Name Server Operations Guide for BIND

Release 4.8

Kevin J. Dunlap*
Michael J. Karels

Computer Systems Research Group
Computer Science Division
Department of Electrical Engineering and Computer Sciences
University of California
Berkeley CA 94720

1. Introduction

The Berkeley Internet Name Domain (BIND) Server implements the DARPA Internet name server for the UNIX† operating system. A name server is a network service that enables clients to name resources or objects and share this information with other objects in the network. This in effect is a distributed data base system for objects in a computer network. BIND is fully integrated into 4.3BSD network programs for use in storing and retrieving host names and address. The system administrator can configure the system to use BIND as a replacement to the original host table lookup of information in the network hosts file */etc/hosts*. The default configuration for 4.3BSD uses BIND.

2. Building A System with a Name Server

BIND is comprised of two parts. One is the user interface called the *resolver* which consists of a group of routines that reside in the C library */lib/libc.a*. Second is the actual server called *named*. This is a daemon that runs in the background and services queries on a given network port. The standard port for UDP and TCP is specified in */etc/services*.

2.1. Resolver Routines in libc

When building your 4.3BSD system you may either build the C library to use the name server resolver routines or use the host table lookup routines to do host name and address resolution. The default resolver for 4.3BSD uses the name server.

Building the C library to use the name server changes the way *gethostbyname*(3N), *gethostbyaddr*(3N), and *sethostent*(3N) do their functions. The name server renders *gethostent*(3N) obsolete, since it has no concept of a next line in the database. These library calls are built with the resolver routines needed to query the name server.

The *resolver* is comprised of a few routines that build query packets and exchange them with the name server.

Before building the C library, set the variable *HOSTLOOKUP* equal to *named* in */usr/src/lib/libc/Makefile*. You then make and install the C library and compiler and then compile the rest of the 4.3BSD system. For more information see section 6.6 of ‘‘Installing and Operating

---

* The author was an employee of Digital Equipment Corporation’s Ultrix Engineering Advanced Development Group and was on loan to CSRG when this work was done. Ultrix is a trademark of Digital Equipment Corporation.

†UNIX is a Trademark of AT&T Bell Laboratories

4.3BSD on the VAX‡''.

2.2. The Name Service

The basic function of the name server is to provide information about network objects by answering queries. The specifications for this name server are defined in RFC1034, RFC1035 and RFC974. These documents can be found in */usr/src/etc/named/doc* in 4.3BSD or *ftp*ed from nic.ddn.mil. It is also recommended that you read the related manual pages, *named* (8), *resolver* (3), and *resolver* (5).

The advantage of using a name server over the host table lookup for host name resolution is to avoid the need for a single centralized clearinghouse for all names. The authority for this information can be delegated to the different organizations on the network responsible for it.

The host table lookup routines require that the master file for the entire network be maintained at a central location by a few people. This works fine for small networks where there are only a few machines and the different organizations responsible for them cooperate. But this does not work well for large networks where machines cross organizational boundaries.

With the name server, the network can be broken into a hierarchy of domains. The name space is organized as a tree according to organizational or administrative boundaries. Each node, called a *domain*, is given a label, and the name of the domain is the concatenation of all the labels of the domains from the root to the current domain, listed from right to left separated by dots. A label need only be unique within its domain. The whole space is partitioned into several areas called *zones*, each starting at a domain and extending down to the leaf domains or to domains where other zones start. Zones usually represent administrative boundaries. An example of a host address for a host at the University of California, Berkeley would look as follows:

monet **.** *Berkeley* **.** *EDU*

The top level domain for educational organizations is EDU; Berkeley is a subdomain of EDU and monet is the name of the host.

3. Types of Servers

There are several types of servers: Master, Caching, Remote, and Slave.

3.1. Master Servers

A Master Server for a domain is the authority for that domain. This server maintains all the data corresponding to its domain. Each domain should have at least two master servers, a primary master and some secondary masters to provide backup service if the primary is unavailable or overloaded. A server may be a master for multiple domains, being primary for some domains and secondary for others.

3.1.1. Primary

A Primary Master Server is a server that loads its data from a file on disk. This server may also delegate authority to other servers in its domain.

3.1.2. Secondary

A Secondary Master Server is a server that is delegated authority and receives its data for a domain from a primary master server. At boot time, the secondary server requests all the data for the given zone from the primary master server. This server then periodically checks with the primary server to see if it needs to update its data.

---

‡VAX is a Trademark of Digital Equipment Corporation

3.2. Caching Only Server

All servers are caching servers. This means that the server caches the information that it receives for use until the data expires. A *Caching Only Server* is a server that is not authoritative for any domain. This server services queries and asks other servers, who have the authority, for the information needed. All servers keep data in their cache until the data expires, based on a time to live field attached to the data when it is received from another server.

3.3. Remote Server

A Remote Server is an option given to people who would like to use a name server on their workstation or on a machine that has a limited amount of memory and CPU cycles. With this option you can run all of the networking programs that use the name server without the name server running on the local machine. All of the queries are serviced by a name server that is running on another machine on the network.

3.4. Slave Server

A Slave Server is a server that always forwards queries it cannot satisfy locally to a fixed list of *forwarding* servers instead of interacting with the master nameservers for the root and other domains. The queries to the *forwarding servers* are recursive queries. There may be one or more forwarding servers, and they are tried in turn until the list is exhausted. A Slave and forwarder configuration is typically used when you do not wish all the servers at a give site to be interacting with the rest of the Internet servers. A typically scenario would involve a number of workstations and a departmental timesharing machine with Internet access. The workstations might be administratively prohibited from having Internet access. To give the workstations the appearance of access to the Internet domain system, the workstations could be Slave servers to the timesharing machine which would forward the queries and interact with other nameservers to resolve the query before returning the answer. An added benefit of using the forwarding feature is that the central machine develops a much more complete cache of information that all the workstations can take advantage of. Slave mode and forwarding is discussed further under the description of the named bootfile commands.

4. Setting up Your Own Domain

When setting up a domain that is going to be on a public network the site administrator should contact the organization in charge of the network and request the appropriate domain registration form. An organization that belongs to multiple networks (such as *CSNET*, *DARPA Internet* and *BITNET*) should register with only one network.

The contacts are as follows:

4.1. DARPA Internet

Sites that are already on the DARPA Internet and need information on setting up a domain should contact HOSTMASTER@NIC.DDN.MIL. You may also want to be placed on the BIND mailing list, which is a mail group for people on the DARPA Internet running BIND. The group discusses future design decisions, operational problems, and other related topics. The address to request being placed on this mailing list is:

bind-request@ucbarpa.Berkeley.EDU.

4.2. CSNET

A *CSNET* member organization that has not registered its domain name should contact the *CSNET* Coordination and Information Center (*CIC*) for an application and information about setting up a domain.

An organization that already has a registered domain name should keep the *CIC* informed about how it would like its mail routed. In general, the *CSNET* relay will prefer to send mail via

CSNET (as opposed to *BITNET* or the *Internet*) if possible. For an organization on multiple networks, this may not always be the preferred behavior. The *CIC* can be reached via electronic mail at *cic@sh.cs.net*, or by phone at (617) 873-2777.

4.3. BITNET

If you are on the BITNET and need to set up a domain, contact INFO@BITNIC.

5. Files

The name server uses several files to load its data base. This section covers the files and their formats needed for *named*.

5.1. Boot File

This is the file that is first read when *named* starts up. This tells the server what type of server it is, which zones it has authority over and where to get its initial data. The default location for this file is */etc/named.boot*. However this can be changed by setting the *BOOTFILE* variable when you compile *named* or by specifying the location on the command line when *named* is started up.

5.1.1. Domain

A default domain may be specified for the nameserver using a line such as

domain Berkeley.Edu

The name server uses this information when it receives a query for a name without a ''**.**'' that is not known. When it receives one of these queries, it appends the name in the second field to the query name. This is an obsolete facility which will be removed from future releases.

5.1.2. Directory

The directory line specifies the directory in which the nameserver should run, allowing the other file names in the boot file to use relative path names.

directory /usr/local/domain

If you have more than a couple of named files to be maintained, you may wish to place the named files in a directory such as /usr/local/domain and adjust the directory command properly. The main purposes of this command are to make sure named is in the proper directory when trying to include files by relative path names with $Include and to allow named to run in a location that is reasonable to dump core if it feels the urge.

5.1.3. Primary Master

The line in the boot file that designates the server as a primary server for a zone looks as follows:

primary Berkeley.Edu ucbhosts

The first field specifies that the server is a primary one for the zone stated in the second field. The third field is the name of the file from which the data is read.

5.1.4. Secondary Master

The line for a secondary server is similar to the primary except that it lists addresses of other servers (usually primary servers) from which the zone data will be obtained.

secondary Berkeley.Edu 128.32.0.10 128.32.0.4 ucbhosts.bak

The first field specifies that the server is a secondary master server for the zone stated in the second field. The two network addresses specify the name servers that are primary for the zone. The secondary server gets its data across the network from the listed servers. Each server is tried in the order listed until it successfully receives the data from a listed server. If a filename is present after the list of primary servers, data for the zone will be dumped into that file as a backup. When the server is first started, the data are loaded from the backup file if possible, and a primary server is then consulted to check that the zone is still up-to-date.

5.1.5. Caching Only Server

You do not need a special line to designate that a server is a caching server. What denotes a caching only server is the absence of authority lines, such as *secondary* or *primary* in the boot file.

All servers should have a line as follows in the boot file to prime the name servers cache:

cache **.** *root*.*cache*

All cache files listed will be read in at named boot time and any values still valid will be reinstated in the cache and the root nameserver information in the cache files will always be used. For information on cache file see section on *Cache Initialization.*

5.1.6. Forwarders Any server can make use of *forwarders*. A *forwarder* is another server capable of processing recursive queries that is willing to try resolving queries on behalf of other systems. The *forwarders* command specifies forwarders by internet address as follows:

forwarders *128.32.0.10* *128.32.0.4*

There are two main reasons for wanting to do so. First, the other systems may not have full network access and may be prevented from sending any IP packets into the rest of the network and therefore must rely on a forwarder which does have access to the full net. The second reason is that the forwarder sees a union of all queries as they pass through his server and therefore he builds up a very rich cache of data compared to the cache in a typical workstation nameserver. In effect, the *forwarder* becomes a meta-cache that all hosts can benefit from, thereby reducing the total number of queries from that site to the rest of the net.

5.1.7. Slave Mode

Slave mode is used if the use of forwarders is the only possible way to resolve queries due to lack of full net access or if you wish to prevent the nameserver from using other than the listed forwarders. Slave mode is activated by placing the simple command

slave

in the bootfile. If *slave* is used, then you must specify forwarders. When in slave mode, the server will forward each query to each of the forwarders until an answer is found or the list of forwarders is exhausted.

5.1.8. Remote Server

To set up a host that will use a remote server instead of a local server to answer queries, the file */etc/resolv.conf* needs to be created. This file designates the name servers on the network that should be sent queries. It is not advisable to create this file if you have a local server running. If this file exists it is read almost every time *gethostbyname()* or *gethostbyaddr()* is called.

5.2. Cache Initialization

5.2.1. root.cache

The name server needs to know the servers that are the authoritative name servers for the root domain of the network. To do this we have to prime the name server's cache with the addresses of these higher authorities. The location of this file is specified in the boot file. This file uses the Standard Resource Record Format (aka. Masterfile Format) covered further on in this paper.

5.3. Domain Data Files

There are three standard files for specifying the data for a domain. These are *named.local*, *hosts* and *host.rev*. These files use the Standard Resource Record Format covered later in this paper.

5.3.1. named.local

This file specifies the address for the local loopback interface, better known as *localhost* with the network address 127.0.0.1. The location of this file is specified in the boot file.

5.3.2. hosts

This file contains all the data about the machines in this zone. The location of this file is specified in the boot file.

5.3.3. hosts.rev

This file specifies the IN-ADDR.ARPA domain. This is a special domain for allowing address to name mapping. As internet host addresses do not fall within domain boundaries, this special domain was formed to allow inverse mapping. The IN-ADDR.ARPA domain has four labels preceding it. These labels correspond to the 4 octets of an Internet address. All four octets must be specified even if an octets is zero. The Internet address 128.32.0.4 is located in the domain 4.0.32.128.IN-ADDR.ARPA. This reversal of the address is awkward to read but allows for the natural grouping of hosts in a network.

5.4. Standard Resource Record Format

The records in the name server data files are called resource records. The Standard Resource Record Format (RR) is specified in RFC1035. The following is a general description of these records:

{name} *{ttl}* *addr-class* *Record Type* *Record Specific data*

Resource records have a standard format shown above. The first field is always the name of the domain record and it must always start in column 1. For some RR's the name may be left blank; in that case it takes on the name of the previous RR. The second field is an optional time to live field. This specifies how long this data will be stored in the data base. By leaving this field blank the default time to live is specified in the *Start Of Authority* resource record (see below). The third field is the address class; currently, only one class is supported: *IN* for internet addresses and other information. The fourth field states the type of the resource record. The fields after that are dependent on the type of the RR. Case is preserved in names and data fields when loaded into the name server. All comparisons and lookups in the name server data base are case insensitive.

The following characters have special meanings:

. A free standing dot in the name field refers to the current domain.

@ A free standing @ in the name field denotes the current origin.

.. Two free standing dots represent the null domain name of the root when used in the name field.

\X Where X is any character other than a digit (0-9), quotes that character so that its special meaning does not apply. For example, ‘‘\.’’ can be used to place a dot character in a label.

\DDD Where each D is a digit, is the octet corresponding to the decimal number described by DDD. The resulting octet is assumed to be text and is not checked for special meaning.

( ) Parentheses are used to group data that crosses a line. In effect, line terminations are not recognized within parentheses.

; Semicolon starts a comment; the remainder of the line is ignored.

* An asterisk signifies wildcarding.

Most resource records will have the current origin appended to names if they are not terminated by a ‘‘**.**’’. This is useful for appending the current domain name to the data, such as machine names, but may cause problems where you do not want this to happen. A good rule of thumb is that, if the name is not in the domain for which you are creating the data file, end the name with a ‘‘**.**’’.

5.4.1. $INCLUDE

An include line begins with $INCLUDE, starting in column 1, and is followed by a file name. This feature is particularly useful for separating different types of data into multiple files. An example would be:

```
$INCLUDE /usr/named/data/mailboxes
```

The line would be interpreted as a request to load the file */usr/named/data/mailboxes*. The $INCLUDE command does not cause data to be loaded into a different zone or tree. This is simply a way to allow data for a given zone to be organized in separate files. For example, mailbox data might be kept separately from host data using this mechanism.

5.4.2. $ORIGIN

The origin is a way of changing the origin in a data file. The line starts in column 1, and is followed by a domain origin. This is useful for putting more then one domain in a data file.

5.4.3. SOA - Start Of Authority

```
name   {ttl}   addr-class   SOA       Origin                   Person in charge
@              IN           SOA       ucbvax.Berkeley.Edu.     kjd.ucbvax.Berkeley.Edu. (
                            1 . 1     ; Serial
                            10800     ; Refresh
                            1800      ; Retry
                            3600000   ; Expire
                            86400 )   ; Minimum
```

The *Start of Authority, SOA,* record designates the start of a zone. The name is the name of the zone. Origin is the name of the host on which this data file resides. Person in charge is the mailing address for the person responsible for the name server. The serial number is the version number of this data file, this number should be incremented whenever a change is made to the data. The name server cannot handle numbers over 9999 after the decimal point. The refresh indicates how often, in seconds, a secondary name servers is to check with the primary name server to see if an update is needed. The retry indicates how long, in seconds, a secondary server is to retry after a failure to check for a refresh. Expire is the upper limit, in seconds, that a secondary name server is to use the data before it expires for lack of getting a refresh. Minimum is the default number of seconds to be used for the time to live field on resource records. There should only be one *SOA* record per zone.

5.4.4. NS - Name Server

```
{name}   {ttl}   addr-class   NS   Name servers name
                 IN           NS   ucbarpa.Berkeley.Edu.
```

The *Name Server* record, *NS,* lists a name server responsible for a given domain. The first name field lists the domain that is serviced by the listed name server. There should be one *NS* record for each Primary Master server for the domain.

5.4.5. A - Address

```
{name}    {ttl}   addr-class   A   address
ucbarpa           IN           A   128.32.0.4
                  IN           A   10.0.0.78
```

The *Address* record, *A*, lists the address for a given machine. The name field is the machine name and the address is the network address. There should be one *A* record for each address of the machine.

5.4.6. HINFO - Host Information

{name}	*{ttl}*	*addr-class*	*HINFO*	*Hardware*	*OS*
		IN	HINFO	VAX-11/780	UNIX

Host Information resource record, *HINFO*, is for host specific data. This lists the hardware and operating system that are running at the listed host. It should be noted that only a single space separates the hardware info and the operating system info. If you want to include a space in the machine name you must quote the name. There should be one *HINFO* record for each host.

5.4.7. WKS - Well Known Services

{name}	*{ttl}*	*addr-class*	*WKS*	*address*	*protocol*	*list of services*
		IN	WKS	128.32.0.10	UDP	who route timed domain
		IN	WKS	128.32.0.10	TCP	(echo telnet discard sunrpc sftp uucp-path systat daytime netstat qotd nntp link chargen ftp auth time whois mtp pop rje finger smtp supdup hostnames domain nameserver)

The *Well Known Services* record, *WKS*, describes the well known services supported by a particular protocol at a specified address. The list of services and port numbers come from the list of services specified in */etc/services.* There should be only one *WKS* record per protocol per address.

5.4.8. CNAME - Canonical Name

aliases	*{ttl}*	*addr-class*	*CNAME*	*Canonical name*
ucbmonet		IN	CNAME	monet

Canonical Name resource record, *CNAME*, specifies an alias for a canonical name. An alias should be the only record associated with the alias name; all other resource records should be associated with the canonical name and not with the alias. Any resource records that include a domain name as their value (e.g. NS or MX) should list the canonical name, not the alias.

5.4.9. PTR - Domain Name Pointer

name	*{ttl}*	*addr-class*	*PTR*	*real name*
7.0		IN	PTR	monet.Berkeley.Edu.

A *Domain Name Pointer* record, *PTR*, allows special names to point to some other location in the domain. The above example of a *PTR* record is used in setting up reverse pointers for the special *IN-ADDR.ARPA* domain. This line is from the example *hosts.rev* file. *PTR* names should be unique to the zone.

5.4.10. MB - Mailbox

name	*{ttl}*	*addr-class*	*MB*	*Machine*
miriam		IN	MB	vineyd.DEC.COM.

MB is the *Mailbox* record. This lists the machine where a user wants to receive mail. The name field is the users login; the machine field denotes the machine to which mail is to be delivered. Mail Box names should be unique to the zone. (These records are currently for experimental use only.)

5.4.11. MR - Mail Rename Name

name	*{ttl}*	*addr-class*	*MR*	*corresponding MB*
Postmistress		IN	MR	miriam

Mail Rename, MR, can be used to list aliases for a user. The name field lists the alias for the name listed in the fourth field, which should have a corresponding *MB* record. (These records are currently for experimental use only.)

5.4.12. MINFO - Mailbox Information

name	*{ttl}*	*addr-class*	*MINFO*	*requests*	*maintainer*
BIND		IN	MINFO	BIND-REQUEST	kjd.Berkeley.Edu.

Mail Information record, *MINFO,* creates a mail group for a mailing list. This resource record is usually associated with a mail group *Mail Group*, but may be used with a *Mail Box* record. The *name* specifies the name of the mailbox. The *requests* field is where mail such as requests to be added to a mail group should be sent. The *maintainer* is a mailbox that should receive error messages. This is particularly appropriate for mailing lists when errors in members names should be reported to a person other than the sender. (These records are currently for experimental use only.)

5.4.13. MG - Mail Group Member

{mail group name}	*{ttl}*	*addr-class*	*MG*	*member name*
		IN	MG	Bloom

Mail Group, MG lists members of a mail group. (These records are currently for experimental use only.)

An example for setting up a mailing list is as follows:

Bind	IN	MINFO	Bind-Request	kjd.Berkeley.Edu.
	IN	MG	Ralph.Berkeley.Edu.	
	IN	MG	Zhou.Berkeley.Edu.	
	IN	MG	Painter.Berkeley.Edu.	
	IN	MG	Riggle.Berkeley.Edu.	
	IN	MG	Terry.pa.Xerox.Com.	

5.4.14. MX - Mail Exchanger

name	*{ttl}*	*addr-class*	*MX*	*preference value*	*mailer exchanger*
Munnari.OZ.AU.		IN	MX	0	Seismo.CSS.GOV.
*.IL.		IN	MX	0	RELAY.CS.NET.

Mail Exchanger records, *MX,* are used to specify a machine that knows how to deliver mail to a machine that is not directly connected to the network. In the first example, above, Seismo.CSS.GOV. is a mail gateway that knows how to deliver mail to Munnari.OZ.AU. but other machines on the network can not deliver mail directly to Munnari. These two machines may have a private connection or use a different transport medium. The preference value is the order that a mailer should follow when there is more then one way to deliver mail to a single machine. See RFC974 for more detailed information.

Wildcard names containing the character ‘‘*’’ may be used for mail routing with *MX* records. There are likely to be servers on the network that simply state that any mail to a domain is to be routed through a relay. Second example, above, all mail to hosts in the domain IL is routed through RELAY.CS.NET. This is done by creating a wildcard resource record, which states that *.IL has an *MX* of RELAY.CS.NET.

5.5. Sample Files

The following section contains sample files for the name server. This covers example boot files for the different types of servers and example domain data base files.

5.5.1. Boot File

5.5.1.1. Primary Master Server

```
;
; Boot file for Primary Master Name Server
;

; type        domain                   source file or host
;
directory     /usr/local/domain
primary       Berkeley.Edu             ucbhosts
primary       32.128.in-addr.arpa      ucbhosts.rev
primary       0.0.127.in-addr.arpa     named.local
cache         .                        root.cache
```

5.5.1.2. Secondary Master Server

```
;
; Boot file for Secondary Master Name Server
;

; type        domain                   source file or host
;
directory     /usr/local/domain
secondary     Berkeley.Edu             128.32.0.4 128.32.0.10 ucbhosts.bak
secondary     32.128.in-addr.arpa      128.32.0.4 128.32.0.10 ucbhosts.rev.bak
primary       0.0.127.in-addr.arpa     named.local
cache         .                        root.cache
```

5.5.1.3. Caching Only Server

```
;
; Boot file for Caching Only Name Server
;

; type        domain                   source file or host
;
directory     /usr/local/domain
cache         .                        root.cache
primary       0.0.127.in-addr.arpa     /etc/named.local
```

5.5.2. Remote Server

5.5.2.1. /etc/resolv.conf

```
domain Berkeley.Edu
nameserver 128.32.0.4
nameserver 128.32.0.10
```

5.5.3. root.cache

```
;
;
; Initial cache data for root domain servers.
;

.                       99999999  IN  NS  NS.NIC.DDN.MIL.
                        99999999  IN  NS  NS.NASA.GOV.
                        99999999  IN  NS  TERP.UMD.EDU.
                        99999999  IN  NS  A.ISI.EDU.
                        99999999  IN  NS  AOS.BRL.MIL.
                        99999999  IN  NS  GUNTER-ADAM.AF.MIL.
                        99999999  IN  NS  C.NYSER.NET.

;  Prep the cache (hotwire the addresses).
ns.NIC.DDN.MIL.         99999999  IN  A   192.67.67.53
NS.NASA.GOV.            99999999  IN  A   128.102.16.10
NS.NASA.GOV.            99999999  IN  A   192.52.195.10
A.ISI.EDU.              99999999  IN  A   26.3.0.103
A.ISI.EDU.              99999999  IN  A   128.9.0.107
AOS.BRL.MIL.            99999999  IN  A   128.20.1.2
AOS.BRL.MIL.            99999999  IN  A   192.5.25.82
GUNTER-ADAM.AF.MIL.     99999999  IN  A   26.1.0.13
C.NYSER.NET.            99999999  IN  A   192.33.4.12
TERP.UMD.EDU.           99999999  IN  A   128.8.10.90
```

5.5.4. named.local

```
@   IN  SOA   ucbvax.Berkeley.Edu. kjd.ucbvax.Berkeley.Edu. (
              1       ; Serial
              10800
              1800
              3600000
              86400 )
    IN  NS    ucbvax.Berkeley.Edu.
1   IN  PTR   localhost.
```

5.5.5. Hosts

```
;
;	@(#)ucb-hosts	1.2	(berkeley)	88/02/05
;
@		IN	SOA	ucbvax.Berkeley.Edu. kjd.monet.Berkeley.Edu. (
				1.2	; Serial
				10800	; Refresh
				1800	; Retry
				3600000	; Expire
				86400 )	; Minimum
		IN	NS	ucbarpa.Berkeley.Edu.
		IN	NS	ucbvax.Berkeley.Edu.
localhost	IN	A	127.1
ucbarpa		IN	A	128.32.4
		IN	A	10.0.0.78
		IN	HINFO	VAX-11/780 UNIX
arpa		IN	CNAME	ucbarpa
ernie		IN	A	128.32.6
		IN	HINFO	VAX-11/780 UNIX
ucbernie	IN	CNAME	ernie
monet		IN	A	128.32.7
		IN	A	128.32.130.6
		IN	HINFO	VAX-11/750 UNIX
ucbmonet	IN	CNAME	monet
ucbvax		IN	A	10.2.0.78
		IN	A	128.32.10
		IN	HINFO	VAX-11/750 UNIX
		IN	WKS	128.32.0.10 UDP syslog route timed domain
		IN	WKS	128.32.0.10 TCP ( echo telnet
				discard sunrpc sftp
				uucp-path systat daytime
				netstat qotd nntp
				link chargen ftp
				auth time whois mtp
				pop rje finger smtp
				supdup hostnames
				domain
				nameserver )
vax		IN	CNAME	ucbvax
toybox		IN	A	128.32.131.119
		IN	HINFO	Pro350 RT11
toybox		IN	MX	0 monet.Berkeley.Edu
miriam		IN	MB	vineyd.DEC.COM.
postmistress	IN	MR	Miriam
Bind		IN	MINFO	Bind-Request kjd.Berkeley.Edu.
		IN	MG	Ralph.Berkeley.Edu.
		IN	MG	Zhou.Berkeley.Edu.
		IN	MG	Painter.Berkeley.Edu.
		IN	MG	Riggle.Berkeley.Edu.
		IN	MG	Terry.pa.Xerox.Com.
```

5.5.6. host.rev

```
;
;	@(#)ucb-hosts.rev	1.1	(Berkeley)	86/02/05
;
@	IN	SOA	ucbvax.Berkeley.Edu. kjd.monet.Berkeley.Edu. (
			1.1	; Serial
			10800	; Refresh
			1800	; Retry
			3600000	; Expire
			86400 )	; Minimum
	IN	NS	ucbarpa.Berkeley.Edu.
	IN	NS	ucbvax.Berkeley.Edu.
0.0	IN	PTR	Berkeley-net.Berkeley.EDU.
	IN	A	255.255.255.0
0.130	IN	PTR	csdiv-net.Berkeley.EDU.
4.0	IN	PTR	ucbarpa.Berkeley.Edu.
6.0	IN	PTR	ernie.Berkeley.Edu.
7.0	IN	PTR	monet.Berkeley.Edu.
10.0	IN	PTR	ucbvax.Berkeley.Edu.
6.130	IN	PTR	monet.Berkeley.Edu.
```

6. Domain Management

This section contains information for starting, controlling and debugging *named*.

6.1. /etc/rc.local

The hostname should be set to the full domain style name in */etc/rc.local* using *hostname (1)*. The following entry should be added to */etc/rc.local* to start up *named* at system boot time:

```
if [ -f /etc/named ]; then
	/etc/named [options] & echo -n ' named'	>/dev/console
fi
```

This usually directly follows the lines that start *syslogd*. **Do Not** attempt to run *named* from *inetd*. This will continuously restart the name server and defeat the purpose of having a cache.

6.2. /etc/named.pid

When *named* is successfully started up it writes its process id into the file */etc/named.pid*. This is useful to programs that want to send signals to *named*. The name of this file may be changed by defining *PIDFILE* to the new name when compiling *named*.

6.3. /etc/hosts

The *gethostbyname ()* library call can detect if *named* is running. If it is determined that *named* is not running it will look in */etc/hosts* to resolve an address. This option was added to allow *ifconfig (8C)* to configure the machines local interfaces and to enable a system manager to access the network while the system is in single user mode. It is advisable to put the local machines interface addresses and a couple of machine names and address in */etc/hosts* so the system manager can rcp files from another machine when the system is in single user mode. The format of */etc/host* has not changed. See *hosts (5)* for more information. Since the process of reading */etc/hosts* is slow, it is not advised to use this option when the system is in multi user mode.

6.4. Signals

There are several signals that can be sent to the *named* process to have it do tasks without restarting the process.

6.4.1. Reload

SIGHUP - Causes *named* to read *named.boot* and reload the database. All previously cached data is lost. This is useful when you have made a change to a data file and you want *named*'s internal database to reflect the change.

6.4.2. Debugging

When *named* is running incorrectly, look first in */usr/adm/messages* and check for any messages logged by *syslog*. Next send it a signal to see what is happening.

SIGINT - Dumps the current data base and cache to */usr/tmp/named_dump.db* This should give you an indication to whether the data base was loaded correctly. The name of the dump file may be changed by defining *DUMPFILE* to the new name when compiling *named*.

Note: the following two signals only work when *named* is built with *DEBUG* defined.

SIGUSR1 - Turns on debugging. Each following USR1 increments the debug level. The output goes to */usr/tmp/named.run* The name of this debug file may be changed by defining *DEBUGFILE* to the new name before compiling *named*.

SIGUSR2 - Turns off debugging completely.

For more detailed debugging, define DEBUG when compiling the resolver routines into */lib/libc.a*.

ACKNOWLEDGEMENTS

Many thanks to the users at U.C. Berkeley for falling into many of the holes involved with integrating BIND into the system so that others would be spared the trauma. I would also like to extend gratitude to Jim McGinness and Digital Equipment Corporation for permitting me to spend most of my time on this project.

Ralph Campbell, Doug Kingston, Craig Partridge, Smoot Carl-Mitchell, Mike Muuss and everyone else on the DARPA Internet who has contributed to the development of BIND. To the members of the original BIND project, Douglas Terry, Mark Painter, David Riggle and Songnian Zhou.

Anne Hughes, Jim Bloom and Kirk McKusick and the many others who have reviewed this paper giving considerable advice.

This work was sponsored by the Defense Advanced Research Projects Agency (DoD), Arpa Order No. 4871 monitored by the Naval Electronics Systems Command under contract No. N00039-84-C-0089. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the Defense Research Projects Agency, of the US Government, or of Digital Equipment Corporation.

REFERENCES

[Birrell] Birrell, A. D., Levin, R., Needham, R. M., and Schroeder, M.D., ‘‘Grapevine: An Exercise in Distributed Computing.’’ In *Comm. A.C.M. 25,* 4:260-274 April 1982.

[RFC819] Su, Z. Postel, J., ‘‘The Domain Naming Convention for Internet User Applications.’’ *Internet Request For Comment 819* Network Information Center, SRI International, Menlo Park, California. August 1982.

[RFC974] Partridge, C., ‘‘Mail Routing and The Domain System.’’ *Internet Request For Comment 974* Network Information Center, SRI International, Menlo Park, California. February 1986.

[RFC1032] Stahl, M., ‘‘Domain Administrators Guide’’ *Internet Request For Comment 1032* Network Information Center, SRI International, Menlo Park, California. November 1987.

[RFC1033] Lottor, M., ‘‘Domain Administrators Guide’’ *Internet Request For Comment 1033* Network Information Center, SRI International, Menlo Park, California. November 1987.

[RFC1034] Mockapetris, P., ‘‘Domain Names - Concept and Facilities.’’ *Internet Request For Comment 1034* Network Information Center, SRI International, Menlo Park, California. November 1987.

[RFC1035] Mockapetris, P., ‘‘Domain Names - Implementation and Specification.’’ *Internet Request For Comment 1035* Network Information Center, SRI International, Menlo Park, California. November 1987.

[RFC1101] Mockapetris, P., ‘‘DNS Encoding of Network Names and Other Types.’’ *Internet Request For Comment 1101* Network Information Center, SRI International, Menlo Park, California. April 1989.

[Terry] Terry, D. B., Painter, M., Riggle, D. W., and Zhou, S., *The Berkeley Internet Name Domain Server.* Proceedings USENIX Summer Conference, Salt Lake City, Utah. June 1984, pages 23-31.

[Zhou] Zhou, S., *The Design and Implementation of the Berkeley Internet Name Domain (BIND) Servers.* UCB/CSD 84/177. University of California, Berkeley, Computer Science Division. May 1984.