Introducción al Network Information Service y Network File System

Autor: José Vicente Núñez Zuleta (jose@ing.ula.ve, josevnz@yahoo.com)

Indice de ilustraciones

Objetivos de la práctica

Está trabajo teórico / práctico persigue los siguientes objetivos:

- Comprensión del funcionamiento de NIS y NFS en conjunto
- Configuración de ambas herramientas para prestar un servicio eficiente y seguro

Durante todo este trabajo se presentan ejemplos prácticos para demostrar los conceptos teóricos.

La plataforma utilizada para la elaboración de los ejercicios fue *Linux Slackware 3.4*, por lo que algunos comandos podrían variar de plataforma en plataforma.

Algunos conceptos fueron omitidos o tratados de manera breve por razones de espacio. Nada puede reemplazar la práctica y la investigación, por lo que se remite al lector interesado a la bibliografía al final de este documento. Se asume que el lector esta familiarizado con el sistema operativo Unix, que sabe trabajar con un editor de texto como Vi, y que conoce algunos archivos de configuración básicos como el /etc/passwd.

Sí encuentra fallas o desea hacer algún comentario adicional puede escribir a:

Jose@ing.ula.ve

José Vicente Núñez Zuleta.

Hace mucho tiempo las salas de computadoras eran pequeñas, con pocos usuarios y recursos que manejar. El entorno de computación era sencillo y su era una tarea relativamente sencilla; Todos trabajaban en el mismo sitio y no importaba mucho el desperdiciar espacio con información duplicada.

Pero la situación cambió rápidamente. Con la llegada de las redes de computadoras la cantidad de usuarios aumento de manera exponencial y los sitios de trabajo se hicieron más alejados. También la cantidad de información almacenada hizo que el aprovechar los recursos se convirtiera en un factor vital.

Se hizo necesario la aparición de mecanismos que permitieran sincronizar esos recursos de manera eficiente. Se buscaban las siguientes cosas:

- Los usuarios podrían trabajar en cualquier máquina con su cuenta, teniendo a la mano sus archivos. El proceso debería ser transparente.
- La administración de cuentas y recursos sería centralizada, lo cual evitaría inconsistencias.
- El uso de estas herramientas debería ser consistente entre las diferentes máquinas para facilitar la administración.

El sistema de archivos de red (Network File System, NFS) y el sistema de información de redes (Network Information Service, NIS) provee mecanismos para resolver estos problemas. Los protocolos NFS y NIS **[6]** fueron desarrollados por Sun Microsystems, y son prácticamente un estándar a la hora de resolver este tipo de problemas.

NIS se encarga de resolver los siguientes problemas:

- Centraliza archivos de configuración replicados como el /etc/passwd en una sola máquina.
- Elimina las copias duplicadas de usuarios e información del sistema, permitiéndole al administrador hacer cambios en un solo sitio.

NFS resuelve los siguientes problemas:

- Hace aparecer los sistemas de archivos remotos como si fueran locales porque oculta su verdadera ubicación física.
- Un usuario puede ver sus archivos, independientemente de donde estén localizados, ya sea que estén en el disco local, en un disco compartido en un servidor o en una máquina que está al otro lado de una red de área ancha.

NIS y NFS se complementan, ya que el exportar un sistema de archivos a una máquina en donde estos no existen violaría las reglas de integridad y seguridad impuestas por Unix. También permiten la creación de servicios sofisticados como buzones de correo únicos en toda la red para un grupo de usuarios.

Finalmente NIS y NFS facilitan la labor de administración de redes, con lo cual se puede prestar un mejor servicio.

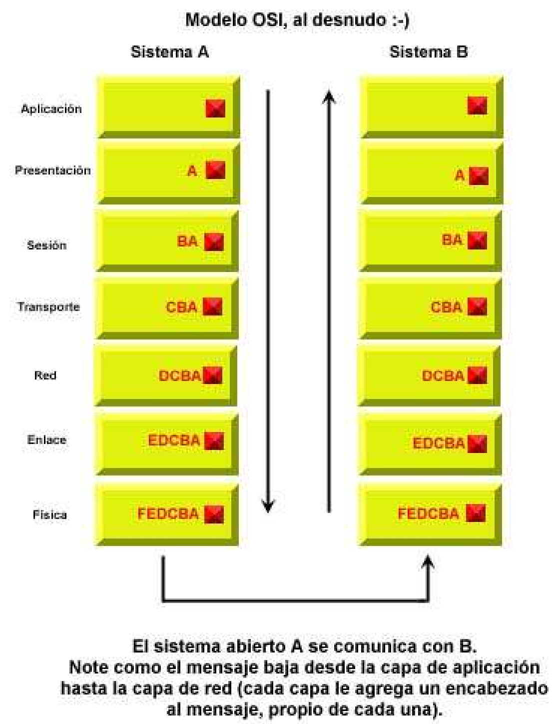
Protocolos de comunicación

NIS y NFS utilizan protocolos de redes para comunicarse con otras máquinas en la red. Se hará a continuación una pequeña introducción al modelo OSI y TCP/IP **[1][2]** y se verá en donde encajan NIS y NFS dentro de ese modelo.

Modelo OSI y TCP/IP

Para poder conectar dos aplicaciones que están separadas físicamente entre sí, se utiliza un lenguaje y reglas de comunicación común a ambas, llamado *protocolo de comunicación*. Existe una abstracción llamada el modelo ***OSI***, la cual permite a aplicaciones remotas comunicarse de manera confiable, a la vez de que su funcionamiento se mantiene independientemente del medio físico por el cual entran en contacto.

El modelo OSI[1] está compuesto de siete capas, cada una de las cuales es recorrida de arriba abajo, las cuales permiten la comunicación entre una aplicación y otra:



[1] OSI es un modelo de referencia, no es una aplicación en particular. Por ejemplo, TCP/IP se basa en este modelo de referencia.

• Ilustración 1: Modelo OSI

A continuación se muestra el significado de cada capa en el modelo OSI y como NIS y NFS encajan en ese modelo:

1. Capa de aplicación. Esta capa trata con los detalles específicos de la aplicación. *NIS* y *NFS* son protocolos que utilizan otras capas inferiores para lograr su cometido[2].
2. Capa de presentación. La capa de presentación trata con la representación de los datos que los componentes de la capa de aplicación usan o refieren en sus comunicaciones. Sun desarrolló un protocolo llamado *XDR (External Data Representation, representación externa de datos)* el cual se encarga de colocar los datos en forma *canónica*[3] para que estos puedan ser llevados de plataforma a plataforma sin ningún problema.
3. Capa de sesión. La capa de sesión provee mecanismos para organizar y sincronizar intercambios de datos con las capas superiores. Esto se debe a las diferencias en como una arquitectura almacena la información respecto a la otra. En esta capa se alojan las llamadas a procedimientos remotos *(RPC, Remote Procedural Call).* En vez de ejecutar un proceso en la máquina local, el proceso es ejecutado en la máquina remota (típicamente porque el recurso que necesita el proceso no se encuentra en la máquina local). *RPC* por lo general utiliza *UDP* para comunicarse ya que es más rápido que *TCP*.
4. Capa de transporte. Le asegura a las capas superiores que los datos serán transmitidos de manera confiable y con el menor costo posible. *TCP* (Uno de los protocolos sobre los cuales trabaja Internet) se encuentra aquí. *UDP* también acompaña a TCP en esta capa. La diferencia básica entre *TCP* y *UDP* es que TCP verifica errores en la transmisión mientras que UDP no. Una máquina puede tener varias conexiones TCP o UDP a la vez, por lo que se utiliza *un puerto* para identificar cada conexión.
5. Capa de red. Se encarga de considerar el enrutamiento de la información. *IP* (Uno de los protocolos sobre los cuales trabaja Internet) se encuentra aquí. El protocolo IP trabaja con el concepto de direcciones IP, las cuales identifican de manera única a una máquina en la red. Una dirección IP está formada por un grupo de *4 octetos*, separados por punto (Por ejemplo *150.185.146.1* es una dirección IP).
6. Capa de enlace. Provee una transferencia punto a punto. También detecta errores provenientes de la capa física. El protocolo Ethernet se encuentra aquí. Ethernet tiene un grupo de direcciones de *48 bits* llamadas *MAC (Media Access Control*). Por ejemplo, *8:0:20:ae:6:1f* es una dirección Ethernet.
7. Capa física. Se encarga de los medios electrónicos y mecánicos que comienzan, mantienen y detienen las conexiones físicas entre dos entidades de enlace. Por ejemplo, la fibra óptica se encuentra en esta capa. Es en esta capa donde se encuentran términos como *MTU* (Maximun Transfer Unit, Unidad Máxima de Transmisión) la cual especifica el tamaño máximo de un paquete que puede ser enviado por la red.

El protocolo TCP/IP no encaja exactamente dentro del modelo de referencia OSI, pero lo sigue de cerca.

Modelo cliente – servidor

NIS y NFS trabajan utilizando el modelo cliente – servidor. Básicamente, un cliente es una entidad que solicita un servicio y un servidor es la entidad que provee el recurso solicitado por el cliente.

En este esquema, RPC juega un papel importante ya que NIS y NFS se basan en sus servicios.

[2] Note como un problema en las capas inferiores puede afectar el funcionamiento de NIS y NFS.

[3] La forma canónica es la forma de ordenamiento de bits que tienen las arquitecturas Motorola y Sparc. El ordenamiento de bits cambia de arquitectura en arquitectura de computadoras.

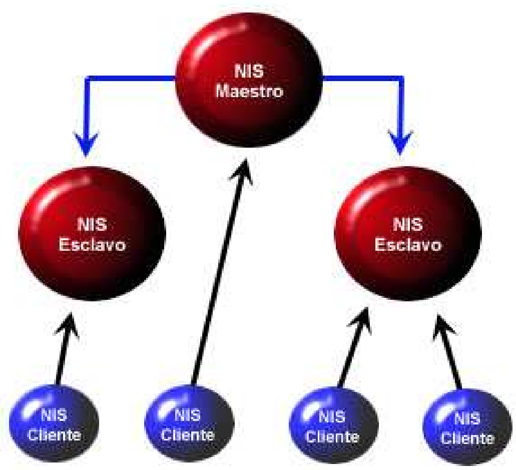
En vez de ejecutar el procedimiento en una máquina local, RPC pasa una serie de argumentos al procedimiento en un datagrama de red. El cliente RPC crea la sesión al localizar al servidor apropiado y se envía el datagrama a un proceso en el servidor que pueda ejecutar la llamada del procedimiento remoto. En el servidor el argumento es desempaquetado, el servidor ejecuta el comando y retorna las respuestas, si hay alguna, al cliente. De vuelta en el cliente, el valor del RPC es convertido a un valor esperado por la función que lo llamo y la aplicación continua como si un procedimiento local hubiera sido llamado.

La ubicación de puertos para una sesión es manejada por un demonio llamado *portmap*[4].

NIS

NIS es una base de datos distribuida que reemplaza copias de archivos de configuración comúnmente replicados por un archivo central. En vez de manejar varias copias de archivos como */etc/hosts*, */etc/passwd*, */etc/group* se utiliza una sola copia que es modificada y almacenada en el servidor y es distribuida entre los clientes. No todos los archivos son candidatos a ser almacenados en un servidor central, como es el caso del */etc/fstab*[5] ya que su contenido varia mucho de equipo en equipo.

Se utilizará como ejemplo la siguiente topología:



• Ilustración 2: Clientes y Servidores utilizando NIS

NIS trabaja bajo el modelo de cliente – servidor. Bajo NIS un *servidor* es una máquina que contiene archivos de datos para NIS, llamados *mapas*. Los *clientes* son máquinas que piden información de esos mapas. Los servidores puede subdividirse aún más en *maestros* y *esclavos*.

Los servidores maestros son los verdaderos poseedores de los mapas y se encargan de su *mantenimiento y distribución*. Los esclavos toman los mapas del servidor y responden preguntas de los clientes (los maestros también lo hacen).

Una vez que se configura a un cliente para que utilice NIS y el servidor está corriendo, algunos archivos dejan de ser utilizados en su totalidad (como él */etc/hosts*) o son

[4] Si desea saber más acerca del demonio *portmap* escriba *man portmap*.

[5] Este archivo se encarga de indicar cuales son los sistemas de archivos que son cargados cuando se arranca la máquina.

complementados por NIS (como él */etc/passwd*); Otros en cambio sólo son útiles si NIS está funcionando (como el /*etc/netroups* del cual se hablará más adelante).

Los mapas NIS no son guardados como archivos ASCII. Los mapas son convertidos a un formato binario llamado DBM, por razones de eficiencia en las búsquedas. NIS puede ser utilizado para algo más que administrar el archivo de password, por ejemplo se puede crear una base de datos de los teléfonos de los empleados de una compañía.

Configuración de un Servidor NIS

Como regla general, es bueno tener un servidor secundario por cada servidor primario en el sistema (*redundancia*). Los siguientes pasos son específicos de la versión de NIS que viene con Slackware 3.4, pero son muy similares entre diversas plataformas.

Configuración de rpc.portmap (No se hará en la práctica)

Para que NIS y NFS funcionen, necesitamos que el demonio [6]*rpc.portmap* funcione. En Slackware está habilitado por omisión en el archivo /etc/rc.d/rc.inet2:

```
if [ -f ${NET}/rpc.portmap ]; then
   echo -n " portmap"
   ${NET}/rpc.portmap
fi
```

Dependiendo de su sistema operativo, deberá tener unas líneas similares a estas.
Verifique que el demonio se encuentre arriba, y si no cárguelo en memoria:

```
leon:/etc/rc.d# ps aux|grep portmap
bin          57  0.0  0.3   848    64  ?  S  May  4   0:01 /usr/sbin/rpc.portmap
root      13270  0.0  1.6   936   316  p1 S   17:19   0:00 grep portmap
```

Existe una manera de evitar que usuarios curiosos o mal intencionados abusen del los servicios del *portmap* en un servidor. Sólo hay que agregar una línea similar a estas en los archivos */etc/hosts.deny* y */etc/hosts.allow*: En el hosts.deny

```
# Negamos el uso de Portmaper a todo el mundo en el hosts.deny
portmap: ALL
```

Y en el hosts.allow:

```
# Todos las maquinas de la red 192.168.1.0 tienen acceso a todos los servicios
#Localhost también tiene permiso
ALL: 127.0.0.1
portmap: 192.168.1.0/255.255.255.0
```

Todas las recomendaciones que se hagan de aquí en adelante respecto a portmap afectan a NIS y a NFS por igual.

Instalación del software para NIS (No se hará en la práctica)

El software para NIS es altamente dependiente del sistema operativo; por ejemplo, Linux Slackware y Solaris traen los demonios y librerías ya precompiladas.

Puede obtener una copia de una implementación de NIS en:

ftp://ftp.kernel.org/pub/linux/utils/net/NIS

Allí encontrará varios paquetes y herramientas para otros programas que trabajan con NIS.

[6] Escriba man 8 portmap para más información.

Netgroup

El archivo Netgroup contiene líneas en el siguiente formato:

```
Grupo (maquina, usuario, nombre de dominio)
```

El uso de este archivo puede ser confuso ya que puede utilizarse para ampliar otros mapas con su información. Por ejemplo, cuando Netgroup es usado para agregar usuarios a un archivo de password, sólo se usa el campo de usuario; en cambio, cuando se usa este archivo para crear una lista de máquinas para controlar el acceso a un sistema de archivos montado por NFS sólo se usa la información de la máquina.

En este ejemplo utilizaremos el archivo /etc/netgroups para administrar un grupo de máquinas:

```
maquinas (leon,,), (pantera,,), (tigre,,), (puma,,)

usuarios(,jose,), (,luised,), (,www,)
```

Usted también puede crear grupos a partir de otros grupos

```
mired maquinas, usuarios
```

Recuerde que el archivo Netgroup sólo funciona con NIS.

Passwd y group

El formato del archivo de password (si no utiliza shadow) es bastante estándar bajo Unix. Dependiendo de la versión de libc que tenga su sistema, este podrá utilizar o no el archivo de shadow de manera transparente. Este no es el caso de Slackware, en el cual el archivo de password no puede ser particionado.

Cuando se instala NIS, se puede escoger entre utilizar el archivo de password del servidor (el cual es exportado a su totalidad, excepto si el *user id* del usuario es menor a cierto número, usualmente 20[7]) o un archivo completamente separado. Es cuestión de preferencias, sin embargo en esta trabajo usaremos como mapa nis el archivo /var/yp/passwd.

La sintaxis del /etc/group se mantiene intacta.

El siguiente es un archivo de ejemplo que muestra el formato del archivo */var/yp/passwd*:

```
josev:1skbWKsmwU:1001:100:Jose Vicente Nunez Gabaldon,,,:/home/josev:/bin/tcsh
luised:g. WKXXrtgsm.2:1002:100:Luis Eduardo Nunez Zuleta,,,:/home/luised:/bin/tcsh
yv2bsv:x:1006:100:Cuenta de radio,,,:/home/yv2bsv:/bin/tcsh
jose:.tArffWKsmsE:1000:100:Jose Vicente Nunez Zuleta,,,:/home/jose:/bin/tcsh
seguridad:WKsm1NHe.my2:1005:100:Seguridad,,,:/home/seguridad:/bin/tcsh
majordomo:EiYcWKsm.my2:1007:2: Majordomo,,,:/usr/local/majordomo:/bin/tcsh
www:*:407:101:Usuario del WWW,,,:/dev/null:/dev/null
```

El archivo /var/yp/*group* quedaría de la siguiente manera:

```
www::101:www
```

[7] Este punto es muy importante. Un error común de los administradores es exportar el archivo de password, con la cuenta de root y otras cuentas importantes a toda la red. *Root tiene el user id igual a 0*.

Configuración de un servidor maestro NIS

NIS utiliza el concepto de dominio al igual que DNS; Sin embargo, ambos dominios son completamente distintos y se recomienda en la práctica que su nombre no sea el mismo que el dominio DNS (para evitar que los mapas NIS sean robados utilizando programas como ypx**[5]**). Un dominio NIS se utiliza para agrupar a un conjunto de máquinas bajo una administración común.

La forma de definir un dominio NIS es con el comando *domainname*:

```
tigre:/etc# /bin/domainname dracula
```

Donde "dracula" es el dominio NIS empleado. Normalmente se colocará una línea similar a esta en el /etc/rc.d/rc.inet2 o similar para que el cambio sea automático cada vez que la máquina arranque:

```
if [ -r /etc/defaultdomain ]; then
  nisdomainname `cat /etc/defaultdomain`
fi
```

Modifique el archivo /var/yp/securenets para asegurarse de que sólo las máquinas autorizadas puedan obtener los mapas. Guíese por el siguiente ejemplo:

```
#Localhost puede todo
host 127.0.0.1
# Todas las máquinas de la red 192.168.1.0 pueden acceder a los mapas
255.255.255.0 192.168.1.0
```

Tanto en el cliente como en el servidor, deberá configurar el archivo */etc/yp.conf* para que apunte a un servidor NIS. Este archivo se puede dejar vacío y el cliente enviará un *broadcast* por la red en busca de un servidor que responda; Sin embargo puede ser un hueco de seguridad ya que un atacante puede configurar a una máquina que responda más rápido (por tener menos carga de trabajo por ejemplo) para que sea servidor NIS y el cliente se conectará a ella cargando, por ejemplo, el archivo de password que el atacante quiere que use.

Un ejemplo de configuración es el siguiente (se puede utilizar una dirección IP si lo desea y puede colocar tantos servidores como desee):

```
# Syntax:
# ypserver <Name_of_ypserver>
ypserver leon.felinos.ve
ypserver leopardo.felinos.ve
```

Si leon.felinos.ve no responde, entonces el cliente intentará con leopardo.felinos.ve

El siguiente paso es decirle al servidor que convierta los archivos ASCII en mapas NIS. Para ello edite el archivo /var/yp/Makefile y verifique lo siguiente:

- Comente la línea que dice *NOPUSH = "True*" si piensa utilizar servidores esclavos en su dominio NIS.
- MINUID controla el valor mínimo de user id que puede aparecer en un mapa NIS. Si el valor MINUID=20 está bien déjelo allí, sino cámbielo de acuerdo a sus necesidades.
- Si decidió utilizar un archivo de password y group distinto y lo guardo en el directorio /var/yp, entonces modifique la línea que dice YPPWDDIR = /etc y cámbiela por YPPWDDIR = /var/yp
- Revise el resto del archivo Makefile por si desea agregar otro mapa.

Escriba lo siguiente:

leon:/var/yp# **/usr/libexec/yp/ypinit –m**

E indique el nombre de los servidores esclavos del dominio. Si no hay ninguno, o no quiere introducir más nombres escriba ^D:

```
At this point, we have to construct a list of the hosts which will run NIS
servers.  leon 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:  leon
        next host to add:  tigre
        next host to add:
The current list of NIS servers looks like this:

leon
tigre

Is this correct?  [y/n: y]
```

El programa mostrará algo como esto:

```
NIS Map update started on Sat Jun 13 20:31:40 GMT-4 1998
make[1]: Entering directory '/var/yp/castor'
Updating passwd.byname...
Updating passwd.byuid...
Updating hosts.byname...
Updating hosts.byaddr...
Updating group.byname...
Updating group.bygid...
Updating netid.byname...
Updating networks.byaddr...
Updating networks.byname...
Updating protocols.bynumber...
Updating protocols.byname...
Updating rpc.byname...
Updating rpc.bynumber...
make[1]: Leaving directory '/var/yp/castor'
NIS Map update completed.
```

Después agregue la siguiente línea en el archivo /etc/rc.d/rc.inet2 o similar; Luego ejecute al demonio ypserv, ypserv es el demonio que se encarga de distribuir los mapas NIS.
:

```
# Start servers
if [ -d /var/yp ] ; then
  echo "Running ypserv..."
  /usr/sbin/ypserv
fi
```

Verifique que ypserv se puede comunicar con *portmaper* utilizando a *rpcinfo*:

```
leon:/var/yp# rpcinfo –u localhost ypserv
program 100004 version 1 ready and waiting
```

Debe tener en cuenta que es recomendable que el servidor maestro debe ser cliente de si mismo, así que siga leyendo la sección de configuración de cliente para completar la configuración del servidor.

Configuración de un cliente NIS

Después de editar el archivo */etc/yp.conf*, defina el dominio NIS con *domainname* y luego ejecute el demonio *ypbind*, el cual hará el enlace con el servidor NIS (edite el archivo /etc/rc.d/rc.inet2):

```
if [ -d /var/yp ] ; then
  echo "Running ypbind..."
  /usr/sbin/ypbind
fi
```

Note que estas líneas deben ir después de cargar a *ypserv* si el servidor es maestro o antes si el servidor es esclavo.

Ejecute el comando *rpcinfo –p localhost o rpcinfo –u localhost ypbind*[8], para ver si ypbind pudo comunicarse con rpc:

```
tigre:/var/yp# rpcinfo -p localhost
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100005    1   udp    684  mountd
    100007    2   udp    694  ypbind
    100007    2   tcp    696  ypbind
```

Como prueba, escriba *ypcat passwd* para ver el contenido del archivo de password.

Adicionalmente deberá configurar el archivo de password para poder utilizar NIS. Se mostrará a continuación un ejemplo y el significado de cada línea:

```
1.  nobody:x:65534:100:nobody:/dev/null:
2.  +jose:::::/dev/null
3.  -luised::::
4.  +@usuarios
5.  +:::::
```

La línea 2 agrega la información del usuario jose al archivo de password, pero este no podrá entrar a la máquina porque el administrador impone que su Shell sea */dev/null*.

La línea 3 no permite la introducción del usuario *luised*, mientras que la línea 4 habilita a todos los usuarios que están en el grupo "*usuarios*".

Finalmente la línea 5 habilita al resto de los usuarios que están en el archivo de password. Note como el orden es importante, si se hubiera colocado la línea 3 después de la 5 está no tendría sentido ya que el usuario luised sería agregado sin ningún problema al archivo de password (la primera regla leída es la que tiene efecto).

Para habilitar el archivo de grupo, sólo tiene que escribir:

```
+:::::
```

al final del /etc/group

Configuración de un servidor esclavo NIS

Primero que nada el servidor esclavo debe estar configurado como cliente del servidor maestro para poder realizar la transferencia de mapas. Luego se debe ejecutar a *ypinit* con la opción *–s:*

tigre:/var/yp# **/usr/libexec/yp/ypinit –s leon**

Cuando se reconstruyen los mapas en el servidor maestro, este sólo actualiza los archivos DBM que estén desfasados con respecto a los nuevos archivos ASCII. El servidor entonces distribuye los mapas con ayuda de la herramienta *yppush* (presente en el archivo Makefile); Sin embargo esa transferencia no siempre funciona, por ejemplo el servidor esclavo puede estar caído en el momento de la transferencia, así que el esclavo utiliza al programa *ypxfr* para traer los mapas actualizados al servidor.

Ypxfr debe volver a construir una copia local de los mapas ya que el formato de la base de datos puede ser distinto en el servidor.

Se recomienda colocar las siguientes líneas en el crontab del servidor esclavo para que este verifique de manera periódica por cambios en los mapas[9]:

[8] Rcpinfo –p indica los programas que están registrados con el demonio rpc. La opción –u indica que se comunique con el programa dado utilizando UDP. Escriba man rpcinfo si desea obtener más detalles.
[9] Utilice para ello crontab –e. Escriba man crontab para más detalles.

```
20 *    * * *    /usr/libexec/yp/ypxfr_1perhour
40 6    * * *    /usr/libexec/yp/ypxfr_1perday
55 6,18 * * *    /usr/libexec/yp/ypxfr_2perday
```

Configuración de yppasswdd en el servidor maestro

La excepción al comportamiento de actualización anterior es el archivo de password, el cual debe poder ser actualizado sin problemas en "tiempo real". Para ello el usuario utiliza la herramienta *yppasswd*[10] para cambiar atributos en el archivo de password y el servidor maestro utiliza al demonio yppasswd para efectuar los cambios. En nuestro caso el archivo de password se encuentra en /var/yp y no en /etc por lo que hay que indicárselo al demonio *rpc.yppasswdd*:

```
# Start servers
if [ -d /var/yp ] ; then
  echo "Running ypserv..."
  /usr/sbin/ypserv
  echo "Running yppasswdd..."
  /usr/sbin/rpc.yppasswdd -D /var/yp
fi
```

El usuario sólo tiene que escribir lo siguiente para cambiar su password:

```
tigre[59] Si maestro? yppasswd
Changing password for jose on leon.
Changing NIS password for jose.
Old password:
New password:
Retype new password:
The NIS password has been changed on leon.
tigre[60] Si maestro?
```

Recuerde escoger un password seguro, no utilice información obvia como su fecha de cumpleaños y trate de que tenga incluidos caracteres alfanuméricos.

Herramientas comunes bajo NIS

Ypmatch

Ypmatch permite encontrar una clave dentro de un mapa NIS. Por ejemplo, busquemos al usuario www dentro del mapa passwd:

```
tigre:/etc# ypmatch www passwd
www:*:407:101:Usuario del WWW,,,:/dev/null:/dev/null
```

Busquemos al la clave usuarios dentro del mapa netgroup:

```
tigre:/etc# ypmatch usuarios netgroup
(,jose,), (,josev,), (,luised,), (,calzul,)
```

Ypcat

Permite mostrar el contenido de un mapa NIS. Si se utiliza con la opción –k, se muestra también la clave asociada, además de su valor:

```
tigre:~# ypcat netgroup
(,jose,), (,josev,), (,luised,), (,calzul,)
(leon,,), (pantera,,), (tigre,,), (puma,,)
tigre:~# ypcat -k netgroup
usuarios (,jose,), (,josev,), (,luised,), (,calzul,)
castor (leon,,), (pantera,,), (tigre,,), (puma,,)
tigre:~#
```

Para ver la lista de servidores NIS escriba lo siguiente:

```
tigre:~# ypcat ypservers
tigre
```

[10] Muchos administradores hacen un alias a passwd para que invoque a yppaswd.

```
leon
tigre:~#
```

Diagnostico básico de problemas en NIS

Las herramientas nombradas con anterioridad sirven para realizar el diagnóstico de problemas bajo NIS. Si estas retornan un mensaje de error diciendo que no pueden conectarse con el servidor puede deberse a las siguientes causas:

- Fallo de conectividad en la red.
- El demonio rpc.portmap no está corriendo en el servidor. Si portmap se queja que el programa no está registrado, entonces falta por correr a ypserv en el servidor.
- Verifique que ypbind se puede comunicar con el servidor utilizando a *ypwhich*:

```
tigre:~# ypwhich
leon.felinos.ve
tigre:~# ypwhich -m
mail.aliases leon
netgroup.byuser leon
netgroup.byhost leon
netgroup leon
services.byname leon
rpc.bynumber leon
rpc.byname leon
protocols.bynumber leon
networks.byname leon
networks.byaddr leon
netid.byname leon
group.bygid leon
group.byname leon
hosts.byaddr leon
hosts.byname leon
passwd.byname leon
protocols.byname leon
ypservers leon
passwd.byuid leon
```

Ypwhich informa a que servidor esta conectado el cliente NIS, la opción –m dice cuales son los mapas que tiene el servidor dado.

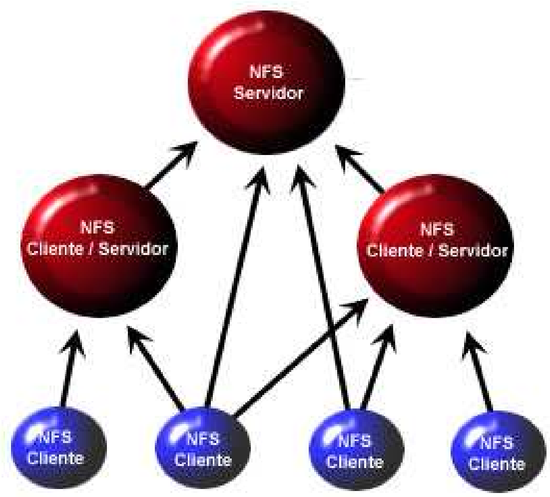
Ypwhich con la opción *–x* muestra que alias existen para los mapas del servidor:

```
leon:/usr/src/ypserv-1.3.2# ypwhich -x
Use "ethers"    for map "ethers.byname"
Use "aliases"   for map "mail.aliases"
Use "services"  for map "services.byname"
Use "protocols" for map "protocols.bynumber"
Use "hosts"     for map "hosts.byaddr"
Use "networks"  for map "networks.byaddr"
Use "group"     for map "group.byname"
Use "passwd"    for map "passwd.byname"
```

NFS es un sistema de archivos distribuido que provee acceso de manera transparente a discos remotos. Así como NIS permite centralizar la administración de un conjunto de archivos, NFS permite la administración centralizada de discos. Por ejemplo, en vez de tener varias copias del directorio */usr/local* se tiene una sola la cual es compartida a través de la red.

NFS también utiliza el protocolo RPC, así que los pasos descritos en la sección de NIS para verificar que RPC este arriba también se aplican aquí. Aparte de eso se necesita que el Kernel tenga soporte para trabajara con NFS, así que quizás tenga que compilarlo[11] con esa opción activada.

Bajo NFS no existe el concepto de cliente o servidor puro. Como puede ver en la ilustración que acompaña este texto, Un servidor puede *exportar* un sistema de archivos y puede *montar* un sistema de archivos distinto a la vez:



• Ilustración 3: Clientes y servidores NFS

Hay dos aspectos básicos en la administración de un sistema de archivos usando NFS: escoger una nomenclatura en la escogencia de los nombres y configurar a los clientes para que se adhieran a ese esquema. Por ejemplo, podría decidirse que los usuarios de una compañía sean agrupados según su departamento, esto implica que todos los clientes deberán montar los directorios remotos siguiendo estas normas; recuerde que NFS fue diseñado para enmascarar las complejidades de la red, no para hacerlas evidentes.

Configuración de los demonios rpc.mountd y rpc.nfsd

El demonio que controla la exportación de archivos bajo Linux es el *rpc.nfsd* y debe ser invocado en alguno de los script de inicio:

```
if [ -f ${NET}/rpc.nfsd ]; then
    echo -n " nfsd"
    ${NET}/rpc.nfsd
fi
```

[11] Se recomienda que consulte el Kernel Howto disponible en http://sunsite.unc.edu/LDP/HOWTO/Kernel–HOWTO.html.

Debido a que el servidor puede convertirse eventualmente en un cliente es bueno asegurarse que el demonio *rpc.mountd* (encargado del montaje de sistemas de archivos) también es invocado (En Slackware está antes del demonio rpc.nfsd):

```
if [ -f ${NET}/rpc.mountd ]; then
    echo -n " mountd"
    ${NET}/rpc.mountd
fi
```

Corra ambos programas y agréguelos a los script de configuración adecuados. Después interrogue al demonio rpc.portmap acerca de su estado:

```
leon:/etc/rc.d# rpcinfo -p
   program vers proto   port
    100000    2   tcp    111  portmapper
    100000    2   udp    111  portmapper
    100005    1   udp    666  mountd
    100005    2   udp    666  mountd
    100005    1   tcp    669  mountd
    100005    2   tcp    669  mountd
    100003    2   udp   2049  nfs
    100003    2   tcp   2049  nfs
    100004    2   udp    677  ypserv
    100004    1   udp    677  ypserv
    100004    2   tcp    680  ypserv
    100007    2   udp    679  ypbind
    100007    2   tcp    681  ypbind
    100002    2   udp   1185  rusersd
    100002    3   udp   1185  rusersd
    100008    1   udp   1190  walld
    100009    1   udp    941  yppasswdd
```

Configuración del servidor

El control sobre los sistemas de archivos que son exportados se realiza por medio del archivo */etc/exports.*[12]. La sintaxis del archivo viene dada de la siguiente manera:

```
Sistema de archivo [máquina | @netgroup | metacaracter| dirección ip/ mascara de red]
opciones
```

Donde:

- Sistema de archivos es el sistema a exportar por la red
- Máquina, netgroup, metacaracter o dirección ip es a quien le es exportado el sistema de archivos.
- Opciones se refiere a los parámetros y condiciones con los que es exportado el sistema de archivos.

Se mostrará a continuación un ejemplo del archivo /etc/exports:

```
1.  /usr/src pantera.felinos.ve(no_root_squash,rw)
2.  /usr/local *.felinos.ve(ro)
3.  /usr/X11 *.felinos.ve(root_squash,ro)
4.  /home *.felinos.ve(rw)
```

La primera línea exporta el directorio /usr/src a pantera.felinos.ve con opciones de escritura y lectura. NFS, por omisión, cuando exporta un sistema de archivos evita que el superusuario del cliente tenga privilegios de lectura y escritura sobre todos los archivos exportados; la opción no_root_squash desactiva ese comportamiento y permite que el superusuario remoto tenga los mismos privilegios que el superusuario normal. Tenga mucho cuidado con esta opción ya que puede convertirse fácilmente en un agujero de seguridad.

En la línea 2 se exporta el directorio /usr/local a toda la red felinos.ve con permiso de sólo lectura.

En la línea 3 se exporta el directorio /usr/X11 a la red felinos con permiso de lectura y escritura, pero desactivando sólo los privilegios de root de manera remota.

[12] Si desea conocer todas las opciones disponibles para exportar un sistema de archivos, escriba man exports

Finalmente en la línea 4 se exporta el directorio /home a toda la red felinos con permiso de lectura y escritura.

Nunca haga algo como esto en el archivo fstab:

```
#Exporta el sistema de archivos a todo el mundo
/
```

Bajo SunOS y Solaris existe un programa llamado *exportfs*[13] el cual vuelve a exportar un sistema de archivos cuando es invocado, útil si se modifica el archivo */etc/exports*. El siguiente script se encarga de volver a exportar los sistemas de archivos una vez terminada la modificación del /etc/exports:

```
#!/bin/sh
killall -HUP /usr/sbin/rpc.mountd
killall -HUP /usr/sbin/rpc.nfsd
echo re-exported file systems
```

Llámelo */sbin/exportfs* y colóquele permiso de ejecución sólo para root.

El programa *showmount –e* le dirá como esta exportando su sistema de archivos:

```
leon:/var/yp# showmount -e leon
Export list for leon:
/home      *.felinos.ve
/usr/X11R6 *.felinos.ve
/usr/local *.felinos.ve
/usr/src   pantera.felinos.ve
```

Y si lo ejecuta sin parámetros le dirá quien está montando sistemas de archivos:

```
leon:~# showmount
Hosts on leon:
pantera.felinos.ve
puma.felinos.ve
tigre.felinos.ve
```

Se recomienda que el lector se familiarice con las opciones del archivo */etc/export* para que la distribución de los archivos por NFS sea más segura.

Configuración del cliente

Bajo Unix el comando *mount*[14] permite montar sistemas de archivos, ya sean remotos o locales:

```
mount -t tipo opciones dispositivo directorio
```

En el caso de NFS, su uso más común sería el siguiente:

```
mount máquina:/directorio/remoto /directorio/local
```

Con lo cual se trabajaría con las opciones por omisión de mount (por ejemplo: lectura y escritura, ejecución de programas, Suid activado, escritura no sincrónica, etc.)

El siguiente ejemplo monta el directorio /usr/local de leon en el directorio /usr/local de tigre:

```
tigre:~# mount leon:/usr/local  /usr/local
```

Note que el directorio /usr/local en tigre debe existir, sino el montaje de los archivos falla.

Si invoca a *mount* sin parámetros podrá ver los sistemas de archivos (remotos o no) actualmente montados:

```
tigre:~# mount
/dev/hda2 on / type ext2 (rw)
/dev/hda1 on /win95 type vfat (rw)
none on /proc type proc (rw)
```

[13] Slackware 3.4 y anteriores no tiene esta facilidad
[14] Escriba man mount para conocer más sobre este comando

```
leon:/usr/local on /usr/local type nfs (rw,addr=192.168.1.2)
tigre:~#
```

El proceso es reversible, es decir, podemos *desmontar* [15] un sistema de archivos previamente montado si el directorio no está siendo utilizado por algún usuario:

```
tigre:~# umount /usr/local
tigre:~#
```

Como puede apreciar, esta manera de montar directorios es incomoda si lo que se desea es que la máquina lo haga cada vez que arranque; Para ello entonces es mejor editar el archivo */etc/fsta*[16]*b*:

```
/dev/hda3       swap        swap        defaults   1   1
/dev/hda2       /       ext2        defaults   1   1
/dev/hda1        /win95        vfat        defaults   1   1
none             /proc    proc        defaults   1   1
leon:/home /home  nfs defaults 0 0
```

En el archivo fstab se especifican en este orden:

- Donde se obtiene el sistema de archivo (*leon:/home*)
- Donde se va a montar   (*/home*)
- Tipo de sistema de archivo (*nfs*)
- Parámetros de montaje, separados por comas (*defaults*)
- Esta columna indica si el sistema debe ser respaldado por la herramienta dump (0 = No respaldar)
- Un número mayor que cero indica el orden en que el programa fsck verificará el estado del sistema de archivos (0 = No verificar).

Una vez introducido en el fstab, el sistema de archivos se puede volver a montar escribiendo sólo el nombre en donde será montado (si no está montado, claro está):

```
tigre:~# umount /home
tigre:~# mount /home
tigre:~#
```

Finalmente, puede volver a montar todos los sistemas de archivos especificados en el archivo */etc/fstab* escribiendo *mount –a* (*umount –a* los desmonta todos sino están ocupados).

A continuación se mostrarán algunas opciones útiles en el montaje de sistemas de archivos.

Parámetros útiles de la herramienta mount

Seguridad

Además de las opciones de exportación dadas por el servidor, el cliente puede reforzarlas con el uso de algunas opciones:

```
1.  leon:/home /home  nfs rw,nosuid,nodev,hard,intr 0 0
2.  leon:/usr/local/src /usr/local/src  nfs ro,nosuid,noexec 0 0
3.  leon:/usr/local/bin /usr/local/bin  nfs ro,exec 0 0
```

La línea 1 permite montar el directorio /home desde leon en /home vía nfs, con permiso de escritura y lectura, pero *sin darle permiso al usuario que el bit Suid tenga efecto y que pueda interpretar dispositivos de caracteres o de bloques*. Un usuario normal no debería poder hacer ninguna de las dos cosas.

[15] Normalmente *sólo root* puede montar o desmontar sistemas de archivos, al menos que se especifique lo contrario en el archivo /etc/fstab o se utilice un programa como SUDO para darle más privilegios a los usuarios.
[16] /etc/fstab define cuales archivos son montados al inicio del sistema.

La opción *hard (activada por omisión)* le dice al cliente que "congele" al programa cliente que hace uso del sistema de archivos si el servidor se cuelga; cuando este vuelva a funcionar el programa continuará como si nada. La opción *intr* permite interrumpir el programa con ^C.

La línea 2 permite compartir el directorio /usr/local/src en modo de sólo lectura sin permitir la ejecución de programas en ese directorio.

La línea 3 permite la ejecución de programas montados desde /usr/local/bin con permiso de sólo lectura.

Estos son sólo algunos parámetros, es conveniente que verifique la documentación de su sistema si desea saber más.

Eficiencia

El rendimiento de NFS puede mejorarse con el uso de algunos parámetros, como *rsize* y *wsize*:

```
leon:/home /home  nfs rw,nosuid,nodev,hard,intr,rsize=8192,wsize=8192 0 0
```

La opción rsize especifica el tamaño del buffer de prelectura (*read ahead*) y cual es el tamaño del buffer de escritura retrasada[17] (*write back*). Los valores mostrados en el ejemplo son los sugeridos por las páginas man, pero el usuario deberá obtener los valores que mejor le convengan de manera experimental. Según Kukuk **[3]**, el tamaño óptimo para máquinas Intel es 4096 bytes.

[17] La prelectura hace más rápido el funcionamiento de NFS ya que la próxima vez que se necesite un dato este quizás ya estará en memoria y no en disco porque ya fue leído. La escritura retrasada trata de demorar lo más posible la escritura de datos para evitar el acceso a disco hasta que haya suficientes datos en el buffer como para escribirlo.

Ejercicio 1: Configuración de NIS

Configure un servidor NIS maestro y varios esclavos para que compartan el archivo de *password* y *netgroups*. El resto de las máquinas deberán ser clientes. Verifique el funcionamiento del servidor y de cada cliente.

Ejercicio 2: Configuración de NFS

Configure varios servidores NFS para que cada uno de ellos pueda exportar su sistema de archivos a todo un dominio, con el directorio hogar de cada una de las cuentas del ejercicio 1. Todas las máquinas deben montar el directorio respectivo. Si lo desea, exporte el directorio */usr/local* a todo el dominio sin que el UID de root sea cambiado durante el proceso, utilice para ello la opción *no_root_squash*. Anote lo que ocurre.

Se recomienda que el lector interesado consulte la siguiente documentación para profundizar los conocimientos acerca de los puntos expuestos en este trabajo.

1. Dawson Terry. " NET–3 HOWTO". Linux Documentation Project. http://sunsite.unc.edu/LDP/HOWTO/NET–3–HOWTO.html.

1. Hunt Craig. "TCP – IP Network Administration". O'Reilly & Associates, Inc. 1993. 449 p.

1. Kukuk Thorsten. "NIS HOWTO". Linux Documentation Project. http://sunsite.unc.edu/LDP/HOWTO/NIS–HOWTO.html

1. Langfeldt Nicolai. "NFS HOWTO". Linux Documentation Project. http://sunsite.unc.edu/LDP/HOWTO/NFS–HOWTO.html.

1. Nauta Rob. J. "YPX – A utility to transfer NIS maps beyond a local (broadcast) network." ftp://coast.cs.purdue.edu/pub/tools/unix/ypx.shar.

1. Stern Hal. "Managin NFS and NIS". O'Reilly & Associates, Inc. 1992. 389 p.