая будет работать в качестве клиента или сервера NFS. В пакет включены программы: lockd, statd, showmount, nfsstat, gssd и idmapd. Просмотрев содержимое скрипта запуска /etc/init.d/nfs-common можно отследить следующую последовательность работы: скрипт проверяет наличие исполняемого бинарного файла /sbin/rpc.statd, проверяет наличие в файлах /etc/default/nfs-common, /etc/fstab и /etc/exportsпараметров, требующих запуск демонов idmapd и gssd, запускает демона /sbin/rpc.statd, далее перед запуском/usr/sbin/rpc.idmapd и /usr/sbin/rpc.gssd проверяет наличие этих исполняемых бинарных файлов, далее для демона /usr/sbin/rpc.idmapd проверяет наличие модулей ядра sunrpc, nfs и nfsd, а так же поддержку файловой системыrpc_pipefs в ядре (то есть наличие ее в файле /proc/filesystems), если все удачно, то запускает/usr/sbin/rpc.idmapd. Дополнительно, для демона /usr/sbin/rpc.gssd проверяет модуль ядра rpcsec_gss_krb5 и запускает демон.

Если просмотреть содержимое скрипта запуска NFS-сервера на Debian (/etc/init.d/nfs-kernel-server), то можно проследить следующую последовательность: при старте, скрипт проверяет существование файла /etc/exports, наличие модуля ядра nfsd, наличие поддержки файловой системы NFS в ядре Linux (то есть в файле/proc/filesystems), если все на месте, то запускается демон /usr/sbin/rpc.nfsd, далее проверяет задан ли параметр NEED_SVCGSSD (задается в файле настроек сервера /etc/default/nfs-kernel-server) и, если задан - запускает демона /usr/sbin/rpc.svcgssd, последним запускает демона /usr/sbin/rpc.mountd. Из данного скрипта видно, что работа сервера NFS состоит из демонов rpc.nfsd, rpc.mountd и если используется Kerberos-аутентификация, то и демон rcp.svcgssd. В краснойшляпе еще запускается демон rpc.rquotad и nfslogd (В Debian я почему-то не нашел информации об этом демоне и о причинах его отсутствия, видимо удален...).

Из этого становиться понятно, что сервер Network File System состоит из следующих процессов (читай - демонов), расположенных в каталогах /sbin и /usr/sbin :

- rpc.statd - Демон наблюдения за сетевым состоянием (он же Network Status Monitor, он же NSM). Он позволяет корректно отменять блокировку после сбоя/перезагрузки. Для уведомления о сбое использует программу /usr/sbin/sm-notify. Демон statd работает как на серверах, так и на клиентах. Ранее данный сервер был необходим для работы rpc.lockd, но за блокировки сейчас отвечает ядро (прим: если я не ошибаюсь ). (RPC программа 100021 и 100024 - в новых версиях)
- rpc.lockd - Демон блокировки lockd (он же NFS lock manager (NLM)) обрабатывает запросы на блокировку файлов. Демон блокировки работает как на серверах, так и на клиентах. Клиенты запрашивают блокировку файлов, а серверы ее разрешают. (устарел и в новых дистрибутивах не используется как демон. Его функции в современных дистрибутивах (с ядром старше 2.2.18) выполняются ядром, точнее модулем ядра (lockd).) (RPC программа 100024)
- rpc.nfsd - Основной демон сервера NFS - nfsd (в новых версиях иногда называется nfsd4). Этот демон обслуживает запросы клиентов NFS. Параметр RPCNFSDCOUNT в файле /etc/default/nfs-kernel-server в Debian иNFSDCOUNT в файле /etc/sysconfig/nfs в RedHat определяет число запускаемых демонов (по-умолчанию - 8).(RPC программа 100003)
- rpc.mountd - Демон монтирования NFS mountd обрабатывает запросы клиентов на монтирование каталогов. Демон mountd работает на серверах NFS. (RPC программа 100005)
- rpc.idmapd - Демон idmapd для NFSv4 на сервере преобразует локальные uid/gid пользователей в формат вида имя@домен, а сервис на клиенте преобразует имена пользователей/групп вида

имя@домен в локальные идентификаторы пользователя и группы (согласно конфигурационному файлу /etc/idmapd.conf, подробней brel="nofollow" title="man idmap.conf" href="http://www.opennet.ru/man.shtml?topic=idmapd.conf&russian=0&category=&submit=%F0%CF%CB%C1%DA%C1%D4%D8+man" target="_blank" style="margin-top: 0px; margin-right: 0px; margin-bottom: 0px; margin-left: 0px; padding-top: 0px; padding-right: 0px; padding-bottom: 0px; padding-left: 0px; color: rgb(0, 94, 172); text-decoration: none; "man idmapd.conf):.

- дополнительно, в старых версиях NFS использовались демоны: nfslogd - демон журналов NFS фиксирует активность для экспортированных файловых систем, работает на серверах NFS и rquotad - сервер удаленных квот предоставляет информацию о квотах пользователей в удаленных файловых системах, может работать как на серверах, так и на клиентах.(RPC программа 100011)

В NFSv4 при использовании Kerberos дополнительно запускаются демоны:

- rpc.gssd - Демон NFSv4 обеспечивает методы аутентификации через GSS-API (Kerberos-аутентификация). Работает на клиенте и сервере.
- rpc.svcgssd - Демон сервера NFSv4, который обеспечивает проверку подлинности клиента на стороне сервера.

portmap и протокол RPC (Sun RPC)

Кроме указанных выше пакетов, для корректной работы NFSv2 и v3 требуется дополнительный пакет portmap (в более новых дистрибутивах заменен на переименован в rpcbind). Данный пакет обычно устанавливается автоматически с NFS как зависимый и реализует работу сервера RPC, то есть отвечает за динамическое назначение портов для некоторых служб, зарегистрированных в RPC сервере. Дословно, согласно документации — это сервер, который преобразует номера программ RPC (Remote Procedure Call) в номера портов TCP/UDP. portmap оперирует несколькими сущностями: RPC-вызовами или запросами, TCP/UDP портами, версией протокола (tcp или udp),номерами программ и версиями программ. Демон portmap запускается скриптом /etc/init.d/portmap до старта NFS-сервисов.

Коротко говоря, работа сервера RPC (Remote Procedure Call) заключается в обработке RPC-вызовов (т.н. RPC-процедур) от локальных и удаленных процессов. Используя RPC-вызовы, сервисы регистрируют или удаляют себя в/из преобразователя портов (он же отображатель портов, он же portmap, он же portmapper, он же, в новых версиях, rpcbind), а клиенты с помощью RPC-вызовов направляя запросы к portmapper получают необходимую информацию. Юзер-френдли названия сервисов программ и соответствующие им номера определены в файле /etc/rpc. Как только какой-либо сервис отправил соответствующий запрос и зарегистрировал себя на сервере RPC в отображателе портов, RPC-сервер присваивает сопоставляет сервису TCP и UDP порты на которых запустился сервис и хранит в себе ядре соответствующую информацию о работающем сервисе (о имени), уникальном номере сервиса (в соответствии с /etc/rpc) , о протоколе и порте на котором работает сервис и о версии сервиса и предоставляет указанную информацию клиентам по запросу. Сам преобразователь портов имеет номер программы (100000), номер версии - 2, TCP порт 111 и UDP порт 111. Выше, при указании состава демонов сервера NFS я указал основные RPC номера программ. Я, наверно, немного запутал Вас данным абзацем, поэтому произнесу основную фразу, которая должна внести ясность: основная функция отображателя портов заключается в том, чтобы по запросу клиента, который предоставил номер RPC-программы (или RPC-номер программы) и версию, вернуть ему (клиенту) порт, на котором работает запрошенная программа. Соответственно, если клиенту нужно обратиться к RPC с конкретным номером программы, он сначала должен войти в контакт с процессом portmap на серверной машине и определить номер порта связи с необходимым ему сервисом RPC.

Работу RPC-сервера можно представить следующими шагами:

1. Преобразователь портов должен стартовать первым, обычно при загрузке системы. При этом создается конечная точка TCP и осуществляется открытие TCP порта 111. Также создается конечная точка UDP, которая находится в ожидании, когда на UDP порт 111 прибудет UDP датаграмма.
2. При старте программа, работающая через сервер RPC создает конечную точку TCP и конечную точку UDP для каждой поддерживаемой версии программы. (Сервер RPC может поддерживать несколько версий. Клиент указывает требуемую версию при посылке RPC-вызова.) Динамически назначаемый номер порта закрепляется за каждой версией сервиса. Сервер регистрирует каждую программу, версию, протокол и номер порта, осуществляя соответствуюоий RPC-вызов.

3. Когда программе клиента RPC необходимо получить необходимую информацию, она вызывает вызовпроцедуру преобразователя портов, чтобы получить динамически назначаемый номер порта для заданной программы, версии и протокола.
4. В ответ на этот запрос север возвращает номер порта.
5. Клиент отправляет сообщение RPC-запрос на номер порта, полученный в пункте 4. Если используется UDP, клиент просто посылает UDP датаграмму, содержащую сообщение RPC-вызова , на номер UDP порта, на котором работает запрошенный сервис. В ответ сервис отправляет UDP датаграмму, содержащую сообщение RPC отклика. Если используется TCP, клиент осуществляет активное открытие на номер TCP порта требуемого сервиса и затем посылает сообщение вызова RPC по установленному соединению. Сервер отвечает сообщением отклика RPC по соединению. Для получения информации от RPC-сервера используется утилита rpcinfo. При указании параметров -p hostпрограмма выводит список всех зарегистрированных RPC программ на хосте host. Без указания хоста программа выведет сервисы на localhost. Пример:

```
ARCHIV ~ # rpcinfo -p
   прог-ма верс прото   порт
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100024    1   udp  59451  status
    100024    1   tcp  60872  status
    100021    1   udp  44310  nlockmgr
    100021    3   udp  44310  nlockmgr
    100021    4   udp  44310  nlockmgr
    100021    1   tcp  44851  nlockmgr
    100021    3   tcp  44851  nlockmgr
    100021    4   tcp  44851  nlockmgr
    100003    2   tcp   2049  nfs
    100003    3   tcp   2049  nfs
    100003    4   tcp   2049  nfs
    100003    2   udp   2049  nfs
    100003    3   udp   2049  nfs
    100003    4   udp   2049  nfs
    100005    1   udp  51306  mountd
    100005    1   tcp  41405  mountd
    100005    2   udp  51306  mountd
    100005    2   tcp  41405  mountd
    100005    3   udp  51306  mountd
    100005    3   tcp  41405  mountd
```

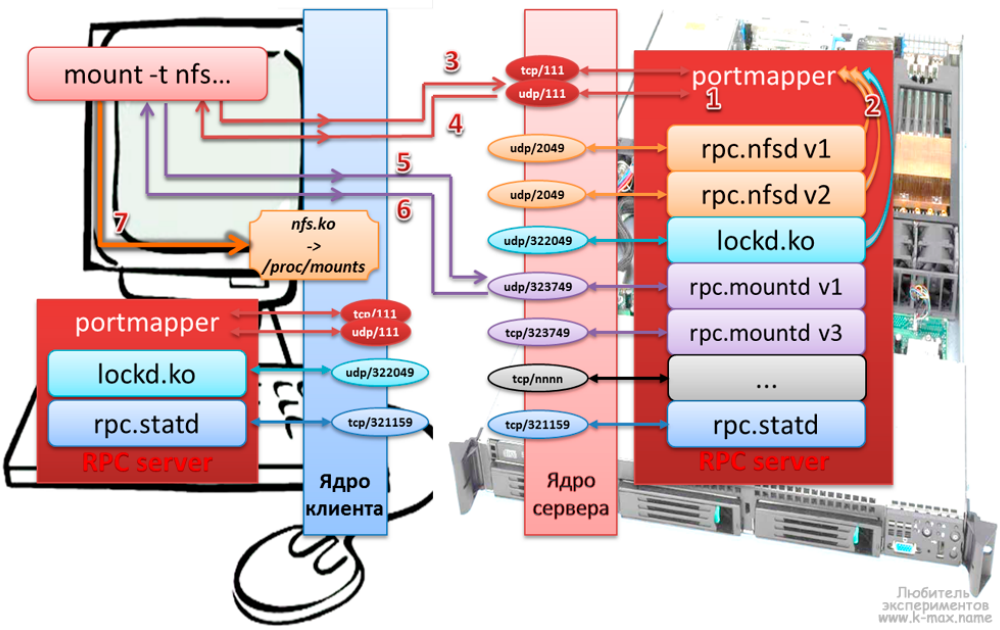
Как видно, rpcinfo отображает (в столбиках слева направо) номер зарегистрированной программы, версию, протокол, порт и название. С помощью rpcinfo можно удалить регистрацию программы или получить информацию об отдельном сервисе RPC (больше опций в rel="nofollow" title="man rpcinfo" href="http://www.opennet.ru/man.shtml?topic=rpcinfo&category=8&russian=0" target="_blank" style="margin-top: 0px; margin-right: 0px; margin-bottom: 0px; margin-left: 0px; padding-top: 0px;

padding-right: 0px; padding-bottom: 0px; padding-left: 0px; color: rgb(0, 94, 172); text-decoration: none; border-bottom-width: 1px; border-bottom-style: dotted; border-bottom-color: rgb(223, 223, 223); "man rpcinfo). Как видно, зарегистрированы демоны portmapper версии 2 на udp и tcp портах, rpc.statd версии 1 на udp и tcp портах, NFS lock manager версий 1,3,4, демон nfs сервера версии 2,3,4, а так же демон монтирования версий 1,2,3.
NFS сервер (точнее демон rpc.nfsd) получает запросы от клиента в виде UDP датаграмм на порт 2049. Несмотря на то, что NFS работает с преобразователем портов, что позволяет серверу использовать динамически назначаемые порты, UDP порт 2049 жестко закреплен за NFS в большинстве реализаций.

Работа протокола Network File System

Монтирование удаленной NFS

Процесс монтирования удаленной файловой системы NFS можно представить следующей схемой:

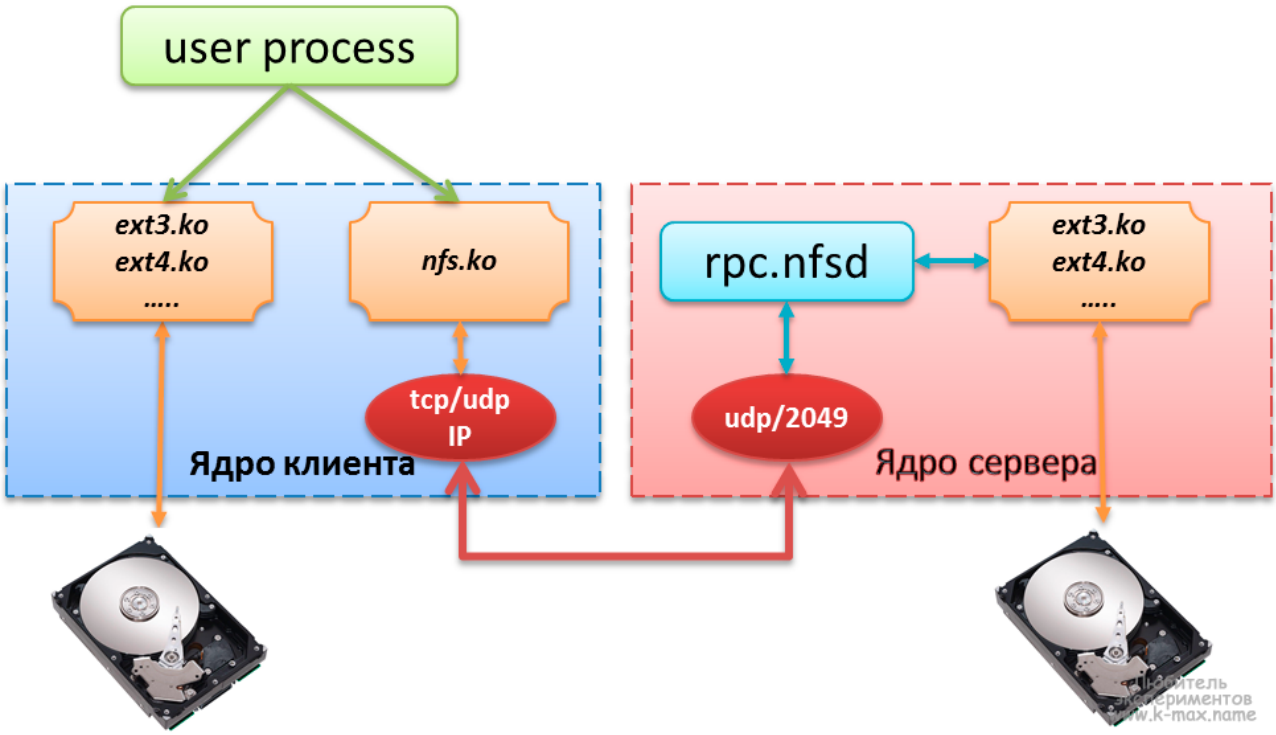


Описание протокола NFS при монтировании удаленного каталога:

1. На сервере и клиенте запускается RPC сервер (обычно при загрузке), обслуживанием которого занимается процесс portmapper и регистрируется на порту tcp/111 и udp/111.
2. Запускаются сервисы (rpc.nfsd,rpc.statd и др.), которые регистрируются на RPC сервере и регистрируются на произвольных сетевых портах (если в настройках сервиса не задан статичный порт).
3. команда mount на компьютере клиента отправляет ядру запрос на монтирование сетевого каталога с указанием типа файловой системы, хоста и собственно - каталога, ядро отправляет формирует RPC-запрос процессу portmap на NFS сервере на порт udp/111 (если на клиенте не задана опция работать через tcp)
4. Ядро сервера NFS опрашивает RPC о наличии демона rpc.mountd и возвращает ядру клиента сетевой порт, на котором работает демон.
5. mount отправляет RPC запрос на порт, на котором работает rpc.mountd. Теперь NFS сервер может проверить достоверность клиента основываясь на его IP адресе и номере порта, чтобы убедиться, можно ли этому клиенту смонтировать указанную файловую систему.
6. Демон монтирования возвращает описание запрошенной файловой системы.
7. Команда mount клиента выдает системный вызов mount, чтобы связать описатель файла, полученный в шаге 5, с локальной точкой монтирования на хосте клиента. Описатель файла хранится в коде NFS клиента, и с этого момента любое обращение пользовательских процессов к файлам на файловой системе сервера будет использовать описатель файла как стартовую точку.

Обмен данными между клиентом и сервером NFS

Типичный доступ к удаленной файловой системе можно описать следующей схемой:



Описание процесса обращения к файлу, расположенному на сервере NFS:

1. Клиенту (пользовательскому процессу) безразлично, получает ли он доступ к локальному файлу или к NFS файлу. Ядро занимается взаимодействием с железом через модули ядра или встроенные системные вызовы.
2. Модуль ядра kernel/fs/nfs/nfs.ko, который выполняет функции NFS клиента отправляет RPC запросы NFS серверу через модуль TCP/IP. NFS обычно использует UDP, однако более новые реализации могут использовать TCP.
3. NFS сервер получает запросы от клиента в виде UDP датаграмм на порт 2049. Несмотря на то, что NFS может работать с преобразователем портов, что позволяет серверу использовать динамически назначаемые порты, UDP порт 2049 жестко закреплен за NFS в большинстве реализаций.
4. Когда NFS сервер получает запрос от клиента, он передаётся локальной подпрограмме доступа к файлу, которая обеспечивает доступ к локальному диску на сервере.
5. Результат обращения диску возвращается клиенту.

Настройка сервера NFS

Настройка сервера в целом заключается в задании локальных каталогов, разрешенных для монтирования удаленными системами в файле /etc/exports. Это действие называется экспорт иерархии каталогов. Основными источниками информации об экспортированных каталогах служат следующие файлы:

- /etc/exports - основной конфигурационный файл, хранящий в себе конфигурацию экспортированных каталогов. Используется при запуске NFS и утилитой exportfs.
- /var/lib/nfs/xtab - содержит список каталогов, монтированных удаленными клиентами. Используется демоном rpc.mountd, когда клиент пытается смонтировать иерархию (создается запись о монтировании).
- /var/lib/nfs/etab - список каталогов, которые могут быть смонтированы удаленными системами с указанием всех параметров экспортированных каталогов.
- /var/lib/nfs/rmtab - список каталогов, которые не разэкспортированы в данный момент.
- /proc/fs/nfsd - специальная файловая система (ядро 2.6) для управления NFS сервером.
- exports - список активных экспортированных иерархий и клиентов, которым их экспортировали, а также параметры. Ядро получает данную информацию из /var/lib/nfs/xtab.
- threads - содержит число потоков (также можно изменять)
- с помощью filehandle можно получить указатель на файл

•и др...
•/proc/net/rpc - содержит "сырую" (raw) статистику, которую можно получить с помощью nfsstat, а также различные кеши.
•/var/run/portmap_mapping - информация о зарегистрированных в RPC сервисах
Прим: вообще, в интернете куча трактовок и формулировок назначения файлов xtab, etab, rmtab, кому верить - не знаю  Даже на http://nfs.sourceforge.net/ трактовка не однозначна.

Настройка файла /etc/exports

В простейшем случае, файл /etc/exports является единственным файлом, требующим редактирования для настройки NFS-сервера. Данный файл управляет следующими аспектами:

•Какие клиенты могут обращаться к файлам на сервере
•К каким иерархиям каталогов на сервере может обращаться каждый клиент
•Как пользовательские имена клиентов будут отображаться на локальные имена пользователей
Каждая строка файла exports имеет следующий формат:
точка_экспорта клиент1(опции) [клиент2(опции) ...]
, где точка_экспорта абсолютный путь экспортируемой иерархии каталогов, клиент1 - n имя одного или более клиентов или IP-адресов, разделенные пробелами, которым разрешено монтировать точку_экспорта. Опцииописывают правила монтирования для клиента, указанного перед опциями.
Вот типичный пример конфигурации файла exports:

```
ARCHIV ~ # cat /etc/exports

/archiv1        files(rw,sync) 10.0.0.1(ro,sync) 10.0.230.1/24(ro,sync)
```

В данном примере компьютерам files и 10.0.0.1 разрешен доступ к точке экспорта /archiv1, при этом, хосту files на чтение/запись, а для хоста 10.0.0.1 и подсети 10.0.230.1/24 доступ только на чтение.

Описания хостов в /etc/exports допускается в следующем формате:

•Имена отдельных узлов описываются, как files или files.DOMAIN.local.
•Описание маски доменов производится в следующем формате: *DOMAIN.local включает все узлы домена DOMAIN.local.
•Подсети задаются в виде пар адрес IP/маска. Например: 10.0.0.0/255.255.255.0 включает все узлы, адреса которых начинаются с 10.0.0.
•Задание имени сетевой группы @myclients имеющей доступ к ресурсу (при использовании сервера NIS)

Общие опции экспорта иерархий каталогов

В файле exports используются следующие общие опции (сначала указаны опции применяемые по-умолчанию в большинстве систем, в скобках - не по-умолчанию):
•auth_nlm (no_auth_nlm) или secure_locks (insecure_locks) - указывает, что сервер должен требовать аутентификацию запросов на блокировку (с помощью протокола NFS Lock Manager (диспетчер блокировок NFS)).
•nohide (hide) - если сервер экспортирует две иерархии каталогов, при этом одна вложенна (примонтированна) в другую. Клиенту необходимо явно смонтировать вторую (дочернюю) иерархию, иначе точка монтирования дочерней иерархии будет выглядеть как пустой каталог. Опция nohide приводит к появлению второй иерархии каталогов без явного монтирования. (прим: я данную опцию так и не смог заставить работать...)
•ro (rw) - Разрешает только запросы на чтение (запись). (в конечном счете - возможно прочитать/записать или нет определяется на основании прав файловой системы, при этом сервер не способен отличить запрос на чтение файла от запроса на исполнение, поэтому разрешает чтение, если у пользователя есть права на чтение или исполнение.)
•secure (insecure) - требует, чтобы запросы NFS поступали с защищенных портов (< 1024), чтобы программа без прав root не могла монтировать иерархию каталогов.
•subtree_check (no_subtree_check) - Если экспортируется подкаталог фаловой системы, но не вся файловая система, сервер проверяет, находится ли запрошенный файл в экспортированном

подкаталоге. Отключение проверки уменьшает безопасность, но увеличивает скорость передачи данных.

- sync (async) - указывает, что сервер должен отвечать на запросы только после записи на диск изменений, выполненных этими запросами. Опция async указывает серверу не ждать записи информации на диск, что повышает производительность, но понижает надежность, т.к. в случае обрыва соединения или отказа оборудования возможна потеря информации.
- wdelay (no_wdelay) - указывает серверу задерживать выполнение запросов на запись, если ожидается последующий запрос на запись, записывая данные более большими блоками. Это повышает производительность при отправке больших очередей команд на запись. no_wdelay указывает не откладывать выполнение команды на запись, что может быть полезно, если сервер получает большое количество команд не связанных друг с другом.

Экспорт символических ссылок и файлов устройств. При экспорте иерархии каталогов, содержащих символические ссылки, необходимо, чтобы объект ссылки был доступен клиентской (удаленной) системе, то есть должно выполняться одно из следующих правил:

- в клиентской файловой системе должен существовать объект ссылки
- необходимо экспортировать и смонтировать объект ссылки

Файл устройства относится к интерфейсу ядра Linux. При экспорте файла устройства экспортируется этот интерфейс. Если клиентская система не имеет устройства такого же типа, то экспортированное устройство не будет работать. В клиентской системе, при монтировании NFS объектов можно использовать опцию nodev, чтобы файлы устройств в монтируемых каталогах не использовались.

Опции по умолчанию в разных системах могут различаться, их можно посмотреть в файле /var/lib/nfs/etab. После описания экспортированного каталога в /etc/exports и перезапуска сервера NFS все недостающие опции (читай: опции по-умолчанию) будут отражены в файле /var/lib/nfs/etab.

Опции отображения (соответствия) идентификаторов пользователей

Для большего понимания нижесказанного я бы посоветовал ознакомиться со статьей Управление пользователями Linux. Каждый пользователь Linux имеет свои UID и главный GID, которые описаны в файлах /etc/passwd и/etc/group. Сервер NFS считает, что операционная система удаленного узла выполнила проверку подлинности пользователей и назначила им корректные идентификаторы UID и GID. Экспортирование файлов дает пользователям системы клиента такой же доступ к этим файлам, как если бы они регистрировались напрямую на сервере. Соответственно, когда клиент NFS посылает запрос серверу, сервер использует UID и GID для идентификации пользователя в локальной системе, что может приводить к некоторым проблемам:

- пользователь может не иметь одни и те же идентификаторы в обеих системах и, соответственно, может получить доступ к фалам другого пользователя.
- т.к. у пользователя root идентификатор всегда 0, то данный пользователь отображается на локального пользователя в зависимости от заданных опций.

Следующие опции задают правила отображения удаленных пользователей в локальных:

- root_squash (no_root_squash) - При заданной опции root_squash, запросы от пользователя root отображаются на анонимного uid/gid, либо на пользователя, заданного в параметре anonuid/anongid.
- no_all_squash (all_squash) - Не изменяет UID/GID подключающегося пользователя.

Опция all_squash задает отображение ВСЕХ пользователей (не только root), как анонимных или заданных в параметре anonuid/anongid.

- anonuid=UID и anongid=GID - Явно задает UID/GID для анонимного пользователя.
- map_static=/etc/file_maps_users - Задает файл, в котором можно задать сопоставление удаленных UID/GID - локальным UID/GID.

Пример использования файла маппинга пользователей:

```
ARCHIV ~ # cat /etc/file_maps_users

# Маппинг пользователей

# remote local comment
```

uid 0-50 1002 # сопоставление пользователей с удаленным UID 0-50 к локальному UID 1002

gid 0-50 1002 # сопоставление пользователей с/span удаленным GID 0-50 к локальному GID 1002

Управление сервером NFS

Управление сервером NFS осуществляется с помощью следующих утилит:

- nfsstat
- showmsecure (insecure)ount
- exportfs

nfsstat: статистика NFS и RPC

Утилита nfsstat позволяет посмотреть статистику RPC и NFS серверов. Опции команды можно посмотреть в rel="nofollow" title="man nfsstat" href="http://www.opennet.ru/man.shtml?topic=nfsstat&category=8&russian=2" target="_blank" style="margin-top: 0px; margin-right: 0px; margin-bottom: 0px; margin-left: 0px; padding-top: 0px; padding-right: 0px; padding-bottom: 0px; padding-left: 0px; color: rgb(0, 94, 172); text-decoration: none; border-bottom-width: 1px; border-bottom-style: dotted; border-bottom-color: rgb(223, 223, 223); "man nfsstat.

showmount: вывод информации о состоянии NFS

Утилита showmount запрашивает демон rpc.mountd на удалённом хосте о смонтированных файловых системах. По умолчанию выдаётся отсортированный список клиентов. Ключи:

- --all - выдаётся список клиентов и точек монтирования с указанием куда клиент примонтировал каталог. Эта информация может быть не надежной.
- --directories - выдаётся список точек монтирования
- --exports - выдаётся список экспортируемых файловых систем с точки зрения nfsd

При запуске showmount без аргументов, на консоль будет выведена информация о системах, которым разрешено монтировать локальные каталоги. Например, хост ARCHIV нам предоставляет список экспортированных каталогов с IP адресами хостов, которым разрешено монтировать указанные каталоги:

FILES ~ # showmount --exports archiv

Export list for archiv:

/archiv-big 10.0.0.2

/archiv-small 10.0.0.2

Если указать в аргументе имя хоста/IP, то будет выведена информация о данном хосте:

ARCHIV ~ # showmount files

clnt_create: RPC: Program not registered

 # данное сообщение говорит нам, что на хосте FILES демон NFSd не запущен

exportfs: управление экспортированными каталогами

Данная команда обслуживает экспортированные каталоги, заданные в файле /etc/exports, точнее будет написать не обслуживает, а синхронизирует с файлом /var/lib/nfs/xtab и удаляет из xtab несуществующие. exportfs выполняется при запуске демона nfsd с аргументом -r. Утилита exportfs в режиме ядра 2.6 общается с демоном rpc.mountd через файлы каталога /var/lib/nfs/ и не общается с ядром напрямую. Без параметров выдаёт список текущих экспортируемых файловых систем.

Параметры exportfs:

- [клиент:имя-каталога] - добавить или удалить указанную файловую систему для указанного клиента)
- -v - выводить больше информации
- -r - переэкспортировать все каталоги (синхронизировать /etc/exports и /var/lib/nfs/xtab)
- -u - удалить из списка экспортируемых

•-a - добавить или удалить все файловые системы
•-o - опции через запятую (аналогичен опциям применяемым в /etc/exports; т.о. можно изменять опции уже смонтированных файловых систем)
•-i - не использовать /etc/exports при добавлении, только параметры текущей командной строки
•-f - сбросить список экспортируемых систем в ядре 2.6;

Клиент NFS

Прежде чем обратиться к файлу на удалённой файловой системе клиент (ОС клиента) должен смонтировать её и получить от сервера указатель на неё. Монтирование NFS может производиться с помощью команды mountили с помощью одного из расплодившихся автоматических монтировщиков (amd, autofs, automount, supermount, superpupermount). Процесс монтирования хорошо продемонстрирована выше на иллюстрации.
На клиентах NFS никаких демонов запускать не нужно, функции клиента выполняет модуль ядрakernel/fs/nfs/nfs.ko, который используется при монтировании удаленной файловой системы. Экспортированные каталоги с сервера могут монтироваться на клиенте следующими способами:
•вручную, с помощью команды mount
•автоматически при загрузке, при монтировании файловых систем, описанных в /etc/fstab
•автоматически с помощью демона autofs
Третий способ с autofs в данной статье я рассматривать не буду, ввиду его объемной информации. Возможно в следующих статьях будет отдельное описание.

Монтирование файловой системы Network Files System командой mount

Пример использования команды mount представлен в посте Команды управления блочными устройствами. Тут я рассмотрю пример команды mount для монтирования файловой системы NFS:

```
FILES ~ # mount -t nfs archiv:/archiv-small /archivs/archiv-small

FILES ~ # mount -t nfs -o ro archiv:/archiv-big /archivs/archiv-big

FILES ~ # mount

.......

archiv:/archiv-small on /archivs/archiv-small type nfs (rw,addr=10.0.0.6)

archiv:/archiv-big on /archivs/archiv-big type nfs (ro,addr=10.0.0.6)
```

Первая команда монтирует экспортированный каталог /archiv-small на сервере archiv в локальную точку монтирования /archivs/archiv-small с опциями по умолчанию (то есть для чтения и записи). Хотя команда mount в последних дистрибутивах умеет понимать какой тип файловой системы используется и без указания типа, все же указывать параметр -t nfs желательно. Вторая команда монтирует экспортированный каталог /archiv-big на серверearchiv в локальный каталог /archivs/archiv-big с опцией только для чтения (ro). Команда mount без параметров наглядно отображает нам результат монтирования. Кроме опции только чтения (ro), возможно задать другиеосновные опции при монтировании NFS:
•nosuid - Данная опция запрещает исполнять setuid программы из смонтированного каталога.
•nodev (no device - не устройство) - Данная опция запрещает использовать в качестве устройств символьные и блочные специальные файлы.
•lock (nolock) - Разрешает блокировку NFS (по умолчанию). nolock отключает блокировку NFS (не запускает демон lockd) и удобна при работе со старыми серверами, не поддерживающими блокировку NFS.
•mounthost=имя - Имя хоста, на котором запущен демон монтирования NFS - mountd.
•mountport=n - Порт, используемый демоном mountd.
•port=n - порт, используемый для подключения к NFS серверу (по умолчанию 2049, если демон rpc.nfsd не зарегистрирован на RPC-сервере). Если n=0 (по умолчанию), то NFS посылает запрос к portmap на сервере, чтобы определить порт.
•rsize=n (read block size - размер блока чтения) - Количество байтов, читаемых за один раз с NFS-сервера. Стандартно - 4096.

- wsize=n (write block size - размер блока записи) - Количество байтов, записываемых за один раз на NFS-сервер. Стандартно - 4096.
- tcp или udp - Для монтирования NFS использовать протокол TCP или UDP соответственно.
- bg - При потери доступа к серверу, повторять попытки в фоновом режиме, чтобы не блокировать процесс загрузки системы.
- fg - При потери доступа к серверу, повторять попытки в приоритетном режиме. Данный параметр может заблокировать процесс загрузки системы повторениями попыток монтирования. По этой причине параметр fg используется преимущественно при отладке.

Опции, влияющие на кэширование атрибутов при монтировании NFS

Атрибуты файлов, хранящиеся в inod (индексных дескрипторах), такие как время модификации, размер, жесткие ссылки, владелец, обычно изменяются не часто для обычных файлов и еще реже - для каталогов. Многи программы, например ls, обращаются к файлам только для чтения и не меняют атрибуты файлов или содержимое, но затрачивают ресурсы системы на дорогостоящие сетевые операции. Чтобы избежать ненужных затрат ресурсов, можно кэшировать данные атрибуты. Ядро использует время модификации файла, чтобы определить устарел ли кэш, сравнивая время модификации в кэше и время модификации самого файла. Кэш атрибутов периодически обновляется в соответствии с заданными параметрами:

- ac (noac) (attrebute cache - кэширование атрибутов) - Разрешает кэширование атрибутов (по-умолчанию). Хотя опция noac замедляет работу сервера, она позволяет избежать устаревания атрибутов, когда несколько клиентов активно записывают информацию в общию иерархию.
- acdirmax=n (attribute cache directory file maximum - кэширование атрибута максимум для файла каталога) - Максимальное количество секунд, которое NFS ожидает до обновления атрибутов каталога (по-умолчанию 60 сек.)
- acdirmin=n (attribute cache directory file minimum - кэширование атрибута минимум для файла каталога) - Минимальное количество секунд, которое NFS ожидает до обновления атрибутов каталога (по-умолчанию 30 сек.)
- acregmax=n (attribute cache regular file maximum - кэширование атрибута максимум для обычного файла) - Максимаьное количество секунд, которое NFS ожидает до обновления атрибутов обычного файла (по-умолчанию 60 сек.)
- acregmin=n (attribute cache regular file minimum- кэширование атрибута минимум для обычного файла) - Минимальное количество секунд, которое NFS ожидает до обновления атрибутов обычного файла (по-умолчанию 3 сек.)
- actimeo=n (attribute cache timeout - таймаут кэширования атрибутов) - Заменяет значения для всех вышуказаных опций. Если actimeo не задан, то вышеуказанные значения принимают значения по умолчанию.

Опции обработки ошибок NFS

Следующие опции управляют действиями NFS при отсутствии ответа от сервера или в случае возникновения ошибок ввода/вывода:

- fg (bg) (foreground - передний план, background - задний план) - Производить попытки монтирования отказавшей NFS на переднем плане/в фоне.
- hard (soft) - выводит на консоль сообщение "server not responding" при достижении таймаута и продолжает попытки монтирования. При заданной опции soft - при таймауте сообщает вызвавшей операцию программе об ошибке ввода/вывода. (опцию soft советуют не использовать)
- nointr (intr) (no interrupt - не прерывать) - Не разрешает сигналам прерывать файловые операции в жестко смонтированной иерархии каталогов при достижении большого таймаута. intr - разрешает прерывание.
- retrans=n (retransmission value - значение повторной передачи) - После n малых таймаутов NFS генерирует большой таймаут (по-умолчанию 3). Большой таймаут прекращает выполнение операций или выводит на консоль сообщение "server not responding", в зависимости от указания опции hard/soft.
- retry=n (retry value - значение повторно попытки) - Количество минут повторений службы NFS операций монтирования, прежде чем сдаться (по-умолчанию 10000).
- timeo=n (timeout value - значение таймаута) - Количество десятых долей секунды ожидания службой NFS до повторной передачи в случае RPC или малого таймаута (по-умолчанию 7). Это значение увеличивается при каждом таймауте до максимального значения 60 секунд или до наступления большого таймаута. В случае занятой сети, медленного сервера или при

прохождении запроса через несколько маршрутизаторов или шлюзов увеличение этого значения может повысить производительность.

Автоматическое монтирование NFS при загрузке (описание файловых систем в /etc/fstab)

Описание файла /etc/fstab я затрагивал в соответствующей статье. В текущем примере я рассмотрю несколько примеров монтирования файловых систем NFS с описанием опций:

```
FILES ~ # cat /etc/fstab | grep nfs

archiv:/archiv-small     /archivs/archiv-small  nfs
rw,timeo=4,rsize=16384,wsize=16384   0       0

nfs-server:/archiv-big   /archivs/archiv-big    nfs     rw,timeo=50,hard,fg
0       0
```

Первый пример монтирует файловую систему /archiv-small с хоста archiv в точку монтирования /archivs/archiv-small, тип файловой системы указан nfs (всегда необходимо указывать для данного типа), файловая система монтирована с опцией для чтения, записи (rw). Хост archiv подключен по быстрому локальному каналу, поэтому для повышения производительности параметр timeo уменьшен и существенно увеличены значения rsize и wsize. Поля для программ dump и fsck заданы в ноль, чтобы данные программы не использовали файловую систему, примонтированную по NFS. Второй пример монтирует файловую систему /archiv-big с хоста nfs-server. Т.к. к хосту nfs-server мы подключены по медленному соединению, параметр timeo увеличен до 5 сек (50 десятых долей сек), а так же жестко задан параметр hard, чтобы NFS продолжала перемонтировать файловую систему после большого таймаута, так же задан параметр fg, чтобы при загрузке системы и недоступности хоста nfs-server не произошло зависания.
Прежде чем сохранять изменения в /etc/fstab, обязательно попробуйте смонтировать вручную и убедитесь, что всё работает!!!

Повышение производительности NFS

На производительность NFS могут влиять несколько параметров, особенно при работе через медленные соединения. При работе с медленными и высоконагруженными соединениями, желательно использовать параметр hard, чтобы таймауты не привели к прекращению работы программ. Но необходимо осознавать, что если смонтировать файловую систему через NFS с параметром hard через fstab, а удаленный хост окажется недоступен, то при загрузке системы произойдет зависание.
Так же, одним из самых легких способов повышения производительности NFS - увеличение количества байтов, передаваемых за один раз. Размер в 4096 байт слишком мал для современных быстрых соединений, увеличивая это значение до 8192 и более можно экспериментальным путем найти оптимальную скорость.

Так же, не стоит упускать из внимания и настройки тайм-аутов. NFS ожидает ответа на пересылку данных в течении промежутка времени, указанного в опции timeo, если ответ за это время не получен, то выполняется повторная пересылка. Но на загруженных и медленных соединениях это время может быть меньше времени реакции сервера и способности каналов связи, в результате чего могут быть излишние повторные пересылки, замедляющие работу.По умолчанию, timeo равно 0,7 сек (700 миллисекунд). после неответа в течении 700 мс сервер совершит повторную пересылку и удвоит время ожидания до 1,4 сек., увеличение timeo будет продолжаться до максимального значения в 60 сек. Далее в зависимости от параметра hard/soft произойдет какое-либо действие (см.выше).
Подобрать оптимальный timeo для определенного значения передаваемого пакета (значений rsize/wsize), можно с помощью команды ping:

```
FILES ~ # ping -s 32768 archiv

PING archiv.DOMAIN.local (10.0.0.6) 32768(32796) bytes of data.

32776 bytes from archiv.domain.local (10.0.0.6): icmp_req=1 ttl=64 time=0.931 ms

32776 bytes from archiv.domain.local (10.0.0.6): icmp_req=2 ttl=64 time=0.958 ms
```

```
32776 bytes from archiv.domain.local (10.0.0.6): icmp_req=3 ttl=64 time=1.03 ms
32776 bytes from archiv.domain.local (10.0.0.6): icmp_req=4 ttl=64 time=1.00 ms
32776 bytes from archiv.domain.local (10.0.0.6): icmp_req=5 ttl=64 time=1.08 ms
^C
--- archiv.DOMAIN.local ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4006ms
rtt min/avg/max/mdev = 0.931/1.002/1.083/0.061 ms
```

Как видно, при отправке пакета размером 32768 (32Kb) время его путешествия от клиента до сервера и обратно плавает в районе 1 миллисекунды. Если данное время будет зашкаливать за 200 мс, то стоит задуматься о повышении значения timeo, чтобы оно превышало значение обмена в три-четыре раза. Соответственно, данный тест желательно делать во время сильной загрузки сети

Запуск NFS и настройка Firewall

Заметка скопипсчена с блога http://bog.pp.ru/work/NFS.html, за что ему огромное спасибо!!!

Запуск сервера NFS, монтирования, блокировки, квотирования и статуса с "правильными" портами (для сетевого экрана)

•желательно предварительно размонтировать все ресурсы на клиентах
•остановить и запретить запуск rpcidmapd, если не планируется использование NFSv4:

```
chkconfig --level 345 rpcidmapd off
service rpcidmapd stop
```

•если нужно, то разрешить запуск сервисов portmap, nfs и nfslock:

```
chkconfig --levels 345 portmap/rpcbind on
chkconfig --levels 345 nfs on
chkconfig --levels 345 nfslock on
```

•если нужно, то остановить сервисы nfslock и nfs, запустить portmap/rpcbind, выгрузить модули

```
service nfslock stop
service nfs stop
service portmap start # service rpcbind start
umount /proc/fs/nfsd
service rpcidmapd stop
rmmod nfsd
service autofs stop # где-то потом его надо запустить rmmod nfs rmmod nfs_acl rmmod lockd
```

•открыть порты в iptables
•для RPC: UDP/111, TCP/111
•для NFS: UDP/2049, TCP/2049
•для rpc.statd: UDP/4000, TCP/4000
•для lockd: UDP/4001, TCP/4001
•для mountd: UDP/4002, TCP/4002
•для rpc.rquota: UDP/4003, TCP/4003
•для сервера rpc.nfsd добавить в /etc/sysconfig/nfs строку

```
RPCNFSDARGS="--port 2049"
```

•для сервера монтирования добавить в /etc/sysconfig/nfs строку

```
MOUNTD_PORT=4002
```

•для настройки rpc.rquota для новых версий необходимо добавить в /etc/sysconfig/nfs строку

```
RQUOTAD_PORT=4003
```

•для настройки rpc.rquota необходимо для старых версий (тем не менее, надо иметь пакет quota 3.08 или свежее) добавить в /etc/services

```
rquotad 4003/tcp

rquotad 4003/udp
```

•проверит адекватность /etc/exports
•запустить сервисы rpc.nfsd, mountd и rpc.rquota (заодно запускаются rpcsvcgssd и rpc.idmapd, если не забыли их удалить)

```
service nfsd start

 или в новых версиях

service nfs start
```

•для сервера блокировки для новых систем добавить в /etc/sysconfig/nfs строки

```
LOCKD_TCPPORT=4001

LOCKD_UDPPORT=4001
```

•для сервера блокировки для старых систем добавить непосредственно в /etc/modprobe[.conf]:

```
options lockd nlm_udpport=4001 nlm_tcpport=4001
```

•привязать сервер статуса rpc.statd к порту 4000 (для старых систем в /etc/init.d/nfslock запускать rpc.statd с ключом -p 4000)

```
STATD_PORT=4000
```

•запустить сервисы lockd и rpc.statd

```
service nfslock start
```

•убедиться, что все порты привязались нормально с помощью "lsof -i -n -P" и "netstat -a -n" (часть портов используется модулями ядра, которые lsof не видит)
•если перед "перестройкой" сервером пользовались клиенты и их не удалось размонтировать, то придётся перезапустить на клиентах сервисы автоматического монтирования ("am-utils, "autofs)

Пример конфигурации NFS сервера и клиента

Конфигурация сервера

Если вы хотите сделать ваш разделённый NFS каталог открытым и с правом записи, вы можете использовать опциюall_squash в комбинации с опциями anonuid и anongid. Например, чтобы установить права для пользователя 'nobody' в группе 'nobody', вы можете сделать следующее:

```
ARCHIV ~ # cat /etc/exports

# Доступ на чтение и запись для клиента на 192.168.0.100, с доступом rw для пользователя 99 с gid 99

/files 192.168.0.100(rw,sync,all_squash,anonuid=99,anongid=99))

# Доступ на чтение и запись для клиента на 192.168.0.100, с доступом rw для пользователя 99 с gid 99

/files 192.168.0.100(rw,sync,all_squash,anonuid=99,anongid=99))
```

Это также означает, что если вы хотите разрешить доступ к указанной директории, nobody.nobody должен быть владельцем разделённой директории:

```
# chown -R nobody.nobody /files
```

Конфигурация клиента

На клиенте необходимо примонтировать удаленный каталогудобным способом, например командой mount:

```
FILES ~ # mount -t nfs archiv:/files /archivs/files
```

Резюме

Фух... Статья завершена. Сегодня мы изучили что такое Network File System и с чем ее едят, в следующей статье попытаюсь сделать HOWTO с аутентификацией Kerberos. Надеюсь материал получился доходчивым и нужным. Буду рад Вашим дополнениям и комментариям!