

Administration système en réseau : synthèse NFS v4 & NIS

Philippe Latu

philippe.latu(at)linux-france.org

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

Historique des versions

\$Revision: 1321 \$ \$Date: 2008-09-24 10:21:50 +0200 (mer 24 sep 2008) \$ \$Author: latu \$

Année universitaire 2007-2008

Résumé

Cette synthèse est une évolution de la version précédente qui traite de la mise en œuvre des services usuels de l'administration système en réseau du monde (Unix|GNU/Linux). On retrouve ici l'authentification NIS et l'automontage de système de fichiers distant autofs. La nouveauté se situe au niveau de la configuration du service de système de fichiers réseau NFS version 4. L'objectif est d'illustrer les nouvelles fonctionnalités et les nouveaux concepts offerts par NFS v4 relativement aux générations précédentes.

Table des matières

1. Copyright et Licence	2
1.1. Meta-information	2
2. Architecture type	3
3. Configuration du service NIS	4
3.1. Configuration du serveur NIS	4
3.2. Création des bases de données du service NIS	5
3.3. Création du compte utilisateur de test NIS	6
3.4. Configuration du client NIS	7
4. Configuration du système de fichiers NFS	9
4.1. Paquets et services communs	9
4.2. Configuration du serveur NFS	9
4.3. Montage manuel NFS	11
5. Configuration de l'automontage NFS	14
5.1. Configuration du serveur NFS & NIS	14
5.2. Configuration du client NFS & NIS	14
6. Test final de connexion	16
7. Documents de référence	16

1. Copyright et Licence

Copyright (c) 2000,2008 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) 2000,2008 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. Meta-information

Cet article est écrit avec *DocBook*¹ XML sur un système *Debian GNU/Linux*². Il est disponible en version imprimable aux formats PDF et Postscript : admin.reseau.synthese-nfs4-nis.pdf³ | admin.reseau.synthese-nfs4-nis.ps.gz⁴.

¹ <http://www.docbook.org>

² <http://www.debian.org>

³ <http://www.linux-france.org/prj/inetdoc/telechargement/admin.reseau.synthese-nfs4-nis.pdf>

⁴ <http://www.linux-france.org/prj/inetdoc/telechargement/admin.reseau.synthese-nfs4-nis.ps.gz>

3. Configuration du service NIS

Le contenu de cette section est identique à la version précédente : *Synthèse NFS v3 & NIS*. On ne reprend que les étapes essentielles de validation du fonctionnement du service.

3.1. Configuration du serveur NIS

Il faut installer le paquet baptisé `nis` sur le serveur et sur le client. C'est en éditant le fichier de configuration du service `/etc/default/nis` que l'on attribue le rôle client ou serveur. Par défaut, c'est le rôle «client» qui est attribué ce qui suppose donc qu'un serveur est déjà en place lors de l'installation.

On attribue le rôle serveur avec l'option `NISSERVER`.

```
# head -10 /etc/default/nis
#
# /etc/default/nis      Configuration settings for the NIS daemons.
#
# Are we a NIS server and if so what kind (values: false, slave, master)
NISSERVER=master
# Are we a NIS client (i.e. start ypbind?)
NISCLIENT=true
```

On désigne manuellement ce serveur en éditant le fichier `/etc/yp.conf`.

```
# cat /etc/yp.conf
<snipped/>
#
# IMPORTANT:      For the "ypserver", use IP addresses, or make sure that
#                 the host is in /etc/hosts. This file is only interpreted
#                 once, and if DNS isn't reachable yet the ypserver cannot
#                 be resolved and ypbind won't ever bind to the server.
#
# ypserver ypserver.network.com
ypserver 127.0.0.1
```

On relance ensuite le servicei.

```
# /etc/init.d/nis restart
Starting NIS services: ypserv yppasswdd ypxfrd ypbind.
```

On contrôle de la liste des services RPC disponibles.

```
# rpcinfo -p
  program no_version protocole  no_port
    100000      2      tcp      111      portmapper
    100000      2      udp      111      portmapper
    100003      2      udp      2049     nfs
    100003      3      udp      2049     nfs
    100003      4      udp      2049     nfs
    100021      1      udp      34319    nlockmgr
    100021      3      udp      34319    nlockmgr
    100021      4      udp      34319    nlockmgr
    100003      2      tcp      2049     nfs
    100003      3      tcp      2049     nfs
    100003      4      tcp      2049     nfs
    100021      1      tcp      53676    nlockmgr
    100021      3      tcp      53676    nlockmgr
    100021      4      tcp      53676    nlockmgr
    100005      1      udp      52890    mountd
    100005      1      tcp      42001    mountd
```

```

100005 2 udp 52890 mountd
100005 2 tcp 42001 mountd
100005 3 udp 52890 mountd
100005 3 tcp 42001 mountd
100004 2 udp 667 ypserv
100004 1 udp 667 ypserv
100004 2 tcp 668 ypserv
100009 1 udp 670 yppasswdd
100004 1 tcp 668 ypserv
600100069 1 udp 673 fypxfrd
600100069 1 tcp 674 fypxfrd
100007 2 udp 682 ypbind
100007 1 udp 682 ypbind
100007 2 tcp 683 ypbind
100007 1 tcp 683 ypbind

```

3.2. Création des bases de données du service NIS

Une fois le serveur en place, il faut créer les bases de données distribuées par le service NIS. Ces bases étant stockées dans le répertoire `/var/yp/`, c'est à partir de ce répertoire que toutes les opérations suivantes doivent être effectuées.

Pour commencer, on édite le fichier `Makefile` qui contient l'ensemble des directives de manipulation des bases. On s'intéresse plus particulièrement à 2 sections du fichier.

UIDs, GIDs

On fixe arbitrairement la valeur minimum des `uid` et `gid` à 2000 de façon à éviter tout «mélange» avec la base de données des comptes utilisateur locaux du serveur NIS.

```

# We do not put password entries with lower UIDs (the root and system
# entries) in the NIS password database, for security. MINUID is the
# lowest uid that will be included in the password maps. If you
# create shadow maps, the UserID for a shadow entry is taken from
# the passwd file. If no entry is found, this shadow entry is
# ignored.
# MINGID is the lowest gid that will be included in the group maps.
MINUID=2000
MINGID=2000

```

`auto.master`, `auto.home`

On complète la liste des bases ou «cartes» gérées par le service NIS en y ajoutant les fichiers de configuration de l'automontage NFS. De cette façon, on évite d'avoir à configurer le service d'automontage sur tous les postes clients.

```

# If you don't want some of these maps built, feel free to comment
# them out from this list.

ALL = passwd group hosts rpc services netid protocols netgrp
ALL += auto.master auto.home

```

Comme la configuration de l'automontage n'est pas encore traitée, on se contente de créer des fichiers vides de façon à ne pas bloquer le fonctionnement du service NIS.

```

rubis:/var/yp# touch /etc/auto.master
rubis:/var/yp# touch /etc/auto.home

```

Une fois le fichier `Makefile` prêt, on lance la création des bases avec la commande `ypinit`.

```

rubis:/var/yp# /usr/lib/yp/ypinit -m

```

```

At this point, we have to construct a list of the hosts which will run NIS
servers. rubis is in the list of NIS server hosts. Please continue to add
the names for the other hosts, one per line. When you are done with the
list, type a <control D>.

```

```

next host to add: rubis

```

```

    next host to add:
The current list of NIS servers looks like this:

rubis

Is this correct? [y/n: y]
We need a few minutes to build the databases...
Building /var/yp/nis.lab/ypservers...
Running /var/yp/Makefile...
make[1]: Entering directory `/var/yp/nis.lab'
Updating passwd.byname...
Updating passwd.byuid...
Updating group.byname...
Updating group.bygid...
Updating hosts.byname...
Updating hosts.byaddr...
Updating rpc.byname...
Updating rpc.bynumber...
Updating services.byname...
Updating services.byservicename...
Updating netid.byname...
Updating protocols.bynumber...
Updating protocols.byname...
Updating netgroup...
Updating netgroup.byhost...
Updating netgroup.byuser...
Updating auto.master...
Updating auto.home...
Updating shadow.byname...
make[1]: Leaving directory `/var/yp/nis.lab'

rubis has been set up as a NIS master server.

Now you can run yppinit -s rubis on all slave server.
```

3.3. Création du compte utilisateur de test NIS

Pour les tests sur les services NIS et NFS on utilise un compte utilisateur spécifique baptisé `etu-nis`.

- ❶ On désigne un nouveau répertoire racine pour les comptes utilisateurs distribués par le service NIS. Ce répertoire sera utilisé par le service d'automontage NFS.
- ❷ Conformément à la règle adoptée lors de la création des bases NIS, on doit fixer une valeur d'uid supérieure ou égale à 2000 pour tous les comptes utilisateurs distribués.

```

rubis:/var/yp# mkdir /home/nis
rubis:/var/yp# adduser --home /home/nis/etu-nis❶ --uid 2000❷ etu-nis
Ajout de l'utilisateur etu-nis...
make: Entering directory `/var/yp'
make[1]: Entering directory `/var/yp/nis.lab'
Updating netid.byname...
make[1]: Leaving directory `/var/yp/nis.lab'
make: Leaving directory `/var/yp'
Adding new group `etu-nis' (2000).
make: Entering directory `/var/yp'
make[1]: Entering directory `/var/yp/nis.lab'
Updating group.byname...
Updating group.bygid...
Updating netid.byname...
make[1]: Leaving directory `/var/yp/nis.lab'
make: Leaving directory `/var/yp'
Adding new user `etu-nis' (2000) with group `etu-nis'.
make: Entering directory `/var/yp'
```

```

make[1]: Entering directory `/var/yp/nis.lab'
Updating passwd.byname...
Updating passwd.byuid...
Updating netid.byname...
Updating shadow.byname...
make[1]: Leaving directory `/var/yp/nis.lab'
make: Leaving directory `/var/yp'
Création du répertoire personnel /home/nis/etu-nis.
Copie des fichiers depuis /etc/skel
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Modification des informations relatives à l'utilisateur etu-nis
Entrez la nouvelle valeur ou « Entrée » pour conserver la valeur proposée
Nom complet []: Etudiant NIS
No de bureau []:
Téléphone professionnel []:
Téléphone personnel []:
Autre []:
Ces informations sont-elles correctes ? [o/N] o

```

D'une façon générale, à chaque modification d'un objet de la base de données du service NIS, il faut relancer la scrutation des directives listées dans le Makefile.

```

rubis:/var/yp# make
make[1]: Entering directory `/var/yp/nis.lab'
Updating passwd.byname...
Updating passwd.byuid...
Updating netid.byname...
Updating shadow.byname...
make[1]: Leaving directory `/var/yp/nis.lab'

```

3.4. Configuration du client NIS

Côté client, il faut aussi installer le paquet Debian `nis` et reprendre le nom de domaine choisi lors de la configuration du serveur.

```

saphir:~# apt-get install nis
saphir:~# nisdomainname
nis.lab
saphir:~# yptest
rubis

```

Une fois le démon `ypbind` lancé, on effectue un test d'accès à une base distribuée via NIS :

```

saphir:~# ypcat hosts
127.0.0.1      localhost
192.168.1.7   saphir
192.168.1.4   rubis

```

On suit ensuite les indications données dans les deux guides de configuration NIS : section *Setting Up the NIS Client* du guide *The Linux NIS(YP)/NYS/NIS+ HOWTO* et/ou section *HOW TO SETUP A LOCAL NIS CLIENT* de la [Documentation du paquet Debian nis](#).

On complète la configuration pour que les utilisateurs NIS puissent se connecter sur le client :

```

saphir:~# echo "+:~::~:" >>/etc/passwd
saphir:~# echo "+:~::~:" >>/etc/shadow
saphir:~# echo "+:~:::" >>/etc/group

```

Pour tester l'authentification de l'utilisateur test `etu-nis`, soit on utilise la commande `su etu-nis`, soit on utilise une console. Voici ce que l'on obtient sur la console n° 2 :

```

Debian GNU/Linux 3.1 saphir tty2

```

```
saphir login: etu-nis
Password:
Linux saphir 2.6.12-rc3 #3 Sun May 1 17:29:27 CEST 2005 i686 GNU/Linux

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
No directory, logging in with HOME=/❶
etu-nis@saphir:/$
```

- ❶ L'authentification est un succès mais le répertoire utilisateur est inaccessible puisqu'il n'a pas été exporté via NFS.

4. Configuration du système de fichiers NFS

4.1. Paquets et services communs

Liste des paquets à installer pour le fonctionnement commun du serveur et du client NFS.

portmap

portmap gère les appels de procédures distantes ou *Remote Procedure Call* (RPC).

nfs-common

nfs-common gère les états des transactions NFS.

État de l'installation des paquets Debian :

```
saphir:~# dpkg -l portmap nfs-common
Souhait=inconnU/Installé/supprimé/Purgé/H=à garder
| État=Non/Installé/fichier-Config/dépaqueté/échec-config/H=semi-installé
|/ Err?=(aucune)/H=à garder/besoin Réinstallation/X=les deux (État,Err: majuscule=mauvais)
||/ Nom                               Version                               Description
+++-----
ii  portmap                               5-10                                  The RPC portmapper
ii  nfs-common                             1.0.7-3                               NFS support files common to client and server
```

Contrôle de l'état des services :

```
saphir:~# ps aux |grep portmap
daemon    4770  0.0  0.1  1684  484 ?      Ss   09:07   0:00 /sbin/portmap -i 127.0.0.1
saphir:~# ps aux |grep rpc
root      5487  0.0  0.2  2452  932 ?      Ss   09:07   0:00 /sbin/rpc.statd
```

❶ Côté client le service de gestion des requêtes RPC n'est en écoute que sur l'interface de boucle locale lo.

```
saphir:~# netstat -autp |grep rpc
tcp        0      0  *:1005                *: *
           LISTEN          5487/rpc.statd
tcp        0      0  localhost.locald:sunrpc *: *
           LISTEN          4770/portmap
udp        0      0  *:999                 *: *
           5487/rpc.statd
udp        0      0  *:1002                *: *
           5487/rpc.statd
udp        0      0  localhost.locald:sunrpc *: *
           4770/portmap
```

Contrôle du fonctionnement local des requêtes RPC :

```
saphir:~# rpcinfo -p
program no_version protocole no_port
 100000      2    tcp      111  portmapper
 100000      2    udp      111  portmapper
 391002      2    tcp      790  sgi_fam
 100024      1    udp     1002  status
 100024      1    tcp     1005  status
```

4.2. Configuration du serveur NFS

Côté serveur, il faut installer le paquet Debian `nfs-kernel-server` en plus des paquets précédents listés dans la [Section 4.1, « Paquets et services communs »](#).

État de l'installation des paquets Debian :

```
rubis:~# dpkg -l portmap nfs-common nfs-kernel-server
Souhait=inconnU/Installé/supprimé/Purgé/H=à garder
| État=Non/Installé/fichier-Config/dépaqueté/échec-config/H=semi-installé
|/ Err?=(aucune)/H=à garder/besoin Réinstallation/X=les deux (État,Err: majuscule=mauvais)
||/ Nom                               Version                               Description
+++-----
ii  portmap                               5-10                                  The RPC portmapper
```

```
ii  nfs-common      1.0.7-3      NFS support files common to client and server
ii  nfs-kernel-server 1.0.7-3      Kernel NFS server support
```

Pour que le service de gestion des requêtes RPC accepte les requêtes émises par le client NFS, il faut que ce service soit en écoute sur le réseau local. On doit donc passer par une reconfiguration du paquet `portmap` et/ou une édition du fichier de configuration du service `/etc/default/portmap`.

```
rubis:~# dpkg-reconfigure portmap
<snipped/>

Portmap doit-il être lié à l'adresse de bouclage ? <Non>
```

Contrôle de l'état des services :

- Liste des processus actifs.

```
rubis:~# ps aux |grep -e rpc -e portmap
root      9511  0.0  0.2  2452  928 ?        Ss   11:51   0:00 /sbin/rpc.statd
daemon    9972  0.0  0.1  1688  464 ?        Ss   11:58   0:00 /sbin/portmap
```

- Liste des ports réseau ouverts.

```
rubis:~# netstat -autp |grep rpc
tcp        0      0  *:sunrpc          *:*        LISTEN     9972/portmap
tcp        0      0  *:789             *:*        LISTEN     9511/rpc.statd
udp        0      0  *:783             *:*        *          9511/rpc.statd
udp        0      0  *:786             *:*        *          9511/rpc.statd
udp        0      0  *:sunrpc          *:*        *          9972/portmap
```

Comme indiqué dans le support de travaux pratiques, on crée un répertoire et on configure son exportation vers le réseau local :

```
rubis:~# mkdir /var/exports
rubis:~# vim /etc/exports
rubis:~# cat /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
#                to NFS clients.  See exports(5).

/var/exports    192.168.1.0/24(sync❶,rw❷,no_root_squash❸,no_subtree_check❹)
```

- La réponse à une requête ne peut être émise que lorsque les opérations sur le support de stockage sont terminées.
- Le répertoire est exporté en lecture et écriture.
- Le super-utilisateur sur le client n'a plus les mêmes droits sur l'arborescence exportée par le serveur.
- La routine `subtree_check` a pour but de contrôler qu'un fichier demandé par le client appartient bien à l'arborescence exportée par le serveur. En règle générale, il faut désactiver cette fonction lorsque l'on exporte des répertoires utilisateurs sur lesquels les modifications et les opérations d'écriture sont fréquentes.

Les explications complètes sont disponibles à partir de la section *Setting Up an NFS Server* du [Linux NFS-HOWTO](#) et des pages de manuels du paquet serveur `nfs-kernel-server` : `man exports`.

```
rubis:~# /etc/init.d/nfs-kernel-server restart
Stopping NFS kernel daemon: mountd nfsd.
Unexporting directories for NFS kernel daemon...done.
Exporting directories for NFS kernel daemon...done.
Starting NFS kernel daemon: nfsd mountd.
```

Contrôle de l'état des services :

- Liste des processus actifs.

```
rubis:~# ps aux |grep -e rpc -e portmap
root      9511  0.0  0.2  2452  928 ?        Ss   11:51   0:00 /sbin/rpc.statd
daemon    9972  0.0  0.1  1688  468 ?        Ss   11:58   0:00 /sbin/portmap
```

```
root    10708  0.0  0.0    0    0 ?      S<   12:11   0:00 [rpciod/0]
root    10710  0.0  0.2  2492  908 ?      Ss   12:11   0:00 /usr/sbin/rpc.mountd
```

- Catalogue des services RPC disponibles.

```
rubis:~# rpcinfo -p
  program no_version  protocole  no_port
  100000    2      tcp      111    portmapper
  100000    2      udp      111    portmapper
  391002    2      tcp      903    sgi_fam
  100024    1      udp      786    status
  100024    1      tcp      789    status
  100003    2      udp      2049   nfs
  100003    3      udp      2049   nfs
  100003    4      udp      2049   nfs
  100003    2      tcp      2049   nfs
  100003    3      tcp      2049   nfs
  100003    4      tcp      2049   nfs
  100021    1      udp      1041   nlockmgr
  100021    3      udp      1041   nlockmgr
  100021    4      udp      1041   nlockmgr
  100021    1      tcp      1029   nlockmgr
  100021    3      tcp      1029   nlockmgr
  100021    4      tcp      1029   nlockmgr
  100005    1      udp      710    mountd
  100005    1      tcp      713    mountd
  100005    2      udp      710    mountd
  100005    2      tcp      713    mountd
  100005    3      udp      710    mountd
  100005    3      tcp      713    mountd
```

4.3. Montage manuel NFS

Contrôle de la disponibilité du service NFS :

```
saphir:~# showmount -e rubis
Export list for rubis:
/var/exports 192.168.1.0/24
```

Suivant l'état du fichier `/etc/hosts` ou du service DNS, il faut éventuellement utiliser la commande avec une adresse IP. Par exemple : `showmount -e 192.168.1.4`.

L'analyse réseau correspondant à la commande ci-dessus illustre les points suivants :

- ❶ L'ensemble des communications utilise le protocole de couche transport TCP. Comme TCP est un protocole orienté connexion garantissant une grande fiabilité de communication, on pourra utiliser les services RPC sur un réseau étendu. On retrouve donc les séquences usuelles d'établissement (`[SYN]`, `[SYN, ACK]`, `[ACK]`) de maintien (`[ACK]`) et de libération (`[FIN, ACK]`, `[ACK]`) de connexion.
- ❷ La première séquence utilise le port client 914 et le port serveur `sunrpc|111`. Elle consiste en un appel de procédure distante `Portmap V2 GETPORT` qui permet d'attribuer un numéro de port pour la séquence suivante.
- ❸ Une fois le port 713 attribué, le client (port 915) peut interroger le serveur NFS pour connaître ses répertoires exportés. L'appel de procédure distante utilisé est `MOUNT V1 EXPORT`.
- ❹ Les évolutions des numéros de séquence et d'acquittement donnent une image des quantités de données échangées. Les données reçues par le client sont celles affichées lors de l'exécution de la commande **showmount**.

```
phil@saphir:~$ sudo tshark -i wlan0
Capturing on wlan0
 1 192.168.1.7 -> 192.168.1.4  TCP❶ 914❷ sunrpc [SYN] Seq=0 Ack=0 Win=5840 Len=0
 2 192.168.1.4 -> 192.168.1.7  TCP sunrpc > 914 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0
 3 192.168.1.7 -> 192.168.1.4  TCP 914 > sunrpc [ACK] Seq=1 Ack=1 Win=5840 Len=0
 4 192.168.1.7 -> 192.168.1.4  Portmap V2 GETPORT Call MOUNT(100005) V:1 TCP
 5 192.168.1.4 -> 192.168.1.7  TCP sunrpc > 914 [ACK] Seq=1 Ack=61 Win=5792 Len=0
 5 192.168.1.4 -> 192.168.1.7  Portmap V2 GETPORT Reply (Call In 10) Port:713❸
```

```

6 192.168.1.7 -> 192.168.1.4 TCP 914 > sunrpc [ACK] Seq=61 Ack=33 Win=5840 Len=0
7 192.168.1.7 -> 192.168.1.4 TCP 914 > sunrpc [FIN, ACK] Seq=61 Ack=33 Win=5840 Len=0
8 192.168.1.7 -> 192.168.1.4 TCP 915 > 713 [SYN] Seq=0 Ack=0 Win=5840 Len=0
9 192.168.1.4 -> 192.168.1.7 TCP sunrpc > 914 [FIN, ACK] Seq=33 Ack=62 Win=5792 Len=0
10 192.168.1.7 -> 192.168.1.4 TCP 914 > sunrpc [ACK] Seq=62 Ack=34 Win=5840 Len=0
11 192.168.1.4 -> 192.168.1.7 TCP 713 > 915 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0
12 192.168.1.7 -> 192.168.1.4 TCP 915 > 713 [ACK] Seq=1 Ack=1 Win=5840 Len=0
13 192.168.1.7 -> 192.168.1.4 MOUNT V1 EXPORT Call
14 192.168.1.4 -> 192.168.1.7 TCP 713 > 915 [ACK] Seq=1 Ack=77 Win=5792 Len=0
15 192.168.1.4 -> 192.168.1.7 MOUNT V1 EXPORT Reply (Call In 20)
16 192.168.1.7 -> 192.168.1.4 TCP 915 > 713 [ACK] Seq=77 Ack=81 Win=5840 Len=0
17 192.168.1.7 -> 192.168.1.4 TCP 915 > 713 [FIN, ACK] Seq=77 Ack=81 Win=5840 Len=0
18 192.168.1.4 -> 192.168.1.7 TCP 713 > 915 [FIN, ACK] Seq=81 Ack=78 Win=5792 Len=0
19 192.168.1.7 -> 192.168.1.4 TCP 915 > 713 [ACK] Seq=78 Ack=82 Win=5840 Len=0

```

Voici un exemple de syntaxe de montage d'un répertoire. Il est volontairement «riche» en options de façon à illustrer les fonctions NFS :

```

saphir:~# mkdir /mnt/nfs
saphir:~# mount -t nfs -o nfsvers=3❶,tcp❷,posix❸ rubis:/var/exports /mnt/nfs
saphir:~# ls /mnt/nfs/
saphir:~# umount /mnt/nfs

```

- ❶ On impose l'utilisation de la version 3 du protocole NFS. Cette version doit normalement être utilisée par défaut avec la distribution Debian.
- ❷ On impose le protocole de transport TCP. Là encore, c'est normalement le protocole de transport par défaut avec la version 3 du protocole NFS.
- ❸ On impose la conformité aux règles POSIX pour les opérations sur le système de fichiers.

Voici une analyse réseau expurgée des établissements, maintiens et libérations de connexion TCP. On y retrouve les appels de procédures distantes correspondant à chaque commande de la capture d'écran ci-avant :

```

1 192.168.1.7 -> 192.168.1.4 Portmap V2 GETPORT Call NFS(100003) V:3 TCP
2 192.168.1.4 -> 192.168.1.7 Portmap V2 GETPORT Reply (Call In 1) Port:2049
3 192.168.1.7 -> 192.168.1.4 Portmap V2 GETPORT Call NFS(100003) V:3 TCP
4 192.168.1.4 -> 192.168.1.7 Portmap V2 GETPORT Reply (Call In 3) Port:2049
5 192.168.1.7 -> 192.168.1.4 NFS V3 NULL Call
6 192.168.1.4 -> 192.168.1.7 NFS V3 NULL Reply (Call In 5)
7 192.168.1.7 -> 192.168.1.4 Portmap V2 GETPORT Call MOUNT(100005) V:3 TCP
8 192.168.1.4 -> 192.168.1.7 Portmap V2 GETPORT Reply (Call In 7) Port:713
9 192.168.1.7 -> 192.168.1.4 Portmap V2 GETPORT Call MOUNT(100005) V:3 TCP
10 192.168.1.4 -> 192.168.1.7 Portmap V2 GETPORT Reply (Call In 9) Port:713
11 192.168.1.7 -> 192.168.1.4 MOUNT V3 NULL Call
12 192.168.1.4 -> 192.168.1.7 MOUNT V3 NULL Reply (Call In 11)
13 192.168.1.7 -> 192.168.1.4 MOUNT V3 MNT Call
14 192.168.1.4 -> 192.168.1.7 MOUNT V3 MNT Reply (Call In 13)
15 192.168.1.7 -> 192.168.1.4 NFS V3 FSINFO Call, FH:0x04780406
16 192.168.1.4 -> 192.168.1.7 NFS V3 FSINFO Reply (Call In 15)
17 192.168.1.7 -> 192.168.1.4 NFS V3 GETATTR Call, FH:0x04780406
18 192.168.1.4 -> 192.168.1.7 NFS V3 GETATTR Reply (Call In 17)
19 192.168.1.7 -> 192.168.1.4 NFS V3 ACCESS Call, FH:0x04780406
20 192.168.1.4 -> 192.168.1.7 NFS V3 ACCESS Reply (Call In 19)
21 192.168.1.7 -> 192.168.1.4 NFS V3 REaddirPLUS Call, FH:0x04780406
22 192.168.1.4 -> 192.168.1.7 NFS V3 REaddirPLUS Reply (Call In 21) . . .
23 192.168.1.7 -> 192.168.1.4 Portmap V2 GETPORT Call MOUNT(100005) V:3 TCP
24 192.168.1.4 -> 192.168.1.7 Portmap V2 GETPORT Reply (Call In 23) Port:713
25 192.168.1.7 -> 192.168.1.4 Portmap V2 GETPORT Call MOUNT(100005) V:3 TCP
26 192.168.1.4 -> 192.168.1.7 Portmap V2 GETPORT Reply (Call In 25) Port:713
27 192.168.1.7 -> 192.168.1.4 MOUNT V3 NULL Call
28 192.168.1.4 -> 192.168.1.7 MOUNT V3 NULL Reply (Call In 27)
29 192.168.1.7 -> 192.168.1.4 MOUNT V3 UMNT Call
30 192.168.1.4 -> 192.168.1.7 MOUNT V3 UMNT Reply (Call In 29)

```

On retrouve le numéro de port 2049 dans les attributions issues des appels de procédures distantes RPC. Il peut être intéressant d'effectuer une capture réseau spécifique à partir de ce numéro de port. Voici un exemple de syntaxe de capture :

```
phil@saphir:~$ sudo tshark -i wlan0 -R "tcp.port==2049"
```

5. Configuration de l'automontage NFS

5.1. Configuration du serveur NFS & NIS

Côté serveur, deux opérations sont à effectuer : exporter les répertoires utilisateur sur le réseau local et distribuer les fichiers de configuration de l'automontage.

Pour l'exportation des répertoires utilisateur, on reprend la syntaxe présentée dans la [Section 4.2, « Configuration du serveur NFS »](#) et on complète le fichier `/etc/exports`, on relance le serveur NFS et on vérifie le résultat.

```
rubis:/var/yp# cat /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
#           to NFS clients.  See exports(5).

/var/exports    192.168.1.0/24(sync,rw,no_root_squash,no_subtree_check)
/home/nis       192.168.1.0/24(sync,rw,no_root_squash,no_subtree_check)
<snipped/>
rubis:/var/yp# /etc/init.d/nfs-kernel-server restart
Stopping NFS kernel daemon: mountd nfsd.
Unexporting directories for NFS kernel daemon...done.
Exporting directories for NFS kernel daemon...done.
Starting NFS kernel daemon: nfsd mountd.
<snipped/>
rubis:/var/yp# exportfs
/var/exports    192.168.1.0/24
/home/nis       192.168.1.0/24
```

Pour les fichiers de configuration de l'automontage, il faut compléter deux fichiers :

`/etc/auto.master`

Le fichier maître du service `autofs` qui désigne la racine de montage des répertoires utilisateur.

```
rubis:/var/yp# cat /etc/auto.master
/home/nis      /etc/auto.home
```

`/etc/auto.home`

Le fichier des répertoires utilisateur qui désigne les utilisateurs dont l'arborescence utilise l'automontage.

```
rubis:/var/yp# cat /etc/auto.home
*      -rw,hard,intr,nosuid,rsize=8192,wsiz=8192      rubis:/home/nis/&
```



Avertissement

Ne pas utiliser les tabulations dans ces fichiers, mais uniquement les espaces !

Surtout, ne pas oublier de mettre à jour les bases NIS :

```
rubis:/var/yp# make
make[1]: Entering directory `/var/yp/nis.lab'
Updating netid.byname...
Updating auto.master...
Updating auto.home...
make[1]: Leaving directory `/var/yp/nis.lab'
```

5.2. Configuration du client NFS & NIS

Côté client, il faut installer le paquet Debian `autofs` et éditer le fichier de configuration principal du service.

État de l'installation du paquet :

```
saphir:~# dpkg -l autofs
```

```

Souhait=inconnU/Installé/suppRimé/Purgé/H=à garder
| État=Non/Installé/fichier-Config/dépaqUeté/échec-conFig/H=semi-installé
|/ Err?=(aucune)/H=à garder/besoin Réinstallation/X=les deux (État,Err: majuscule=mauvais)
||/ Nom                Version                Description
++-----
ii  autofs                4.1.3+4.1.4beta2-7  kernel-based automounter for Linux

```

Fichier de configuration `/etc/auto.master` complété :

```

saphir:~# cat /etc/auto.master
+auto.master

```



Avertissement

Il ne doit pas y avoir de fichier `/etc/auto.home` sur le client. Le fichier `/etc/auto.master` doit être le seul fichier consulté.

On relance ensuite le service `autofs` et on contrôle son état :

```

saphir:~# /etc/init.d/autofs restart
Stopping automounter: done.
Starting automounter: done.
<snipped/>
saphir:~# ps aux |grep automount
root      20083  0.0  0.2   2504   964 pts/2    S   17:59   0:00 /usr/sbin/automount \
  --pid-file=/var/run/autofs/_home_nis.pid --timeout=300 /home/nis yp auto.home
root      20089  0.0  0.1   2152   776 pts/2    S+  17:59   0:00 grep automount
<snipped/>
saphir:~# /etc/init.d/autofs status
Configured Mount Points:
-----
/usr/sbin/automount --timeout=300 /home/nis yp auto.home

Active Mount Points:
-----
/usr/sbin/automount --pid-file=/var/run/autofs/_home_nis.pid \
  --timeout=300 /home/nis yp auto.home

```

6. Test final de connexion

Après toutes ces étapes de configuration longues et douloureuses, on termine par un test final de connexion. Notre utilisateur `etu-nis` bénéficie d'un service totalement transparent. Il peut se connecter (via NIS) sur n'importe quel poste client et retrouver son répertoire utilisateur (via NFS) dans l'état où il l'avait laissé lors de sa connexion précédente.

Voici le résultat d'un premier test de connexion à la console :

```
Debian GNU/Linux 3.1 saphir tty2

saphir login: etu-nis
Password:
Last login: Sun May  8 18:09:40 2005 on :0
Linux saphir 2.6.12-rc3 #3 Sun May  1 17:29:27 CEST 2005 i686 GNU/Linux

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
etu-nis@saphir:~$ pwd
/home/nis/etu-nis
etu-nis@saphir:~$
```

Et pour finir, quelque chose de beaucoup plus joli, une capture d'écran avec le gestionnaire graphique KDE.

7. Documents de référence

Synthèse NFS v3 & NIS

*Administration système en réseau : synthèse NFS v3 & NIS*⁵ : Version «historique» de ce document utilisant la version précédente du protocole de système de fichiers réseau NFS.

Administration système en réseau : architecture réseau

*Architecture réseau des travaux pratiques*⁶ : présentation de l'architecture réseau de la salle de travaux pratiques. On y trouve les éléments sur la topologie physique avec les emplacements des commutateurs dans l'armoire de brassage et sur la topologie logique avec les correspondances entre adresses de réseaux IP et numéros de VLANs.

Virtualisation système et enseignements pratiques

*Virtualisation système et enseignements pratiques*⁷ : présentation de la solution de virtualisation KVM/QEMU permettant d'héberger plusieurs instances de systèmes d'exploitation invités sur un même système hôte. On y trouve les différentes méthodes de d'interconnexion réseau utilisables pour l'illustration des séries de travaux pratiques.

Linux NFS-HOWTO

*Linux NFS-HOWTO*⁸ : documentation complète sur la configuration d'un serveur et d'un client NFS.

The Linux NIS(YP)/NYS/NIS+ HOWTO

*The Linux NIS(YP)/NYS/NIS+ HOWTO*⁹ : documentation complète sur la configuration d'un serveur et d'un client NIS.

Documentation du paquet Debian `nis`

Le fichier `/usr/share/doc/nis/nis.debian.howto.gz` contient une description pas à pas de la marche à suivre pour configurer le service NIS.

⁵ <http://www.linux-france.org/prj/inetdoc/cours/admin.reseau.synthese-nfs3-nis/>

⁶ <http://www.linux-france.org/prj/inetdoc/cours/archi.tp/>

⁷ <http://www.linux-france.org/prj/inetdoc/articles/vm/>

⁸ <http://nfs.sourceforge.net/nfs-howto/>

⁹ <http://www.linux-nis.org/nis-howto/HOWTO/>

Configuration d'une interface réseau

*Configuration d'une interface réseau*¹⁰ : tout sur la configuration des interfaces réseau ; notamment les explications sur les opérations «rituelles» de début de travaux pratiques :

```
# /etc/init.d/networking stop
# ifconfig lo up
# ifconfig eth0 192.168.0.2 netmask 255.255.255.240
# route add default gw 192.168.0.1
# ping 192.168.0.1
# ping 172.16.80.1
# ping www.cict.fr
```

¹⁰ <http://www.linux-france.org/prj/inetdoc/cours/config.interface.lan/>