OpenStep Journal, Summer 1995 (Volume 1, Issue 2).

The Village Smythy

Written by **Alan M. Marcum**

> *Under the spreading chestnut-tree,*
> *The village smithy stands ¼*
> *ÐHenry Wadsworth Longfellow*

(Note that sidebars and marginal notes in the printed journal are denoted here by smaller type with bars above and below the item.)

In the last issue, we explored the portion of NEXTSTEP network startup through the end of **/etc/rc.net**'s execution. Let's continue our exploration, after ¼

A BRIEF REVIEW OF NETWORK STARTUP

The **/etc/rc.net** script con®gures the network interface and sets a hostname. It does this by processing **/etc/iftab** and **/etc/hostcon®g**, with the various settings there affecting whether, for example, things such as the IP address or hostname are hard-coded or are obtained from the network.You can set the values of the various parameters in **hostcon®g** using HostManager; **iftab** must be editedÐcarefullyÐby hand (normally, you won't need to change **iftab**).

Questions? Send your questions to **smythy@NeXT.COM**. In future editions of this column, I'll address some of those I think are most interesting.

AND NOW, ON WITH THE SHOW ¼

Following processing **rc.net**, **/etc/rc** invokes **rc.swap**. This anachronism is a throwback to the days when NEXTSTEP on NeXT hardware supported a swapdiskÐa small SCSI disk used for paging, typically when booting from an optical disk or from the network. Swapdisks are rarely used anymore, so we won't cover **rc.swap** in this article.

The **rc** mounts the local ®lesystems with the command:

```
mount -vat 4.3
```

The **-v** ¯ag speci®es verbose operations (what happens is reported to stdout); **-a** signi®es that all known ®lesystems without the **noauto** keyword should be mounted; **-t 4.3** indicates that **-a** is restricted to ®lesystems of type 4.3 (that is, UNIX ®lesystems, speci®cally BSD 4.3 ®lesystems). Note that, because the NetInfo™ tool is not yet started, this command obtains information from the ®le **/etc/fstab**, *not* from NetInfo.

Sprinkled through **rc** are calls to the **fbshow** program, which displays a message during graphical startup.

Because this series of articles focuses on network startup, we'll skip over much of the nonnetworking portions of **/etc/rc**. Refer to the comments in the script itself, the UNIX manual pages, various articles such as ªoptimizing virtual memory with swaptabº in the Winter 1993 issue (Volume 3, Issue 1) of the *support bulletin*, and the *UNIX System Administrators Handbook* (*USAH*), 2nd edition, by Evi Nemeth, Garth Snyder, John Seabass, and Trent Hahn.

Let's skip down to where **syslogd** is started:

```
(echo -n Starting early daemons:)             >/dev/console
fbshow -B -I "Starting background services" -z 27
if [ -f /usr/etc/syslogd -a -f /etc/syslog.conf ]; then
    /bin/rm -f /dev/log
    # If you want a timestamp to be logged periodically,
    # modify the invocation of syslogd below. For example,
    # for a half-hourly timestamp, add the argument "-m30".
    /usr/etc/syslogd  && (echo -n ' syslogd')   >/dev/console
fi
```

(echo '.') >/dev/console

syslogd is the system logging daemon; it provides a centralized collection and dissemination point for system messages. syslog is used by most of the system daemons, including NetInfo, so it should be started early. For more information on syslog, see ªprowling about in syslogº in the Winter 1993 issue of the *support bulletin*, the various UNIX manual pagesÐincluding sections 3 (library routines), 5 (®le formats), and 8 (administrative utilities)Ðand the *USAH*.

After reading in **/etc/hostcon®g**, **rc** determines whether NIS should be consulted.This is done by examining the **YPDOMAIN** parameter in **hostcon®g**; if the parameter's value is **-NO-**, the NIS will not be consulted for system information, and no NIS domain name is set. Any other value will lead to that value being used as the NIS domain name and to NIS being consulted for system information using the normal NIS semantics. (You can modify the **YPDOMAIN** parameter using HostManager.)

STARTING RPC AND NETWORK SERVICES

What are ªRPCº services? They are those services that use the SunRPC protocol for providing a mechanism for remote procedure invocation. The ®rst of these, and one on which all others are based, is the portmapper. As described in ªNetInfo Binding and Connecting,º *NEXTSTEP In Focus*, Summer 1993 (Volume 3, Issue 3):

> The portmapper is like directory assistance, only instead of matching names to telephone numbers, it translates SunRPC program numbers into port numbers. In portmapper parlance, this is the operation **PMAPPROC_GETPORT**. The portmapper allows clients to ®nd many SunRPC programs without having to reserve a well-known port for each one.

The portmapper is itself a SunRPC-based program. It happens to have a well-known port number, just as the telephone system's directory assistance service has a well-known number (411 in the United States, for example). The portmapper is used extensively by NetInfo, NFS, and NIS, among other services.

(If you're following along in **/etc/rc**, you probably noticed that there are some lines relating to **nmserver** that are commented out, just before the portmapper is started. These should remain commented out [or even be removed]: they are vestiges of times long past, when **nmserver** was started at this point, rather than earlier.)

The **rc** script con®gures IP routing next, so that IP packets can be sent to computers on other networks. Two types of routing are supported through the default mechanisms: ªdynamicº routing, using the **routed** program and the RIP protocol, and ªstaticº routing, where an explicit default route is speci®ed. In the former case, **routed** builds a routing table based on the RIP packets obtained from network broadcasts. In the latter case, all remote networks are reached by forwarding the packet to the speci®ed router.

Note that when dynamic routing is speci®ed, the **-q** ¯ag is used when invoking **routed**. This informs **routed** that it should not supply routing information to the network.

Note that NEXTSTEP does not support routing of packets from one interface to another and, indeed, does not support multiple network interfaces as of Release 3.3.

If static routing is requested, a command similar to the following is invoked (this command assumes that **192.42.172.1** is the address of the router for this network):

```
/usr/etc/route add default 192.42.172.1 1        >/dev/console 2>&1
```

What does this call to **/usr/etc/route** mean? It directs **route** to add a route to the kernel's routing table (the **add** keyword). This route should be used if no explicit routes are available (**default**). The router's address is speci®ed next (**192.42.172.1** in this example), and the router is onªhopº away (**1** in this example). For details, see the UNIX manual pages for **route**(8); for more information on routing in general, see the networking references in last issue's column.

Routing is con®gured *before* NetInfo is started. This allows **netinfod** to bind and **lookupd** to connect across networks (subnets), although it forces you to use explicit IP addresses, rather than hostnames, for static routing con®guration. (If you insist on using a symbolic hostname for routing, just ensure that it's in **/etc/hosts**, along with its IP address, and that you keep **/etc/hosts** current.)

TIME TO START NETINFO

Once the portmapper and routing are con®gured, it's time to start NetInfo. This is done by invoking **nibindd**. The following lines from **/etc/rc** do this:

```
if [ -f /usr/etc/nibindd ]; then
    fbshow -B -I "Starting network services" -z 32
```

```
    /usr/etc/nibindd && (echo -n ' netinfo')       >/dev/console 2>&1
fi
```

When **nibindd** starts, it does the following:

- Creates a ªpid ®leº (a ®le containing its process ID number)
- Locks the **/etc/netinfo** directory, so that no other **nibindd**s will run
- Registers the RPC service with the RPC protocol-dispatching mechanism
- Registers its UDP and its TCP services with the portmapper
- Runs a **netinfod** for each NetInfo database in **/etc/netinfo**

Once started, **nibindd** awaits requests from clients.

How does **nibindd** determine which NetInfo databases exist in **/etc/netinfo**? It looks for entries in **/etc/netinfo** that have a suf®x of **.nidb**, **.move**, or **.temp**. These entries represent NetInfo databases; those with a **.move** or a **.temp** suf®x are temporary databases.

For details about the differences between **.move** and **.temp**, consult ªNetInfo Binding and Connectingº in *NEXTSTEP In Focus*, Summer 1993.

For each **netinfod** started by **nibindd**, the following steps are accomplished:

1. Check the database for consistency, if needed, calculating its checksum while so doing. Consistency checking will be needed if the **checksum** ®le is missing from the database, indicating that the previous **netinfod** running on that database didn't terminate normally. This consistency checking veri®es the internal structure of the NetInfo database.
2. Register the RPC service with the RPC protocol-dispatching mechanism.
3. Bind to a parent server if **netinfod**'s tag is **local** (if its tag isn't **local**, parent binding is done lazily, only when necessary). See ªNetInfo Binding and Connectingº from *NEXTSTEP In Focus*, Summer 1993, for details on children binding to a parent.
4. Register its UDP and TCP ports with **nibindd**.
5. If running as a clone, send a **readall** request to the master, to ensure its database is synchronized with the master's. If running as a master, send a **crashed** request to each

clone, informing the clones that the master has restarted (if the clone's database is out of date, the clone will request a new database from the master, using the **readall** operation). See the section ªWhen a Clone Misses an Updateº in ªPropagating Updatesº from *NEXTSTEP In Focus*, Summer 1993, for more information on database synchronization in the face of failures.

6 If NetInfo from NEXTSTEP Release 3.3 is running (NetInfo version 58), and if running a clone, check to ensure that it's not a ªrogue clone.º See ªDetecting Rogue Serversº in *NEXTSTEP In Focus*, Spring 1994, for additional information on rogue clones.

If there's a problem in step 3 (**netinfod local** binding to its parent), you'll see the following message on your console:

Still searching for parent network administration (NetInfo) server.
Please wait, or press 'c' to continue without network user accounts.

See your system administrator if you need help.

This is the standard message; it's con®gurable, so yours might even be completely different from this. (The message resides in the ®le **NetInfo.strings** in the various *language*.**lproj** directories within **/usr/lib/NextStep/Resources**; edit the portion of the message to the *right* of the equals sign if you want to see what's displayed.) If you see this message, it's because **netinfod local** is looking for a parent and not getting an answer. It'll keep lookingÐforever, or until it ®nds a parent, or until you tell it to continue without a parent. If your network just crashed (maybe there was a sitewide power failure, for example), it might take 10 or 20 minutes after all the disks are checked for NetInfo to recover completely from the failure.

OTHER DATA-ACCESSING DAEMONS

If an NIS domain name was speci®ed in **/etc/hostconfig** (using HostManager), and if the network is up, then binding to an NIS server is performed. The following lines invoke the NIS services:

```
# If we are in an NIS domain, start up the appropriate services.

if [ "$YPDOMAIN" != "-NO-" -a $NETWORKUP = "-YES-" ]; then
    fbshow -B -I "Starting YP services" -z 36
    # ypserv is run on NIS servers - machines with an /etc/yp/XXX dir
    if [ -f /usr/etc/ypserv -a -d /etc/yp/$YPDOMAIN ]; then
```

```
        /usr/etc/ypserv && (echo -n ' ypserv')              >/dev/console
    fi
    if [ -f /usr/etc/ypbind ]; then
        /usr/etc/ypbind && (echo -n ' ypbind')              >/dev/console
    fi
fi
```

There are two steps in this code fragment. The ®rst, which invokes **/usr/etc/ypserv**, is performed only if this computer should run an NIS server; this is determined based on the presence or absence of an appropriate subdirectory in **/etc/yp**. The second, which invokes **/usr/etc/ypbind**, binds the data lookup mechanisms to an NIS server.

NetInfo's **lookupd** is started next. By default, **lookupd** will load its cache of all known users (called the **pwent** cache). This entails iterating over all the users in the local NetInfo domainÐall subdirectories of **/users**Ðthen all the users in the second-level domain, and so on through the root domain, and then iterating over all the users in NIS, if NIS is con®gured.

For details on **lookupd**, see ªThe NetInfo Lookup ServerÐ**lookupd**º in the Summer 1993 issue of *NEXTSTEP In Focus*.

To iterate over the users in a domain, **lookupd** must connect to the domain. To start, **lookupd** connects to the local domainÐthe domain tagged **local** on the current machineÐusing the loopback interface (127.0.0.1). Note that the start of the domain hierarchyÐthe ªaddressº of **netinfod local** on the local hostÐis always known: **127.0.0.1/local** in NetInfo parlance.

After connecting to the local domain and exhausting its user information, **lookupd** sends **netinfod local** an **rparent** message, requesting the address and tag of **netinfod local**'s parent domain, if it has one. If there's no parentÐbecause this machine's NetInfo is standalone and has no parent domain, even if it's on a networkÐthe search through NetInfo ends. If there's a parent, **lookupd** gets the users from that parent domainÐthe ªsecond-level domain.º Because **netinfod local** always binds to its parent, if it has one, before completing its startup sequence (step 3 in the **netinfod** startup list, above), **local**'s parent is always available at this point, if there is one.

When **lookupd** exhausts the information in the second-level domain, it asks the server providing that information for its parentÐfor the address of a server for the third-level domain. If the domain hierarchy is two deep, the second-level server replies that it's running the root domain; if the hierarchy is deeper than two, and if the second-level server has bound to its parent already, then the address of a

server for the third-level domain will be returned.

Recall in step 3 from the **netinfod** startup list that only **netinfod local** binds during its startup sequence. But the second-level server is never tagged **local**, so there's a chance that it won't have bound to its parent when the information is requested. If this is the case, the server will initiate binding at the time it receives the **rparent** message requesting the address and tag of its bound parent, and it will respond with the answer.

Before returning information about a bound parent, a **netinfod** verifies that the parent is still available. As an example, let's say that **netinfod network** on a host called **sabre** is asked for its parent, and that it had previously bound to **netinfod Rhino** on host **mustang** (also known by the notation **mustang/Rhino**). **sabre/network** will verify that **mustang/Rhino** is up before responding with the information. It does this by contacting the **nibindd** running on **mustang**, verifying that the parent's tag, **Rhino**, is still registered. Note that it does *not*, in fact, contact the parent server, **mustang/Rhino**, itself. If **mustang** is down, or if **mustang**'s **nibindd** has crashed, or if the tag **Rhino** is no longer registered with **mustang**'s **nibindd**, then **netinfod network** on **sabre** will rebind, repeating exactly the initial binding sequence.

TIMEOUTS, TIMEOUTS, TIMEOUTS

Let's revisit the previous example, where **sabre** is starting up. Let's say that **lookupd** on **sabre** is ®lling its **pwent** cache. It has gathered all the information available from **sabre/local**, asked **sabre/local** for its parent and received the response **sabre/network**, and exhausted **sabre/network**'s available users information. It then asks **sabre/network** for its parent. In response to this, **sabre/network** attempts to bind.

It gets no response. So it tries again. Still no response.

And **sabre/network** keeps on trying until it *gets* a response.

After a suf®cient period of no response from **sabre/network**, the NetInfo client library routine invoked by **lookupd** will send a message to syslog. Before Release 3.3, this message would be ªnetinfo timeout, sleeping.º Starting with 3.3, the message will be ªNetInfo timeout ®nding server for parent of 192.42.172.66/network, sleeping.º This message will appear only once for each time the remote parent is required.

Let's turn this around for a moment. Let's assume that your computer is hanging during startup, and

you see a message like ªNetInfo timeout ®nding server ¼º You can be pretty well assured that the problem is in binding of the higher-level domains: second level to its parent, or third to its parent, and so on.

STILL TO COME ¼

Although we're only about half-way through **/etc/rc**, we're past most of the dif®cult (and most of the interesting) stuff. We'll wrap up this tour in the next issue.

Alan M. Marcum is a member of NeXT's Premium Support Team, specializing in the management of large networks.

Next Article NeXTanswer #2042 **appDidInit:**
Previous article NeXTanswer #2038 **NICE to Be Sure**
Table of contents

http://www.next.com/HotNews/Journal/OSJ/SummerContents95.html