NAME

yp - description of the YP/NIS system

SYNOPSIS

yp

DESCRIPTION

The **YP** subsystem allows network management of passwd, group, netgroup, hosts, services, rpc, bootparams and ethers file entries through the functions getpwent(3), getgrent(3), getnetgrent(3), gethostent(3), getnetent(3), getrpcent(3), and ethers(3). The bootparamd(8) daemon makes direct NIS library calls since there are no functions in the standard C library for reading bootparams. NIS support is enabled in nsswitch.conf(5).

The **YP** subsystem is started automatically in */etc/rc* if it has been initialized in */etc/rc.conf* and if the directory */var/yp* exists (which it does in the default distribution). The default NIS domain must also be set with the domainname(1) command, which will happen automatically at system startup if it is specified in */etc/rc.conf*.

NIS is an RPC-based client/server system that allows a group of machines within an NIS domain to share a common set of configuration files. This permits a system administrator to set up NIS client systems with only minimal configuration data and add, remove or modify configuration data from a single location.

The canonical copies of all NIS information are stored on a single machine called the NIS *master server*. The databases used to store the information are called NIS *maps*. In FreeBSD, these maps are stored in */var/yp/<domainname>* where <*domainname*> is the name of the NIS domain being served. A single NIS server can support several domains at once, therefore it is possible to have several such directories, one for each supported domain. Each domain will have its own independent set of maps.

In FreeBSD, the NIS maps are Berkeley DB hashed database files (the same format used for the passwd(5) database files). Other operating systems that support NIS use old-style **ndbm** databases instead (largely because Sun Microsystems originally based their NIS implementation on **ndbm**, and other vendors have simply licensed Sun's code rather than design their own implementation with a different database format). On these systems, the databases are generally split into *.dir* and *.pag* files which the **ndbm** code uses to hold separate parts of the hash database. The Berkeley DB hash method instead uses a single file for both pieces of information. This means that while you may have *passwd.byname.dir* and *passwd.byname.pag* files on other operating systems (both of which are really parts of the same map), FreeBSD will have only one file called *passwd.byname*. The difference in format is not significant: only the NIS server, ypserv(8), and related tools need to know the database format of the NIS maps. Client NIS systems receive all NIS data in ASCII form.

There are three main types of NIS systems:

1. NIS clients, which query NIS servers for information.

2. NIS master servers, which maintain the canonical copies of all NIS maps.

3. NIS slave servers, which maintain backup copies of NIS maps that are periodically updated by the master.

A NIS client establishes what is called a *binding* to a particular NIS server using the ypbind(8) daemon. The ypbind(8) utility checks the system's default domain (as set by the domainname(1) command) and begins broadcasting RPC requests on the local network. These requests specify the name of the domain for which ypbind(8) is attempting to establish a binding. If a server that has been configured to serve the requested domain receives one of the broadcasts, it will respond to ypbind(8), which will record the server's address. If there are several servers available (a master and several slaves, for example), ypbind(8) will use the address of the first one to respond. From that point on, the client system will direct all of its NIS requests to that server. The ypbind(8) utility will occasionally "ping" the server to make sure it is still up and running. If it fails to receive a reply to one of its pings within a reasonable amount of time, ypbind(8) will mark the domain as unbound and begin broadcasting again in the hopes of locating another server.

NIS master and slave servers handle all NIS requests with the ypserv(8) daemon. The ypserv(8) utility is responsible for receiving incoming requests from NIS clients, translating the requested domain and map name to a path to the corresponding database file and transmitting data from the database back to the client. There is a specific set of requests that ypserv(8) is designed to handle, most of which are implemented as functions within the standard C library:

yp_order()
: check the creation date of a particular map

yp_master()
: obtain the name of the NIS master server for a given map/domain

yp_match()
: lookup the data corresponding to a given in key in a particular map/domain

yp_first() obtain the first key/data pair in a particular map/domain

yp_next() pass ypserv(8) a key in a particular map/domain and have it return the key/data pair immediately following it (the functions **yp_first**() and **yp_next**() can be used to do a

sequential search of an NIS map)

yp_all() retrieve the entire contents of a map

There are a few other requests which ypserv(8) is capable of handling (i.e., acknowledge whether or not you can handle a particular domain (YPPROC_DOMAIN), or acknowledge only if you can handle the domain and be silent otherwise (YPPROC_DOMAIN_NONACK)) but these requests are usually generated only by ypbind(8) and are not meant to be used by standard utilities.

On networks with a large number of hosts, it is often a good idea to use a master server and several slaves rather than just a single master server. A slave server provides the exact same information as a master server: whenever the maps on the master server are updated, the new data should be propagated to the slave systems using the yppush(8) command. The NIS *Makefile* (*/var/yp/Makefile*) will do this automatically if the administrator creates */var/yp/Makefile.local* and empties the *NOPUSH* variable:

NOPUSH=

(*NOPUSH* is set to true by default because the default configuration is for a small network with only one NIS server). The yppush(8) command will initiate a transaction between the master and slave during which the slave will transfer the specified maps from the master server using ypxfr(8). (The slave server calls ypxfr(8) automatically from within ypserv(8); therefore it is not usually necessary for the administrator to use it directly. It can be run manually if desired, however.) Maintaining slave servers helps improve NIS performance on large networks by:

- Providing backup services in the event that the NIS master crashes or becomes unreachable
- Spreading the client load out over several machines instead of causing the master to become overloaded
- Allowing a single NIS domain to extend beyond a local network (the ypbind(8) daemon might not be able to locate a server automatically if it resides on a network outside the reach of its broadcasts. It is possible to force ypbind(8) to bind to a particular server with ypset(8) but this is sometimes inconvenient. This problem can be avoided simply by placing a slave server on the local network.)

The FreeBSD ypserv(8) is specially designed to provide enhanced security (compared to other NIS implementations) when used exclusively with FreeBSD client systems. The FreeBSD password database system (which is derived directly from 4.4BSD) includes support for *shadow passwords*. The standard password database does not contain users' encrypted passwords: these are instead stored (along with other information) in a separate database which is accessible only by the super-user. If the encrypted password database were made available as an NIS map, this security feature would be totally

disabled, since any user is allowed to retrieve NIS data.

To help prevent this, FreeBSD's NIS server handles the shadow password maps (*master.passwd.byname*, *master.passwd.byuid*, *shadow.byname* and *shadow.byuid*) in a special way: the server will only provide access to these maps in response to requests that originate on privileged ports. Since only the super-user is allowed to bind to a privileged port, the server assumes that all such requests come from privileged users. All other requests are denied: requests from non-privileged ports will receive only an error code from the server. Additionally, FreeBSD's ypserv(8) includes support for Wietse Venema's tcp wrapper package; with tcp wrapper support enabled, the administrator can configure ypserv(8) to respond only to selected client machines.

While these enhancements provide better security than stock NIS, they are by no means 100% effective. It is still possible for someone with access to your network to spoof the server into disclosing the shadow password maps.

On the client side, FreeBSD's getpwent(3) functions will automatically search for the *master.passwd* maps and use them if they exist. If they do, they will be used, and all fields in these special maps (class, password age and account expiration) will be decoded. If they are not found, the standard *passwd* maps will be used instead.

COMPATIBILITY

When using a non-FreeBSD NIS server for passwd(5) files, it is unlikely that the default MD5-based format that FreeBSD uses for passwords will be accepted by it. If this is the case, the value of the *passwd_format* setting in login.conf(5) should be changed to "des" for compatibility.

Some systems, such as SunOS 4.x, need NIS to be running in order for their hostname resolution functions (**gethostbyname**(), **gethostbyaddr**(), etc.) to work properly. On these systems, ypserv(8) performs DNS lookups when asked to return information about a host that does not exist in its *hosts.byname* or *hosts.byaddr* maps. FreeBSD's resolver uses DNS by default (it can be made to use NIS, if desired), therefore its NIS server does not do DNS lookups by default. However, ypserv(8) can be made to perform DNS lookups if it is started with a special flag. It can also be made to register itself as an NIS v1 server in order to placate certain systems that insist on the presence of a v1 server (FreeBSD uses only NIS v2, but many other systems, including SunOS 4.x, search for both a v1 and v2 server when binding). FreeBSD's ypserv(8) does not actually handle NIS v1 requests, but this "kludge mode" is useful for silencing stubborn systems that search for both a v1 and v2 server.

(Please see the ypserv(8) manual page for a detailed description of these special features and flags.)

SEE ALSO

domainname(1), ypcat(1), ypmatch(1), ypwhich(1), nsswitch.conf(5), yp_mkdb(8), ypbind(8), ypinit(8),

yppoll(8), yppush(8), ypserv(8), ypset(8), ypxfr(8)

HISTORY

The **YP** subsystem was written from the ground up by Theo de Raadt to be compatible to Sun's implementation. Bug fixes, improvements and NIS server support were later added by Bill Paul. The server-side code was originally written by Peter Eriksson and Tobias Reber and is subject to the GNU Public License. No Sun code was referenced.

BUGS

While FreeBSD now has both NIS client and server capabilities, it does not yet have support for ypupdated(8) or the **yp_update**() function. Both of these require secure RPC, which FreeBSD does not support yet either.

The getservent(3) and getprotoent(3) functions do not yet have NIS support. Fortunately, these files do not need to be updated that often.

Many more manual pages should be written, especially ypclnt(3). For the time being, seek out a local Sun machine and read the manuals for there.

Neither Sun nor this author have found a clean way to handle the problems that occur when ypbind cannot find its server upon bootup.