

Introduction au système de fichiers réseau NFS

Philippe Latu

philippe.latu(at)linux-france.org

<http://www.linux-france.org/prj/inetdoc/>

| | | |
|---|---|-------------------|
| Historique des versions | | |
| \$Revision: 1636 \$ | \$Date: 2011-04-29 23:42:34 +0200 (ven. 29 avril 2011) \$ | \$Author: latu \$ |
| Année universitaire 2010-2011 | | |
| Résumé | | |
| <p>L'objectif de ce premier support de travaux pratiques de la série est l'étude du système de fichiers réseau NFS. Il illustre les accès en «mode fichier» à une unité de stockage réseau. Ce mode d'accès correspond à un stockage de type NAS (Network Attached Storage). On débute avec l'étude du principe de fonctionnement des appels de fonctions RPC (Remote Procedure Call) puis on poursuit avec la configuration d'un serveur NFS qui exporte une arborescence de comptes utilisateurs. Côté client, on étudie les accès au système de fichiers réseau NFS suivant deux modes distincts : le montage manuel puis l'automontage.</p> | | |

Table des matières

| | |
|---|----|
| 1. Copyright et Licence | 1 |
| 1.1. Méta-information | 2 |
| 2. Adressage IP des postes de travail | 2 |
| 3. Protocole NFS et topologie de travaux pratiques | 2 |
| 4. Configuration commune au client et au serveur NFS | 4 |
| 4.1. Gestion des appels RPC | 4 |
| 4.2. Gestion des paquets NFS | 7 |
| 5. Configuration du client NFS | 9 |
| 5.1. Opérations manuelles de (montage démontage) NFS | 9 |
| 5.2. Opérations automatisées de (montage démontage) NFS | 11 |
| 6. Configuration du serveur NFS | 13 |
| 7. Gestion des droits sur le système de fichiers NFS | 17 |
| 8. Système de fichiers NFS & sécurité | 18 |
| 9. Documents de référence | 19 |

1. Copyright et Licence

Copyright (c) 2000,2011 Philippe Latu.
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Copyright (c) 2000,2011 Philippe Latu.
Permission est accordée de copier, distribuer et/ou modifier ce document selon les termes de la Licence de Documentation Libre GNU (GNU Free Documentation License), version 1.2 ou toute version ultérieure publiée par la Free Software Foundation ; sans Sections Invariables ; sans Texte de Première de Couverture, et sans Texte de Quatrième de Couverture. Une copie de la présente Licence est incluse dans la section intitulée « Licence de Documentation Libre GNU ».

1.1. Méta-information

Cet article est écrit avec *DocBook*¹ XML sur un système *Debian GNU/Linux*². Il est disponible en version imprimable au format PDF : [admin.reseau.nfs.synthese.pdf](#)³.

2. Adressage IP des postes de travail

Tableau 1. Affectation des adresses IP des postes de travaux pratiques

| Poste 1 | Poste 2 | Passerelle par défaut |
|----------|-----------|-----------------------|
| alderaan | bespin | 172.19.116.1/26 |
| centares | coruscant | 10.0.119.65/27 |
| dagobah | endor | 10.0.121.129/27 |
| felucia | geonosis | 172.19.114.129/26 |
| hoth | mustafar | 192.168.108.129/25 |
| naboo | tatooine | 10.5.6.1/23 |

Pour ces travaux pratiques, de nombreuses questions peuvent être traitées à l'aide du document de référence : *Nfsv4 configuration*⁴. Il faut cependant faire correspondre les configurations décrites dans ce document avec les configurations proposées avec les paquets de la distribution *Debian GNU/Linux*.

Pour chaque paire de postes de travaux pratiques, il faut attribuer les rôles serveur et client. Le serveur doit exporter une partie de son arborescence locale de système de fichiers et le client doit pouvoir y accéder de façon transparente via un montage du système de fichiers distant. Ce support de travaux pratiques fait suite à la présentation : *Systèmes de fichiers réseau*⁵.

3. Protocole NFS et topologie de travaux pratiques

Cette section reprend les éléments spécifiques au protocole NFS introduits lors de la présentation *Systèmes de fichiers réseau*⁶.

Plusieurs versions du protocole de système de fichiers réseau NFS sont disponibles. Chacune correspond à une «époque» ou à un mode d'exploitation. La vue ci-dessous illustre la distribution des fonctionnalités de la version 4 entre les espaces noyau et utilisateur. Elle est extraite de la page *NFSv4 Testing for Linux*⁷.

¹ <http://www.docbook.org>

² <http://www.debian.org>

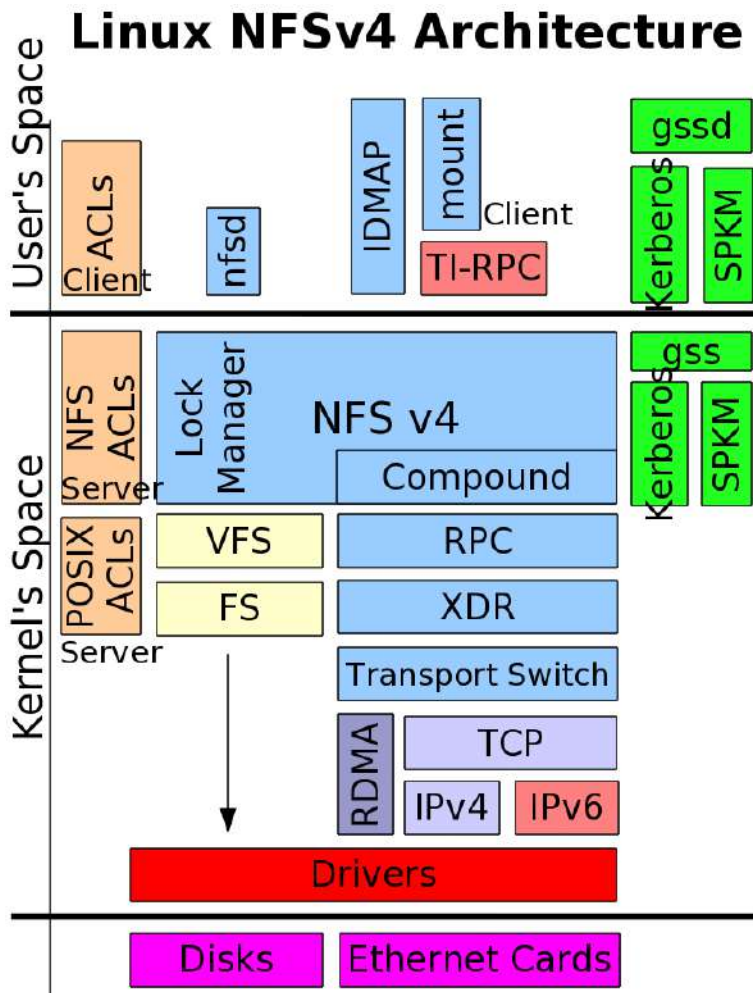
³ <http://www.linux-france.org/prj/inetdoc/telechargement/admin.reseau.nfs.synthese.pdf>

⁴ https://wiki.linux-nfs.org/wiki/index.php/Nfsv4_configuration_fr

⁵ <http://www.linux-france.org/prj/inetdoc/cours/admin.reseau.fs/>

⁶ <http://www.linux-france.org/prj/inetdoc/cours/admin.reseau.fs/>

⁷ <http://devresources.linuxfoundation.org/dev/nfsv4/site/architecture/>



Architecture NFSv4 Linux - vue complète⁸

La version 2 du protocole NFS a été la première à être largement adoptée à la fin des années 80. Elle a été conçue pour fournir un service de partage de fichiers entre les hôtes d'un même réseau local. Elle s'appuie sur le protocole UDP au niveau transport et sur le mécanisme d'appel de procédure distant (RPC) aux niveaux supérieurs.

La version 3 du protocole, introduite au milieu des années 90, a apporté de nombreuses améliorations en termes de fiabilité et de performances relativement à la précédente. Avec la version 3 du protocole :

- La taille maximum de fichier n'est plus limitée à 2Go.
- Les écritures asynchrones sur le serveur sont possibles ; ce qui améliore beaucoup les performances. Les requêtes en écriture des clients sont gérées en mémoire cache. Le client n'a plus à attendre que les demandes d'écritures soient effectivement appliquées sur les disques ce qui améliore les temps de réponse.
- Les contrôles d'accès sont effectués avant les manipulations sur les fichiers.
- La taille des données transférées n'est plus limitée à 8Ko.
- Il est possible d'utiliser le protocole TCP au niveau transport.

La version 4 du protocole apporte de nouvelles fonctionnalités relativement aux précédentes.

Les identifiants d'utilisateur et de groupe (*uid/gid*) sont représentés par des chaînes de caractères. Un nouveau service, baptisé *idmapd*, est utilisé sur le serveur *et* le client pour faire les correspondances entre les valeurs numériques locales et les chaînes de caractères. Ces nouvelles correspondances permettent d'utiliser de nouveaux contrôles d'accès indépendants entre clients et serveurs.

Les serveurs maintiennent un pseudo système de fichiers qui assure la cohérence du système de nommage avec les clients. Ainsi, un objet est nommé de façon identique entre le serveur et ses clients. Pour respecter les spécifications

⁸ <http://www.linux-france.org/prj/inetdoc/cours/admin.reseau.nfs/images/NFSv4.Schema.png>

POSIX, un client qui a accès à un niveau d'arborescence peut parcourir tous les niveaux inférieurs. Il n'est pas nécessaire d'exporter les sous arborescences.

Les appels de procédures distants n'utilisent plus le multiplexage de ports. Un numéro de port unique a été attribué à la version 4 du protocole NFS : tcp/2049. La version 3 doit utiliser plusieurs ports pour les traitements de ses protocoles complémentaires ; ce qui donne un assemblage plutôt complexe de ports et de couches avec des problèmes de sécurité propres. Aujourd'hui, ce mode de fonctionnement est abandonné et toutes les opérations de mise en œuvre de protocole complémentaire précédemment exécutées via des ports individuels sont maintenant traitées directement à partir d'un port unique connu.

Désormais, le mécanisme d'appel RPC n'est plus aussi important et sert essentiellement d'enveloppe pour les opérations encapsulées dans la pile NFSv4. Ce changement rend le protocole beaucoup moins dépendant de la sémantique du système de fichiers sous-jacent. Pour autant, les opérations de système de fichiers d'autres systèmes d'exploitation n'ont pas été négligées. Par exemple, les systèmes Windows exigent des appels *stateful* ouverts. Le mécanisme de suivi d'état de communication (*statefulness*) facilite l'analyse de trafic et rend les opérations de système de fichiers beaucoup plus simples à interpréter. Ce même mécanisme permet aux clients de gérer les données «en l'état» en mémoire cache.

La version 4 simplifie les requêtes en utilisant des opérations composées ou groupées (*compound*) qui englobent un grand nombre de traitements sur les objets du système de fichiers. L'effet immédiat est, bien sûr, une diminution très importante des appels RPC et des données qui doivent parcourir le réseau. Bien que chaque appel RPC transporte beaucoup plus de données en accomplit beaucoup plus de traitements, on considère qu'une requête composée de la version 4 du protocole exige cinq fois moins d'interactions client serveur qu'avec la version 3.

L'objectif des manipulations qui sont demandées dans ce document est d'illustrer les nouvelles fonctionnalités apportées par la dernière version du protocole NFS. Le séquençement des opérations à réaliser lors de la séance de travaux pratiques est décrit dans le tableau ci-dessous. Après le traitement de la première partie commune, les deux postes occupent chacun un rôle distinct.

Tableau 2. Attribution des rôles

| Client | Serveur |
|--|---------|
| Identification du mécanisme des appels RPC. Installation et configuration des paquets communs. | |

4. Configuration commune au client et au serveur NFS

Plusieurs services communs doivent être actifs pour que les accès au système de fichiers réseau NFS soient utilisables. Le mécanisme de gestion des appels de procédures distants appelé RPC ou *Remote Procedure Call* constitue le point de départ dans la mise œuvre de ces services communs.

Le logiciel de gestion des appels de procédures distants a évolué avec les différentes versions du système de fichiers NFS et l'arrivée du protocole réseau IPv6. La configuration étudiée ici doit permettre de fonctionner de la façon la plus transparente possible avec les versions 3 et 4 du système de fichiers NFS.



Note

Les manipulations présentées ici ne traitent pas le volet authentification et chiffrement des échanges sur le réseau. On considère que les services *Kerberos*, *SPKM-3* et *LIPKEY* ne sont pas actifs sur les systèmes étudiés.

4.1. Gestion des appels RPC

1. Quels sont les deux logiciels disponibles chargés de la gestion des appels RPC ? Qu'est-ce qui les distinguent ?

La présentation *Systèmes de fichiers réseau*⁹ introduit les principes de fonctionnement des appels de procédures distants.

⁹ <http://www.linux-france.org/prj/inetdoc/cours/admin.reseau.fs/>

Dans un premier temps, rechercher dans le support *Linux NFS-HOWTO*¹⁰ le service «historique» utilisé par NFS pour le multiplexage des appels de procédures distants. Dans un second temps, consulter la page *TI-RPC / rpcbind support*¹¹ pour identifier les évolutions apportées.

Le support *Linux NFS-HOWTO*¹² présente le service «historique» utilisé par NFS pour le multiplexage des appels de procédures distants : `portmap`. Ce service est fourni par le paquet du même nom et est limité au protocole réseau IPv4.

La page *TI-RPC / rpcbind support*¹³ présente un nouveau logiciel de multiplexage des mêmes appels de procédures distants : `rpcbind`. Ce nouveau démon est aussi fourni par le paquet du même nom. Il se veut plus évolutif que le précédent et supporte le protocole réseau IPv6.

2. Quels sont les paquets qui correspondent à ces logiciels ? Installer le paquet ouvrant les services de transport universels.

Utiliser les outils de recherche dans les répertoires de noms de paquets et dans leurs descriptions : **apt-cache**, **dpkg**, **aptitude**.

Comme indiqué dans la documentation, on recherche un paquet portant le nom `rpcbind`.

```
# aptitude search rpcbind
p  rpcbind - converts RPC program numbers into universal addresses
```

Une fois l'existence du paquet confirmée, on l'installe en acceptant la suppression de l'ancien paquet `portmap` qui est en conflit avec cette nouvelle version du même service.

```
# aptitude install rpcbind
Les NOUVEAUX paquets suivants vont être installés :
  libgssgluel{a} libtirpc1{a} rpcbind{b}
0 paquets mis à jour, 3 nouvellement installés, 0 à enlever et 0 non mis à jour.
Il est nécessaire de télécharger 155 ko d'archives. Après dépaquetage, 504 ko seront utilisés.
Les paquets suivants ont des dépendances non satisfaites :
  rpcbind: Est en conflit avec: portmap mais 6.0.0-3 est installé.
Les actions suivantes permettront de résoudre ces dépendances :

  Supprimer les paquets suivants :
1)   portmap

Accepter cette solution ? [Y/n/q/?]
Les NOUVEAUX paquets suivants vont être installés :
  libgssgluel{a} libtirpc1{a} rpcbind
Les paquets suivants seront ENLEVÉS :
  portmap{a}
0 paquets mis à jour, 3 nouvellement installés, 1 à enlever et 0 non mis à jour.
Il est nécessaire de télécharger 155 ko d'archives. Après dépaquetage, 319 ko seront utilisés.
Voulez-vous continuer ? [Y/n/?]
```

3. Quel est le numéro de port utilisé par le service ? Quel est le principe de fonctionnement du service pour le traitement des appels de procédures distants ?

Utiliser les commandes qui permettent d'obtenir les informations sur :

- La liste des processus actifs sur le système,
- Les numéros de ports en écoute sur les interfaces réseau,
- Les pages de manuels des applications utilisées.
- La liste des processus actifs sur le système,

```
# ps aux | grep rpc[b]ind
```

¹⁰ <http://nfs.sourceforge.net/nfs-howto/>

¹¹ http://nfsv4.bullopen-source.org/doc/tirpc_rpcbind.php

¹² <http://nfs.sourceforge.net/nfs-howto/>

¹³ http://nfsv4.bullopen-source.org/doc/tirpc_rpcbind.php

```
root      1988  0.0  0.1  18772   760 ?          Ss   23:49   0:00 /sbin/rpcbind -w
```

- Les numéros de ports en écoute sur les interfaces réseau,

```
# lsof -i | grep rpc[b]ind
rpcbind 1988      root    6u  IPv4  4884      0t0  UDP *:sunrpc
rpcbind 1988      root    7u  IPv4  4889      0t0  UDP *:891
rpcbind 1988      root    8u  IPv4  4890      0t0  TCP *:sunrpc (LISTEN)
rpcbind 1988      root    9u  IPv6  4893      0t0  UDP *:sunrpc
rpcbind 1988      root   10u  IPv6  4896      0t0  UDP *:891
rpcbind 1988      root   11u  IPv6  4897      0t0  TCP *:sunrpc (LISTEN)
```

On obtient la correspondance entre numéro de port et nom de service en consultant le fichier `/etc/services`.

```
# grep sunrpc /etc/services
sunrpc      111/tcp      portmapper   # RPC 4.0 portmapper
sunrpc      111/udp      portmapper
```

Le principe de fonctionnement des appels de procédures distants veut que tous ces appels soient reçus sur un numéro de port unique ; `sunrpc/111` dans le cas présent. Ces appels, une fois identifiés, sont transmis aux programmes concernés pour être traités.

- Les pages de manuels des applications utilisées.

```
# man rpcbind
```

4. Quelle est la commande qui permet de lister les services accessibles via un appel RPC ? À quel paquet appartient cette commande ?

Rechercher dans le support [Linux NFS-HOWTO](#)¹⁴ et dans la liste des fichiers du paquet sélectionné pour la gestion des appels RPC.

La commande présentée dans le support [Linux NFS-HOWTO](#)¹⁵ est appelée **rpcinfo**. On vérifie sa présence sur le système étudié de la façon suivante.

```
# dpkg -S `which rpcinfo`
rpcbind: /usr/sbin/rpcinfo
```

Dans la version la plus récente du programme, c'est l'option `-s` qui permet d'obtenir la présentation la plus synthétique des services accessibles par appel RPC.

```
# rpcinfo -s
program version(s) netid(s)          service  owner
100000  2,3,4      local,udp,tcp,udp6,tcp6      portmapper  superuser
```

La copie d'écran ci-dessus montre que le gestionnaire d'appel `portmapper` est le seul service ouvert. On relève l'ordre de priorité des différentes versions du service supportées par le système ainsi que les versions des protocoles de couche transport.

5. Donner deux exemples d'exécution : un en local et un sur le poste de travaux pratiques voisin.

L'exemple d'exécution de la commande en local est donné dans la copie d'écran de la question précédente. Pour connaître les services accessibles sur un autre poste, on utilise la même commande suivie de l'adresse IP de cet hôte.

```
# rpcinfo -s 192.200.0.4
program version(s) netid(s)          service  owner
100000  2,3,4      local,udp,tcp,udp6,tcp6      portmapper  superuser
```

Cette copie d'écran montre la même liste de paramètres que lors de l'exécution de la commande en local. Les configurations sur les deux hôtes sont donc identiques.

6. Réaliser une capture à l'aide de l'analyseur réseau lors de l'exécution de la commande et relever : le protocole de transport utilisé, les numéros de ports caractéristiques de cette transaction ainsi que le nom de la procédure RPC utilisée.

¹⁴ <http://nfs.sourceforge.net/nfs-howto/>

¹⁵ <http://nfs.sourceforge.net/nfs-howto/>



Voici un exemple de capture en mode console qui donne les éléments demandés.

```
# tshark -i eth0 not port 22
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:45: dofile has been disabled
Running as user "root" and group "root". This could be dangerous.
Capturing on eth0
192.200.0.3 -> 192.200.0.4  TCP 876 > sunrpc [SYN] Seq=0 Win=14600 Len=0 MSS=1460
192.200.0.4 -> 192.200.0.3  TCP sunrpc > 876 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460
192.200.0.3 -> 192.200.0.4  TCP 876 > sunrpc [ACK] Seq=1 Ack=1 Win=14624 Len=0
192.200.0.3 -> 192.200.0.4  Portmap V3 DUMP Call
192.200.0.4 -> 192.200.0.3  TCP sunrpc > 876 [ACK] Seq=1 Ack=45 Win=14496 Len=0
192.200.0.4 -> 192.200.0.3  Portmap V3 DUMP Reply (Call In 4)
192.200.0.3 -> 192.200.0.4  TCP 876 > sunrpc [ACK] Seq=45 Ack=697 Win=16000 Len=0
192.200.0.3 -> 192.200.0.4  TCP 876 > sunrpc [FIN, ACK] Seq=45 Ack=697 Win=16000 Len=0
192.200.0.4 -> 192.200.0.3  TCP sunrpc > 876 [FIN, ACK] Seq=697 Ack=46 Win=14496 Len=0
192.200.0.3 -> 192.200.0.4  TCP 876 > sunrpc [ACK] Seq=46 Ack=698 Win=16000 Len=0
```

- Le protocole de couche transport utilisé est TCP.
- Le numéro de port utilisé correspond bien au service enregistré sunrpc/111.
- Le sous-programme distant appelé est : Portmap V3 DUMP Call.

4.2. Gestion des paquets NFS

1. Quel est le paquet commun au client et au serveur ? Identifier le jeu de commandes fournies par ce paquet.

Rechercher dans la liste des paquets disponibles, ceux dont le nom débute par nfs.

```
# aptitude search ?name"^(nfs)"
v  nfs-client          -
p  nfs-common          - NFS support files common to client and server
p  nfs-kernel-server   - support for NFS kernel server
v  nfs-server          -
p  nfs4-acl-tools      - Commandline and GUI ACL utilities for the NFSv4 client
p  nfswatch            - Program to monitor NFS traffic for the console
```

Dans la liste ci-dessus, on identifie le paquet `nfs-common` qui correspond bien aux fonctions communes au client et au serveur NFS.

Une fois le paquet installé, la liste des programmes fournis par ce paquet est extraite de la liste de ses fichiers à l'aide de la commande suivante.

```
# dpkg -L nfs-common | grep bin
/sbin
/sbin/showmount
/sbin/rpc.statd
/sbin/mount.nfs
/sbin/sm-notify
/usr/sbin
/usr/sbin/nfsiostat
/usr/sbin/gss_clnt_send_err
/usr/sbin/gss_destroy_creds
/usr/sbin/rpcdebug
/usr/sbin/rpc.idmapd
/usr/sbin/mountstats
/usr/sbin/start-statd
/usr/sbin/rpc.gssd
/usr/sbin/nfsstat
/sbin/umount.nfs
/sbin/umount.nfs4
```

```
/sbin/mount.nfs4
```

Dans cette liste on trouve les commandes de montage, de démontage et de suivi d'état du système de fichiers réseau.

2. Quels sont les différents moyens qui permettent d'identifier l'ouverture du nouveau service suite à l'installation du paquet ?

Passer en revue les commandes qui listent les processus, les sockets (unix|inet) ouverts en écoute et les appels RPC.

La liste des processus actifs sur le système donne les informations suivantes.

```
# ps aux | grep [r]pc
root      1988  0.0  0.1 18772   972 ?    Ss   Apr13   0:00 /sbin/rpcbind -w
statd    2763  0.0  0.2 22948  1128 ?    Ss   00:40   0:00 /sbin/rpc.statd
root     2769  0.0  0.0     0     0 ?    S<   00:40   0:00 [rpciod]
root     2778  0.0  0.0 31352   436 ?    Ss   00:40   0:00 /usr/sbin/rpc.idmapd
```

Dans cette liste, on identifie les processus `rpc.statd` et `rpc.idmapd`.

Le premier est un démon qui implémente le protocole NSM (*Network Status Monitor*). Ce protocole est chargé de la gestion de l'état des verrous lors des échanges réseau.

Le second est chargé de faire la correspondance entre les identifiants numériques des utilisateurs (`uid|gid`) et les noms qui correspondent.

Pour en savoir plus sur les relations entre `rpc.idmapd` et les autres fonctions ou bibliothèques du système, on peut utiliser une commande du type `# lsof | grep rpc\.id`.

3. Réaliser une capture réseau lors de l'exécution des commandes et relever les protocoles et les numéros de ports caractéristiques de ces transactions. Relativement aux questions sur [Section 4.1, « Gestion des appels RPC »](#), est-ce que de nouveaux ports (tcp|udp) en écoute sont apparus ? Pourquoi ?

```

Poste 1                                     Poste 2
-----
<commande>      --- requête --->          <processus>
                                     <-- réponse ----
```

La capture réseau en mode console telle qu'elle est pratiquée dans la question ci-dessus ne montre aucune différence quant à l'utilisation du protocole de couche transport et des numéros de ports utilisés. La différence se situe dans le contenu au niveau de la couche application. La réponse à l'appel de sous-programme distant `Portmap V3 DUMP Call` contient des éléments supplémentaires relatifs aux services ouverts `idmapd` et `statd`.

Pour visualiser la liste des services accessibles via RPC avec l'analyseur réseau, il est préférable de passer en mode graphique. On peut réaliser la capture en mode console en stockant les résultats dans un fichier de capture et procéder à l'analyse en mode graphique à partir de ce fichier.

```
# tshark -i eth0 -w rpcinfo.pcap not port 22
```

4. Quel fichier de configuration faut-il éditer pour privilégier l'utilisation de la version 4 du protocole NFS ?

Rechercher le répertoire commun à l'ensemble des services du système dans lequel on trouve les fichiers de paramétrage de ces services. Une fois le répertoire identifié, on doit y trouver un fichier portant le nom du paquet `nfs-common`.

En amont des scripts de démarrage responsables de l'initialisation des services sur un système GNU/Linux (*runlevels*), le répertoire `/etc/default` contient des fichiers texte qui servent à positionner des variables d'environnement. Ces variables précisent les conditions dans lesquelles un service doit être exécuté.

```
# find /etc -type f -name nfs-common
/etc/default/nfs-common
```



```
/etc/init.d/nfs-common
```

5. Quels sont les paramètres à éditer pour privilégier l'utilisation de la version 4 du protocole NFS ? Éditez le fichier de configuration en conséquence et relancez le service

Rechercher dans les différences entre les versions du protocole NFS les éléments sur les échanges *stateless* et *stateful*.

Voici un *patch* des modifications apportées au fichier `/etc/default/nfs-common`.

```
# diff -uBb nfs-common.dist nfs-common
--- nfs-common.dist      2011-04-14 10:50:16.000000000 +0200
+++ nfs-common          2011-04-14 10:51:33.000000000 +0200
@@ -3,7 +3,7 @@
 # for the NEED_ options are "yes" and "no".

 # Do you want to start the statd daemon? It is not needed for NFSv4.
-NEED_STATD=
+NEED_STATD=no

 # Options for rpc.statd.
 # Should rpc.statd listen on a specific port? This is especially useful
@@ -13,7 +13,7 @@
 STATDOPTS=

 # Do you want to start the idmapd daemon? It is only needed for NFSv4.
-NEED_IDMAPD=
+NEED_IDMAPD=yes

 # Do you want to start the gssd daemon? It is required for Kerberos mounts.
NEED_GSSD=
```

Les choix effectués ici permettent de désactiver le processus `rpc.statd` et d'activer le processus `rpc.idmapd`.

Une fois le fichier édité, il est nécessaire de redémarrer le service pour que les changements de configuration soient pris en compte.

```
# /etc/init.d/nfs-common stop
Stopping NFS common utilities: idmapd.
# killall rpc.statd
# /etc/init.d/nfs-common start
Starting NFS common utilities: idmapd.
# rpcinfo -s
program version(s) netid(s)          service      owner
100000  2,3,4      local,udp,tcp,udp6,tcp6      portmapper  superuser
```

5. Configuration du client NFS

Le rôle du client est d'intégrer un accès au système de fichiers d'un hôte distant dans son arborescence locale. On parle de «montage NFS». Dans un premier temps, on teste les opérations de montage manuel. Bien sûr, ces tests ne peuvent aboutir que si une arborescence a été exportée par un serveur.

Ensuite, on teste les opérations de montage automatisées ou «automontage». Si le serveur NFS n'est pas encore disponible au moment des tests de montage manuel, il faut préparer les fichiers de configuration du service d'automontage.

5.1. Opérations manuelles de (montage|démontage) NFS

1. Quelle est la commande qui permet de tester la disponibilité du service de montage NFS sur un hôte distant ?

Reprenre l'utilisation de la commande identifiée dans la section précédente.

Relativement aux résultats de la section précédente, la liste des services accessibles via RPC s'est étoffée et le service NFS apparaît cliarement.

```
# rpcinfo -s 192.200.0.3
```

| program | version(s) | netid(s) | service | owner |
|---------|------------|-------------------------|------------|-----------|
| 100000 | 2,3,4 | local,udp,tcp,udp6,tcp6 | portmapper | superuser |
| 100003 | 4,3,2 | udp6,tcp6,udp,tcp | nfs | unknown |
| 100227 | 3,2 | udp6,tcp6,udp,tcp | - | unknown |
| 100021 | 4,3,1 | tcp6,udp6,tcp,udp | nlockmgr | unknown |
| 100005 | 3,2,1 | tcp6,udp6,tcp,udp | mountd | superuser |

2. Quelle est la commande qui permet d'identifier l'arborescence disponible à l'exportation sur le serveur NFS ?

Rechercher dans la liste des fichiers du paquet de service commun NFS.

Dans la liste des commandes fournies avec le paquet `nfs-common`, on trouve un programme appelé **showmount**. Après consultation des pages de manuels, on relève l'option `-e` qui permet de consulter l'arborescence exportée par un serveur depuis un client. Voici un exemple d'exécution.

```
# showmount -e 192.200.0.3
Export list for 192.200.0.3:
/home/exports/home 192.200.0.0/27
/home/exports      192.200.0.0/27
```

3. Quelle est la commande à utiliser pour les opérations de montage manuel ? À quel paquet appartient cette commande ? Cette commande est-elle exclusivement liée au protocole NFS ?

Après avoir consulté le support *Linux NFS-HOWTO*¹⁶, interroger la base de données des paquets, rechercher dans le contenu des paquets et consulter les pages de manuels.

La documentation indique que c'est la commande **mount** qui nous intéresse. On effectue ensuite les recherches avec le gestionnaire de paquets.

```
# dpkg -S `which mount`
mount: /bin/mount
```

La commande appartient au paquet du même nom. La consultation des pages de manuels `# man mount` montre que cette commande n'est pas réservée au seul protocole NFS mais à l'ensemble des opérations de montage pour tous les systèmes de fichiers utilisables.

4. Créer le répertoire `/ahome` destiné à «recevoir» le contenu répertoires utilisateurs exportés depuis le serveur NFS. Quelle est la syntaxe de la commande permettant de *monter* le répertoire exporté par le serveur NFS sur ce nouveau répertoire ?

Rechercher dans le support *Linux NFS-HOWTO*¹⁷.

Dans le contexte de ces manipulations, il est important de préciser la version du protocole NFS lors du montage manuel.

```
# mkdir /ahome
# mount -t nfs4 192.200.0.3:/home /ahome
# mount | grep nfs
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
192.200.0.3:/home on /ahome type nfs4 (rw,addr=192.200.0.3,clientaddr=192.200.0.4)
```

5. Quelles sont les options de montage disponibles avec le protocole NFS ? Relever la signification des options principales ?

Rechercher dans le support *Linux NFS-HOWTO*¹⁸. Les options caractéristiques sont : choix du protocole de transport, taille des blocs de données et version NFS. On peut aussi consulter les pages de manuels de la catégorie 5 concernant les formats de fichiers à l'aide de la commande `man 5 nfs`.

6. Réaliser une capture lors de l'exécution des commandes et relever les numéros de ports caractéristiques de ces transactions. Est-il possible de retrouver les informations échangées dans les données de capture ?

¹⁶ <http://nfs.sourceforge.net/nfs-howto/>

¹⁷ <http://nfs.sourceforge.net/nfs-howto/>

¹⁸ <http://nfs.sourceforge.net/nfs-howto/>

| Client | | Serveur |
|------------------|------------------------|------------|
| mount | --- requête RPC ---> | portmapper |
| mount | <--- numéro port --- | portmapper |
| mount | --- requête RPC ---> | mountd |
| mount | <-- réponse ----- | mountd |
| lecture/écriture | ---- I/O -----> | nfsd |
| lecture/écriture | <- ACK fin opération - | nfsd |

La marche à suivre est identique à celle de la **même question côté serveur** NFS.

7. Quelles seraient les opérations à effectuer pour configurer le système et rendre un montage NFS statique permanent ?

Rechercher le fichier de configuration système responsable des montages statiques des partitions.

Il est inutile de modifier les fichiers de configuration du système sachant que l'on change de méthode de montage dans la section suivante.

Il faudrait éditer le fichier `/etc/fstab` pour effectuer un montage statique à chaque initialisation du système. On pourrait par exemple insérer une ligne du type suivant à la fin du fichier.

```
192.200.0.3:/home /ahome nfs4 0 0
```

5.2. Opérations automatisées de (montage|démontage) NFS



Note

Il existe plusieurs implémentations libres pour le service d'automontage. On se limite ici au logiciel lié au noyau Linux.

Dans cette section, on reprend le processus de montage précédent en utilisant le service d'automontage. L'objectif étant de rendre les opérations d'accès au système de fichiers réseau totalement transparentes pour l'utilisateur, le recours au montage manuel doit être évité le plus possible.

1. Quel est le paquet qui contient les outils nécessaires au fonctionnement de l'automontage ?

Interroger les méta données dans le cache du gestionnaire de paquets en cherchant le mot clé **automount**.

La recherche dans le champ description du catalogue des paquets disponibles donne les résultats suivants.

```
# aptitude search "?description(automount)"
p  afuse          - automounting file system implemented in user-space using FUSE
p  am-utils       - automounter utilities from 4.4BSD (includes amd)
p  am-utils-doc   - automounter utilities documentation
p  autodir        - Automatically creates home and group directories for LDAP/NIS/SQL/local account
p  autofs5        - kernel-based automounter for Linux, version 5
p  autofs5-hesiod - Hesiod map support for autofs, version 5
p  autofs5-ldap   - LDAP map support for autofs, version 5
p  libamu-dev     - Support library for amd the 4.4BSD automounter (development)
p  libamu4        - Support library for amd the 4.4BSD automounter (runtime)
p  ltspfsd        - Fuse based remote filesystem hooks for LTSP thin clients
p  pmount         - mount removable devices as normal user
p  vfu            - A versatile text-based filemanager
```

Dans le contexte de ces manipulations, c'est le paquet `autofs5` qui nous intéresse.

2. Comment créer un compte utilisateur local baptisé `etu-nfs` avec un répertoire utilisateur situé sous la racine `/ahome` dont les fichiers et répertoires sont placés sur le serveur NFS ?

Après consultation des pages de manuels de la commande **adduser**, on dispose des options de création de compte respectant les deux critères énoncés. L'option `--home` permet de désigner le répertoire utilisateur dans l'arborescence système et l'option `--no-create-home` évite la création de ce répertoire sur le système local.

```
# adduser --no-create-home --home /ahome/etu-nfs etu-nfs
# id etu-nfs
uid=1001(etu-nfs) gid=1001(etu-nfs) groupes=1001(etu-nfs)
```

Les identifiants numériques `uid/gid` jouent un rôle important dans la suite des manipulations. Voir [Section 7, « Gestion des droits sur le système de fichiers NFS »](#).

3. Quels sont les fichiers de configuration du service d'automontage à éditer ou créer pour que l'utilisateur `etu-nfs` ait accès à ses données personnelles ?

Utiliser les fichiers exemples fournis avec le paquet, les pages de manuels associées et créer un fichier spécifique pour la gestion des comptes utilisateurs.

La liste des fichiers du paquet `autofs5` montre qu'il existe une page de manuel consacrée au fichier principal de configuration du service : `/etc/auto.master`. Ces informations permettent de configurer un point de montage au dessous duquel doivent se trouver les répertoires utilisateurs. Ces derniers utilisent un fichier de configuration propre : `/etc/auto.home`.

1. On définit la racine de montage `/ahome` dans le fichier de configuration principal `/etc/auto.master`. Cette racine de montage pointe vers le fichier de configuration dédié au montage automatique des répertoires des utilisateurs.

```
# grep -v ^# /etc/auto.master
/ahome /etc/auto.home
```

2. Le fichier `/etc/auto.home` utilise une syntaxe particulière pour que le montage du système de fichiers du serveur soit générique et indépendant du nombre des comptes utilisateurs.

```
# cat /etc/auto.home
* -fstype=nfs4 192.200.0.3:/home/&
```

- Le premier paramètre est le symbole `*` qui se substitue au nom d'utilisateur : `etu-nfs` dans notre exemple.
- Le deuxième paramètre `-fstype=nfs4` correspond à une option de montage qui privilégie la version 4 du protocole NFS. Le jeu des options de montage est le même que pour un montage statique.
- Le troisième paramètre est l'adresse IP du serveur. Comme on ne dispose pas d'un service DNS à ce stade de la progression des travaux pratiques, on utilise directement les adresses IP.
- Le répertoire `/home/` correspond à la configuration de l'exportation NFS [sur le serveur](#). Le répertoire `/home/` est situé sous la racine d'exportation qui est uniquement connue du serveur.
- Le symbole `&` indique la répétition du premier paramètre : le nom d'utilisateur.

4. Quelles sont les conditions à respecter sur le client et le serveur NFS pour que l'utilisateur `etu-nfs` ait la capacité à écrire dans son répertoire personnel ?

Rechercher les attributs d'un compte utilisateur qui correspondent aux propriétés des objets d'un système de fichiers au sens général.

Les identifiants numériques `uid/gid` doivent nécessairement être identiques sur le client et le serveur NFS. Toute la gestion des droits sur le système de fichiers est conditionnée par ces valeurs.

5. Comment prendre l'identité de l'utilisateur `etu-nfs` pour tester la validité du montage ?

Cette validation suppose que l'utilisateur puisse atteindre son répertoire et que l'on visualise l'automontage avec les commandes **mount** et **df**.

C'est la commande **su** qui permet de «changer d'identité» sur le système. On l'utilise donc pour prendre l'identité de l'utilisateur dont le répertoire est situé sur le serveur NFS. Pour que l'opération de montage automatique ait lieu, il suffit de se placer dans ce répertoire.

```
root@clnt:/home/etu# su etu-nfs
etu-nfs@clnt:/home/etu$ cd
```

```

etu-nfs@clnt:~$ pwd
/ahome/etu-nfs
etu-nfs@clnt:~$ df -h
Sys. de fichiers      Taille  Uti. Disp.  Uti% Monté sur
/dev/mapper/vm0-root  30G    806M   28G    3% /
tmpfs                 249M   4,0K   249M   1% /lib/init/rw
udev                 243M   160K   243M   1% /dev
tmpfs                 249M   0      249M   0% /dev/shm
/dev/vdal             228M   19M   197M   9% /boot
192.200.0.3:/home/etu-nfs
                    30G    806M   28G    3% /ahome/etu-nfs
etu-nfs@clnt:~$ mount
/dev/mapper/vm0-root on / type ext3 (rw,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
/dev/vdal on /boot type ext2 (rw)
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw)
192.200.0.3:/home/etu-nfs on /ahome/etu-nfs type nfs4 \
(rw,addr=192.200.0.3,clientaddr=192.200.0.4)

```

Bien sûr, ces manipulations ne sont possibles que si la **configuration du serveur** est effective.

- Réaliser une capture réseau lors de l'exécution des commandes et relever les numéros de ports caractéristiques de ces transactions. Est-il possible de retrouver les informations échangées dans les données de capture ?

La marche à suivre est identique à celle de la **même question côté serveur** NFS.

6. Configuration du serveur NFS

Le rôle du serveur NFS est de mettre à disposition sur le réseau une partie de son arborescence locale de système de fichiers. On parle d'«exportation».



Note

Il existe plusieurs implémentations libres de serveur NFS. On se limite ici à l'utilisation du logiciel lié au noyau Linux.

Cette section traite de l'installation d'un serveur NFS en version 4 dont le but est d'exporter le contenu des répertoires utilisateurs vers les clients.

- Quel est le paquet qui contient les outils nécessaires au fonctionnement du serveur NFS ? Installez ce paquet.

Interroger les méta données du gestionnaire de paquets pour identifier le nom du paquet à installer.

La recherche des mots clés `nfs` et `server` donne les résultats suivants.

```

# aptitude search '?and(nfs, server)'
p  nfs-kernel-server  - support for NFS kernel server
v  nfs-server

```

Les informations données par la commande `# aptitude show nfs-kernel-server` permettent de confirmer qu'il s'agit bien du paquet à installer.

```

# aptitude install nfs-kernel-server

```

- Quel est le fichier de configuration principal de gestion des exportations NFS ?

Rechercher dans le support *Linux NFS-HOWTO*¹⁹.

¹⁹ <http://nfs.sourceforge.net/nfs-howto/>

Quelles que soient les versions du protocole, c'est toujours le fichier `/etc/exports` qui est utilisé. Ce fichier est présenté dans le support *Linux NFS-HOWTO*²⁰. Le fichier livré avec le paquet contient, en commentaires, deux exemples complets de configuration NFSv3 et NFSv4. C'est ce dernier exemple que l'on adapte pour traiter les questions suivantes.

3. Créer le répertoire `/home/exports/home`. Quelle est la syntaxe à utiliser dans le fichier de configuration pour «exporter» ce répertoire ?

Rechercher dans les supports *Linux NFS-HOWTO*²¹ et *Nfsv4 configuration*²². On peut aussi utiliser les pages de manuels fournies avec le paquet du serveur NFS.

En exploitant la documentation *Nfsv4 configuration*²³ et l'exemple donné dans le fichier de configuration, on applique la configuration suivante.

```
# mkdir -p /home/exports/home
# grep -v ^# /etc/exports
/home/exports          192.200.0.0/27(rw, sync, fsid=0, crossmnt, no_subtree_check)
/home/exports/home     192.200.0.0/27(rw, sync, no_subtree_check)
```

Pour les besoins de ces travaux pratiques, les fonctions de sécurité *Kerberos* ne sont pas utilisées. On utilise l'appartenance au réseau IP comme critère de contrôle d'accès ; ce qui correspond à un niveau de sécurité faible.



Note

Du point de vue pédagogique, le choix d'une progression en séances de travaux pratiques autonomes et indépendantes implique que l'étude de la configuration *Kerberos* soit repoussée en dernière étape. En effet, le service *Kerberos* intervient à tous les niveaux : LDAP, NFS et authentification. Il peut faire l'objet d'une étude de synthèse supplémentaire une fois que les configurations des différentes fonctions ont été validées l'une après l'autre.

En ce qui concerne les options entre parenthèses, elles sont documentées dans les pages de manuels `exports` : `# man 5 exports`. Les éléments suivants en sont extraits.

- `rw` : autoriser les requêtes en lecture et en écriture sur le volume NFS. Le comportement par défaut est d'interdire toute requête qui modifierait le système de fichiers.
- `sync` : ne répondre aux requêtes qu'après l'exécution de tous les changements sur le support réel.
- `fsid=0` : avec NFSv4, un système de fichiers particulier est la racine de tous les systèmes de fichiers partagés. Il est défini par `fsid=root` ou `fsid=0`, qui veulent tous deux dire exactement la même chose.
- `crossmnt` : cette option permet aux clients de se déplacer du système de fichiers marqué `crossmnt` aux systèmes de fichiers partagés montés dessus. Voir l'option `nohide`.
- `no_subtree_check` : cette option neutralise la vérification de sous-répertoires, ce qui a des subtiles implications au niveau de la sécurité, mais peut améliorer la fiabilité dans certains cas. Si un sous-répertoire dans un système de fichiers est partagé, mais que le système de fichiers ne l'est pas, alors chaque fois qu'une requête NFS arrive, le serveur doit non seulement vérifier que le fichier accédé est dans le système de fichiers approprié (ce qui est facile), mais aussi qu'il est dans l'arborescence partagée (ce qui est plus compliqué). Cette vérification s'appelle `subtree_check`.

4. Qu'est-ce qui distingue l'exportation d'une arborescence entre les versions 3 et 4 du protocole NFS ?

Rechercher dans les différences relatives à la notion de nommage dans les manipulations proposées dans les supports *Linux NFS-HOWTO*²⁴ et *Nfsv4 configuration*²⁵.

Donner la signification du paramètre `fsid=0` dans la documentation relative à la version 4. Proposer une analogie avec le fonctionnement d'un serveur Web.

²⁰ <http://nfs.sourceforge.net/nfs-howto/>

²¹ <http://nfs.sourceforge.net/nfs-howto/>

²² https://wiki.linux-nfs.org/wiki/index.php/Nfsv4_configuration_fr

²³ https://wiki.linux-nfs.org/wiki/index.php/Nfsv4_configuration_fr

²⁴ <http://nfs.sourceforge.net/nfs-howto/>

²⁵ https://wiki.linux-nfs.org/wiki/index.php/Nfsv4_configuration_fr

Au delà des évolutions du protocole, c'est la cohérence du système de nommage qui distingue la version 4 du système de fichiers réseau. Il s'agit de garantir qu'un objet (fichier ou répertoire) soit représenté de la même manière sur un serveur et sur ses clients.

Dans le contexte de ces travaux pratiques les répertoires utilisateurs doivent être référencés à partir d'une racine nommée `/ahome/`.

Du point de vue infrastructure, l'utilisation de cette référence de nommage unique présente un avantage non négligeable. En effet, les répertoires d'exportation tels qu'ils ont été définis dans le fichier `/etc/exports` donné ci-dessus désignent un espace de stockage physique. La racine `/ahome/` désigne un espace de stockage logique. Ce schéma de nommage logique doit rester constant alors que les volumes de stockages physique peuvent migrer et se déplacer, être étendus, etc. sans qu'il soit nécessaire de remettre en question la configuration des clients.

Les différences entre les manipulations proposées dans les supports *Linux NFS-HOWTO*²⁶ et *Nfsv4 configuration*²⁷ traduisent les différences de conception entre les deux générations du protocole NFS. On peut relever deux paramètres importants sur le serveur.

- L'option `fsid=0`, présente dans le fichier `/etc/exports/`, permet de définir une *racine de montage* tout comme on le verrait sur un serveur Web. Le paramètre de configuration `DocumentRoot /var/www` du serveur *apache2* désigne la racine à partir de laquelle les pages Web publiées sont référencées. Cette racine est indépendante de l'arborescence du système de fichier local du serveur.
- L'utilisation d'un montage local avec l'option `bind` de la commande **mount** permet de mettre en cohérence l'arborescence du serveur et de ses clients. Ainsi, le répertoire `/ahome/` présente les mêmes objets que l'on soit connecté sur le serveur ou sur un client. Le schéma de nommage est donc cohérent.

Le montage local peut se faire manuellement sur le serveur avec la syntaxe suivante.

```
# mount --bind /home/exports/home /ahome
```

Une fois la configuration validée, on peut intégrer ce montage local dans la configuration système pour que l'opération soit effectuée à chaque initialisation. Il faut alors éditer le fichier de configuration dédié aux montages des volumes locaux du système : `/etc/fstab`. Voici un exemple donnant les dernières lignes d'un fichier `/etc/fstab` de serveur.

```
# tail -5 /etc/fstab
/dev/mapper/vm0-tmp      /tmp      ext3      defaults      0          2
/dev/mapper/vm0-usr     /usr      ext4      defaults      0          2
/dev/mapper/vm0-var     /var      ext4      defaults      0          2
/dev/mapper/vm0-swap    /none     swap      sw            0          0
/home/exports/home     /ahome    none      defaults,bind 0          0
```

5. Quelle est la commande qui permet de visualiser l'état courant de l'arborescence exportée ?

Rechercher dans la liste des fichiers du paquet relatif au serveur NFS.

La liste des commandes fournies avec le paquet `nfs-kernel-server` est la suivante.

```
# dpkg -L nfs-kernel-server | grep bin
/usr/sbin
/usr/sbin/exportfs
/usr/sbin/rpc.mountd
/usr/sbin/rpc.nfsd
/usr/sbin/rpc.svcgssd
```

Chacune de ces commandes dispose de pages de manuels. En consultant ces pages, on relève que la commande **exportfs** est chargée de la gestion de la liste des systèmes de fichiers partagés par NFS. L'exécution de cette commande sans argument affiche la liste des répertoires exportés. Dans notre cas, on obtient le résultat suivant.

```
# exportfs
/home/exports 192.200.0.0/27
/home/exports/home
```

²⁶ <http://nfs.sourceforge.net/nfs-howto/>

²⁷ https://wiki.linux-nfs.org/wiki/index.php/Nfsv4_configuration_fr

192.200.0.0/27

On peut ainsi vérifier que les directives données dans le fichier `/etc/exports` sont effectivement appliquées.

6. Quelles sont les principales options disponibles pour l'exportation d'une arborescence ? Relever la signification des paramètres.

Rechercher dans le support *Linux NFS-HOWTO*²⁸. On doit s'intéresser plus particulièrement aux options : `(rw|ro)`, `(sync|async)` et `*squash`.

Voici quelques éléments de réponse issus des pages de manuels : `# man 5 exports`

- L'option `rw` autorise les requêtes en lecture et en écriture sur le volume NFS alors que l'option `ro` interdit toute requête qui modifierait le système de fichiers.
- L'option `async` permet au serveur de transgresser le protocole NFS en répondant aux requêtes avant que tous les changements impliqués par la requête en cours n'aient été effectués sur le support réel (par exemple, le disque dur). L'utilisation de cette option améliore généralement les performances, mais au risque de perdre ou de corrompre des données en cas de redémarrage brutal du serveur. À l'opposé, l'option `sync` impose de ne répondre aux requêtes qu'après l'exécution de tous les changements sur le support réel.
- Les options `*_squash` sont relatives aux transformations des identifiants `uid` et `gid` entre le serveur NFS et ses clients. Par exemple, l'option `root_squash` transforme les requêtes avec un couple `uid/gid` à 0 (ie. le super-utilisateur) en un couple `uid/gid` anonyme.

7. Comment créer un compte utilisateur local baptisé `etu-nfs` avec un répertoire utilisateur situé sous la racine `/ahome` ?

Après consultation des pages de manuels de la commande `adduser`, on dispose des options de création de compte respectant le critère énoncé. L'option `--home` permet de désigner le répertoire utilisateur dans l'arborescence système.

```
# adduser --home /ahome/etu-nfs etu-nfs
# id etu-nfs
uid=1001(etu-nfs) gid=1001(etu-nfs) groupes=1001(etu-nfs)
```

Les identifiants numériques `uid/gid` jouent un rôle important dans la suite des manipulations. Voir [Section 7, « Gestion des droits sur le système de fichiers NFS »](#).

8. Réaliser une capture et relever les numéros de ports caractéristiques de des transactions de montage. Est-il possible de retrouver les informations échangées dans les données de capture ?

Pour réaliser cette capture, il faut synchroniser les opérations entre les postes client et serveur. On commence par le lancement de l'analyseur réseau puis on effectue un montage manuel par exemple pour caractériser les transactions réseau.

Voici un extrait de capture en mode console qui illustre la séquence de commande suivante exécutée sur le poste client.

```
showmount -e 192.200.0.3
mount -t nfs4 192.200.0.3:/home /ahome
ll /ahome/
umount /ahome/
```

Côté serveur, la capture réseau donne les résultats suivants.

```
Capturing on eth0
192.200.0.4 -> 192.200.0.3 Portmap V2 GETPORT Call MOUNT(100005) V:3 TCP
192.200.0.3 -> 192.200.0.4 Portmap V2 GETPORT Reply (Call In 1) Port:45696
192.200.0.4 -> 192.200.0.3 TCP apex-mesh > 45696 [SYN] Seq=0 Win=14600 Len=0 MSS=1460
192.200.0.3 -> 192.200.0.4 TCP 45696 > apex-mesh [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460
192.200.0.4 -> 192.200.0.3 TCP apex-mesh > 45696 [ACK] Seq=1 Ack=1 Win=14624 Len=0
192.200.0.4 -> 192.200.0.3 MOUNT V3 EXPORT Call
```

²⁸ <http://nfs.sourceforge.net/nfs-howto/>


```

192.200.0.3 -> 192.200.0.4 TCP 45696 > apex-mesh [ACK] Seq=1 Ack=73 Win=14496 Len=0
192.200.0.3 -> 192.200.0.4 MOUNT V3 EXPORT Reply (Call In 6)
192.200.0.4 -> 192.200.0.3 TCP apex-mesh > 45696 [ACK] Seq=73 Ack=141 Win=15680 Len=0
192.200.0.4 -> 192.200.0.3 TCP apex-mesh > 45696 [FIN, ACK] Seq=73 Ack=141 Win=15680 Len=0
192.200.0.3 -> 192.200.0.4 TCP 45696 > apex-mesh [FIN, ACK] Seq=141 Ack=74 Win=14496 Len=0
192.200.0.4 -> 192.200.0.3 TCP apex-mesh > 45696 [ACK] Seq=74 Ack=142 Win=15680 Len=0
192.200.0.4 -> 192.200.0.3 TCP 883 > nfs [SYN] Seq=0 Win=14600 Len=0 MSS=1460
192.200.0.3 -> 192.200.0.4 TCP nfs > 883 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460
192.200.0.4 -> 192.200.0.3 TCP 883 > nfs [ACK] Seq=1 Ack=1 Win=14624 Len=0
192.200.0.4 -> 192.200.0.3 NFS V4 NULL Call
192.200.0.3 -> 192.200.0.4 TCP nfs > 883 [ACK] Seq=1 Ack=45 Win=14496 Len=0
192.200.0.3 -> 192.200.0.4 NFS V4 NULL Reply (Call In 16)
192.200.0.4 -> 192.200.0.3 TCP 883 > nfs [ACK] Seq=45 Ack=29 Win=14624 Len=0
192.200.0.4 -> 192.200.0.3 NFS V4 COMPOUND Call <EMPTY> PUTROOTFH;GETFH;GETATTR
192.200.0.3 -> 192.200.0.4 NFS V4 COMPOUND Reply (Call In 20) <EMPTY> PUTROOTFH;GETFH;GETATTR
192.200.0.4 -> 192.200.0.3 NFS V4 COMPOUND Call <EMPTY> PUTFH;GETATTR
192.200.0.3 -> 192.200.0.4 NFS V4 COMPOUND Reply (Call In 22) <EMPTY> PUTFH;GETATTR
192.200.0.4 -> 192.200.0.3 NFS V4 COMPOUND Call <EMPTY> PUTFH;GETATTR
192.200.0.3 -> 192.200.0.4 NFS V4 COMPOUND Reply (Call In 24) <EMPTY> PUTFH;GETATTR
192.200.0.4 -> 192.200.0.3 NFS V4 COMPOUND Call <EMPTY> PUTFH;GETATTR
192.200.0.3 -> 192.200.0.4 NFS V4 COMPOUND Reply (Call In 26) <EMPTY> PUTFH;GETATTR
192.200.0.4 -> 192.200.0.3 NFS V4 COMPOUND Call <EMPTY> PUTFH;GETATTR
192.200.0.3 -> 192.200.0.4 NFS V4 COMPOUND Reply (Call In 28) <EMPTY> PUTFH;GETATTR
192.200.0.4 -> 192.200.0.3 NFS V4 COMPOUND Call <EMPTY> PUTFH;GETATTR
192.200.0.3 -> 192.200.0.4 NFS V4 COMPOUND Reply (Call In 30) <EMPTY> PUTFH;GETATTR
192.200.0.4 -> 192.200.0.3 NFS V4 COMPOUND Call <EMPTY> PUTFH;GETATTR
192.200.0.3 -> 192.200.0.4 NFS V4 COMPOUND Reply (Call In 32) <EMPTY> PUTFH;GETATTR
192.200.0.4 -> 192.200.0.3 NFS V4 COMPOUND Call <EMPTY> PUTFH;GETATTR
192.200.0.3 -> 192.200.0.4 NFS V4 COMPOUND Reply (Call In 34) <EMPTY> PUTFH;GETATTR
192.200.0.4 -> 192.200.0.3 NFS V4 COMPOUND Call <EMPTY> PUTFH;ACCESS;GETATTR
192.200.0.3 -> 192.200.0.4 NFS V4 COMPOUND Reply (Call In 36) <EMPTY> PUTFH;ACCESS;GETATTR
192.200.0.4 -> 192.200.0.3 NFS V4 COMPOUND Call <EMPTY> PUTFH;LOOKUP;GETFH;GETATTR
192.200.0.3 -> 192.200.0.4 NFS V4 COMPOUND Reply (Call In 38) <EMPTY> PUTFH;LOOKUP;GETFH;GETATTR
192.200.0.4 -> 192.200.0.3 TCP 883 > nfs [ACK] Seq=1205 Ack=1541 Win=18912 Len=0
192.200.0.4 -> 192.200.0.3 NFS V4 COMPOUND Call <EMPTY> PUTFH;ACCESS;GETATTR
192.200.0.3 -> 192.200.0.4 NFS V4 COMPOUND Reply (Call In 43) <EMPTY> PUTFH;ACCESS;GETATTR
192.200.0.4 -> 192.200.0.3 TCP 883 > nfs [ACK] Seq=1337 Ack=1773 Win=19968 Len=0
192.200.0.4 -> 192.200.0.3 NFS V4 COMPOUND Call <EMPTY> PUTFH;GETATTR
192.200.0.3 -> 192.200.0.4 NFS V4 COMPOUND Reply (Call In 46) <EMPTY> PUTFH;GETATTR
192.200.0.4 -> 192.200.0.3 NFS V4 COMPOUND Call <EMPTY> PUTFH;REaddir
192.200.0.3 -> 192.200.0.4 NFS V4 COMPOUND Reply (Call In 48) <EMPTY> PUTFH;REaddir
192.200.0.4 -> 192.200.0.3 TCP 883 > nfs [ACK] Seq=1609 Ack=2265 Win=22112 Len=0
192.200.0.4 -> 192.200.0.3 TCP 883 > nfs [FIN, ACK] Seq=1609 Ack=2265 Win=22112 Len=0
192.200.0.3 -> 192.200.0.4 TCP nfs > 883 [FIN, ACK] Seq=2265 Ack=1610 Win=17696 Len=0
192.200.0.4 -> 192.200.0.3 TCP 883 > nfs [ACK] Seq=1610 Ack=2266 Win=22112 Len=0

```

Les points caractéristiques illustrés par cette capture sont : l'utilisation du protocole TCP, l'utilisation du port enregistré 2049/nfs, les appels de sous-programmes par lots.

7. Gestion des droits sur le système de fichiers NFS

Le contrôle des droits sur les objets de l'arborescence exportée par le serveur NFS est limité au masque de permissions de ces objets. Il est donc important de faire correspondre les identifiants `uid` et `gid` entre le client et le serveur.

Les manipulations suivantes sont à réaliser en «concertation» entre les administrateurs des postes client et serveur. Le compte utilisateur `etu-nfs` doit avoir été créé sur le **serveur** et sur le **client**.



Note

Ces manipulations se font sans système de gestion centralisé de l'authentification. L'utilisation d'un annuaire LDAP pour fournir une base de comptes utilisateurs fait l'objet d'un support de travaux pratiques qui vient après celui-ci. Ce support se concentre sur le volet système de fichiers réseau.

1. Quelles sont les valeurs numériques des identifiants `uid` et `gid` du compte utilisateur `etu-nfs` sur le client et sur le serveur NFS ?

Si les valeurs diffèrent entre le client et le serveur, il faut détruire ces comptes utilisateurs et reprendre les options de la commande **adduser** pour fournir ces valeurs de façon explicite.

L'extrait du résultat de l'instruction # **adduser --help** ci-dessous montre les options utiles.

```
adduser [--home DIR] [--shell SHELL] [--no-create-home] [--uid ID]
[--firstuid ID] [--lastuid ID] [--gecos GECOS] [--ingroup GROUP | --gid ID]
[--disabled-password] [--disabled-login] USER
Ajoute un utilisateur normal
```

2. Sur quel poste peut-on créer des fichiers et des répertoires avec des masques de permissions ayant d'autres valeurs `uid` et `gid` que celles de l'utilisateur `etu-nfs` ? Quelles sont les options des commandes **chmod** et **chown** à utiliser pour réaliser ces opérations ?

Utiliser les pages de manuels des commandes.

C'est sur le serveur que le super-utilisateur a la possibilité de créer n'importe quel objet avec n'importe quel propriétaire dans la mesure où le système de fichiers est local et non réseau.

```
# cd /ahome/etu-nfs/
root@srvr:/ahome/etu-nfs# touch ThisOneIsMine
root@srvr:/ahome/etu-nfs# chown etu-nfs.etu-nfs ThisOneIsMine
root@srvr:/ahome/etu-nfs# touch ThisOneIsNotMine
root@srvr:/ahome/etu-nfs# chown 2000.2000 ThisOneIsNotMine
root@srvr:/ahome/etu-nfs# ll This*
-rw-r--r-- 1 etu-nfs etu-nfs 0 21 avril 00:36 ThisOneIsMine
-rw-r--r-- 1 2000 2000 0 21 avril 00:37 ThisOneIsNotMine
```

Côté client, les objets créés sont biens visibles mais la vue réseau du système de fichiers NFS passe par une correspondance des propriétaires.

```
root@clnt:/home/etu# su etu-nfs
etu-nfs@clnt:/home/etu$ cd
etu-nfs@clnt:~$ ll This*
-rw-r--r-- 1 etu-nfs etu-nfs 0 21 avril 00:36 ThisOneIsMine
-rw-r--r-- 1 nobody nogroup 0 21 avril 00:37 ThisOneIsNotMine
etu-nfs@clnt:~$ touch ThisOneIsMine
etu-nfs@clnt:~$ touch ThisOneIsNotMine
touch: impossible de faire un touch « ThisOneIsNotMine »: Permission non accordée
```

3. Quel est le service qui assure la conformité des identifiants entre serveur et client NFS ?

Reprendre la liste des service RPC actifs sur les deux systèmes.

Voir [Section 3, « Protocole NFS et topologie de travaux pratiques »](#). Le démon `rpc.idmapd` est fourni avec le paquet `nfs-common`.

8. Système de fichiers NFS & sécurité

Lors de leur conception, au début des années 80, la sécurité des mécanismes RPC la sécurité n'était pas une préoccupation. Il a donc fallu appliquer des fonctions de sécurité sur des protocoles qui n'étaient pas prévus pour.

Avec les versions 2 et 3 du protocole NFS, le service `portmap` ne dispose d'aucun mécanisme interne de sécurité. C'est la raison pour laquelle on lui associe les utilitaires *TCP wrapper* qui «encadrent» les accès aux appels RPC. Cette sécurisation à minima est très limitée puisqu'elle se limite à définir les adresses IP des hôtes qui peuvent accéder au service. Il n'est donc pas réaliste d'utiliser les versions 2 et 3 du protocole NFS sur un réseau étendu sans passer par des tunnels. De plus, l'affectation dynamique de numéro de port pour les montages avec ces versions du protocole ne facilite pas la configuration des pare feux.

Avec le développement de la version 4 du protocole NFS, des fonctions d'authentification basées sur la technologie *Kerberos* ont été introduites. L'affectation dynamique est abandonnée au profit d'un numéro de port unique : `tcp/2049`.

9. Documents de référence

Systèmes de fichiers réseau : NFS & CIFS

*Systèmes de fichiers réseau*²⁹ : présentation des modes de fonctionnement des systèmes de fichiers réseau NFS & CIFS. Cette présentation est à consulter avant d'aborder la [Section 3, « Protocole NFS et topologie de travaux pratiques »](#).

Linux NFS-HOWTO

*Linux NFS-HOWTO*³⁰ : documentation historique complète sur la configuration d'un serveur et d'un client NFS jusqu'à la version 3 incluse.

Nfsv4 configuration

*Nfsv4 configuration*³¹ : traduction française extraite des pages du projet CITI de l'université du Michigan.

NFSv4 Testing for Linux

*NFSv4 Testing for Linux*³² : Page du site *linuxfoundation.org* consacrée à la publication des tests de performances de la version 4 du système de fichiers réseau NFS. On y trouve les schémas sur l'architecture de la pile des protocoles utilisées par NFS et repris dans la [Section 3, « Protocole NFS et topologie de travaux pratiques »](#).

Autres liens

*Marque-pages Delicious sur NFSv4*³³

²⁹ <http://www.linux-france.org/prj/inetdoc/cours/admin.reseau.fs/>

³⁰ <http://nfs.sourceforge.net/nfs-howto/>

³¹ https://wiki.linux-nfs.org/wiki/index.php/Nfsv4_configuration_fr

³² <http://devresources.linuxfoundation.org/dev/nfsv4/site/architecture/>

³³ <http://delicious.com/phlatu/nfsv4>