USERS

Probably the most time-consuming aspect of administration is managing users ....

You need some sort of database where the system can find out information about a user. And we can think of two types of user accounts, those local to a system and a "network" account. That is an account which is valid on multiple systems.

- Local User Accounts
  - Managed in /etc/passwd
  - Always need the "root" user
  - Other users in the local file (these differ from system to system, but generally you will have adm, bin, daemon, sys, uucp)

```
adm, bin, daemon, hpdb, listen, lp,
noaccess, nobody, nobody4,
nuucp, sys, uucp
```

- "Network" User Accounts
  - When you have more than one computer that users log into, you may want to centralize your administration and have a global database. This way you only have to manage the user information in one place.
  - Managed through some network database
    - NIS - Network Information Service (Yellow Pages)
    - NIS+ - Network Information Service Plus. It was designed to replace NIS
    - LDAP - **L**ightweight **D**irectory **A**ccess **P**rotocol

The anatomy of a user account

No matter where or how your user account is managed, certain information needs to be present for each user.

- Login-Name
- Password Information
- User ID
- Group ID
- Description of the User -- (User's Real Name)
- Home Directory
- Shell

See the descriptions in the password file for more details.

/etc/passwd contains one line for each user account, with seven fields delimited by colons (“:”).

Password File

```
NAME
     passwd - password file

SYNOPSIS
     /etc/passwd

DESCRIPTION
     /etc/passwd is a local source of  information  about  users'
     accounts.  The password file can be used in conjunction with
     other password sources, including the NIS maps
      passwd.byname and  passwd.bygid and the NIS+ table  passwd.
     Programs use the getpwnam(3C) routines to access this infor-
     mation.

     Each  passwd entry is a single line of the form:


          username:password:uid:
          gid:gcos-field:home-dir:
          login-shell


     where

     username
           is the user's login name. It is recommended that  this
           field conform to the checks performed by  pwck(1M).

     password
           is an empty field. The encrypted password for the user
           is in the corresponding entry in the /etc/shadow file.
           pwconv(1M) relies on a special value  of  'x'  in  the
           password  field  of  /etc/passwd. If this value of 'x'
           exists in the  password  field  of  /etc/passwd,  this
           indicates that the password for the user is already in
           /etc/shadow and should not be modified.

     uid   is the user's unique numerical ID for the system.

     gid   is the unique numerical ID of the group that the  user
           belongs to.

     gcos-field
           is the user's real name,  along  with  information  to
           pass  along in a mail-message heading.   (It is called
           the gcos-field  for  historical  reasons.)   An  ``&''
           (ampersand)  in  this  field stands for the login name
           (in cases where the login name  appears  in  a  user's
           real name).
```

home-dir
is the pathname to the directory in which the user is initially positioned upon logging in.

login-shell
is the user's initial shell program. If this field is empty, the default shell is /usr/bin/sh.

The maximum value of the *uid* and *gid* fields is 2147483647. To maximize interoperability and compatibility, administrators are recommended to assign users a range of UIDs and GIDs below 60000 where possible.

The password file is an ASCII file. Because the encrypted passwords are always kept in the shadow file, /etc/passwd has general read permission on all systems and can be used by routines that map between numerical user IDs and user names.

Previous releases used a password entry beginning with a `+' (plus sign) or `-' (minus sign) to selectively incorporate entries from NIS maps for password. If still required, this is supported by specifying ``passwd : compat'' in **nsswitch.conf(4)**. The "compat" source may not be supported in future releases. The preferred sources are, "files" followed by "nisplus". This has the effect of incorporating the entire contents of the NIS+ passwd table after the password file.

EXAMPLES

Example 1: A sample passwd file.

Here is a sample passwd file:

```
root:q.mJzTnu8icF.:0:10:God:/:/bin/csh
fred:6k/7KCFRPNVXg:508:10:& Fredericks:/usr2/fred:/bin/csh
```

and the sample password entry from nsswitch.conf:

```
passwd: files nisplus
```

In this example, there are specific entries for users root and fred to assure that they can login even when the system is running single-user. In addition, anyone in the NIS+ table passwd will be able to login with their usual password, shell and home directory.

If the password file is:

```
root:q.mJzTnu8icF.:0:10:God:/:/bin/csh
fred:6k/7KCFRPNVXg:508:10:& Fredericks:/usr2/fred:/bin/csh
+
```

and the password entry from nsswitch.conf is:

passwd: compat

then all the entries listed in the NIS passwd.byuid and passwd.byname maps will be effectively incorporated after the entries for root and fred.

FILES

/etc/nsswitch.conf

/etc/passwd

/etc/shadow

SEE ALSO

chgrp(1), **chown(1)**, **groups(1)**, **login(1)**, **newgrp(1)**, **nispasswd(1)**, **passwd(1)**, **sh(1)**, **sort(1)**, **chown(1M)**, **domainname(1M)**, **getent(1M)**, **in.ftpd(1M)**, **passmgmt(1M)**, **pwck(1M)**, **pwconv(1M)**, **su(1M)**, **useradd(1M)**, **userdel(1M)**, **usermod(1M)**, **a64l(3C)**, **crypt(3C)**, **getpw(3C)**, **getpwnam(3C)**, **getspnam(3C)**, **putpwent(3C)**, **group(4)**, **hosts.equiv(4)**, **nsswitch.conf(4)**, **shadow(4)**, **environ(5)**, unistd(3HEAD)

System Administration Guide, Volume 1

Shadow File

NAME

shadow - shadow password file

DESCRIPTION

/etc/shadow is an access-restricted ASCII system file that stores users' encrypted passwords and related information. The shadow file can be used in conjunction with other shadow sources, including the NIS maps passwd.byname and passwd.byuid and the NIS+ table passwd. Programs use the **getspnam(3C)** routines to access this information.

The fields for each user entry are separated by colons. Each user is separated from the next by a newline. Unlike the /etc/passwd file, /etc/shadow does not have general read permission.

Each entry in the shadow file has the form:

username:*password*:*lastchg*: *min*:*max*:*warn*: *inactive*:*expire*:*flag*

The fields are defined as follows:

username
The user's login name (UID).

password
A 13-character encrypted password for the user, a

lock string to indicate that the login is not accessible, or no string, which shows that there is no password for the login.

lastchg
The number of days between January 1, 1970, and the date that the password was last modified.

min The minimum number of days required between password changes.

max The maximum number of days the password is valid.

warn The number of days before password expires that the user is warned.

inactive
The number of days of inactivity allowed for that user.

expire
An absolute date specifying when the login may no longer be used.

flag Reserved for future use, set to zero. Currently not used.

The encrypted password consists of 13 characters chosen from a 64-character alphabet (., /, 0-9, A-Z, a-z). To update this file, use the **passwd(1)**, **useradd(1M)**, **usermod(1M)**, or **userdel(1M)** commands.

In order to make system administration manageable, /etc/shadow entries should appear in exactly the same order as /etc/passwd entries; this includes ``+'' and ``-'' entries if the compat source is being used (see **nsswitch.conf(4)**).

FILES

/etc/shadow
shadow password file

/etc/passwd
password file

/etc/nsswitch.conf
name-service switch configuration file

SEE ALSO

login(1), **passwd(1)**, **useradd(1M)**, **userdel(1M)**, **usermod(1M)**, **getspnam(3C)**, **putspent(3C)**, **nsswitch.conf(4)**, **passwd(4)**

NOTES

If password aging is turned on in any name service the *passwd*: line in the /etc/nsswitch.conf file must have a format specified in the **nsswitch.conf(4)** man page.

```
If the /etc/nsswitch.conf passwd policy is not in one of the
supported  formats, logins will not be allowed upon password
expiration because the software does not know how to  handle
password     updates     under     these     conditions.     See
nsswitch.conf(4) for additional information.
```

INTRO TO NIS

WHAT IS IT

NIS stands for Network Information Service. Its purpose is to provide information, that has to be known throughout the network, to all machines on the network. Information likely to be distributed by NIS is:

- login names/passwords/home directories (/etc/passwd)
- group information (/etc/group)

If, for example, your password entry is recorded in the NIS passwd database, you will be able to login on all machines on the network which have the NIS client programs running.

HOW IT WORKS

Within a network there must be at least one machine acting as a NIS server. You can have multiple NIS servers, each serving different NIS "domains" - or you can have cooperating NIS servers, where one is the master NIS server, and all the other are so-called slave NIS servers (for a certain NIS "domain", that is!) - or you can have a mix of them...

Slave servers only have copies of the NIS databases and receive these copies from the master NIS server whenever changes are made to the master's databases. Depending on the number of machines in your network and the reliability of your network, you might decide to install one or more slave servers. Whenever a NIS server goes down or is too slow in responding to requests, a NIS client connected to that server will try to find one that is up or faster.

NIS databases are in so-called DBM format, derived from ASCII databases. For example, the files /etc/passwd and /etc/group can be directly converted to DBM format using ASCII-to-DBM translation software (makedbm, included with the server software). The master NIS server should have both, the ASCII databases and the DBM databases.

Slave servers will be notified of any change to the NIS maps, (via the yppush program), and automatically retrieve the necessary changes in order to synchronize their databases. NIS clients do not need to do this since they always talk to the NIS server to read the information stored in it's DBM databases.

Old ypbind versions do a broadcast to find a running NIS server. This is insecure, due the fact that anyone may install a NIS server and answer the broadcast queries. Newer Versions of ypbind (ypbind-3.3 or ypbind-mt) are able to get the server from a configuration file - thus no need to broadcast.

Details of NIS from Sun

Network Information Service (NIS) (Overview)

This chapter provides an overview of the Network Information Service (NIS).

NIS is a distributed naming service. It is a mechanism for identifying and locating network objects and resources. It provides a uniform storage and retrieval method for network-wide information in a transport-protocol and media-independent fashion.

NIS Introduction

By running NIS, the system administrator can distribute administrative databases, called **maps**, among a variety of servers (**master** and **slaves**). The administrator can update those databases from a centralized location in an automatic and reliable fashion to ensure that all clients share the same naming service information in a consistent manner throughout the network.

NIS was developed independently of DNS and has a slightly different focus. Whereas DNS focuses on making communication simpler by using machine names instead of numerical IP addresses, NIS focuses on making network administration more manageable by providing centralized control over a variety of network information. NIS stores information not only about machine names and addresses, but also about users, the network itself, and network services. This collection of network **information** is referred to as the NIS **namespace**.

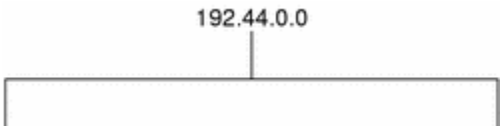
Note -

In some contexts, **machine** names are referred to has **host** names or **machine** names. This discussion uses **machine**, but some screen messages or NIS map names might use **host** or **machine**.

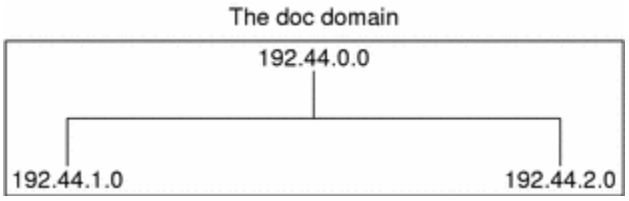
NIS Architecture

NIS uses a client-server arrangement. NIS servers provide services to NIS clients. The principal servers are called **master** servers, and for reliability, they have backup, or **slave** servers. Both master and slave servers use the NIS information retrieval software and both store NIS maps.

NIS uses domains to arrange the machines, users, and networks in its namespace. However, it does not use a domain hierarchy; an NIS namespace is flat. Thus, this physical network:



would be arranged into one NIS domain:



An NIS domain cannot be connected directly to the Internet using just NIS. However, organizations that want to use NIS and also be connected to the Internet can combine NIS with DNS. You can use NIS to manage all local information and use DNS for Internet host lookup. NIS provides a forwarding service that forwards host lookups to DNS if the information cannot be found in an NIS map. The Solaris operating environment also allows you to set up the `nsswitch.conf` file so that hosts lookup requests go only to DNS, or to DNS and then NIS if not found by DNS, or to NIS and then DNS if not found by NIS

NIS Machine Types

There are three types of NIS machines:

- Master server

- Slave servers
- Clients of NIS servers

Any machine can be an NIS client, but only machines with disks should be NIS servers, either master or slave. Servers are also clients, typically of themselves.

NIS Servers

The NIS server does not have to be the same machine as the NFS file server.

NIS servers come in two varieties, master and slave. The machine designated as master server contains the set of maps that the system administrator creates and updates as necessary. Each NIS domain must have one, and only one, master server, which can propagate NIS updates with the least performance degradation.

You can designate additional NIS servers in the domain as slave servers. A slave server has a complete copy of the master set of NIS maps. Whenever the master server maps are updated, the updates are propagated among the slave servers. Slave servers can handle any overflow of requests from the master server, minimizing "server unavailable" errors.

Normally, the system administrator designates one master server for all NIS maps. However, because each individual NIS map has the machine name of the master server encoded within it, you could designate different servers to act as master and slave servers for different maps. To minimize confusion, designate a single server as the master for all the maps you create within a single domain. The examples in this chapter assume that one server is the master for all maps in the domain.

NIS Clients

NIS clients run processes that request data from maps on the servers. Clients do not make a distinction between master and slave servers, since all NIS servers should have the same information.

NIS Elements

The NIS naming service is composed of the following elements:

- Domains
- Maps
- Daemons
- Utilities

- NIS Command Set

The NIS Domain

An NIS **domain** is a collection of machines which share a common set of NIS maps. Each domain has a domain name and each machine sharing the common set of maps belongs to that domain.

Any machine can belong to a given domain, as long as there is a server for that domain's maps in the same network. An NIS client machine obtains its domain name and binds to an NIS server as part of its boot process.

NIS Daemons

NIS service is provided by five daemons as shown in Table 7-1.

Table 7-1 NIS Daemons

Daemon	**Function**
`ypserv`	Server process
`ypbind`	Binding process
`ypxfr`	High speed map transfer
`rpc.yppasswdd`	NIS password update daemon ** See NOTE below.**
`rpc.ypupdated`	Modifies other maps such as `publickey`

Note -

`rpc.yppasswdd` considers all shells that begin with an `r` to be restricted. This means that users who have a shell that begins with an `r`. For example, if you are in `/bin/rksh`, you are not allowed to change from that shell to another one.

NIS Utilities

NIS service is supported by nine utilities as shown in Table 7-2.

Table 7-2 NIS Utilities

Utility	**Function**
`makedbm`	Creates `dbm` file for an NIS map
`ypcat`	Lists data in a map
`ypinit`	Builds and installs an NIS database and initializes NIS client's `ypservers` list.
`yppmatch`	Finds a specific entry in a map
`yppoll`	Gets a map order number from a server
`yppush`	Propagates data from NIS master to NIS slave server
`ypset`	Sets binding to a particular server
`ypwhich`	Lists name of the NIS server and nickname translation table
`ypxfr`	Transfers data from master to slave NIS server

NIS Maps

The information in NIS maps is stored in `ndbm` format. ypfiles(4) and ndbm(3C) explain the format of the map file.

NIS maps were designed to replace UNIX `/etc` files, as well as other configuration files, so they store much more than names and addresses. On a network running NIS, the NIS master server for each NIS domain maintains a set of NIS maps for other machines in the domain to query. NIS slave servers also maintain duplicates of the

master server's maps. NIS client machines can obtain namespace information from either master or slave servers.

NIS maps are essentially two-column tables. One column is the **key** and the other column is information related to the key. NIS finds information for a client by searching through the keys. Some information is stored in several maps because each map uses a different key. For example, the names and addresses of machines are stored in two maps: `hosts.byname` and `hosts.byaddr`. When a server has a machine's name and needs to find its address, it looks in the `hosts.byname` map. When it has the address and needs to find the name, it looks in the `hosts.byaddr` map.

An NIS `Makefile` is stored in the `/var/yp` directory of machines designated as an NIS server at installation time. Running `make` in that directory causes `makedbm` to create or modify the default NIS maps from the input files.

Note -

Always create maps on the master server, as maps created on a slave will not automatically be pushed to the master server.

Default NIS Maps

A default set of NIS maps are provided in the Solaris operating environment. You might want to use all these maps or only some of them. NIS can also use whatever maps you create or add when you install other software products.

Default maps for a NIS domain are located in each server's `/var/yp/`*domainname* directory. For example, the maps that belong to the domain `test.com` are located in each server's `/var/yp/test.com` directory.

Table 7-3 describes the default NIS maps, information they contain, and whether the software consults the corresponding administrative files when NIS is running.

Table 7-3 NIS Map Descriptions

Map Name	Corresponding NIS Admin File	Description
`bootparams`	`bootparams`	Contains path names of files clients need during boot: root, swap, possibly others.
`ethers.byaddr`	`ethers`	Contains machine names and Ethernet addresses. The Ethernet address is the key in the map.
`ethers.byname`	`ethers`	Same as `ethers.byaddr`, except the key is machine name instead of the Ethernet address.
`group.bygid`	`group`	Contains group security information with group ID as key.
`group.byname`	`group`	Contains group security information with group name as key.
`hosts.byaddr`	`hosts`	Contains machine name, and IP address, with IP address as key.
`hosts.byname`	`hosts`	Contains machine name and IP address, with machine (host) name as key.
`mail.aliases`	`aliases`	Contains aliases and mail addresses, with aliases as key.
`mail.byaddr`	`aliases`	Contains mail address and alias, with mail address as key.
`netgroup.byhost`	`netgroup`	Contains group name, user name and machine name.

Map Name	Corresponding NIS Admin File	Description
`netgroup.byuser`	`netgroup`	Same as `netgroup.byhost`, except that key is user name.
`netgroup`	`netgroup`	Same as `netgroup.byhost`, except that key is group name.

`netid.byname`	`passwd, hosts`
	
	`group`
		Used for UNIX-style authentication. Contains machine name and mail address (including domain name). If there is a `netid` file available it is consulted in addition to the data available through the other files.

`netmasks.byaddr`	`netmasks`	Contains network mask to be used with IP submitting, with the address as key.
`networks.byaddr`	`networks`	Contains names of networks known to your system and their IP addresses, with the address as key.
`networks.byname`	`networks`	Same as `networks.byaddr`, except key is name of network.
`passwd.adjunct. byname`	`passwd` and `shadow`	Contains auditing information and the hidden password information for C2 clients.
`passwd.byname`	`passwd` and `shadow`	Contains password information with user name as key.

Map Name	Corresponding NIS Admin File	Description
passwd.byuid	passwd and shadow	Same as passwd.byname, except that key is user ID.
protocols.byname	protocols	Contains network protocols known to your network.
protocols.bynumber	protocols	Same as protocols.byname, except that key is protocol number.
rpc.bynumber	rpc	Contains program number and name of RPCs known to your system. Key is RPC program number.
services.byname	services	Lists Internet services known to your network. Key is port or protocol.
services.byservice	services	Lists Internet services known to your network. Key is service name.
ypservers	N/A	Lists NIS servers known to your network.

New ipnodes maps (ipnodes.byaddr and ipnodes.byname) are added to NIS. The maps store both IPv4 and IPv6 addresses. See ipnodes(4). NIS clients and servers can communicate using either IPv4 or IPv6 RPC transports.

Using NIS Maps

NIS makes updating network databases much simpler than with the /etc files system. You no longer have to change the administrative /etc files on every machine each time you modify the network environment.

For example, when you add a new machine to a network running NIS, you only have to update the input file in the master server and run `make`. This automatically updates the `hosts.byname` and `hosts.byaddr` maps. These maps are then transferred to any slave servers and are made available to all of the domain's client machines and their programs. When a client machine or application requests a machine name or address, the NIS server refers to the `hosts.byname` or `hosts.byaddr` map as appropriate and sends the requested information to the client.

You can use the `ypcat` command to display the values in a map. The `ypcat` basic format is the following.

% **`ypcat`** ***mapname***

where *mapname* is the name of the map you want to examine or its **nickname**. If a map is composed only of keys, as in the case of `ypservers`, use `ypcat` **`-k`**. Otherwise, `ypcat` prints blank lines. The `ypcat` man page describes more options for `ypcat`.

You can use the `ypwhich` command to determine which server is the master of a particular map. Type the following.

% **`ypwhich -m`** ***mapname***

where *mapname* is the name or the nickname of the map whose master you want to find. `ypwhich` responds by displaying the name of the master server. For complete information, refer to the `ypwhich` man page.

NIS Map Nicknames

Nicknames are aliases for full map names. To obtain a list of available map nicknames, such as `passwd` for `passwd.byname`, type `ypcat` **`-x`** or `ypwhich` **`-x`**.

Nicknames are stored in the `/var/yp/nicknames` file, which contains a map nickname followed by the fully specified name for the map, separated by a space. This list might be added to or modified. Currently, there is a limit of 500 nicknames.

NIS-Related Commands

The NIS service includes specialized daemons, system programs, and commands, which are summarized in the following table.

Table 7-4 NIS Command Summary

Command	Description
`ypserv`	Services NIS clients' requests for information from an NIS map. `ypserv` is a daemon that runs on NIS servers with a complete set of maps. At least one `ypserv` daemon must be present on the network for NIS service to function.
`ypbind`	Provides NIS server binding information to clients. It provides binding by finding a `ypserv` process that serves maps within the domain of the requesting client. `ypbind` must run on all servers and clients.
`ypinit`	Automatically creates maps for an NIS server from the input files. It is also used to construct the initial `/var/yp/binding/`*domain*`/ypservers` file on the clients. Use `ypinit` to set up the master NIS server and the slave NIS servers for the first time.
`make`	Updates NIS maps by reading the `Makefile` (when run in the `/var/yp` directory). You can use `make` to update all maps based on the input files or to update individual maps. The `ypmake(1M)` man page describes the functionality of `make` for NIS.
`makedbm`	`makedbm` takes an input file and converts it into `dbm.dir` and `dbm.pag` files--valid `dbm` files that NIS can use as maps. You can also use `makedbm` **`-u`** to disassemble a map, so that you can see the key-value pairs that comprise it.
`ypxfr`	Pulls an NIS map from a remote server to the local `/var/yp/`*domain* directory, using NIS itself as the transport medium. You can run `ypxfr` interactively, or periodically from a `crontab` file. It is also called by `ypserv` to initiate a transfer.
`ypxfrd`	Provides map transfers service for `ypxfr` requests (generally slave servers). It is run only on the master server.
`yppush`	Copies a new version of an NIS map from the NIS master server to its slaves. You run it on the master NIS server.
`ypset`	Tells a `ypbind` process to bind to a named NIS server. This is not for casual use and its use is discouraged because of security implications. See the `ypset(1M)` and `ypbind(1M)` man pages for information about the **`ypset`** and **`ypsetme`**options to the `ypbind` process.
`yppoll`	Tells which version of an NIS map is running on a server that you specify. It also lists the master server for the map.

Command	Description
`ypcat`	Displays the contents of an NIS map.
`ypmatch`	Prints the value for one or more specified keys in an NIS map. You cannot specify which version of the NIS server map you are seeing.
`ypwhich`	Shows which NIS server a client is using at the moment for NIS services, or, if invoked with the **`-m`** *mapname* option, which NIS server is master of each of the maps. If only **`-m`** is used, it displays the names of all the maps available and their respective master servers.

NIS Binding

NIS clients get information from an NIS server through the binding process, which can work in one of two modes: server-list or broadcast.

- Server-list. In the server-list mode, the `ypbind` process queries the `/var/yp/binding/`*domain*`/ypservers` list for the names of all of the NIS servers in the domain. The `ypbind` process binds only to servers in this file. The file is created by running `ypinit` **`-c`**.
- Broadcast. The `ypbind` process can also use an RPC broadcast to initiate a binding. Since broadcasts are only local subnet events that are not routed further, there must be at least one server (master or slave) on the same subnet as the client. The servers themselves might exist throughout different subnets since map propagation works across subnet boundaries. In a subnet environment, one common method is to make the subnet router an NIS server. This allows the domain server to serve clients on either subnet interface.

Server-List Mode

The binding process in server-list mode works as follows:

1. Any program, running on the NIS client machine that needs information provided by an NIS map, asks `ypbind` for the name of a server.
2. `ypbind` looks in the `/var/yp/binding/`*domainname*`/ypservers` file for a list of NIS servers for the domain.

3. `ypbind` initiates binding to the first server in the list. If the server does not respond, `ypbind` tries the second, and so on, until it finds a server or exhausts the list.
4. `ypbind` tells the client process which server to talk to. The client then sends the request directly to the server.
5. The `ypserv` daemon on the NIS server handles the request by consulting the appropriate map.
6. `ypserv` sends the requested information back to the client.

Broadcast Mode

The broadcast mode binding process works as follows:

1. `ypbind` must be started with the broadcast option set (**`broadcast`**).
2. `ypbind` issues an RPC broadcast in search of an NIS server.

Note -

In order to support such clients, it is necessary to have an NIS server on each subnet requiring NIS service.

---

3. `ypbind` initiates binding to the first server that responds to the broadcast.
4. `ypbind` tells the client process which server to talk to. The client then sends the request directly to the server.
5. The `ypserv` daemon on the NIS server handles the request by consulting the appropriate map.
6. `ypserv` sends the requested information back to the client.

Normally, once a client is bound to a server it stays bound to that server until something causes it to change. For example, if a server goes out of service, the clients it served will then bind to new servers.

To find out which NIS server is currently providing service to a specific client, use the following command.

%**`ypwhich`** ***machinename***

Where *machinename* is the name of the client. If no machine name is mentioned, `ypwhich` defaults to the local machine (that is, the machine on which the command is run).