CONVEX
Internet Services
System Manager's Guide

Order No. DSW-142

First Edition
October 1990

CONVEX Computer Corporation
Richardson, Texas USA

CONVEX
Internet Services
System Manager's Guide

Order No. DSW-142

Printed in the United States of America

Revision Information for

CONVEX
Internet Services
System Manager's Guide

Edition	Document No.	Description
First	710-002430-204	Released with CONVEX Internet Services V9.0, October 1990. This manual was previously entitled *CONVEX Networking Utilities System Manager's Guide.* The manual was updated and reorganized, and new material was added. Major additions include: Chapter 2, "Configuring Network Hardware and Software," describes /ioconfig file entries for network hardware and summarizes network-related boot-time parameters. Chapter 3, "Setting up the Host Name Database," introduces host naming and addressing concepts, and describes how to choose and implement a host naming and addressing structure. Chapter 5, "Installing and Configuring a HYPERchannel Interface," describes how to install, configure, and troubleshoot a HYPERchannel interface. Chapter 6, "Configuring a CONVEX UltraNet Interface," contains updated information. Chapter 9, "Using the Berkeley Internet Name Domain Server," describes how to configure and use BIND. Chapter 10, "Configuring Serial Line Internet Protocol (SLIP)," describes how to configure SLIP and use it for point-to-point communication over serial lines. Chapter 11, "Testing and Troubleshooting the Configured Network," includes information on new `netstat` commands.

Table of Contents

CONVEX Internet Services System Manager's Guide

Configuring an Ethernet Interface

Installing and Configuring a HYPERchannel Interface

Configuring a CONVEX UltraNet Interface

Routing Packets on the Network

Configuring Network Sevices

Using the Berkeley Internet Name Domain Server

Configuring Serial Line Internet Protocol (SLIP)

Testing and Troubleshooting the Configured Network

Appendix A: Network System Files

Appendix B: Reporting Problems

Using This Guide

Purpose and Audience

This document provides system managers or operators with information needed to configure and maintain a TCP/IP Ethernet, HYPERchannel, or UltraNet local area network (LAN). The reader is assumed to be an experienced system manager or operator familiar with the ConvexOS operating system, but not necessarily familiar with network management. Persons unfamiliar with CONVEX Internet Services will benefit from reading *CONVEX Networking Concepts* prior to using this guide.

Organization

This guide is organized as follows:

- Chapter 1, "Introduction," summarizes network management tasks.
- Chapter 2, "Configuring Network Hardware and Software," describes how to integrate network devices into ConvexOS and customize networking boot-time parameters.
- Chapter 3, "Setting Up the Host Name Database," defines host names and addresses, and describes how to create the host name database.
- Chapter 4, "Configuring an Ethernet Interface," describes how to configure an Ethernet interface using `ifconfig` and ARP.
- Chapter 5, "Installing and Configuring a HYPERchannel Interface," describes how to install and configure HYPERchannel hardware and software.
- Chapter 6, "Configuring a CONVEX UltraNet Interface," guides you through the steps used to configure CONVEX UltraNet Interface software.
- Chapter 7, "Routing Packets on the Network," describes how packets are routed on the network and guides you through the steps used to set up Internet routing.
- Chapter 8, "Configuring Network Utilities," guides you through the steps used to configure and start Internet server daemons and utilities.
- Chapter 9, "Using the Berkeley Internet Name Domain Server," describes how to configure systems to use the Berkeley Internet Name Domain Server (BIND).
- Chapter 10, "Configuring Serial Line Internet Protocol (SLIP)," guides you through the steps used to configure the Serial Line Internet Protocol.
- Chapter 11, "Testing and Troubleshooting the Configured Network," describes the use of tools for testing and troubleshooting the network.

- Appendix A, "Network System Files," describes the format and content of system files used by CONVEX Internet Services.
- Appendix B, "Reporting Problems," provides instructions for reporting software or documentation problems to the CONVEX Technical Assistance Center (TAC).

Technical Assistance

If you have questions that are not answered in this book, contact the CONVEX Technical Assistance Center (TAC).

- Within the continental U.S., call 1(800)952-0379.
- From Canada, call 1(800)345-2384.
- All other locations, contact the local CONVEX office.

Associated Documents

Using this software successfully may require more information than is contained in this manual.

CONVEX Computer Corporation provides the following documents to help you with the ConvexOS operating system:

- *CONVEX UNIX Primer* (DSW-133) introduces new users to the ConvexOS operating system.
- *ConvexOS Programmer's Reference* (DSW-332) is the standard reference for the ConvexOS operating system. It contains man pages describing ConvexOS and CONVEX Internet Services software.
- *Managing ConvexOS: Configuration Guide* (DSW-031) and *Managing ConvexOS: Operations Guide* (DSW-030) contain information needed to manage and maintain a CONVEX supercomputer.
- *CONVEX UNIX Tutorial Papers* (DSW-002) is a collection of previously published papers that provide instruction in document preparation, programming, text editing, supporting tools and languages, system maintenance, and system implementation.

CONVEX Computer Corporation provides the following documents to help you with CONVEX Internet Services:

- *CONVEX Networking Concepts* (DSW-128) provides an overview of all CONVEX networking products.
- *CONVEX Internet Services User's Guide* (DSW-141) explains how to use CONVEX Internet Services to get work done on a day-to-day basis. In particular, it describes the network utilities that enable you to log in to remote systems, execute commands on different machines, and transfer files across the network.
- *CONVEX Interprocess Communication (IPC) Programming Guide* (DSW-143) provides application programmers with the necessary information to develop network applications.

Ordering Documentation

To order the current edition of this or any other CONVEX document, send requests to:

CONVEX Computer Corporation
Customer Service
P.O. Box 833851
Richardson, TX 75083-3851 USA

Include the order number or the exact title, as listed on the front cover.

Notational Conventions

This section discusses notational conventions used in this guide.

Command Syntax

Consider this example:

```
COMMAND input_file [...] {a | b} [output_file]
   ①        ②       ③      ④         ⑤
```

1. **COMMAND** must be typed as it appears.
2. *input_file* indicates a file name that must be supplied by the user.
3. The horizontal ellipsis in brackets indicates that additional input file names may be supplied.
4. Either `a` or `b` must be supplied.
5. *output_file* indicates an optional file name.

General Conventions

In general, the following conventions are used in this guide:

- **`Bold constant-width font`** identifies user input in examples.
- *Italics*
  - Designate user-supplied variables in a command-line example.
  - Introduce new and important terms.
  - Identify variables in mathematical equations.
  - Indicate titles of documents.
- `Constant-width font` is used to designate input and output, including:
  - Command names and options.
  - System calls.
  - Data structures and types.
  - Directives, program statements, display examples, printout examples, and error messages.
- Horizontal ellipsis (...) shows repetition of the preceding item(s).
- Vertical ellipses show that lines of code have been left out of an example.
- Words and abbreviations that indicate keyboard keys you press are identified in a distinctive bold type. For example, **RETURN** refers to the carriage return key. Words separated by a hyphen indicate two keys that you press simultaneously. For example, **CTRL-X** indicates that you press and hold down the **CTRL** key and then press the **X** key.
- The word "enter" in a phrase such as "enter **`ls`**" means that you type the command and then press **RETURN**.
- References to the *ConvexOS Programmer's Reference* appear in the form `adb(1)`, where the name of the man page is followed by its section number enclosed in parentheses.

- The ConvexOS prompt is printed as a percent sign (%). The pound sign (#) signifies the superuser prompt.

Note

A note highlights information that may be of particular interest.

Reader Response

If you have comments or questions about the contents of this book, please notify the CONVEX documentation department by using the Reader's Comments form at the end of this document.

Introduction 1

This guide describes optional software products. To acquire the software, you must purchase a CONVEX Internet Services or CONVEX UltraNet Interface license. The license includes both the network software and a current set of release notes and installation procedures.

CONVEX UltraNet Interface software includes the UltraNet device driver and sockets compatibility library. To obtain the full set of utilities discussed in this book, you must purchase a CONVEX Internet Services License in addition to a CONVEX UltraNet Interface license.

This chapter lists prerequisites to configuring CONVEX Internet Services or CONVEX UltraNet Interface software and presents an overview of tasks used to configure and test the network. Subsequent chapters explain how to perform these tasks.

Prerequisites

Network software includes communication protocols, socket commands, utilities, and device drivers. You install network software when you install ConvexOS and Utilities, as outlined in the installation procedure you receive when you purchase a CONVEX Internet Services or CONVEX UltraNet Interface license. This book assumes you have already installed network hardware and software and are now ready to configure the network. Therefore, before beginning network configuration, make sure that you can answer "yes" to the following questions:

1. Has the hardware (cables, controllers, adapters) been completely installed and tested?
2. Have you run diagnostics on the SPU? In particular, if you have installed an Ethernet, has the Ethernet Controller Functional Test (`dev4500`) run successfully? (Instructions for running diagnostic tests are included in the *CONVEX Processor Diagnostics Manual*.)
3. Have you started the system in multi-user mode without problems?

Task Summary

Other network software components include commands, daemons, and system files. You use these commands to define network interfaces, assign host names and addresses, test the network, and set up routing; you configure daemons to service network communication requests; and you modify system files to reflect network configuration.

To install and configure CONVEX Internet Services, complete the tasks listed below.

1. Install network hardware and integrate it into ConvexOS by editing the /ioconfig file on the SPU.
2. Install network software according to the installation procedure shipped with the software.
3. Customize boot-time parameters for your particular network interface.
4. Assign local host names and internet addresses and set up the host name database.
5. Configure network interfaces.
6. Optionally, configure the Serial Line Internet Protocol (SLIP).
7. Configure and start network utilities.
8. Set up and enable routing.
9. Optionally, configure the Berkeley Internet name server (BIND).
10. Test the configured network and troubleshoot problems if necessary.

Subsequent chapters describe these tasks in detail.

Configuring Network Hardware and Software

2

When you add network hardware to your system, you must integrate the new device into ConvexOS and customize the operating system for the interface's particular characteristics. This chapter describes how to integrate network devices into ConvexOS and define parameters for the operating system to use when the system is booted.

Adding CONVEX supported devices is primarily a hardware task. Refer to the documentation for the hardware you are adding, as well as the *CONVEX Guide to Attaching Multibus Peripherals*, and the *CONVEX VMEbus Service Documentation Kit* if you are adding VME peripherals.

If you are adding a device that is not supported by CONVEX, refer to the *CONVEX Guide to Writing Device Drivers*. This guide is part of the optional User Written Device Drivers package; it includes instructions about writing and installing device drivers and generating a new operating system.

Managing ConvexOS: Configuration Guide describes device configuration in detail. You can find supplemental information in the man pages for `ex(4)`, `hy(4)`, `hv(4)`, and `uv(4)`, and in the *Release Notice, CONVEX Internet Services*.

The /ioconfig File

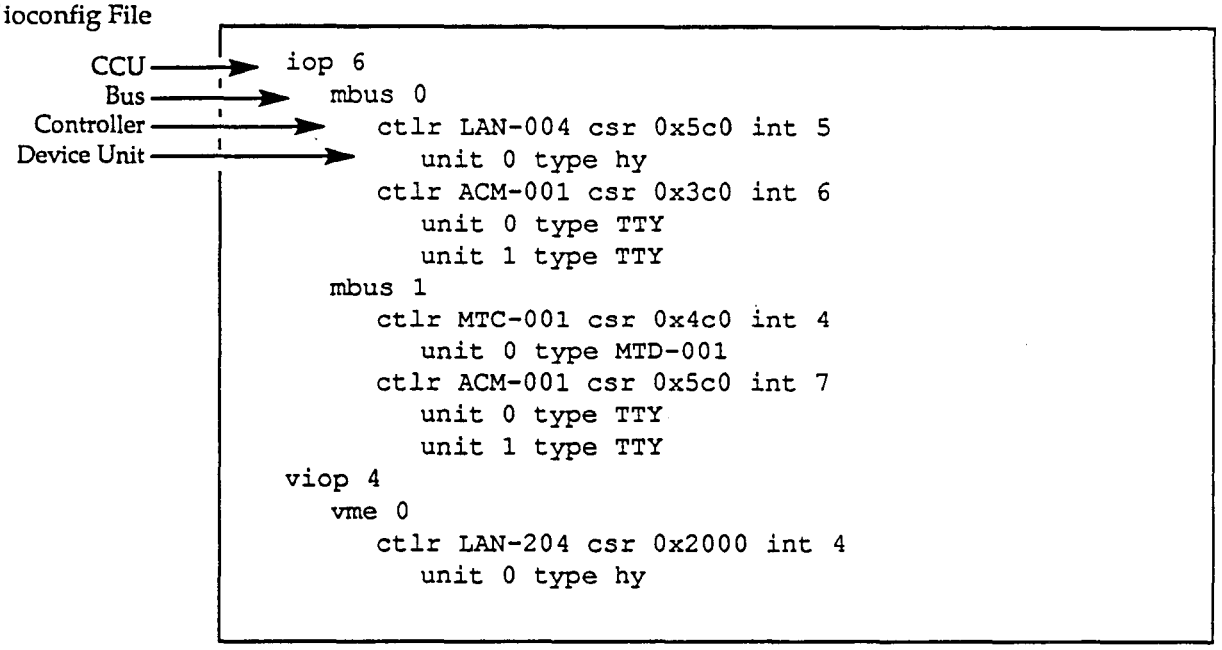
The /ioconfig file contains a description of all the Channel Control Units (CCUs), interfaces, controller boards, and peripheral devices for your system. The boot process reads this file to determine what devices are present.

The /ioconfig file is located on the Service Processor Unit (SPU) disk. To display this file, enter the following command:

```
spu -r /ioconfig
```

Figure 2-1 on the following page shows a sample /ioconfig file.

Figure 2-1
Example /ioconfig File



The /ioconfig file contains an entry for each CCU connected to the system. Each entry contains several levels of information, usually distinguished by indention levels for readability. The following sections describe the different levels of information in the /ioconfig file.

CCU Description

Beginning at the left margin, the CCU description identifies the CCU type and slot number. Supported CCU types for network devices include:

- IOP—For multibus Ethernet and HYPERchannel interfaces.
- VIOP—For VME Ethernet, HYPERchannel, and UltraNet interfaces.

Slot numbers range from 0 to 7, corresponding to the I/O slot to which the CCU is connected.

Bus or Interface Description

Information at the first level of indention identifies the type and chassis number of the bus or interface. Supported types for network interfaces are:

- Multibus (mbus) for IOP-type CCUs.
- VMEbus (vme) for VIOP-type CCUs.

IOP- and VIOP-type CCUs each support two chassis, numbered zero and one.

Controller or Driver Description

At the second level of indention, the controller description specifies I/O controller type, unique control and status register (CSR) address, and interrupt level.

The format for a network interface description is:

`ctlr` *type* `csr` *address* `int` *level*

where

`ctlr` *type*	Type of controller. Table 2-1 lists controller types and descriptions.
`csr` *address*	Control and status register (CSR) address in hexadecimal format, corresponding to the board strapping for the specific controller type.
`int` *level*	Controller's interrupt number. It can be any number between 0 and 7.

The CSR address and interrupt level for a controller are provided in the specific controller documentation and in the *CONVEX Guide to Attaching Multibus Peripherals*.

Table 2-1
Network Controller Types

Controller Type	Description
LAN-001	Excelan multibus Ethernet controller
LAN-002	HYPERchannel multibus device driver
LAN-004	HYPERchannel multibus network interface
LAN-007	Excelan VMEbus Ethernet controller
LAN-202	UltraNet VMEbus controller
LAN-204	HYPERchannel VMEbus network interface

Device Unit Description

At the third level of indention, the device unit description specifies device unit number and type.

For IOP- and VIOP-type CCUs, the format for the unit description is:

`unit` *number* `type` *name*

where

`unit` *number* is the number assigned to the unit. Units are numbered sequentially beginning with 0 for each controller type on a bus. The number of units supported by a single controller depends upon the controller type. You can find this information in the documentation for the specific controller.

`type` *name* is the type of unit connected to the controller. Table 2-2 lists unit types and descriptions.

Table 2-2
Network Device Unit Types

Unit Type	Description
ex	Multibus or VMEbus Excelan Ethernet interface
HYP-001	Multibus HYPERchannel device driver
hy	Multibus or VMEbus HYPERchannel network interface
unet	VMEbus UltraNet network interface

Adding Network Interfaces

CONVEX Internet Services provides network interfaces for Ethernet, HYPERchannel, and UltraNet hardware. To add a network interface, include its entry in the /ioconfig file. Interface type determines content of the /ioconfig file entry. The following sections contain examples of /ioconfig file entries for each type of network interface.

Ethernet Interface /ioconfig File Entries

CONVEX Internet Services supports both a multibus and a VMEbus Ethernet network interface. Figure 2-2 shows a /ioconfig file entry for a multibus Excelan Ethernet interface.

Figure 2-2
Multibus Ethernet Network Interface /ioconfig Entry

```
iop 6
   mbus 0
      ctlr LAN-001 csr 0x5c0 int 5
         unit 0 type ex
```

Figure 2-2 shows a /ioconfig file entry for a VMEbus Ethernet network interface.

Figure 2-3
VMEbus Ethernet Network Interface /ioconfig Entry

```
viop 3
   vme 0
      ctlr LAN-007 csr 0x4c0 int 4
         unit 0 type ex
```

HYPERchannel Interface and Device Driver /ioconfig File Entries

CONVEX Internet Services supports both a multibus and a VMEbus HYPERchannel network interface, as well as support for a user-written device driver. Figure 2-4 shows a /ioconfig file entry for a multibus HYPERchannel network interface.

Figure 2-4
Multibus HYPERchannel Network Interface /ioconfig Entry

```
iop 3
   mbus 0
      ctlr LAN-004 csr 0x3c0 int 3
         unit 0 type hy
```

Figure 2-5 shows a /ioconfig file entry for a VMEbus HYPERchannel network interface.

Figure 2-5
VMEbus HYPERchannel Network Interface /ioconfig Entry

```
viop 2
   vme 0
      ctlr LAN-204 csr 0x2000 int 4
         unit 0 type hy
```

Figure 2-6 shows a /ioconfig file entry for a multibus HYPERchannel device driver.

Figure 2-6
Multibus HYPERchannel Device Driver /ioconfig Entry

```
iop 1
   mbus 0
      ctlr LAN-002 csr 0x5c0 int 5
         unit 0 type HYP-001
```

See the *CONVEX Guide to Writing Device Drivers* for information about writing your own drivers.

UltraNet Interface /ioconfig File Entry

The CONVEX UltraNet Interface supports a VMEbus UltraNet network interface. An example /ioconfig file entry is shown in Figure 2-7.

Figure 2-7
VMEbus UltraNet Network Interface /ioconfig Entry

```
viop 1
   vme 0
      ctlr LAN-202 csr 0x7740 int 3
         unit 0 type unet
```

Networking Boot-Time Parameters

This section describes boot-time parameters related to network performance. For a discussion about how to change boot-time parameters, see *Managing ConvexOS: Configuration Guide.*

When you boot your system, the boot process reads a file containing parameters that control the way the operating system handles CPUs and CCUs at your site. You can customize these parameters for the network interface you have installed.

Table 2-3
Networking Boot-Time Parameters

Parameter	Definition
`gateway`	Enables sending Internet Control Message Protocol (ICMP) errors if both of the following conditions are true: ❑ The system has a single network interface, or IP forwarding is disabled (see `ipforwarding` parameter). ❑ The received IP packet is not for the system that has `gateway` enabled. If `gateway` is disabled (set to 0) under these circumstances, errors are not set to the source machine and the packet is dropped. See `ipforwarding`, `ipsendredirects`, and `subnetsarelocal` parameters. default = 0, off = 0, on = 1
`ipforwarding`	Enables Internet protocol (IP) packets to be forwarded. Packets are forwarded when the IP address does not correspond to any of the Internet addresses for the machine's network interfaces. If this parameter is disabled (set to 0), packets can be dropped. See `ipsendredirects`, `gateway`, and `subnetsarelocal` parameters. default = 1, off = 0, on = 1
`ipsendredirects`	Enables sending Internet Control Message Protocol (ICMP) redirects. Redirects inform the sending machine of the direct address of the machine to which packets were forwarded. If this option is disabled (set to 0), redirects are not sent when packets are forwarded. See `ipforwarding`, `gateway`, and `subnetsarelocal` parameters. default = 1, off = 0, on = 1

Parameter	Definition
`subnetsarelocal`	Considers an Internet address local even if it belongs to another subnet. A different network number means the address is remote, that is, accessible through a gateway. This option is used only during determination of the TCP maximum segment size. TCP uses a small maximum segment size (536 bytes) for remote destinations. For local destinations, TCP uses the maximum transmission unit of outgoing network interface. See `ipforwarding`, `ipsendredirects`, and `gateway` parameters. default = 1, off = 0, on = 1
`udpcksum`	Enables checksumming of User Datagram Protocol (UDP) datagrams. UDP checksumming incurs substantial overhead because each transmitted and received UDP datagram is checksummed. Turning the parameter off increases the risk of allowing bad packets farther up in the protocols. default = 0, off = 0, on = 1
`viop_enet_proc`	Sets the number of send and receive processes for a particular VIOP Ethernet interface. This parameter is used to reduce the amount of memory needed by the VIOP when servicing multiple Ethernet interfaces. While lowering the number of processes helps the VIOP avoid running out of memory, it also lowers network performance, so you should decrease the number of processes only if your system fails to boot because of insufficient memory. default = 4 processes per Ethernet interface

Setting Up the Host Name Database

3

This chapter describes how to set up a host name database for your local network. It contains three sections:

- An introduction to host naming and addressing concepts, as well as descriptions of ways to represent host names and addresses.
- A discussion on choosing an address structure.
- A step-by-step guide to creating the host name database.

Use the procedure described in this chapter to set up a host name database for a TCP/IP network, that is, a network that uses DARPA Internet protocols. The CONVEX UltraNet Interface, which uses its own protocols in addition to TCP/IP and has its own naming and addressing requirements, is discussed further in Chapter 6, "Configuring a CONVEX UltraNet Interface."

Readers familiar with internet naming and addressing may want to skip to the section, "Creating the Host Name Database," which guides you through steps for implementing an address structure.

Identifying Hosts on a Network

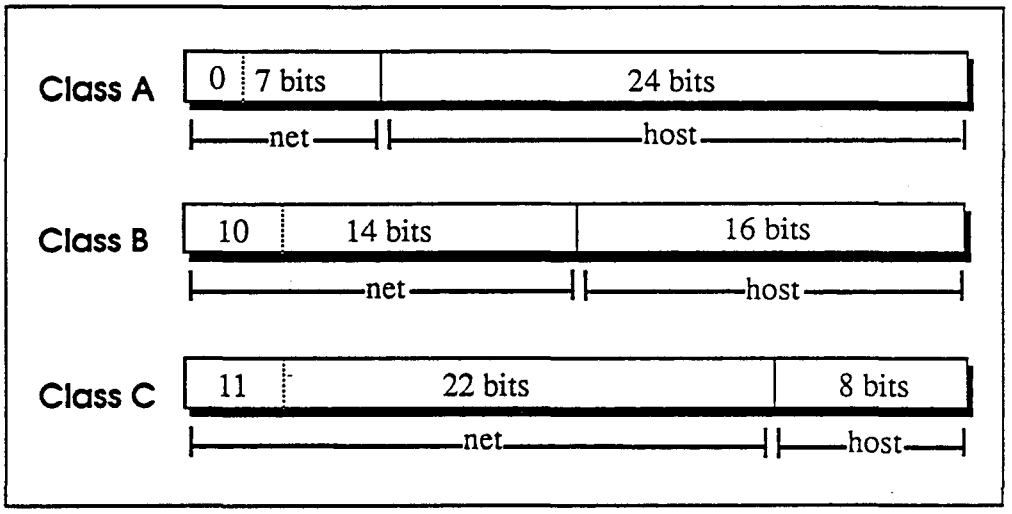
Internet addresses, also known as *network addresses* or *host addresses*, are numeric, universal identifiers assigned to machines on a TCP/IP network. While software works well with these compact addresses, people find them cumbersome and difficult to remember. *Host names* also identify machines on a network, but in a text form most people find easier to use. Network software translates host names to internet addresses before using them internally.

Internal Representation of Internet Addresses

Internet addresses are expressed as 32-bit values representing two fields: *network number*, which includes *address class*, and *host number*. Address class determines the size (often called *address space* or *name space*) of the network in terms of how many hosts it can support. Network number identifies the interface used by the host to communicate on the network. Host number refers to a specific machine on the network interface.

As Figure 3-1 on the following page illustrates, distribution of the 32 bits of address depends upon the address class defined in the high-order bits.

Figure 3-1
Internet Addresses by Class



Users typically designate hosts by name, and need not work with addresses. When configuring the network, however, the system manager must work with both names and addresses.

Dot Notation

In situations where you identify hosts or networks by address, as in the /etc/hosts and /etc/networks files, you normally use *dot notation*. In dot notation, you specify an address as a series of decimal numbers (usually four) separated by periods, as in

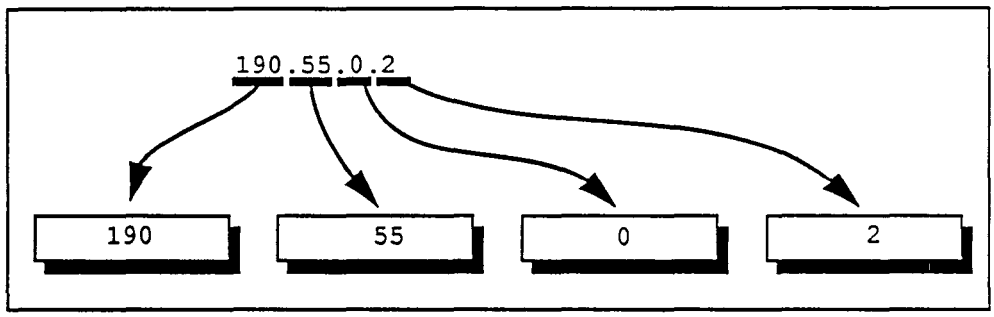
```
128.50.10.1
```

Note

In addition to dot notation, you can specify internet addresses as decimal, octal, or hexadecimal numbers. Use a leading 0x or 0X to signify a hexadecimal address, and a leading 0 to signify an octal address. Numbers without prefixes are interpreted as decimal.

When the address is specified in four parts, each 8-bit value is encoded into the corresponding octet of the internet address. Figure 3-2 illustrates this correspondence.

Figure 3-2
Four-part Dot Notation Address



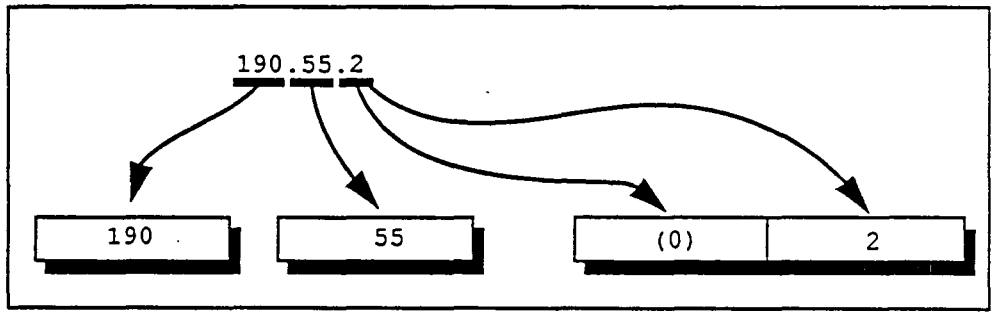
Address `190.55.0.2` has binary value `10` in its address class to show that it is a class B address. Network software interprets the first two octets of a class B address as the network number and the second two octets as the host number.

Three-part dot notation conveniently represents class B host addresses in the form,

```
net.net.host
```

Figure 3-2 shows how host address, `190.55.2`, is encoded into a four-octet internet address.

Figure 3-3
Three-part Dot Notation Address



As you can see from the previous two figures, internal representation of host address `190.55.2` is equivalent to that of host address `190.55.0.2`.

Note

If you specify a network address in three-part dot notation, as in /etc/networks, the address will be left-justified in four octets. For example, for network address 190.50.5, the first octet receives 190, the second receives 50, the third receives 5, and the rightmost octet is set to zero.

You can also use three-part dot notation to distinguish subnet numbers from network numbers. Subnet addresses are discussed in the section, "Will You Divide Your Address Space into Subnets?".

Reserved Addresses

Certain addresses have special meanings to network software and to other hosts on the network.

Broadcast Addresses

By default, a host number field of all ones signifies a *broadcast* address. A broadcast is a message sent to all hosts on the network. For example, networked hosts share routing information with each other by broadcasting their routing tables.

Note

Because some network interfaces, for example, HYPERchannel and UltraNet, do not support broadcasts, using a broadcast address does not guarantee that broadcasts will actually take place.

Refer to the section, "Defining Broadcast Addresses," in Chapter 4, "Configuring an Ethernet Interface," for details about setting up broadcast addresses.

Network Addresses

An internet address with a host number of all zeros refers to the network itself, rather than to a particular host. Class A network addresses are normally expressed in one-part dot notation. Class B network addresses are normally expressed in two-part dot notation, as in `140.55.` Class C network addresses are normally written in three parts, as in `140.55.20`, with each decimal value mapped to an octet of the network field; the host number field is set to zero.

Host Naming Conventions

Thus far, to keep matters simple, this discussion of names and addresses has implied that a host address identifies a unique computer. This is only partially true, because a host can communicate over more than one network interface. In such cases, a single address is not enough.

Termed *multihomed hosts*, machines connected to multiple network interfaces have multiple addresses, one per interface. This situation makes it apparent that an internet address does not actually refer to a host computer, but to a connection—a pathname that specifies a particular network interface on a particular machine.

Naming conventions help minimize confusion caused by multihomed hosts. One such convention consists of a root name identifying the host computer (for example, `hamlet`) followed by suffixes for the network connection (for example, `hamlet-ex` for accessing `hamlet` over an Ethernet; and `hamlet-hy` for accessing it via a HYPERchannel connection). This naming convention is illustrated in Figure 3-4.

Figure 3-4
Naming Hosts With Multiple Network Connections

```
#
# Host Database
#
127.0.0.1    localhost
#
# Local Ethernet
#
190.55.10.2    ariel-ex air
190.55.10.3    puck-ex
190.55.11.1    hamlet-ex son
#
# HYPERchannel
#
190.56.10.2    ariel-hy  wind
190.56.11.1    hamlet-hy pop
```

Unfortunately, this naming convention sometimes results in names that are less than user-friendly. To compensate for this awkwardness, assign aliases that are both meaningful and friendly, as in the example above.

Note

Assigning names that identify both host and network interface becomes even more helpful when connecting to a CONVEX UltraNet interface. Refer to Chapter 6, "Configuring a CONVEX UltraNet Interface," for further discussion on multihomed hosts.

Choosing an Address Structure

The first step in configuring your network is to select names and addresses for hosts on your LAN.

If you were setting up a LAN that you did not want to connect to any other network, you could pick addresses at random or invent your own numbering scheme. However, if you choose addresses randomly and later, after your small, simple network has grown in size and complexity, you find that you want to connect to other networks, you face the difficult task of changing your address structure to conform to the expectations of external networks. For this and other reasons, you should consider several factors before assigning addresses to hosts on your LAN. Among these factors are:

- Whether your LAN will connect to external networks.
- The number of host addresses you need, with room for expansion.
- Whether you will divide your network address space into subnets.

Do You Need Access to Other Networks?

Internet protocols require each host to have a unique address. Unless you use "official" internet addresses, you cannot guarantee uniqueness. Therefore, it pays to obtain official addresses before you set up your network. To do so costs nothing, and it permits you to expand the service area of your network as your needs grow.

After you have determined which address class you need, your site administrator should contact the organization in charge of the network and request the appropriate domain registration form. An organization that connects to multiple networks (such as CSNET, DARPA Internet and BITNET) should register with only one network. Contacts are as follows:

- **DARPA Internet**—Sites that are already on the DARPA Internet and need information about setting up a domain should contact `HOSTMASTER@NIC.DDN.MIL`. You can do so by writing to DDN Network Information Center, SRI International, 333 Ravenswood Avenue, Menlo Park, CA 94025, or by sending `email` to the network address below. You may also want to be placed on the BIND mailing list, a mail group for people on the Internet who run BIND. The group discusses future design decisions, operational problems, and other related topics. The network address to contact to request being added to this mailing list is:

  `bind-request@ucbarpa.Berkeley.EDU`.

- 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 prefers 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 method. The CIC can be reached via electronic mail at `cic@sh.cs.net`, or by phone at (617) 873-2777.

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

How Much Address Space Do You Need?

The potential size of your network influences how you choose addresses. Because your choice of address class imposes a hard limit on address space, you should choose a class generous enough to meet your needs as your network grows. Few networks require class A addresses (16,777,214 hosts), but many need a larger address space than the 254 hosts available on class C networks (host numbers of all zeros and all ones are reserved, as discussed above, under "Reserved Addresses"). Choosing an address class is a relatively easy decision to make before the network is set up, and a difficult one to change later.

Because the total number of available network numbers in each class is limited (127 class A; 16,383 class B; and 4,194,303 class C), you should choose the largest class that can support your present needs and your expected growth.

Will You Divide Your Address Space Into Subnets?

Before you assign addresses, you should decide whether you want to use *subnetting*. Subnetting is the partitioning of the network address space defined by a single network number into multiple virtual networks called *subnets* or *subnetworks*. With subnetting, local networks can include multiple physical network interfaces, yet appear as a single entity to remote networks. Figure 3-5 illustrates a typical way in which a class B network address is partitioned into subnet addresses.

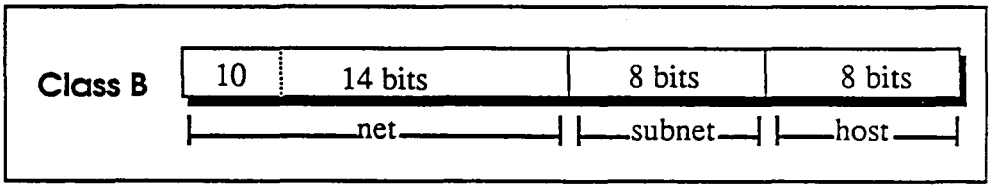


Figure 3-5
Subnet Address Partitioning

While this division into 8 bits for subnet number and 8 bits for host number is common for class B networks, you can partition the address any way you choose. The number of bits you give each field depends upon your needs. Will you have just a few large subnets, or is your installation more suited to having many small subnets?

Subnetted class B host addresses are commonly written in four-part dot notation as

```
net.net.subnet.host
```

When setting up subnets, assign addresses to your subnets as well as to the hosts they serve. Three-part dot notation conveniently represents a subnet address as

```
net.net.subnet
```

Note

Use of the subnet field is different for addresses assigned to hosts on an UltraNet network. Chapter 6, "Configuring a CONVEX UltraNet Interface," discusses these differences.

You might choose to logically divide an Ethernet LAN into subnets so that you can switch to multiple cables relatively easily if the network grows beyond the capacity of a single Ethernet cable.

More commonly, subnets correspond to separate physical networks. For example, an installation may use one Ethernet cable to connect machines used by engineering, another to serve the accounting department's machines, another for marketing, and so forth, each one assigned its own subnet number. A single gateway, assigned a single network number, could then be used for communication between hosts on different subnets, as well as between hosts on all subnets and the rest of the internet. (Gateways are discussed further in Chapter 7, "Routing Packets on the Network.")

Figure 3-6 depicts one possible use of subnets.

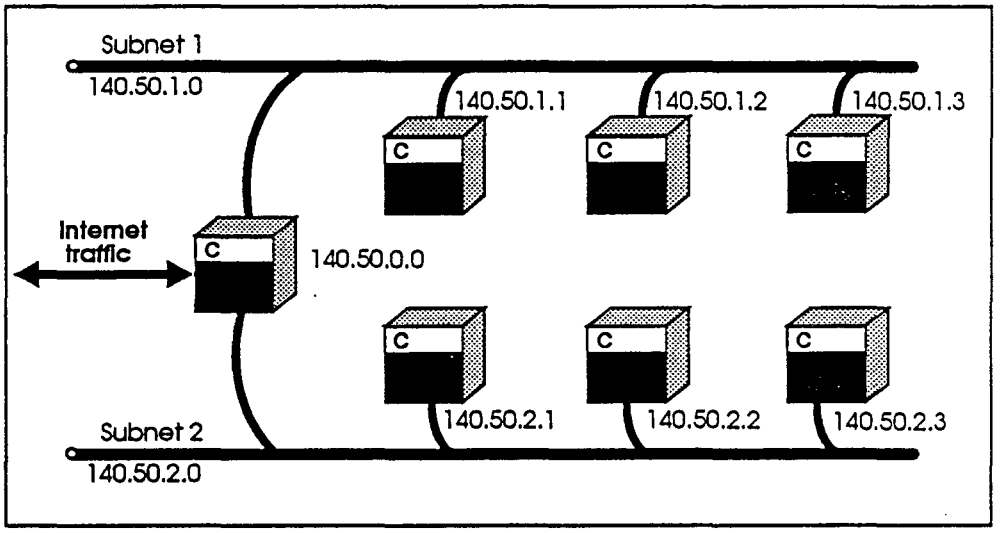


Figure 3-6
Subnets Connected to the Internet by a Gateway

You could achieve the same logical partitioning of a large local network by assigning different network numbers to each department's LAN. However, this method has disadvantages. Take the situation where a network consists of several physical LANs. You could obtain one network number and partition it into subnets, or you could request a different network number for each LAN.

Though the decision to use subnets or multiple network numbers is yours to make, be aware that it has an impact on systems beyond your own. Each address class has a fixed number of networks it can support, limited by the number of bits in the network number field. If you use multiple network addresses, you may unnecessarily deplete the pool of network numbers.

Once you have made the decision to divide your address space into subnets, you must tell network software which bits to use as the network and subnet numbers. You do this with the `netmask` parameter when you configure the network interface with `ifconfig`.

Methods for partitioning address space into subnets are discussed later in this chapter. Chapter 4, "Configuring an Ethernet Interface," explains how to set the subnet mask with `ifconfig`.

Other Considerations

How you structure your addresses also affects routing, because routing decisions are based on the network number portion of the address. All hosts on the same LAN should have the same network number.

Creating the Host Name Database

Previous sections described ways in which hosts on a TCP/IP network are identified, and discussed points to consider before deciding upon an address structure.

After choosing an address structure, the next step is to create the host name database for your name space and enable a method of associating host names with addresses. Host names are associated with internet addresses through a mechanism called *name resolution.* Network software uses one of two methods to resolve host names:

- Host table lookup method, whereby network software uses a static host name database containing explicit name-to-address translations. If you choose to use the host table lookup method to resolve names, your primary task is to maintain the /etc/hosts and /etc/networks files. This method requires you to update the /etc/hosts file on every networked host each time a host is added, removed, or changes its address. Consequently, this method is practical only when your network is relatively small and stable.
- Name server method, whereby dynamic host name and address information is shared across the network. CONVEX Internet Services provides the Berkeley Internet Name Domain (BIND) server system, consisting of a name server and an address resolver. When you use the name server method, you need only set up the host name database on a single host and configure other machines on the LAN to send queries to that host.

Regardless of which method you choose, you must create and maintain the /etc/hosts and /etc/networks files on at least one host on your LAN. The host name database must include all machines on your network.

To use the name server, you must also configure hosts on your LAN as described in Chapter 9, "Using the Berkeley Internet Name Domain Server."

Modifying /etc/hosts

The /etc/hosts file contains the database of host names and addresses for all hosts on your local network, as well as hosts on networks interconnected with yours with which your network needs to communicate.

The first few lines of the /etc/hosts file distributed with CONVEX Internet Services are similar to those in Figure 3-7 on the following page.

Figure 3-7
Sample /etc/hosts File

```
#
# Host Database
#
127.0.0.1  localhost
#
#
# The format is:
#      Internet-address official-hostname aliases...
#
# Local Net -- Ethernet
#
140.50.0.4     acme1 dopey
140.50.0.5     acme2 sleepy
140.50.0.6     acme3 sneezy
140.50.0.7     acme4 doc
140.50.0.9     acme5 bashful
140.50.0.10    acme6 happy cad
```

The /etc/hosts file contains two types of data: dot-notation addresses (`140.50.0.9`) and logical host names (`acme5 and bashful`).

The first number of the addresses in Figure 3-7 is between 128 and 191, signifying that the hosts belong to a class B network. This means that the 32-bit internet address is divided into 16 bits of network number and 16 bits of host number.

Note

In addition to dot notation, you can use decimal, octal, or hexadecimal numbers to specify Internet addresses. Use a leading 0x or 0X to signify a hexadecimal address, and a leading 0 to signify an octal address. Numbers without prefixes are interpreted as decimal.

Host names in Figure 3-7 are specified two ways: official host name and aliases. The official host name is listed first; aliases, if any, follow on the same line. For example, in the entry:

```
acme6 happy cad
```

`acme6` is the official name, and `happy` and `cad` are aliases. Aliases enable you to reference hosts by symbolic names. For example, the command sequences `rlogin cad` and `rlogin acme6` connect you to the same machine.

The version of /etc/hosts initially installed on your system is large, containing many groups of host names. It is intended to serve as an example and to provide addresses of hosts you may want to access if you connect to the DARPA Internet. Because most of these host names have little to do with names and addresses you choose for hosts on your LAN, feel free to delete those you do not need. Then, modify /etc/hosts to reflect your own network configuration.

Add addresses, names, and aliases for each host to which your hosts need access. Internet Requests for Comments (RFCs) require all networks to register with NIC.DDN.MIL. However, if your LAN is not connected to external networks, you are free to choose any network and host numbers; just make sure that each host on any given network has the same network number and a unique host number.

Note

If an official host name appears on more than one entry in the /etc/hosts file, the system uses only the first name. For example, if the /etc/hosts file contains the entries

```
140.50.0.4    acme1
140.50.1.4    acme1
```

networking software will send messages destined for `acme1` to internet address 140.50.0.4.

An example of a customized host database is shown in Figure 3-8. The file contains names and addresses for machines on two LANs.

Figure 3-8
Customized /etc/hosts File

```
#
# Host Database
#
127.0.0.1 localhost
#
# local Ethernet; network number 140.50
#
140.50.0.1 acmeq   obewan    acmeq-ex
140.50.0.2 acmer   vader     acmer-ex
140.50.0.3 acmes   leia      acmes-ex
140.50.0.4 acmet   luke      acmet-ex
#
# local Ethernet; network number 140.51
#
140.51.0.1 acmea   jason     acmea-ex
140.51.0.2 acmeb   medea     acmeb-ex
140.51.0.3 acmec   argos     acmec-ex
140.51.0.4 acmed   athens    acmed-ex
#
```

Host names shown in Figure 3-8 consist of a machine name (for example, `acmeq`), followed by two aliases (`obewan` and `acmeq-ex`). The second alias has an appended interface designator (`-ex`). Though you are not required to adhere to this naming convention, you may find it useful if you have multiple network interfaces, because it serves to identify both the machine and the interface.

Read more about the /etc/hosts file in Appendix A of this manual and in the `hosts`(5) man page.

Note

The /etc/hosts file will probably be identical for all hosts on the local network. If this is the case on your network, you can create the /etc/hosts file once and transfer it to all the other hosts.

Modifying /etc/networks

Just as the /etc/hosts file contains logical names, addresses, and aliases for each host to which your network is connected, the /etc/networks file contains logical names, addresses, and aliases for each network with which your LAN communicates. Several network utilities, including `netstat` and `routed`, use this file for mapping between network numbers and names. A typical /etc/networks file is shown in Figure 3-9.

Figure 3-9
Sample /etc/networks File

```
#
# Network Database
#
# The format is:
#     network-name Internet-address aliases...
#
acme-ex0      140.50  starnet
acme-ex1      140.51  greeknet
loopback-net  127     localnet
#
```

Network numbers shown in the example correspond to network numbers given in the sample /etc/hosts file in Figure 3-8.

Modify the /etc/networks file to include a logical name and network number for each network to be referenced from your machine. Be sure to use the same network numbers that you used in the /etc/hosts file. If your LAN has connections to other large networks such as the DARPA Internet, you must use officially assigned network numbers.

Read more about the /etc/networks file in Appendix A of this manual and in the `networks`(5) man page.

Configuring the Loopback Network

The file shown in Figure 3-7 contains entries for both a *loopback* network (`localhost`) and a local Ethernet connecting several hosts. The loopback network is a network interface to the local machine. It is used for, among other things, testing network connections. You must configure the loopback network; several network daemons need it to operate.

Three conditions must be met for the loopback interface to work:

1. You must configure the loopback interface, `lo0`, explicitly using `ifconfig`.
2. `lo0` should be the last interface configured, because some protocols use the first interface as the default.
3. You must configure the loopback interface before starting any network daemons.

There is an `ifconfig` command for `lo0` in the autoboot script, /etc/rc.local, which is installed automatically with CONVEX Internet Services. The loopback interface is configured in /etc/rc.local before network daemons are started.

If you add `ifconfig` commands to /etc/rc.local for other network interfaces, make sure you add them before the `ifconfig` command for `lo0`.

You can modify the configuration of the loopback interface in /etc/rc.local as long as the above requirements are met. Refer to Chapter 4, "Configuring an Ethernet Interface," for more information about /etc/rc and /etc/rc.local.

By convention, the address used for the loopback interface is `127.0.0.1`. You may use this address along with the symbolic name "localhost," or you may choose another address and name.

Creating Subnets

Subnetting allows interconnected local networks to share a single network number, thus reducing the demand on the available pool of network numbers and simplifying routing for external hosts and gateways. With subnets, local networks can include multiple physical network interfaces, yet appear as a single entity to remote networks.

To implement subnets, divide the host number field of your LAN's internet address into subnet and host portions by specifying a *subnet mask* with the `ifconfig` command used to configure the network interface. The subnet mask determines which part of the 32-bit internet address is combined with the network number to identify the subnet. By default, the mask is set to the size of the normal network number field for the particular network class (A, B, or C with 8, 16, or 24 bits of network part, respectively).

Within a subnetted local network, the subnet field is considered part of the network number. Outside the local network, the subnet field is not included in the network number.

Set the subnet mask at boot time in /etc/rc.local with `ifconfig`. Chapter 4, "Configuring an Ethernet Interface," explains how to use `ifconfig` for this purpose. The /etc/networks and /etc/hosts files in Figure 3-10 illustrate how a network number is logically partitioned into three subnets.

Figure 3-10
Subnet Partitioning in the /etc/networks and /etc/hosts Files

```
#
# Network Database (/etc/networks)
#
# The format is:
#     network-name Internet-address aliases. . .
#
# Class B Network - 140.50.
# subnet mask - 255.255.255.0 (0xffffff00)
# 255 subnets with 255 hosts available
#
acme-ex0        140.50.100   fishnet
acme-ex1        140.50.200   astronet
acme-ex2        140.50.300   faulknet
#
loopback-net    127          localnet
#
```

```
#
# Host Database (/etc/hosts)
#
# The format is:
#     Internet-address official-hostname aliases. . .
#
127.1           localhost
#
# Engineering Dept. (fishnet 140.50.100.hh)
#
140.50.100.10  eng1a   carp
140.50.100.20  eng1b   tuna
140.50.100.30  eng1c   sushi      squishy
140.50.100.40  eng1d   squid      inky
#
# Marketing Dept. (astronet 140.50.200.hh)
#
140.50.200.10  eng2a   altair
140.50.200.20  eng2b   sirius
140.50.200.30  eng2c   betelgeuse beetle
#
# Documentation Dept. (faulknet 140.68.300.hh)
#
140.50.300.10  doc1a   caddy
140.50.300.20  doc1b   benjy
140.50.300.30  doc1c   jason
140.50.300.40  doc1d   quentin
#
```

Regenerating /etc/hosts and /etc/networks Files

Internet host name databases are normally derived from a file retrieved from the Internet Network Information Center at SRI. The *gettable* program retrieves the NIC host database, and the *htable* program converts the data to the format library routines use to resolve addresses. Both programs are located in the /usr/etc directory.

To retrieve and reformat the internet host database table, change to the directory where you maintain local additions to the host table and execute the commands shown in Figure 3-11.

Figure 3-11
Using the `gettable` and `htable` Commands

```
# /usr/etc/gettable nic.ddn.mil
Connection to sri-nic.arpa opened.
Host table received.
Connection to sri-nic.arpa closed.
# /usr/etc/htable hosts.txt
Warning, no localgateways file.
#
```

The `htable` program generates three data files: /etc/hosts, /etc/networks, and /etc/gateways. (Gateways are discussed in Chapter 7, "Routing Packets on the Network.") If a file named localhosts is present in the working directory, it is copied to the output file preceding the /etc/hosts file created by the `htable` program. Similarly, a file named localnetworks is copied to the /etc/networks file preceding the networks data created by `htable`, and the localgateways file is copied to the front of the /etc/gateways data. Before installing new host and network databases in the /etc directory, run the `diff` file comparator on the old and new host databases to ensure that the files are complete.

If you use the name server for address resolution, you need only install the /etc/networks file and a small /etc/hosts file describing your local hosts. You may want to place the full host table elsewhere for reference by users.

If you are connected to the DARPA Internet, it is recommended that you use the name server, because it provides access to a much larger set of hosts than are in the host table. Many large organizations on the network that run the name server currently have only a small percentage of their hosts listed in the host table retrieved from the NIC.

Completing Host Name Database Setup

If your system uses the host table lookup method of name resolution, your host name database is configured and ready to use upon completion of procedures outlined in this chapter. To use the name server, however, you must also complete tasks described in Chapter 9, "Using the Berkeley Internet Name Domain Server." Before starting the name server, configure your network interface(s) according to procedures discussed in Chapter 4, "Configuring an Ethernet Interface," Chapter 5, "Installing and Configuring a HYPERchannel Interface," or Chapter 6, "Configuring a CONVEX UltraNet Interface."

Configuring an Ethernet Interface

4

Before running network software, you must configure each network interface connected to your system. This chapter describes how to use the *ifconfig* command to customize your Ethernet network configuration.

You also configure the internet (TCP/IP) interface for the CONVEX UltraNet Interface according to the procedures outlined in this chapter. Chapter 6 describes the CONVEX UltraNet Interface, which uses its own internal protocols in addition to TCP/IP and has its own configuration program. Perform the procedures in Chapter 6 to configure the "native" UltraNet interface, then configure the internet interface as instructed in this chapter.

You configure a HYPERchannel TCP/IP interface using a procedure similar to the one described in this chapter. Refer to Chapter 5, "Installing and Configuring a HYPERchannel Interface," for further information.

Using the ifconfig Command

You must be logged in as `root` to use `ifconfig`. Interfaces are numbered in the order in which they are listed in the /ioconfig file. For instance, the first Ethernet controller is assigned the unit name, `ex0`, the second, `ex1`, and so forth.

The command syntax is:

`/etc/ifconfig` *interface host_name* **`{up | down} [arp] [trailers]`**
`[netmask` *mask*`] [broadcast`** *address*`]`

where:

interface	Name and number of the network interface. (Refer to the `ex`(4), `hy`(4), `lo`(4), and `uv`(4) man pages.)
host_name	Local host name; for example, `acme1`, or internet address in dot notation.
`up`	Start the network interface. The complementary argument, `down`, shuts down the interface.
`arp`	Enable use of the Address Resolution Protocol (ARP), a mechanism for dynamically mapping between internet and Ethernet addresses. (Refer to "Using the Address Resolution Protocol (ARP)," below.)
`-arp`	Disable use of the Address Resolution Protocol (ARP).

trailers Enable use of *trailer* link-level encapsulation of outgoing messages (enabled by default). If a network interface supports trailers, the system, when possible, encapsulates outgoing messages in a way that minimizes the number of memory-to-memory copy operations performed by the receiver. On networks that support ARP, this option indicates that the system should request other systems to use trailers when sending to this host. Similarly, trailer encapsulations are sent to other hosts that have made such requests.

All machines that communicate with each other must use the same `trailers` setting; that is, they must all use trailers or they must all have the `trailers` option disabled.

-trailers Disable trailer link-level encapsulation of outgoing messages.

netmask *mask* specifies the portion of the internet address to reserve for the combined network and subnet fields. (Refer to "Defining Subnet Masks," below.)

broadcast *address* used to represent network broadcasts. By default, the broadcast address has a host part of all ones. (Refer to "Defining Broadcast Addresses," below.)

Note

The `ifconfig` command has parameters in addition to those discussed here. Refer to the `ifconfig`(8C) man pages for more information.

You may want to run the `ifconfig` command from the command line to ensure that it works properly before you edit /etc/rc.local to run it automatically upon system startup. Verify that the system accepted the information by entering:

`ifconfig` *interface*

For the specified *interface* (for example, `ex0`), the system displays the host address, network status (up or down), network mask, current trailers and ARP selections, and so forth.

Defining Broadcast Addresses

The default broadcast address contains a host part of all ones. You can change the broadcast address for an interface by using the `ifconfig` command. The system accepts broadcasts with a host part of all zeros (for compatibility with systems that use BSD 4.2 broadcasts) but transmits broadcasts with the destination address set to the broadcast address assigned with `ifconfig`.

If a machine on a network does not understand the broadcast address you select, some utilities, such as `rwho`, fail. If this happens, use `ifconfig` to set the network interface broadcast address to the old broadcast address (all zeros) for all machines on the network.

Note

All machines that communicate with each other must use the same broadcast address, either all zeros or all ones. The preferred method is a broadcast address of all ones, as in `broadcast 128.194.255.255`.

Defining Subnet Masks

Ones in the subnet mask indicate bit positions to use for the combined network and subnet fields; zeros mark the positions of bits in the host field. You specify the mask as a single hexadecimal number with a leading 0x, or in dot notation. For example, specifying

```
netmask 0xffffff00
```

or

```
netmask 255.255.255.0
```

both indicate you want 24 bits of combined network and subnet fields, and 8 bits of host number. For a class B network, this mask logically partitions your 16 bits of host number into an 8-bit subnet field and an 8-bit host field. If you do not supply `netmask`, the mask is set according to the network class (A, B, or C with 8, 16, or 24 bits of network part, respectively).

Note

To avoid confusion with broadcast addresses, do not use subnet numbers of all zeros or all ones.

Using the Address Resolution Protocol (ARP)

ARP maps logical internet addresses to physical Ethernet addresses by caching the logical mappings between dot-notation addresses (as in /etc/hosts) and physical Ethernet addresses. If an interface requests mapping for an address not cached, ARP queues the message that requires the mapping, then broadcasts a message on the associated network to request the address. If a response is received, the new mapping is cached, and the queued message is transmitted.

If you want to communicate with a machine on a LAN that does not use the ARP protocol, you must use the `arp` program to manually add address mapping information to the CONVEX ARP table. The `arp` program forces caching of an ARP table entry for a specific host, so that the ARP protocol does not transmit a packet to a host to get its Ethernet address.

To add an ARP entry for a new host, enter:

`arp -s` *host_name e_address* **`[temp] [pub]`**

where:

`-s`	Adds an entry to the ARP table.
host_name	Remote host as listed in the /etc/hosts file.
e_address	Physical Ethernet address of the remote host. This address is displayed by most systems at boot time and is usually printed on the network controller. You specify it as six hexadecimal numbers separated by colons, as in: 08:00:20:06:dd:42 The entry will be permanent unless you also specify temp.
`temp`	Specifies that the ARP table entry is temporary.
`pub`	Specifies that the entry will be "published"; that is, this system will act as an ARP server, responding to requests for *host_name* even though the host address is not its own.

You can check the current status of the ARP table by entering

`arp -a`

The `arp` command has more options than are discussed here. For a complete summary, refer to the `arp`(8C) man page.

Configuring at Start-Up

When you are confident that the network is properly configured, add `ifconfig` commands to your /etc/rc.local file for each network interface on your system, so that the network(s) will be brought up at boot time. An example is shown in italics in Figure 4-1. This line causes the network to be configured automatically each time the system is started.

Figure 4-1
Sample /etc/rc.local File

```
#
/bin/hostname muse
#
if [ -f /etc/ifconfig ]; then
        /etc/ifconfig ex0 `/bin/hostname` up arp -trailers netmask 0xffffff00
        #
        # All physical interfaces MUST be configured before lo0 is configured
        #
        /etc/ifconfig lo0 localhost up
fi
#
# syslogd must be started before any other daemons.
#
if [ -f /usr/etc/syslogd ]; then
        rm -f /dev/log
        /usr/etc/syslogd & echo 'starting system logger'
        if [ -f /usr/adm/bin/errlogd -a -f /usr/local/bin/perl ] ; then
                /usr/adm/bin/errlogd 2>&1 & echo ' errlogd' 2>&1
        fi
fi
#
if [ -f /etc/routed ]; then
        /etc/routed & echo -n ' routed'
fi
#
echo -n 'checking quotas: '
        /usr/etc/quotacheck -p -a
        /usr/etc/quotaon -a
echo 'done.'
#
echo -n 'local daemons:'
#
if [ -f /etc/rwhod ]; then
        /etc/rwhod & echo -n ' rwhod'
fi
#
echo '.'
```

Verifying Network Configuration with netstat

Bringing the machine up in multiuser mode is good evidence that you have correctly configured network interfaces. Usually, the machine simply does not run in multiuser mode if you make a mistake during the configuration process. Of course, you should test the network after the system is up and running in multiuser mode.

You must configure the network interface as described in this chapter before configuring network utilities according to the procedures discussed in Chapter 8, "Configuring Network Services." After you have configured network utilities, you should be able to use the network and its associated utilities. As a check, exercise the utilities associated with each of the daemons you have installed. You can check the network configuration by entering:

```
netstat -i
```

If the network is properly configured, the system displays output similar to that shown in Figure 4-2.

Figure 4-2
Configuration Check: Sample `netstat -i` Output

```
Name  Mtu   Network      Address    Ipkts   Ierrs Opkts    Oerrs Collis   Queue
ex0   1500  acme-net     acmes      1520512   0   1327049  0     293      0
lo0   1536  loopback-n   localhost  434254    0   434254   0     0        0
```

The displayed host name and network name are specific to your installation. The name of the interface you just configured should appear. If it does not, you have a problem with your installation or your configuration of the network. In either case, use the troubleshooting procedures outlined in Chapter 11, "Testing and Troubleshooting the Configured Network," to find the problem. The chapter on troubleshooting also contains more complete instructions for using `netstat`.

Installing and Configuring a HYPERchannel Interface

5

Because the HYPERchannel interface supports Internet protocols (TCP/IP), you configure it in much the same way as you configure an Ethernet interface. However, HYPERchannel interface software has its own set of configuration prerequisites, as well as its own address resolution program. This chapter describes procedures used to install, configure, and test a HYPERchannel interface.

You can find supplemental information in the *ConvexOS Programmer's Reference* under `hy`(4), `tcp`(4P), `hyroute`(1C), and `intro`(4N), and in the *Release Notice, CONVEX Internet Services*.

Prerequisites to Software Configuration

This chapter assumes you have completed installation procedures that came with your CONVEX Internet Services software. In addition, before you can begin configuring HYPERchannel network software, you must complete the following hardware installation and configuration tasks:

- ❑ Install HYPERchannel hardware.
- ❑ Modify the /ioconfig file to integrate network hardware into ConvexOS.
- ❑ Test the hardware configuration.

Installing HYPERchannel Hardware

CONVEX Internet Services provides a device driver or network interface (`hy`) for the Network Systems Corporation A400 or NB400 HYPERchannel Adapter through the IKON 10077-NSC Multibus or the IKON 10090 VMEbus Interface. To complete installation of network hardware, perform the steps below.

1. Log in to the system as `root`.
2. You must have an IKON 10077-NSC Multibus or IKON 10090 VMEbus Interface card installed in your system. Make sure the strappings are correct.

Note

On the IKON multibus board, output connector J1 connects to HYPERchannel adapter input. Input connector J2 connects to HYPERchannel adapter output. On the IKON VMEbus board, output connector J1 connects to the HYPERchannel adapter output. Input connector J2 connects to the HYPERchannel adapter input.

3. Set the `timeout` value for the A400 Adapter to 78 by using thumbwheels on the back of the adapter. On the NB400, this setting is stored into EEPROM via the diagnostic console.
4. Set the `unit` value into thumbwheels on the back of the A400 Adapter. Called the *adapter address*, this value either translates to a field in the internet address or is used in the `hyroute` database. On the NB400, this setting is stored into EEPROM via the diagnostic console.

Note

Any change to the unit value on the thumbwheels requires you to reboot the system.

Configuring the VMEbus Controller

The `csr` value is determined by controller switches U56, U57, and U58, as documented in the *IKON 10090 Hardware/Software Manual*. The interrupt level (`int`) is selected by jumpers W41-58 and W55-68. Other important controller switch settings are shown in Table 5-1.

Table 5-1
VMEbus
Controller Switch
Settings

Controller Switch	Setting
Address Modifier	3D
Bus Priority Level	3
DMA Burst Length	16
Range Counter Extension	16 bits wide
Device Flag	0
TEST ENB	off
SWAP DMA BYTES	on
SWAP P I/O BYTES	off

When you use configuration programs such as `ifconfig` and `hyroute`, specify the device prefix as `hv`. For example, to configure the device, `hostname-h`, enter the command:

```
ifconfig hv0 hostname-h up
```

Integrating HYPERchannel Hardware into ConvexOS

After installing network hardware, you must reconfigure ConvexOS to recognize the new device. Edit the /ioconfig file on the SPU to insert the device definition under the appropriate IOP and bus numbers. To edit the /ioconfig file for your system, first boot to the SPU prompt, `(spu)>`. Refer to the *CONVEX Processor Operation Guide* for booting instructions.

For a multibus HYPERchannel interface, insert an entry similar to:

```
iop 6
   mbus 0
      ctlr LAN-004 csr 0x5c0 int 5
         unit 0 type hy
```

Set the interrupt level (`int`) according to the IKON board's interrupt dipswitch setting.

A software-only release of the HYPERchannel driver for the IKON 10090 VMEbus Network Adapter Interface is available. The interface supports the NSC A400 and NB400 Adapters. Installation of this controller is similar to that of the multibus controller. To integrate the VIOP HYPERchannel driver, add an entry similar to the following to the /ioconfig file:

```
viop 6
   vme 0
      ctlr LAN-204 csr 0x2000 int 4
         unit 0 type hy
```

Reboot the CONVEX system and bring it up in multiuser mode. This forces the system to autoconfigure the newly-installed network hardware.

Testing the Hardware Configuration

Bringing the machine up in multiuser mode is good evidence that you have correctly installed network hardware. Usually, I/O configuration fails and the system sends error messages to the /mnt/errlog file on the SPU if you make a mistake during the installation and configuration processes.

After modifying the /ioconfig file and rebooting the system, test the hardware configuration by performing the steps below.

1. If the system does not `autoconfig` correctly (as indicated by a timeout on the device following the HYPERchannel adapter in the /ioconfig file), the I/O cables from the A400 Adapter to the IKON board may have been switched (that is, output connected to input and vice versa). Check connections on both the IKON board and the A400 Adapter.
2. Bring up any network interfaces by running `ifconfig`. Until you run `ifconfig` to bring up a specific network interface (for example, `ex0`, `lo0`, or `hy0`), you cannot display its status.
3. Enter **`netstat -i`** to verify that the entry, `hy0`, exists for the HYPERchannel interface. If **`netstat -i`** does not display an entry for the HYPERchannel interface, you will likely find that either the /ioconfig file is incorrect or the IKON board is not installed properly. Recheck the hardware installation and the /ioconfig file.

Configuring HYPERchannel Network Software

Once you are confident that you have properly installed and configured network hardware, the next task is to configure network software.

Configuring HYPERchannel TCP/IP software is similar to configuring a TCP/IP Ethernet. Among the differences are the ability to specify the maximum transmission unit (MTU) for communication with remote hosts and the method used to map internet addresses to hardware addresses. If you wish to use a larger MTU, or if your HYPERchannel network includes internet addresses that do not conform to the structure shown in Figure 5-1, you must configure the software to use `hyroute` after you assign host names and addresses. `hyroute` is discussed in the section, "Mapping Internet Addresses to HYPERchannel Hardware Addresses."

To configure CONVEX HYPERchannel Interface software, perform the steps below.

1. Assign host names and addresses.
2. Configure the HYPERchannel network interface.
3. Map internet addresses to HYPERchannel hardware addresses.

Assigning Host Names and Internet Addresses

As with an Ethernet LAN, you can use either the host table lookup or name server method to assign HYPERchannel host names and associate them with internet addresses. The difference is in the format of the addresses themselves, not in the method of name resolution.

Regardless of which method you choose, read this section to get an understanding of how HYPERchannel hardware addresses are structured. Then, if you choose to use the name server, turn to Chapter 9, "Using the Berkeley Internet Name Domain Server," for a complete discussion on how to configure the name server.

If you choose the table-lookup method for associating host names and addresses, modify the /etc/hosts file to include hosts on the HYPERchannel network. Add an entry to the /etc/hosts file for each host connection on the HYPERchannel network.

Unless you structure your addresses as shown in Figure 5-1, you must use `hyroute` to translate the internet addresses you assign into HYPERchannel hardware addresses.

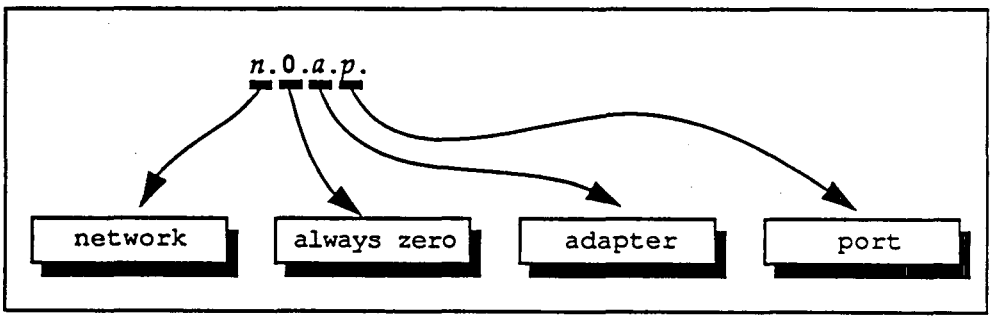


Figure 5-1
HYPERchannel Hardware Address Structure

Entries in the /etc/hosts file take one of the following forms:

n.0.a.p. *local_host_name*
n.0.a.p. *remote_host_name*

where:

n	Network number (1-63, 64-127, 128-255).
0	Second octet must be zero.
a	HYPERchannel adapter address, normally expressed in hexadecimal to match the `unit` value selected with thumbwheels on the back of the A400 adapter to which the host is connected, or via the diagnostic console for an NB400 adapter. (See Figure 5-2 for examples.)
p	HYPERchannel port number (0-3).
local_host_name	Name of the local CONVEX host.
remote_host_name	Name of a remote host.

Figure 5-2 shows typical /etc/hosts file entries for a CONVEX system with both an Ethernet and a HYPERchannel interface.

Figure 5-2
Typical /etc/hosts File with HYPERchannel Interface

```
#
# Host Database
#
127.1        localhost
#
#
# The format is:
#    internet-address official-hostnamealiases...
#
# Ethernet
140.50.4       acme1 dopey
140.50.5       acme2 sleepy
140.50.6       acme3 sneezy
#
# HYPERchannel
#
128.0.0x12.1   acme4
128.0.0xc2.0   cray2
128.0.0x05.3   vax-1
#
```

You must also modify the /etc/networks file to include the address of the HYPERchannel network. To do so, add an entry similar to

netname 128

where *netname* is any name you assign to the HYPERchannel network. The network address you assign (for example, 128) must match the network address field of the HYPERchannel hosts identified in the /etc/hosts file.

Note

If your HYPERchannel addresses do not conform to the structure shown in Figure 5-1, or if you wish to change the MTU, you must configure the software to use `hyroute` after you assign host names and addresses.

For more information about /etc/hosts and /etc/networks, refer to Chapter 3, "Setting Up the Host Name Database."

Configuring the HYPERchannel Network Interface

You define a HYPERchannel TCP/IP interface and customize network software the same way you define an Ethernet interface, by using the `ifconfig` command. Use a command similar to the following:

```
ifconfig hy0 host_name up
```

If the system responds with the error message, "Could not assign requested address," one of the following is probably true:

- The CONVEX system is not connected to the correct HYPERchannel adapter and port. Verify that the address you assigned to the interface matches the one selected with thumbwheels on the back of the A400 Adapter. If your system uses an NB400 Adapter, verify that the address you assigned to the interface matches the one stored in EEPROM. Correct the address if necessary.
- You assigned an incorrect network number to the HYPERchannel interface. Verify that the network number you entered is what you intended.
- HYPERchannel adapter hardware is malfunctioning. Consult the *NSC A400 HYPERchannel Adapter Hardware Reference Manual* or the *NSC NB400 HYPERchannel Adapter Hardware Reference Manual* for information about diagnosing the problem.

If `ifconfig` is successful, test the network configuration by entering

```
/usr/etc/ping host_name
```

You should get a series of messages that indicate data is being sent and received over the network. If not, one of the following may be true:

- *host_name* is not defined in the host name database.
- HYPERchannel adapter hardware is malfunctioning. Consult the *NSC A400 HYPERchannel Adapter Hardware Reference Manual* or the *NSC NB400 HYPERchannel Adapter Hardware Reference Manual* for information about diagnosing the problem.

Once you are confident that the network functions properly, add the `ifconfig` command to your /etc/rc.local file so that network software is configured each time the system starts. Then complete the software configuration by following the steps outlined in Chapters 7 through 11.

Mapping Internet Addresses to HYPERchannel Hardware Addresses

You must configure network software to use `hyroute` if your HYPERchannel host addresses do not conform to the structure shown in Figure 5-1, or if you wish to change the MTU.

`hyroute` allows you to assign arbitrary internet addresses (that is, addresses that do not conform to the HYPERchannel hardware address structure), or to assign an MTU other than the default of 4096. For example, you could use `hyroute` to map internet address `128.4.2.13` to `128.0.0x44.3` so that the host is assigned to adapter number 0x44, port 3.

HYPERchannel network software uses a default MTU of 4096 bytes. You can use `hyroute` to change the MTU for particular routes.

To use `hyroute`, follow the steps listed below.

1. Add commands to the /etc/rc.local file similar to the following:

```
/etc/ifconfig hy0 acme1 up # bring up the interface
/etc/hyroute -s /etc/hymap # load the hyroute tables
```

2. Modify the /etc/hosts file to include HYPERchannel host names and addresses. Because you are using `hyroute`, these addresses do not need to conform to HYPERchannel hardware addresses. Figure 5-3 shows a typical /etc/hosts file for a HYPERchannel network.

Figure 5-3
Sample /etc/hosts file Configured for HYPERchannel

```
#
# Host Database
#
127.1        localhost
#
#
# The format is
#    internet-address official-hostnamealiases...
#
128.50.0.0    acme1     # map to adapter 0x24, port 0 with hyroute
128.50.0.1    acme2     # map to adapter 0x24, port 1 with hyroute
128.50.0.2    acme3     # map to adapter 0x48, port 0 with hyroute
98.50.0.1     acme4     # map to adapter 0xc2, port 3 with hyroute
                        #  (this computer is not directly accessible
                        #   because it's on a different network (98))
#
```

3. Define host name to hardware address mappings. The `-s` option allows you to pass a file of mappings to `hyroute`. Figure 5-4 on the following page shows what the mappings might be for the hosts in Figure 5-3.

Figure 5-4
Host Name to HYPERchannel Address Mapping File

```
#
# Lines starting with  "#"  are comments
# The format is
# type   host    dest    control  access    MTU
#                (gate1) (gate2)           (* 1024)
#
direct   acme2   2401    1100     0         4;
direct   acme3   4801    2200     0         16;
gateway  acme4   acme3   acme2
```

The three entries in Figure 5-4 specify the following:

1. Any messages going to `acme2` go out over trunk 1 and any messages coming from `acme2` are received from trunk 1. This information is encoded in the high-order octet of the control field, where bits 7-4 encode transmit trunks 0-3, and bits 3-0 encode receive trunks 0-3.
2. `acme3` maps to port 1 on adapter 0x48 instead of port 0 as specified in the /etc/hosts file in Figure 5-3. This entry also causes all messages to be transmitted over trunk 2 of the HYPERchannel adapter.
3. `acme4` can be reached by going through either `acme2` or `acme3`.

The access field allows the local host to send the message to the receiving adapter. This access code must match the receiving adapter's access code as selected on the thumbwheels on the back of the adapter. Generally, the access code is not used, but it still must be included in the 64-byte header of the HYPERchannel protocol.

Specify the MTU field in 1024-byte units. The actual MTU will be slightly larger than this (refer to the definition of `HYMTU` in `<dev_iop/if_hyreg.h>` for the actual value).

Note

Because of the limited amount of buffer space available on the A400 Adapter, it is recommended that you do not set the MTU greater than 4096 for transfers between hosts directly connected to a single A400 Adapter.

If you need to change the mappings after the system has been booted, edit the mappings file and reload the mapping table by entering the command

```
/etc/hyroute hy0 -s hymap
```

Installing a Non-TCP/IP Device Driver

You may install a device driver to communicate with networks that use protocols other than TCP/IP. Refer to the *CONVEX Guide to Writing Device Drivers* for information about how to write your own drivers.

To install the device driver, complete the steps listed below.

1. Insert the following lines in the /ioconfig file on the SPU under the appropriate IOP and multibus entry:

```
ctlr LAN-002 csr 0x5c0 int 5
unit 0 type HYP_001
```

2. Reboot the system to cause an autoconfiguration.
3. Enter the command

```
mknod /dev/hy0 c 15 0
```

   to create a device entry in the operating system for the HYPERchannel device.

To use the port (`/dev/hy0`), you must have the correct software. The HYPERchannel device driver can only send and receive messages that contain a 64-byte HYPERchannel header block. Refer to the *NSC A400 HYPERchannel Adapter Hardware Reference Manual* or the *NSC NB400 HYPERchannel Adapter Hardware Reference Manual* for specifications.

Installation Problems

After completing software configuration, you should be able to use `rlogin`, `ftp`, or `telnet` to any host on the TCP/IP LAN. If you experience problems, use this section to help diagnose and solve them.

- Sometimes, the HYPERchannel adapter itself may be faulty. Try more than one adapter when you are having trouble getting a HYPERchannel link established.
- Make sure that the strappings on the IKON board are correct based on the most current documentation.

Configuring a CONVEX UltraNet Interface

6

Previous chapters described procedures used to configure a TCP/IP network, that is, one that uses DARPA Internet protocols. To configure an UltraNet network, you perform the same basic tasks, plus a few more. This chapter describes procedures for configuring a CONVEX UltraNet Interface. Before beginning the tasks in this chapter, you may want to read *CONVEX Networking Concepts* for a description of the CONVEX UltraNet Interface.

As in previous chapters, it is assumed here that you have completed the installation procedures that came with your CONVEX Internet Services and CONVEX UltraNet Interface software.

Introduction

CONVEX's high-speed UltraNet interface achieves its high throughput by using its own protocols, which run primarily on the UltraNet host adapter rather than on the CONVEX. While UltraNet's network architecture allows it to operate at high speeds, its internal protocols do not support facilities such as internet routing, subnetting, and broadcasting, nor do they communicate with other types of networks.

Because most LAN installations require connections to other networks, CONVEX UltraNet Interface software includes an internet interface module to allow it to participate in a TCP/IP network. An UltraNet network, therefore, has two network interfaces: a *native* UltraNet interface that is completely isolated from the internet, and an internet interface, which is a full member of the internet.

If you wish to connect your UltraNet network to other networks, you must configure both the internet interface and the native UltraNet interface. What this means is that you must configure UltraNet as if it were two networks, an internet network and a native UltraNet network.

CONVEX UltraNet Interface software uses the destination internet address of a packet to determine which interface and which set of protocols to use. If the destination address is the one you assigned to the native UltraNet interface, the software uses native UltraNet protocols. Otherwise, the software uses Internet protocols, complete with full internet functionality. Therefore, you assign two host names and two internet addresses to your UltraNet network. You then configure the native UltraNet interface according to the procedures in this chapter, and the internet interface according to procedures described in Chapter 4, "Configuring an Ethernet Interface." This chapter refers you to Chapter 4 when you need to perform a task discussed there.

Configuration Task Summary

To configure a CONVEX UltraNet Interface, complete the tasks listed below.

1. Load UltraNet adapter firmware.
2. Assign host names and addresses for both interfaces.
3. Map internet addresses to UltraNet station addresses.
4. Define network interfaces.
5. Set up routing.
6. Configure and start network servers.
7. Optionally, configure anonymous `ftp` accounts.
8. Test the configured network.

The remainder of this chapter discusses procedures for completing these tasks.

Loading Adapter Firmware

The second step in configuring the CONVEX UltraNet Interface is to load UltraNet adapter firmware from the host CPU. You must perform this step before running `unetcf` commands. Load firmware with the following command:

```
/usr/etc/uvmload /dev/ruv0 /usr/lib/firmware/uv/uvm32.ult
```

The file containing UltraNet adapter firmware, uvm32.ult, is installed on the /usr/lib/firmware/uv directory. For more information about loading adapter firmware, refer to the *CONVEX UltraNet Installation Procedures*.

Assigning Host Names and Internet Addresses

Because UltraNet has two network interfaces, you assign it two host names and two internet addresses. Each interface must have its own network number. The network will not operate unless you assign different network numbers to the internet and the native UltraNet interfaces.

You must assign each UltraNet host a unique subnet number and a host number of one. A sample /etc/hosts file reflecting this requirement is shown in Figure 6-1.

Figure 6-1
UltraNet Host Database

```
#
# Host Database (/etc/hosts)
#
127.1          localhost
#
# Native UltraNet
#
128.57.11.1    hamlet-ultra    ultra
128.57.12.1    romeo-ultra     romeo
#
# UltraNet internet Interface
#
128.58.11.1    hamlet-uip      ultraip
128.58.12.1    romeo-uip       romeoip
#
# excelan internet interface
128.50.11.1    hamlet
128.50.12.1    romeo
#
```

Each internet address in this example takes the form `net.net.unique_subnet.1` and the native UltraNet interface is assigned a different network number than the internet interface.

Naming conventions help minimize confusion caused by UltraNet's multiple interfaces. In the above example, the suffixes `-ultra` and `-uip` and the aliases `ultra` and `ultraip` serve to distinguish the native from the internet interfaces.
Once you have selected names and addresses, include them in the host database according to one of the methods described in Chapter 3, "Setting Up the Host Name Database" and Chapter 9, "Using the Berkeley Internet Name Domain Server." Figure 6-2 shows what the /etc/networks file might be for hosts in Figure 6-1.

Figure 6-2
Network Database with UltraNet Interfaces

```
#
# Network Database (/etc/networks)
#
#
# The format is:
#      network-name internet-address aliases...
#
acme-ultra     128.57   fastnet
acme-uip       128.58   internet
loopback       127      localnet
#
```

Mapping Internet Addresses to UltraNet Station Addresses

UltraNet adapters use 16-bit station addresses internally. These are equivalent to the 48-bit hardware addresses used by Ethernet controllers and the 16-bit hardware addresses used by HYPERchannel adapters. Because the Address Resolution Protocol (ARP) depends on broadcasts, it is not supported by UltraNet, which has no broadcast capabilities. Therefore, you must specify the translation between internet addresses and UltraNet station addresses explicitly.

If your UltraNet network uses a class A or class B network address, both of which have at least 16 bits of host number, mapping between internet addresses and UltraNet station addresses is one-to-one. No explicit translations are necessary. However, if your UltraNet network is assigned a class C network address, or a class B network address partitioned into subnets, there are not enough bits for the host number portion of the address to contain the full 16-bit UltraNet station address. In this case, you must use the *unetcf madd* command to define translations between internet addresses and UltraNet station addresses.

Note

If your network uses class C or subnetted class B addresses, you must include `unetcf madd` commands for both the Internet and the native UltraNet interfaces. Both addresses must map to the same station (adapter/unit) address (see below).

Because every destination host on an UltraNet network must have an address translation defined, numerous `unetcf madd` commands are usually required. To simplify the process of specifying these translations, you can create a file and pass it to the `unetcf madd` command. This file will probably be identical for all hosts on the UltraNet, so you can create it once and transfer it to all other hosts.

The syntax of the `unetcf madd` command is:

`/usr/etc/unetcf madd` *net-addr station-addr* **`|-f`** *filename*

where:

net-addr — Internet address of the host being mapped. You may specify the host by name (for example, `hamlet-ultra`), in standard dot notation (for example, 128.57.11.1), or as an eight-digit hexadecimal value beginning with **`0x`**.

station-addr — UltraNet station address corresponding to *net-addr*. Refer to the `unetcf`(8C) man pages for valid station addresses.

filename — Use the `-f` switch to pass `unetcf madd` a file containing translations. Each line in this file must contain exactly two fields separated by white space. The first field specifies *net-addr* and the second field specifies *station-addr*, as described above.

Note

You must use the `unetcf madd` command before adding adapters with the `unetcf aadd` command. If you define an adapter with `unetcf aadd` before mapping its address with `unetcf madd`, the adapter will incorrectly set the station address.

The commands

```
/usr/etc/unetcf madd hamlet-ultra 7/35
/usr/etc/unetcf madd hamlet-uip 7/35
```

define address mappings for host interfaces `hamlet-ultra` and `hamlet-uip` on Unit 35 in Adapter Group 7.

You can either add multiple `unetcf madd` commands to your /etc/rc.local file, or use a single command and pass it a file. To pass `unetcf madd` a file of address mappings, enter a command similar to

```
/usr/etc/unetcf madd -f /etc/ultra-map
```

Each line in the file of address mappings must be either a comment, indicated by "#" in column one, or contain exactly two fields separated by white space. The first field specifies *net-addr*; the second field specifies *station-addr*, both described above.

To delete an UltraNet mapping, enter

```
/usr/etc/unetcf mdel [net-addr]
```

Note

If you omit *net-addr*, all mappings will be deleted.

Defining UltraNet Adapters

Use the *unetcf aadd* command to define the native UltraNet interface and the `ifconfig` command to set up the internet interface. The `unetcf aadd` command is similar to `ifconfig up`. It declares the existence of a network path (in this case, an adapter), and activates it. The command syntax is:

```
/usr/etc/unetcf aadd adapter_id host_address in_dev
```

where:

adapter_id — A name from 1-9 characters you assign to the adapter. You use this name when you set up routing.

host_address — The address of the host on which the adapter resides, that is, the host on which you are configuring the network. You may specify it by name (for example, `hamlet-ultra`), in standard dot notation (for example, 128.57.11.1), or as an eight-digit hexadecimal value beginning with `0x`.

in_dev — The input block special device assigned to the adapter, generally of the form `/dev/uv0`.

Note

You must use the `unetcf madd` command before adding adapters with the `unetcf aadd` command. If you define an adapter with `unetcf aadd` before mapping its address with `unetcf madd`, the adapter will incorrectly set the station address.

The command

```
/usr/etc/unetcf aadd vme0 hamlet-ultra /dev/uv0
```

assigns the name `vme0` to the adapter connected to host `hamlet-ultra`.

The `unetcf aadd` command has several optional parameters. They are discussed in the `unetcf`(8C) man pages.

Verify that the information was accepted by entering:

```
/usr/etc/unetcf adis [adapter_id]
```

The system displays information about the adapter. If you omit *adapter_id*, the system displays information about all currently-defined adapters.

To delete an UltraNet adapter, enter

```
/usr/etc/unetcf adel adapter_id
```

Note

If you omit *adapter_id*, all adapters will be deleted.

Occasionally, a malfunctioning adapter cannot be deleted unless you use the `-f` option with the `unetcf aadd` command. To force deletion of the adapter, enter:

```
/usr/etc/unetcf adel -f adapter_id
```

Setting up UltraNet Routing

Because native UltraNet is completely independent of internet routing, you must define its routes explicitly using the *unetcf radd* command. To set up routing for the internet interface, use one of the methods described in Chapter 7, "Routing Packets on the Network."

UltraNet Routing Algorithm

Each native UltraNet route has associated with it a network address prefix and a *mask*, as described above. When deciding which adapter to use to get to a particular destination network address, routes are considered in the order you define them. For each route, the destination address is masked with the route's *mask* value, then the result is compared to the route's *net-addr-prefix*. If the two values match, the route is taken; if not, the next route in sequence is considered.

Any route with a mask of zero will always be selected. Such a route is termed the *default* route and must be defined last. Routes with masks of all ones will match only a single destination address.

For example, the command:

```
/usr/etc/unetcf radd ultranet vme0 0xffff0000 128.57.11.1
```

assigns the name `ultranet` to the route accessed through the adapter `vme0` and masks the upper 16 bits of each destination address. If the address after masking is `128.57.0.0`, the route will be taken.

Defining UltraNet Routes

The *unetcf radd* command is similar to `/etc/route add`. It specifies the adapter to use to access a given UltraNet host. The destination network need not be attached directly, and you may specify multiple `unetcf radd` commands for an adapter if there are several independent UltraNet networks accessible through the adapter.

The syntax of the `unetcf radd` command is

`/usr/etc/unetcf radd` *route_id adapter_id* [*mask*] [*net-addr-prefix*] [**`-i`**]

where:

route_id — A name from 1 to 9 characters in length that you assign to the route. Use this name to display information about the route or to delete it.

adapter_id — The name assigned to the adapter using the `unetcf aadd` command.

mask — A mask to be applied to destination UltraNet addresses when considering them for this route (see below). The mask is normally specified as a hexadecimal value beginning with **`0x`**. If you omit this argument, a mask of zero is used.

Normally, you would use the same mask you specify on the `ifconfig` command for the corresponding internet interface.

net-addr-prefix The network address prefix for the route (see below). You may specify this argument by name (as in the /etc/hosts database), in standard dot notation (for example, 128.57.11.1), or as an eight-digit hexadecimal value beginning with `0x`. The value you specify is masked with *mask* to yield the final network address prefix. If you omit this argument, the default is the network address of the corresponding adapter, masked by *mask*.

`-i` Specified if *net-addr-prefix* is an internet address (that is, an address defined for using UltraNet as an internet datagram delivery service). This option is required to enable network software to differentiate between the native UltraNet address and the UltraNet internet interface.

Verify that the system accepted the information by entering:

`/usr/etc/unetcf rdis` [*route-id*]

The system displays information about the route. If you omit *route-id*, information about all currently-defined routes is displayed.

After running `unetcf aadd` to define the native UltraNet interface and `unetcf radd` to set up routing for it, run `ifconfig` as described in Chapter 4, "Configuring an Ethernet Interface," to configure the internet interface. When using `ifconfig`, specify the network name associated with the internet interface, that is, `unet0`.

Deleting UltraNet Routes

Because the routing daemon, `routed`, does not automatically maintain native UltraNet routes, you must manually update your routes with `unetcf radd` and `unetcf rdel` commands. Add routes as described above and delete them as follows:

`/usr/etc/unetcf rdel` [*route-id*] [*adapter-id*]

You can omit either or both arguments.

Note

If you omit both arguments, all currently-defined routes are deleted. If you omit *route-id*, you must supply a null argument (by entering "" or ''); all routes for the adapter are deleted. For example, use the command:

`/usr/etc/unetcf rdel "" vme0`

to delete all the routes defined for adapter `vme0`.

Configuring UltraNet Servers

The next step is to configure and start network servers. UltraNet uses its own set of network daemons, started by its own superserver daemon, *unetd*, to handle the services it supports—`ftp` and `rcp`. The internet superserver daemon, `inetd`, and its associated daemons handle the services not supported by the CONVEX UltraNet Interface.

Like `inetd`, `unetd` reads a configuration file (/usr/etc/unetd.conf) and starts the daemons associated with the services listed there. Figure 6-3 shows the contents of unetd.conf.

Figure 6-3
Sample unetd.conf File

```
#
# Copyright 1990 Convex Computer Corp.
#
# internet server configuration database
#
ftp    stream tcp  nowait  root  /usr/etc/in.ftpd  ftpd
shell  stream tcp  nowait  root  /etc/in.rshd      rshd
```

(The `shell` port is used for `rcp` as well as for `rsh`.)

Entries for the services shown in Figure 6-3 are also included in the /etc/inetd.conf file (see Figure 8-1). To configure network daemons for UltraNet, edit the `inetd` configuration file, /etc/inetd.conf. Turn the entries for those services also included in unetd.conf into comments by adding a # at the beginning of each line. You must comment out each entry in the /etc/inetd.conf file that has a corresponding entry in the /usr/etc/unetd.conf file.

To start `unetd`, ensure that your /etc/rc.local file contains the following entry:

```
if [ -f /usr/etc/unetd ]; then
   /usr/etc/unetd /usr/etc/unetd.conf & echo -n ' unetd'
fi
```

Note

After you have commented out the `ftp` and `shell` services in the /etc/inetd.conf file, you must start `unetd` even if you do not bring up the UltraNet interface; otherwise, `ftp`, `rsh`, and `rcp` will not work.

`unetd` services requests from the internet interface regardless of whether the UltraNet interface is up and running.

After modifying the /etc/inetd.conf file to nullify entries for services supported by UltraNet, configure and start the internet superserver daemon, `inetd`, by following the procedure described in Chapter 8, "Configuring Network Services."

Configuring Anonymous ftp Accounts

Configure anonymous `ftp` accounts for a CONVEX UltraNet Interface as described in Chapter 8, "Configuring Network Services."

Configuring at Start-Up

When you are confident the network is properly configured, add commands to load adapter firmware, configure network interfaces, start network daemons, and set up routing to your /etc/rc.local file. Commands included in /etc/rc.local execute each time the system is booted.

For the CONVEX UltraNet Interface V1.1 and later, a skeleton rc.ultra file is loaded into the /etc directory when you install the software. This file should require minimal editing to configure it for your system.

Figure 6-4 shows a typical sequence of commands used to configure the CONVEX UltraNet Interface at boot time.

Figure 6-4
Starting UltraNet from /etc/rc.local

```
#
# Load and start the UltraNet host adapter
#
if [ -f /usr/etc/uvmload -a /dev/ruv0 ]; then
        #
        # Make sure UltraNet is deactivated to handle case of coming up
        # without a reboot. Ignore errors.
        #
        if [ -f /usr/etc/unetcf ]; then
                /usr/etc/unetcf rdel
                /usr/etc/unetcf adel
                /usr/etc/unetcf mdel
        fi
        #
        # Reset adapter to ensure that uvmload will succeed. This
        # takes about 5 seconds (uncomment commands for restarting
        # UltraNet without rebooting).
        #
        #if [ -f /usr/etc/uvmreset ]; then
        #       /usr/etc/uvmreset /dev/ruv0
        #fi
        #
        # Load UltraNet host adapter firmware
        #
        echo "Loading UltraNet(tm) host adapter software."
        /usr/etc/uvmload /dev/ruv0 /usr/lib/firmware/uv/uvm32.ult
        #
        # Activate the UltraNet if the firmware load succeeded.
        #
        if [ $? -eq 0 -a -f /usr/etc/unetcf ]; then
                #
                # Load the UltraNet station address mappings. See the
                # unetcf(8) man page for the format of this file. Note
                # that this operation is optional.
                #
                echo "Loading the UltraNet(tm) station addresses."
                /usr/etc/unetcf madd -f /etc/ultra-map
```

Figure 6-4
Starting UltraNet from /etc/rc.local (continued)

```
                if [ $? -eq 0 ]; then
                        echo "Activating UltraNet(tm) Interface."
                        /usr/etc/unetcf aadd vme0 dhostwo-ultra /dev/uv0

                        /usr/etc/unetcf adis
                        /usr/etc/unetcf radd ultranet vme0 0xffffff00
                        /usr/etc/unetcf radd ultra-ip vme0 0xffffff00 -i
                        /usr/etc/unetcf rdis
                        #
                        # If the UltraNet was successfully activated, then
                        # activate the ifnet (TCP/IP) interface.
                        #
                        if [ $? -eq 0 ]; then
                                echo "Activating UltraNet(tm) ifnet interface."
                                /etc/ifconfig un0 dhostwo-uip up netmask 0xffffff00
                        fi
                else
                        echo "Unetcf madd failed."
                        echo "UltraNet(tm) not activated."
                fi
        else
                echo "UltraNet(tm) host adapter load failed."
                echo "UltraNet(tm) not activated."
        fi
fi
#
# Start unetd (UltraNet version of inetd)
#
# This daemon must be started regardless of the success of activating
# UltraNet because without it, ftp, rcp and rsh will not work.
#
if [ -f /usr/etc/unetd -a -f /usr/etc/unetd.conf ]; then
                echo "Starting unetd."
                /usr/etc/unetd /usr/etc/unetd.conf
fi
#
```

Testing the Configured Network

Use `netstat` as described in Chapter 4, "Configuring an Ethernet Interface," to test the configuration of the internet interface. However, neither of these utilities operates on the native UltraNet interface. To test the native UltraNet interface, use the `tsock` network exerciser described in Chapter 11, "Testing and Troubleshooting the Configured Network."

Routing Packets on the Network

7

Previous chapters discussed host names and internet addresses as the means by which hosts on a network are identified. Addresses tell networking software where to send packets. Simply saying where to send a packet, however, is often not enough. Sometimes the sender does not know exactly where to find the destination host because the two machines are connected to different physical networks. In such cases, the sender needs a way of determining how to get the packet to its destination. This chapter discusses methods of *routing*, the task of finding the path over which to send packets to their destination.

Use procedures described in this chapter to set up routing on a TCP/IP network; that is, a network that uses DARPA Internet protocols. The CONVEX UltraNet Interface, which uses its own protocols in addition to TCP/IP and has its own routing configuration program, is discussed in Chapter 6, "Configuring a CONVEX UltraNet Interface."

Routes, Bridges, and Gateways

The simplest *route* connects one host to another on the same LAN. Packets never leave the physical network where they originate; they reach their destination directly over the LAN hardware. The process of routing becomes complicated when a host needs to send a packet to a machine on a different physical network. Transferring packets across physical network boundaries requires services provided by gateways and bridges.

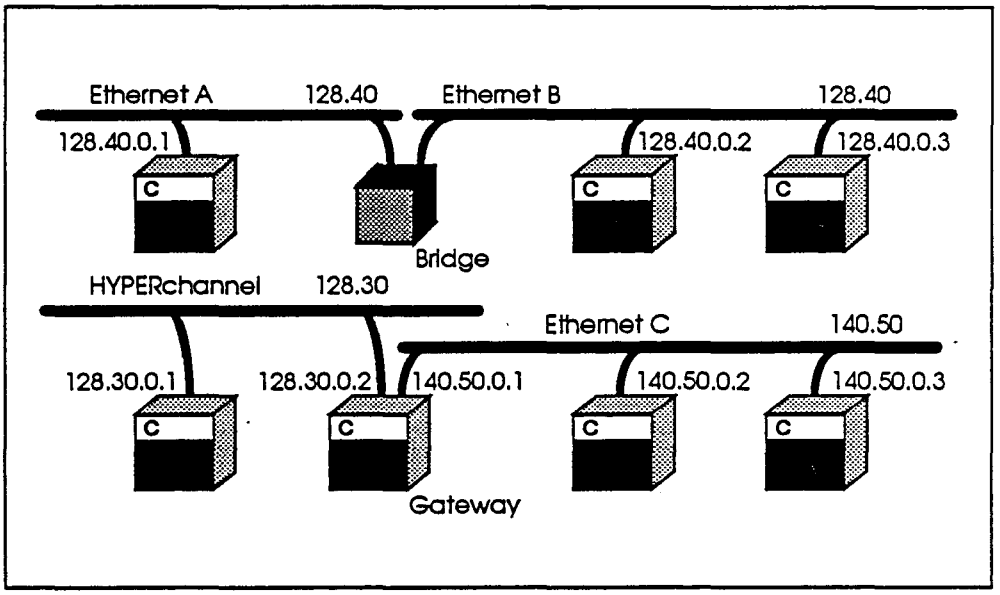
Gateways connect networks. Some are special-purpose packet-switching computers; others are simply hosts connected to multiple network interfaces, which transfer packets across networks in addition to serving as general purpose computers. Gateways can connect similar or different types of networks. For example, you might use a gateway to connect an Ethernet to a HYPERchannel.

Gateways that exchange routing information are called *active*, or *smart*, gateways; those that do not are called *passive* gateways. Active gateways keep dynamically-created routing tables up to date for use by themselves and by passive gateways.

Bridges also connect networks, but at the physical level. They normally connect networks of the same type. For example, you can use a bridge to catenate Ethernet cables so that they appear to network software as a single physical entity. Unlike gateways, bridges are transparent to network software.

Figure 7-1 illustrates the difference between bridges and gateways, and the purpose each serves in connecting networks.

Figure 7-1
The Role of Bridges and Gateways



As you can see from the diagram, gateways have two internet addresses (one for each network interface), whereas bridges have none. Because bridges are transparent to networking software, hosts on Ethernet A and Ethernet B view the LAN as a single physical network. Hosts on Ethernet C, however, must use different network numbers to reach hosts on the HYPERchannel or the Ethernet network.

Setting Up Internet Routing

As installed, CONVEX Internet Services software can route packets only to networks directly attached to your host. If you plan to connect your LAN to other networks, you must set up a method for determining routes to hosts on those other networks.

For your network to successfully route packets that originate within it, each machine needs to know the route to all possible destinations. Network software bases routing decisions on information stored in *routing tables*. Routing table entries include destination internet address, address of the gateway to the destination network, address of the network interface, and other information.

You can use one of three methods to define and maintain routing tables:

- Configure the system to use the network routing daemon, `routed`, to maintain routes dynamically.
- Use `the route` command to create a set of static routes the system loads into routing tables at startup. Using static routes has the same disadvantage as using the table-lookup method to resolve host names—it requires that you update information on all networked hosts each time any route changes.
- Define a *default*, or *wildcard*, route to a smart gateway and rely on that gateway for routing information. This method allows you to use one host on your LAN to handle all traffic to and from other networks. The disadvantage of this method is that if the smart gateway goes down, your network becomes isolated from other networks.

Note

Regardless of which routing method you choose, you must configure network interfaces before enabling routing.

Using the Routing Daemon

Unless your network is small and relatively stable, your best option is to use the routing table management daemon, `routed`, to maintain system routing tables. The routing daemon maintains up-to-date routing tables in a cluster of LANs. If you add suitable entries to /etc/gateways, you can also use `routed` to initialize static routes to distant networks.

You normally start `routed` at boot time from the /etc/rc.local file. To use `routed`, your /etc/rc.local must include the following lines (Chapter 4, "Configuring an Ethernet Interface," describes how to set up your /etc/rc.local file to configure the system at boot time.):

```
if [ -f /etc/routed ]; then
      /etc/routed & echo -n ' routed'
fi
```

These lines are included in the /etc/rc.local file created as a function of installing CONVEX Internet Services.

When `routed` first starts, it reads /etc/gateways and installs routes defined there. The routing daemon then attempts to find peer routing daemons on other hosts by sending out a broadcast message on each network interface to which the host is attached. If `routed` receives any responses, the local daemon cooperates with remote daemons to maintain a globally consistent view of routing in the local environment. You can also add entries to /etc/gateways if you want to communicate with remote sites running the routing daemon. If you find that /etc/gateways does not exist on your system, create it following the format described in Appendix A of this document and in the `routed`(8) man pages.

Creating Static Routes

Another way to route packets is to create static routes using the `route` command. To do this, edit your /etc/rc.local file, adding the `route` commands you require. You can add and delete routes to specific hosts or to networks. Figure 7-2 illustrates typical `route` commands.

Entries specify the internet address of the gateway used to reach the host or network. You can also identify a *default* gateway to send packets to if no other route is known (the next section discusses default routes more thoroughly).

Figure 7-2
Sample `route` Commands

```
#
# Internet routes
#
# the format is:
# /etc/route [add|delete] [net|host] destination gateway
#     [metric]
#
/etc/route add host     128.70.50.1  129.50.1.1  2
/etc/route add net      128.68.1     128.68.4.1  1
/etc/route add default  129.50.1.2
#
```

Note

If you decide to create and maintain static routes, be aware that as your network grows, so will the task of updating routing tables on all computers it serves.

Defining Default Routes

The third approach to routing combines `route` and `routed`. You define a default or wildcard route to a smart gateway, then depend on that gateway to provide routing redirect information used to dynamically create and update routing tables. Define a wildcard route by adding an entry similar to the following to /etc/rc.local:

```
/etc/route add default smart_gateway 1
```

The variable, *smart_gateway,* is the host name of the gateway you wish to access; 1 specifies the number of hops (gateway machines) in the path to the destination. Specifying **default** tells networking software to take the route if no other routes to the destination are known. The `route`(8C) man page contains a more complete explanation of this command and its syntax.

The system uses the default route as a "last resort." If the gateway to which packets are directed is able to generate the proper routing redirect messages, the system updates routing table entries based on the information supplied.

You cannot use the wildcard method in an environment that contains only bridges (which do not generate routing-redirect messages) and no gateways. Furthermore, if the smart gateway crashes, there is no way to maintain service other than manually altering the routing table entry.

Because the system always listens for and processes routing-redirect messages, it is possible to use `routed` to maintain up-to-date information about routes to geographically local networks and the wildcard method for routing to "distant" networks.

Setting Boot-time Routing Options

Three boot-time parameters control additional routing options: *gateway*, *ipsendredirects*, and *ipforwarding*.

If you set the `gateway` option, network software uses a larger routing table, and forwards packets even if the host has only a single, non-loopback interface. If the `ipsendredirects` option is set, network software sends Internet Control Message Protocol (ICMP) redirect messages for use in routing. Setting `ipforwarding` enables IP packets to be forwarded when the internet address does not correspond to any of the internet addresses for the machine's network interfaces. Refer to the section, "Networking Boot-Time Parameters," in this book and Chapter 14, "Customizing Kernel Boot-Time Parameters," in *Managing ConvexOS: Configuration Guide* for a complete explanation of these parameters and how to change them.

Verifying Routes with netstat

Use the *netstat* program to display routing table contents and statistics. For example, use the following command to display the contents of the routing tables:

```
netstat -r
```

Typical output is shown in Figure 7-3.

Figure 7-3
Sample Output from `netstat -r`

```
Routing tables
Destination  Gateway    Flags   Refcnt   Use       Interface
localhost    localhost  UH      2        3736      lo0
acme-net     acmes      U       49       1050480   ex0
dragon-net   acmee      UG      0        20834     ex0
mktg-net     acme1-ex0  UG      1        1095      ex0
```

The following command displays the number of routing tables dynamically created as the result of routing redirect messages:

```
netstat -r -s
```

Typical output is shown in Figure 7-4.

Figure 7-4
Sample Output from `netstat -r -s`

```
routing:
        0 bad routing redirects
        9 dynamically created routes
        3 new gateways due to redirects
        0 destinations found unreachable
        16 uses of a wildcard route
```

Configuring Network Services

8

This chapter describes procedures for configuring network servers and for controlling access to services. Use procedures described in this chapter to configure servers for a TCP/IP network, that is, a network that uses DARPA Internet protocols. The CONVEX UltraNet Interface, which uses its own protocols in addition to TCP/IP and has additional network servers, is discussed in Chapter 6, "Configuring a CONVEX UltraNet Interface."

The /etc/services File

The /etc/services file defines services provided by the network. Entries in the /etc/services file specify a service name and its aliases, associated port number, and protocol name.The format for records in the /etc/services file is:

official_service_name port_number/protocol_name [aliases] [#*comment*]

where:

service_name — Name of the command associated with the service.

port_number/protocol_name — Number of the port associated with the service and name of the protocol it uses. These parameters are expressed as a single item with a "/" separating them, as in `512/tcp`.

aliases — Any other names you wish to call the server.

#comment — All characters right of the "#" are ignored by routines that use the file.

Figure 8-1 on the following page shows a sample /etc/services file.

Figure 8-1
Sample /etc/services File

```
#
# Network services, Internet style
#
echo            7/udp
echo            7/tcp
discard         9/udp           sink null
discard         9/tcp           sink null
systat          11/tcp
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp          mail
uucp            33/tcp          uucpd
time            37/tcp          timserver
time            37/udp          timserver
name            42/tcp          nameserver
name            42/udp          nameserver
whois           43/tcp          nicname
hostnames       101/tcp         hostname        # usually from sri-nic
#
# Host specific functions
#
tftp            69/udp
finger          79/tcp
#
# UNIX specific services
#
exec            512/tcp
login           513/tcp
shell           514/tcp         cmd             # no passwords used
biff            512/udp         comsat
who             513/udp         whod
syslog          514/udp
talk            517/udp
route           520/udp         router routed   # 521 also
timed           525/udp         timed           # 4.3 timed
llockmgr        536/udp         lockf           # for lockf
#
# service for time delivery system
#
batch           666/tcp         batchd
nqs             607/tcp                         # nqs batch (tcp)
nqs             607/udp                         # nqs batch (udp)
crashdump       670/udp
```

For more information about /etc/services, refer to the `services(5)` man page.

Configuring the Internet Superserver Daemon

Each internet utility, such as `ftp`, `rcp`, and `rlogin`, uses a pair of processes to communicate over the network. A *client* process running in the sending host initiates a request for a connection to the destination host. Its corresponding *server* running on the destination host awaits such requests and performs necessary actions—initiate the transfer of a source program, list the contents of a file, execute a command, etc.—to complete the request. This section describes procedures for configuring internet servers to process requests for networking services.

To minimize the number of networking processes running on the system, CONVEX Internet Services uses a single daemon to service multiple network requests. This daemon is called *inetd*, or the "internet superserver." `inetd` is started automatically at boot time from the /etc/rc script. Once started, `inetd` reads a configuration file called /etc/inetd.conf and monitors a set of ports associated with each internet service defined there. When `inetd` receives a connection request on one of its ports, it activates the appropriate daemon to service the request.

Use the following procedure to ensure that necessary network daemons start at boot time:

1. All daemons except those beginning with `rpc` should be listed in the /etc/inetd.conf file. `rpc` daemons are started only if your site runs `NFS`. If you have any `rpc` daemons listed in the configuration file, but the `rpc` superserver, `portmap`, has not been started, `inetd` logs an error message using `syslog` and continues without starting up any `rpc` servers. Once `portmap` has been started, `inetd` sets up the daemons listed in the configuration file if it receives a SIGHUP signal.

   If sent a SIGHUP signal, `inetd` rereads the configuration file and reconfigures itself. If you do not want a particular daemon started, edit the /etc/inetd.conf file, turning the appropriate line into a comment by adding a "#" at the beginning of the line.

   The format for records in the /etc/inetd.conf file is:

   service_name {**stream** | **dgram** | **raw**} *protocol* {**wait** | **nowait**} *user server_program* [*arguments*]

   where:

service_name	Name of a valid service in the /etc/services or /etc/rpc file (if your site runs `nfs`.) See below.
stream **dgram** **raw**	Socket type.
protocol	Name of a valid protocol listed in the /etc/protocols file.
wait **nowait**	Flag indicating whether the server processes all incoming messages on the socket (**wait**), or frees the socket after receiving a message (**nowait**). This parameter applies only to datagram (**dgram**) sockets.
user	User name under which the server runs. This parameter allows the system manger to give servers less permission than `root`.
server_program	Pathname of the program `inetd` executes when it receives a request on the socket. If `inetd` provides the service itself, this field should be "internal."
arguments	Arguments passed to the server program.

Figure 8-2 shows the contents of the /etc/inetd.conf file installed as a function of installing network software.

(nfs is an optional product. For more information about configuring rpc daemons, refer to the *CONVEX NFS System Manager's Guide*, and the *CONVEX Remote Procedure Call Programming Guide*.)

Figure 8-2
Sample /etc/inetd.conf File

```
#
# Copyright 1990 Convex Computer Corp.
#
# Internet server configuration database
#
ftp         stream   tcp     nowait   root    /usr/etc/in.ftpd          ftpd
telnet      stream   tcp     nowait   root    /usr/etc/in.telnetd       telnetd
shell       stream   tcp     nowait   root    /etc/in.rshd              rshd
login       stream   tcp     nowait   root    /etc/in.rlogind           rlogind
exec        stream   tcp     nowait   root    /usr/etc/in.rexecd        rexecd
syslog      dgram    udp     wait     root    /usr/etc/in.syslog        syslog
comsat      dgram    udp     wait     root    /usr/etc/in.comsat        comsat
talk        dgram    udp     wait     root    /usr/etc/in.talkd         talkd
crashdump   dgram    udp     wait     root    /usr/etc/in.crashreceive  crashrcv
rusers      dgram    udp     wait     root    1-2 /usr/etc/rpc.rusersd  rusersd
rup         dgram    udp     wait     root    1-3 /usr/etc/rpc.rstatd   rstatd
rwall       dgram    udp     wait     root    1 /usr/etc/rpc.rwalld     rwalld
ount        dgram    udp     wait     root    1 /usr/etc/rpc.mountd     mountd
quota       dgram    udp     wait     root    1 /usr/etc/rpc.rquotad    rquotad
spray       dgram    udp     wait     root    1 /usr/etc/rpc.sprayd     sprayd
rex         stream   tcp     wait     root    1 /usr/etc/rpc.rexd       rexd
echo        stream   tcp     nowait   root    internal
discard     stream   tcp     nowait   root    internal
chargen     stream   tcp     nowait   root    internal
daytime     stream   tcp     nowait   root    internal
time        stream   tcp     nowait   root    internal
echo        dgram    udp     wait     root    internal
discard     dgram    udp     wait     root    internal
chargen     dgram    udp     wait     root    internal
daytime     dgram    udp     wait     root    internal
time        dgram    udp     wait     root    internal
```

For more information about inetd and the /etc/inetd.conf file, refer to the inetd(8C) man page.

2. If you have added any daemons, you must start them by running them from the command line or shutting down to single-user mode (using `reboot` or `shutdown -r`) and rebooting according to instructions in the *CONVEX Processor Operation Guide.*

After completing the steps above, you should be able to use the network and its associated services. As a check, exercise utilities associated with each daemon you have installed. For example, enter the command

`telnet` *`hostname`*

The host you specified should respond with a login prompt. If it does not, use the troubleshooting procedures outlined in Chapter 11, "Testing and Troubleshooting the Configured Network."

Controlling Access to the Network

Each machine on the network can accept or refuse logins by remote users. The system manager specifies whether users logging in from other machines must supply a password, or even if remote logins are permitted on a machine. Two files control this type of access: */etc/hosts.equiv* and *.rhosts*. (If your system runs NFS, the Yellow Pages Service (YP) uses the /etc/netgroup file to control user access. `nfs` is an optional product. For information about configuring YP, refer to the *CONVEX Network File System System Manager's Guide.*)

The /etc/hosts.equiv File

The /etc/hosts.equiv file contains a list of machine names that you can access from the local machine without using a password (using the same login name on the remote machines). When you use `rlogin` to remotely log in to a machine that is not listed in the /etc/hosts.equiv file, you are prompted for a password unless the local machine is listed in the ~/.rhosts file. Figure 8-3 shows a sample /etc/hosts.equiv file.

Figure 8-3
Sample /etc/hosts.equiv File

```
acme1
dopey
sleepy
sneezy
doc
bashful
grumpy
```

Note

For security reasons, access to the root account is not controlled by the /etc/hosts.equiv file. You must either have an appropriate entry in root's /.rhosts file, or you must supply the root password when you log in remotely. For security reasons, it is suggested that read access to the /etc/hosts.equiv file be limited to root.

The .rhosts File

Each user's home directory may also contain a private equivalence list in a file called .rhosts for use by `rlogin`. Entries in this file contain remote host names (not aliases) and user names specifying which users on which machines are permitted to log in without supplying passwords. For example, if you use different login names on different machines, add entries to your .rhosts file containing your other login names and names of machines from which you want to access your local account. The example in Figure 8-4 shows a .rhosts file for a user named `rsmith` on the local machine who uses login names `smith` and `bobsmith` on remote machines.

Figure 8-4
Sample .rhosts File

```
aries smith
aries bobsmith
```

In the above example, both `smith` and `bobsmith` may access `rsmith`'s account on `aries` without entering a password.

Note

Your .rhosts file will not be honored if it has either group or world write permissions. You must use official host names (the first name listed in /etc/hosts file entries), not aliases, in the .rhosts file.

Configuring Anonymous ftp Accounts

This section describes an optional procedure for enabling remote users to transfer files using `ftp`.

The `ftp` file server provides support for an anonymous `ftp` account. This account enables users to transfer files between machines without using a password. Files transferred are written to a common user account, enabling users to transfer files to a secure or heavily protected host without gaining unrestricted access to the host's file system.

Note

There are inherent security problems with any user account of this type. Therefore, before you install this account, make sure that benefits to your users outweigh possible security risks. If you decide to install this account, read this chapter carefully. By following the procedures described here, you will make the installation much safer than it might otherwise be.

Enable the anonymous account by creating a user named `ftp`. (You can use `nu` to create the `ftp` user. Refer to the `nu`(8) man page for details.) When a client uses the anonymous account, the `ftp` server performs a `chroot` system call. In this way, the user is restricted from moving outside the part of the file system containing the common `ftp` directory.

For this system to work properly, you must ensure that all directories and executable images associated with the `ftp` account are restricted to reading and executing (not writing), except for the directory into which you allow anonymous users to transfer files (`pub`). Furthermore, you must ensure that certain programs and files are supplied to the server. Enter the following sequence of commands to allocate files and directories needed. The subdirectory, `~ftp/pub`, is allocated as the common `ftp` directory.

Figure 8-5 shows the sequence of commands used to set up an anonymous `ftp` account.

Figure 8-5
Setting Up Anonymous `ftp` Accounts

```
# cd ~ftp
# chmod 555.
# chown ftp .
# chgrp ftp .
# mkdir bin etc pub
# chown root bin etc
# chmod 555 bin etc
# chown ftp pub
# chmod 777 pub
# cd bin
# cp /bin/sh /bin/ls .
# chmod 111 sh ls
# cd ../etc
# cp /etc/passwd /etc/group .
# chmod 444 passwd group
```

When performing the last command in Figure 8-5, be sure to edit the `passwd` and `group` files to contain only `root` and `ftp`. Doing so helps minimize security risks. The system described here is compromised if you allow access to this account by users logged in as `uucp` or `root`.

You may also run into problems if you allow logins by accounts with nonstandard shells and no passwords (for example, `rwho` or `finger`). To avoid these types of problems, make sure to add these names to the /etc/ftpusers file (or create the file if it does not exist). This file is checked each time `ftp` is used. If the requested user name is in the file, the request for service is denied. Read more about the /etc/ftpusers file in Appendix A of this guide.

Using the Berkeley Internet Name Domain Server

9

This chapter describes how to configure a system to use the Berkeley Internet Name Domain server (BIND). If you choose to use the name server, your primary task as system manager is to configure the system to use BIND as a replacement for /etc/hosts table lookup. Configuring the name server is not a simple task; it involves setting up several complex files. The benefit comes from having to do this on only one machine and letting the name server keep the files up-to-date on other networked machines.

This chapter contains three sections:

- An introduction to name server concepts for the reader who is unfamiliar with the internet name server.
- Descriptions of files used by the name server system.
- A step-by-step guide to configuring each type of name server.

Specifications for the name server are defined by Requests for Comments (RFCs), which you can find in /usr/lib/conf/bind. Familiarity with the material in these documents will aid you in configuring the name server for your system. It is also recommended that you read related man pages, `named`(8), `resolver`(3), and `resolver`(5).

Introduction

The Berkeley Internet Name Domain (BIND) server implements the DARPA Internet name server. A name server is a network service that enables clients to name resources or hosts and share this information with other hosts in the network. This in effect is a distributed database system for hosts in a computer network.

Internet Domain Names

The internet name server views the network as a hierarchy of *domains*. A domain is a tree structure that can be organized according to organizational or administrative boundaries. Each domain in the hierarchy has a *label*. A *fully-qualified domain name* is the concatenation of all labels of domains from the root (highest level in the hierarchy) to the current domain, listed from right to left and separated by dots. A label need only be unique within its domain.

Name space is partitioned into areas called *zones*, usually represented by administrative boundaries. Each zone starts at a domain and extends down to the leaf domain (lowest level in the hierarchy) or to domains where other zones start. A given name server is said to have *authority* over a particular zone, meaning that the server maintains all data corresponding to that zone.

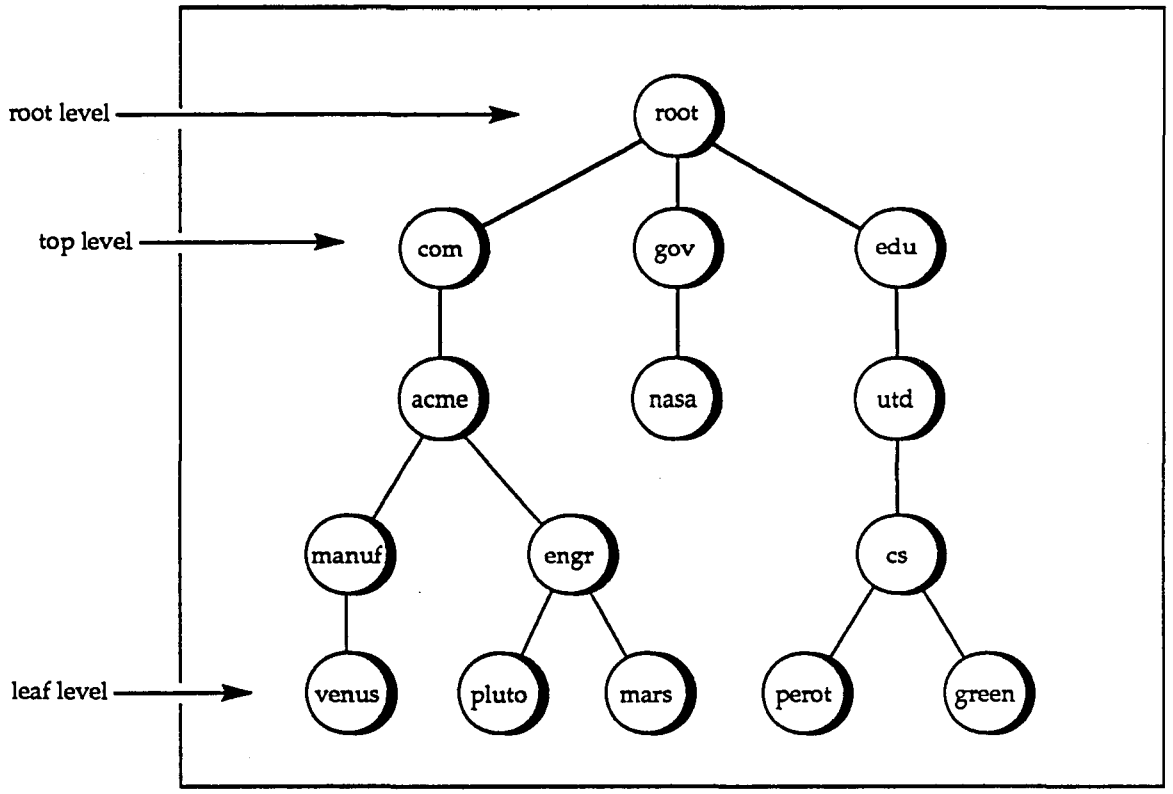
A domain name for a host at a university might be as follows:

```
perot.cs.utd.edu
```

The *top level* domain for educational organizations is `edu`; `utd` is a *subdomain* of `edu`, and `perot` is the name of the host used by the computer science (`cs`) department. A name server running on `perot` services all names in its zone of authority, which may be all the names for machines on the campus network or a subset of those names.

Figure 9-1 shows a hierarchical name structure.

Figure 9-1
Domain Name Tree Structure



In the example above, fully-qualified domain names for hosts at the leaf level are `venus.manuf.acme.com`, `pluto.engr.acme.com`, `mars.engr.acme.com`, `perot.cs.utd.edu`, and `green.cs.utd.edu`.

Internet Domain Name Server Software

BIND software comprises two parts: *named* and the *resolver.* `named` is a daemon that services queries on a given network port (as defined in the /etc/services file). The `resolver` consists of a few routines that build query packets and exchange them with the name server.

You can configure your system to use any of several types of servers. The type you choose to run on a given system depends upon factors such as the amount of disk and memory space available on each system, and the extent to which you want users on the system to have access to the network.

BIND provides the following types of servers:

- **Master Server**—A *master server* for a domain has authority over that domain. This server maintains all data within its domain. Each domain should have at least two master servers, a primary master and at least one secondary master to provide backup service if the primary is unavailable or overloaded. A server may be a master for multiple domains; it may serve as primary for some domains and secondary for others.
  - **Primary**—A *primary master server* loads its data from a file on disk. This server may also delegate authority to other servers in its domain.
  - **Secondary**—The primary master server delegates authority to and provides data for a secondary master server. At boot time, the secondary server requests all 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.
- **Caching-Only Server**—A *caching* server is one that caches information it receives. Master servers and *caching-only* servers both cache information; however, a *caching-only server* is not authoritative for any domain. It services queries and asks other servers who have authority for the information it needs. All caching servers keep data in their caches until the data expires, based on a time-to-live field attached to the data when it is received from another server.
- **Resolving-Only Server**—You can use a name server on a workstation or on a machine that has a limited amount of memory and CPU cycles by choosing the *resolving-only* server option. This option allows you to run all networking programs that use the name server without running the name server on the local machine. Queries are serviced by a name server that is running on another machine on the network.

  Queries by resolving-only servers are recursive. The resolving-only server sends queries to each server in a list of a available servers until the list is exhausted.

  You typically use a resolving-only configuration when you do not wish all servers at a given site to interact with the rest of the internet servers. For example, suppose your network consists of a number of workstations and a departmental timesharing machine with internet access. You might want to prohibit workstations from having internet access. To give workstations the appearance of access to the internet domain system, you could configure them as resolving-only servers to the timesharing machine. The timesharing machine would then forward queries and interact with other name servers.

  An added benefit of using the forwarding feature is that the central machine develops a much more complete cache of information of which all workstations can take advantage. The use of slave servers and forwarding is discussed further under the description of the `named` boot file commands.

System Files

Configuring the name server requires you to create or modify several control and data files. Control files influence execution of named. Data files make up the name server's database. This section describes the content and format of files used by the name server system.

File Summary

The name server system uses the following files (Refer to "File Descriptions," below, for detailed descriptions and examples of each file):

/etc/named.boot	Name server configuration boot file
/usr/etc/named.reload	Script used to reload and restart named.
/usr/etc/named.restart	Script used to restart named.
/etc/named.pid	Contains the process id of named.
/etc/resolv.conf	Resolver configuration file.
/etc/use_nameserver	File to indicate named is in use.
/etc/hosts	Host name database.
<directory>/named.hosts	Contains data about all hosts in this zone.
<directory>/named.local	Contains the address of the local loopback interface.
<directory>/named.rev	Contains internet host addresses in reverse order.
<directory>/root.cache	Contains addresses of authoritative name servers for the root domain.
/usr/lib/conf/bind	Contains referenced RFCs and other documentation.

Control File Descriptions

This section contains detailed descriptions and formats for files used to control the execution of the name server.

Boot File

The boot file, /etc/named.boot, contains name server configuration information. The name server daemon, named, reads this file when it first starts up. The file specifies the server's type, zones of authority, and location of its initial data. You can change the pathname for the boot file from the default, /etc/named.boot, by specifying another pathname on the command line that starts named. (See Figure 9-22 for an example of the commands used to start up named.)

Figure 9-2 on the following page shows a sample boot file.

Figure 9-2
Sample named.boot File

```
; named.boot
;
directory /usr/local/domain

; type      domain                    source host/file          backup file

cache       .                         root.cache
primary     acme.com                  named.hosts
primary     32.128.IN-ADDR.ARPA       named.rev
secondary   mars.acme.com             128.32.137.8 128.32.137.3  mars.hosts.bak
secondary   6.32.128.IN-ADDR.ARPA     128.32.137.8 128.32.137.3  mars.rev.bak
primary     0.0.127.IN-ADDR.ARPA      localhost.rev
forwarders  10.0.0.78 10.2.0.78
;
```

The boot file may contain the following command lines, depending on how you wish to configure your name server:

- **directory**—Specifies the root directory for the name server. The `directory` command allows you to specify other file names in the boot file relative to this pathname. You should always include a `directory` command in your boot file. The format of the `directory` command is:

  `directory` */root_directory*

  If you have more than a couple of `named` data files to maintain, you can place files in a directory such as /usr/local/domain and specify that pathname with the `directory` command.

  This command also ensures that `named` is in the proper directory to locate `$INCLUDE` and source/host files specified with relative pathnames.

- **primary**—Designates the server as a primary master server. The format of the `primary` command is:

  `primary` *domain_name* *hosts_file*

  The first field specifies that the server is a primary master for the domain named in the second field. The third field is the pathname of the file containing the domain data, relative to the pathname specified with the `directory` command.

  In the example in Figure 9-2, the first `primary` command specifies that the database for domain `acme.com` is located in the file, named.hosts. Records in named.hosts contain data in the master file format described in "Standard Resource Record Format," below.

  The second `primary` command in Figure 9-2 specifies that authoritative data for the domain `32.128.IN-ADDR.ARPA` is located in the file, named.rev (reverse). The name server uses data in named.rev to translate addresses in network 128.32 to host names. This file is described in the section, "named.rev."

- secondary—Designates the server as a secondary master server. The `secondary` command is similar to the `primary` command except that it lists addresses of other servers (usually primary master servers) from which the name server running on the local host obtains its domain data. The format of the `secondary` command is:

  `secondary` *domain_name* *primary_name_servers* *hosts.backup*

  The first field identifies the server as a secondary master for the domain named in the second field. The *primary_name_servers* field specifies names of primary servers for the domain. The secondary server obtains its data across the network from the listed servers. Secondary masters try each primary server in the order listed until they successfully receive data from a listed server. If you specify a file name after the list of primary servers, the name server will dump data for the domain into that file as a backup. When the server first starts, it loads data from the backup file if possible, then consults a primary server to ensure that data is still up-to-date.

- cache—Specifies the name of the file used to prime the name server's cache. All servers should have at least one `cache` command in the boot file. The format of the command is:

  `cache` . *root.cache*

  At boot time, the name server reads the files listed and reinstates in the cache all values that are still valid.

- sortlist—Lists in order of preference addresses of other name servers to query. Queries for host addresses from hosts on the same network as the local server will receive responses with local network addresses listed first, then addresses in the sort list, then other addresses. This line has meaning only at initial start-up. When the name server is reloaded with a SIGHUP signal, it ignores this line. the format of the `sortlist` command is:

  `sortlist` *preferred_servers*

named.pid

When `named` successfully starts, it writes its process id into the /etc/named.pid file. In addition to `named.reload` and `named.restart`, programs that need to send signals to `named` use this file. You can change the name of this file by defining the `PIDFILE` parameter before compiling `named`. (Refer to "Building a System with a Name Server.")

named.reload

The shell script, /usr/etc/named.reload, causes the server to read the boot file and reload the database. All previously cached data is lost.

If `named.reload` does not exist on your system, create it as shown in Figure 9-3.

Figure 9-3
Sample named.reload File

```
#!/bin/csh
#
# named.reload
#
kill -HUP `cat /etc/named.pid`
```

named.restart

The shell script, /usr/etc/named.restart, kills the active `named` process, and causes the server to start without reloading the database. Previously cached data is preserved.

If `named.restart` does not exist on your system, create it as shown in Figure 9-4.

Figure 9-4
Sample named.restart File

```
#!/bin/csh
#
# named.restart
#
kill -9 `cat /etc/named.pid`
/usr/etc/named
```

resolv.conf

To set up a host that will use a remote server instead of a local server to answer queries, create the file, /etc/resolv.conf. This file specifies the name servers on the network that should be sent queries. You should always create the resolv.conf file to include, at minimum, the `domain` directive.

The resolv.conf file may contain two types of directives: `domain` and `nameserver`. The `domain` directive allows you to specify the default domain name to append to names that are not fully qualified.

The `nameserver` directive allows you to specify the internet address of a name server for the local resolver to query. If no `nameserver` is specified, the system uses the name server on the local machine.

use_nameserver

The existence of this file on a system activates the name server resolver for that system. Create /etc/use_nameserver by using the `touch` command. To deactivate the name server, remove this file. (The /etc/use_nameserver file is discussed further under "Building a System with a Name Server.")

Note

In the ConvexOS implementation, /etc/use_nameserver is the key to using the name server. Create it to activate the server and remove it to deactivate the server.

Data File Descriptions

The name server processes two kinds of data: *authoritative* and *cached*. Authoritative data is the complete database for a particular domain over which the name server has authority. The name server loads this data from master files or obtains it from another name server. A local *resolver* acquires and maintains cached data.

Three standard files specify data for a domain: named.local, named.hosts, and named.rev. Name server data files contain directives and resource records. A description of each of these files, as well as the name server cache file, root.cache, follows the discussion on standard resource record formats.

Data File Directives

In addition to records in the standard resource record format, name server data files may contain the following directives:

- **$INCLUDE**—Requests that the name server load the data in the named file. This directive begins with `$INCLUDE` in column 1, followed by the name of files to load.

  This feature is primarily used to separate different types of data into multiple files. For example, you might want to keep mailbox data separate from host data. You could do this by specifying

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

  The name server interprets this line as a request to load the /usr/named/data/-mailboxes file. `$INCLUDE` does not cause data to be loaded into a different domain.

- **$ORIGIN**—Change the origin in a data file. This directive begins with `$ORIGIN` in column 1, followed by a domain origin. The `$ORIGIN` feature is used for putting more than one domain in a data file. Each domain in the file should have a corresponding origin specified for it.

Standard Resource Record Format

Data records in the name server data files are called resource records (RR). The standard resource record format is specified in detail in RFCs. The following is a general description of resource records:

[record_name] ttl addr_class record_type record_specific_data

where:

record_name	Specifies the name of the domain record. If present, it must start in column 1. If you leave the name blank, the record takes on the name of the previous RR.
ttl	Specifies an optional time-to-live value. This value defines how long data will be stored in the database. If you leave this field blank, the name server uses the default time-to-live specified on the *Start Of Authority* (SOA) resource record.
addr_class	Specifies the address class. Currently, only the internet (IN) class is supported.
record_type	Specifies the resource record type. See "Record Types," below.
record_specific_data	Contents depend on record type. Case is preserved in names and data fields. All comparisons and lookups in the name server database are case insensitive.

The following characters have special meanings:

.	In the name field, a "." refers to the current domain.
@	In the name field, an "@" sign denotes the current origin.
..	In the name field, two dots represent the null domain name of the root.
\X	Where X is any character other than 0-9. A backslash tells the name server to ignore any special meaning X normally carries. For example, "\." can be used to place a dot character in a label.
\DDD	Where each D is a decimal digit. The resulting octet is assumed to be text and is not checked for special meaning.
( )	Used to group data specified on more than one line. In effect, line terminations are not recognized within parentheses.
;	Signals the start of a comment; the remainder of the line is ignored.
*	Signifies use of wildcards.

The current origin is 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 if you do not want this to happen. In general, if the name is not in the domain for which you are creating the data file, end the name with a ".".

Resource Record Types

This section describes and gives examples of each type of resource record. Files are listed in the order in which they normally appear in data files.

- **Start of Authority (SOA)**—Designates the beginning of a domain. There should be only one SOA record per domain. Figure 9-5 shows the format and an example of an SOA record.

Figure 9-5
Start of Authority (SOA) Record

```
[name] ttl   class  SOA    origin/data    administrator

;
@            IN     SOA    mars.acme.com.root.mars.acme.com(
                    1.1      ; serial
                    3600     ; refresh
                    300      ; retry
                    3600000  ; expire
                    3600 )   ; minimum
```

name	Domain name.
origin	Domain name of host on which this data file resides.
administrator	Address of person responsible for this name server.
serial	Data file version number. Increment this number whenever you make a change to the data.
refresh	Interval, in seconds, at which a secondary name server checks with the primary name server to see if it needs to update its data.
retry	Interval, in seconds, at which a secondary server retries after failing to get a response when checking for a refresh.
expire	Maximum duration in seconds that a secondary name server uses its data after it fails to get a refresh.
minimum	Default, in seconds, for the time-to-live interval.

- **Name Server (NS)**—Specifies the name of the server that has authority over the domain. There should be one NS record for each primary master server for a domain. Figure 9-6 shows the format and an example of an NS record.

Figure 9-6
Name Server (NS) Record

```
[name] ttl    class NS      server name

;
              IN    NS      mars.acme.com.
```

server name Name of the primary master server for the domain.

- **Address (A)**—Specifies the internet address(es) of a given host. There should be one A record for each network address assigned to the host. A host with multiple network interfaces will have multiple A records. Figure 9-7 shows the format and examples of the A record.

Figure 9-7
Address (A) Record

```
[name] ttl    class A       address

;
mars          IN    A       128.32.0.4
              IN    A       10.0.0.78
```

name Host name. The first A record for a host must include the host name.

address Internet address of the host.

- **Host Information (HINFO)**—Specifies host-specific information. There should be one HINFO record for each host. Host name and address are specified on a preceding A record. Figure 9-8 shows the format and examples of HINFO records.

Figure 9-8
Host Information (HINFO) Record

```
[name] ttl    class HINFO   host-specific information

;
altair        IN    A       140.150.70.1
              IN    HINFO   VAX-11/780 UNIX
deneb         IN    A       140.150.70.2
              IN    HINFO   C240 ConvexOS
sirius        IN    A       140.150.70.4
              IN    HINFO   machine_room
```

The first two HINFO records in the above example specify the hardware and operating system of the listed hosts. Only a single space separates the hardware and the operating system information. If you want to include a space in the machine name, you must quote the name. The third HINFO record specifies the location of the host. You can use HINFO to specify any information you want to associate with the host.

- **Well-Known Services (WKS)**—Lists well-known services supported by a particular protocol at a specific address. The list of services and port numbers are defined in the /etc/services file. There should be only one WKS record per protocol per address. Figure 9-9 shows the format and examples of the WKS record.

Figure 9-9
Well-Known Services (WKS) Record

```
[name] ttl  class  WKS  address      prot  services

;
            IN     WKS  128.32.0.10  UDP   who route timed
            IN     WKS  128.32.0.10  TCP   ( echo telnet
                                            discard daytime
                                            netstat ftp auth
                                            time whois finger
                                            smtp hostname )
;
```

- **Canonical Name (CNAME)**—Specifies an alias for a canonical name. A CNAME record should be the only record associated with the alias name. All other resource records that include a domain name as their value (for example, NS or MX) should be associated with the canonical name, not with the alias.

  Figure 9-10 shows the format and an example of a CNAME record.

Figure 9-10
Canonical Name (CNAME) Record

```
aliases ttl   class CNAME  canonical name

;
cave          IN    CNAME  saturn
```

- **Domain Name Pointer (PTR)**—Allows special names to point to some other location in the domain. PTR names should be unique within a domain.

  Figure 9-11 shows the format and an example of a PTR record.

Figure 9-11
Domain Name Pointer (PTR) Record

```
name     ttl   class PTR    domain name

;
70.1          IN    PTR    saturn.acme.com.
```

In the above example, a PTR record is used to set up reverse addresses for the special `IN-ADDR.ARPA` domain. The record in the example is from the named.rev file shown in Figure 9-19. Host `saturn` has an internet address of `140.50.70.1`; the PTR record points to subdomain `70.1`.

- **Mailbox (MB)**—Designates the host on which a user wishes to receive mail. Mailbox names should be unique within the domain.

  Figure 9-12 shows the format and an example of an MB record.

Figure 9-12
Mailbox (MB) Record

```
name      ttl    class MB     host

;
jones             IN    MB     mars.acme.com.
```

name User's login name.

host Denotes the machine to which the server is to deliver the user's mail.

- **Mail Rename (MR)**—Associates aliases with a user name. Figure 9-13 shows the format and an example of an MR record.

Figure 9-13
Mail Rename (MR) Record

```
alias     ttl    class MR     MB name

;
postman           IN    MR     jones
```

alias An alias for the user name listed in the fourth field.

MB name Name on the corresponding MB record.

- **Mailbox Information (MINFO)**—Creates a mail group for a mailing list. This record is usually associated with the MG (mail group), but may be used with an MB record. Figure 9-14 shows the format and an example of a MINFO record.

Figure 9-14
Mailbox Information (MINFO) Record

```
name      ttl    class MINFO  requests      maintainer

;
BIND              IN    MINFO  bind-request  jj.acme.com.
```

name Name of the mailbox.

requests Name to which mail such as requests to be added to a mail group should be sent.

maintainer Mailbox to which the server should send error messages. This is particularly appropriate for mailing lists when errors in members' names should be reported to a person other than the sender.

- **Mail Group Member (MG)**—Specifies members of a mail group. Figure 9-15 shows the format of the MG record and a sample mailing list.

Figure 9-15
Mail Group Member (MG) Record

```
[group]  ttl    class MG     member name

;
Bind             IN    MINFO  bind-request     jj.acme.com.
                 IN    MG     ralph.acme.com.
                 IN    MG     barney.acme.com.
                 IN    MG     uriah.acme.com.
                 IN    MG     jones.acme.com.
                 IN    MG     white.acme.com.
```

- **Mail Exchanger (MX)**—Designates a host to serve as a gateway. Figure 9-16 shows the format and sample MX records.

Figure 9-16
Mail Exchanger (MX) Record

```
name      ttl    class MX   preference   mail gateway

;
munnari.oz.au  IN    MX    0            seismo.css.gov.
*.IL.          IN    MX    0            relay.cs.net.
```

preference Order in which a mailer should attempt to forward mail when there is more then one route to a single machine. (Refer to RFCs for more detailed information.)

The first MX record in the example above designates `seismo.css.gov.` as a mail gateway to `munnari.oz.au`. This MX record allows hosts that are not directly connected to the same network as `munnari.oz.au` to forward mail through the gateway, `seismo.css.gov`. Hosts `seismo` and `munnari` may have a private connection or use a different transport medium.

The second record in the example above shows the use of wildcard names for mail routing with MX records. There are usually servers on the network that simply state that any mail to a domain is to be routed through a relay. In the example, all mail to hosts in the domain `IL` is routed through `relay.cs.net`. This is done by creating a wildcard record that states that *.IL has an `MX` of `relay.cs.net`.

/etc/hosts

When you configure your system to use the name server, the /etc/hosts file is only used for setting interface addresses, and at times when the name server is not running. Therefore, /etc/hosts need only contain addresses for local hosts.

Set up the /etc/hosts file for the hosts on your LAN according to procedures outlined in Chapter 3, "Setting Up the Host Name Database."

named.hosts

The named.hosts file contains all authoritative data for hosts in a domain. You specify the name of this file in the boot file; its location is relative to the pathname specified with the `directory` command in the boot file. Records in the file follow the standard resource record format.

Figure 9-17 shows an example of a named.hosts file.

Figure 9-17
Sample named.hosts File

```
; named.hosts

@           IN    SOA pluto.acme.comroot.pluto.acme.com. (
                  1.1     ; Serial
                  10800   ; Refresh 3 hours
                  3600    ; Retry 1 hour
                  3600000; Expire 1000 hours
                  86400 ); Minimum 24 hours
            IN    NS      pluto.acme.com.

localhost   IN    A       127.0.0.1
saturn      IN    A       140.50.70.1
            IN    HINFO   machine_room c1e22
cave        IN    CNAME   saturn
timehost    IN    CNAME   saturn
andrewhost  IN    CNAME   saturn
antares     IN    A       140.50.70.2
            IN    HINFO   machine_room c1e22
fontshost   IN    CNAME   antares
localserver IN    CNAME   antares
snoopy      IN    A       140.50.70.4
            IN    HINFO   cbrown c2w47-1
solo        IN    A       140.50.70.100
            IN    HINFO   jinx c1e70-r
hercules    IN    A       140.50.70.101
            IN    HINFO   ralph c1e69-1
harpo       IN    A       140.50.70.102
            IN    HINFO   adolph c1e69-r
chocolate   IN    A       140.50.70.106
            IN    HINFO   grant c2w46-1
choc        IN    CNAME   chocolate
cumulus     IN    A       140.50.70.107
            IN    HINFO   doobey c2w44-1
daffy       IN    A       140.50.70.108
            IN    HINFO   binaca c2w44-r
holmes      IN    CNAME   daffy
shylock     IN    A       140.50.70.109
            IN    HINFO   bluto c1e70-1
veggy       IN    A       140.50.70.113
            IN    HINFO   katey c2w47-r
vege        IN    CNAME   veggy
inanna      IN    A       140.50.71.174
            IN    HINFO   jimjones c2w34-r
juno        IN    A       140.50.71.177
            IN    HINFO   klutz c2w41-r
eddie       IN    CNAME   juno
jambox      IN    A       140.50.73.1
            IN    HINFO   machine_room c1e22
            IN    MX      0       jambox
postmaster  IN    MR      root
;
```

named.local

The named.local file specifies the network address for the local loopback interface. By convention, the loopback is usually named `localhost` and has network address `127.0.0.1`. You specify the name of this file in the boot file; its location is relative to the pathname specified with the `directory` command. Data records in the file follow the standard resource record format.

Figure 9-18 shows an example of a named.local file.

Figure 9-18
Sample named.local File

```
; named.local

@       IN    SOA   pluto.acme.com. root.pluto.acme.com. (
                    1.1 ; Serial
                    3600 ; Refresh
                    300 ; Retry
                    3600000 ; Expire
                    14400 ) ; Minimum
        IN    NS    pluto.acme.com.

1       IN    PTR   localhost.
```

named.rev

The named.rev (reverse) file specifies host addresses in the IN-ADDR.ARPA domain. This is a special domain for allowing internet address-to-name mapping. Because 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, which correspond to the four octets of an internet address. You must specify all four octets even if an octet is zero. Internet address 128.32.0.4 is located in the domain, 32.128.IN-ADDR.ARPA. This reversal of the address is awkward to read but allows for natural grouping of hosts in a network.

You specify the name of this file in the boot file; its location is relative to the pathname specified with the `directory` command. Data records in the file follow the standard resource record format.

Local hosts will be listed in the named.rev file as well as the named.hosts file.

Figure 9-19 shows an example of a named.rev file. Data records in the named.rev file correspond to entries in the named.hosts file shown in Figure 9-17.

Figure 9-19
Sample named.rev File

```
; named.rev
;
@       IN   SOA    pluto.acme.com. root.pluto.acme.com.(
                    1.1             ; Serial
                    10800           ; Refresh 3 hours
                    3600            ; Retry 1 hour
                    3600000         ; Expire 1000 hours
                    86400 )         ; Minimum 24 hours
        IN   NS     pluto.acme.com.
;
1.70    IN   PTR    saturn.acme.com.
2.70    IN   PTR    antares.acme.com.
4.70    IN   PTR    snoopy.acme.com.
100.70  IN   PTR    solo.acme.com.
101.70  IN   PTR    hercules.acme.com.
102.70  IN   PTR    harpo.acme.com.
106.70  IN   PTR    chocolate.acme.com.
107.70  IN   PTR    cumulus.acme.com.
108.70  IN   PTR    daffy.acme.com.
109.70  IN   PTR    shylock.acme.com.
113.70  IN   PTR    veggy.acme.com.
174.71  IN   PTR    inanna.acme.com.
177.71  IN   PTR    juno.acme.com.
1.73    IN   PTR    jambox.acme.com.
;
```

root.cache

The root.cache file is used to prime the name server's cache with addresses of authoritative name servers for the root domain of the network. You specify the name of this file in the boot file; its location is relative to the pathname specified with the `directory` command. Data records in the file follow the standard resource record format.

Figure 9-20 shows an example of a root.cache file.

Figure 9-20
Sample root.cache File

```
; 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    BRL-AOS.ARPA.
                99999999 IN  NS    GUNTER-ADAM.ARPA.
                99999999 IN  NS    C.NYSER.NET.
;
; Prep the cache (hotwire the addresses). Order does not
matter
;
NS.NIC.DDN.MIL. 99999999 IN  A     192.67.67.53
SRI-NIC.ARPA.   99999999 IN  CNAME NS.NIC.DDN.MIL.
A.ISI.EDU.      99999999 IN  A     26.3.0.103
                99999999 IN  A     128.9.0.107
BRL-AOS.MIL.    99999999 IN  A     128.20.1.2
                99999999 IN  A     192.5.25.82
C.NYSER.NET.    99999999 IN  A     192.33.4.12
GUNTER-ADAM.ARPA.99999999IN A     26.1.0.13
NS.NASA.GOV.    99999999 IN  A     128.102.16.10
                99999999 IN  A     192.52.195.10
TERP.UMD.EDU.   99999999 IN  A     128.8.10.90
;
```

Time-to-live values in the example are set very high to ensure that data is not discarded because of a time-out.

Building a System with a Name Server

BIND software consists of two parts. A group of user interface routines, collectively called the `resolver`, resides in the C library, /lib/libc.a. The resolver builds query packets and exchanges them with the name server. The actual name server, `named`, 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.

Resolver Routines in libc

The C library uses either name server resolver routines or host table lookup routines to resolve host names and addresses.

If the file, /etc/use_nameserver, exists, networking software uses the name server to resolve host names. Otherwise, the host table lookup method of name resolution will be used.

Setting up Your Own Domain

When you set up a domain that is going to be on a public network, your 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.

Contacts are as follows:

- **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, a mail group for people on the Internet who run BIND. The group discusses future design decisions, operational problems, and other related topics. The network address to contact to request being added to this mailing list is:

  `bind-request@ucbarpa.Berkeley.EDU.`

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

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

Name Server Configuration

This section describes procedures for configuring each type of name server and gives examples of files.

The exact steps you perform in configuring the name server depend upon the type of server—primary master, secondary master, caching-only, or resolving-only—that will run on the host on which you are installing it.

To configure a primary master, secondary master, or caching-only server, perform the following steps:

- ❑ Set up the boot file for your particular configuration.
- ❑ Create named.local.
- ❑ Add hosts to named.hosts and named.rev.
- ❑ Create resolv.conf.
- ❑ Add commands to the rc.local file to start the name server.
- ❑ Activate the name server.

To configure a resolving-only server, you need only create resolv.conf and activate the name server. The sections that follow describe the outlined tasks in detail.

Configuring a Primary Master Server

Follow steps outlined in this section to configure a primary master server. The directory, /usr/lib/conf/bind/setup, contains scripts and files to help you install a primary master server.

1. Change directory into /usr/lib/conf/bind/setup.
2. Read the comments at the beginning of the Makefile, then edit the necessary definitions to tailor the file to your system.

   Figure 9-21 shows the first few lines of the Makefile installed with the software.

Figure 9-21
Sample Primary Master Server Makefile

```
#
HOSTSRC     = cat /etc/hosts
DESTDIR     = /usr/etc/domain
DOMAIN      = berkeley.edu
HOSTADDR    = 128.32.140.6
HOSTNAME    = monet
ROOT        = root
#
```

Customize the following definitions:

HOSTSRC—Enter the name of the file containing host data. Set HOSTSRC to cat /etc/hosts.

DESTDIR—Enter the pathname of the directory to contain all configuration files except named.boot. This is the pathname you also specify in the boot file with the directory command.

DOMAIN—Enter the name of the default domain, for example, acme.com.

HOSTADDR—Enter the internet address of the local host.

HOSTNAME—Enter the name of the local host.

ROOT—Enter the login name of the network administrator.

3. Create customized name server configuration files by entering the following commands:

```
make clean
make
```

These commands use the definitions in the Makefile to create customized named.boot, named.hosts, named.local, named.rev, and resolv.conf files from the host database contained in the /etc/hosts file.

Note

The Makefile is only an aid to creating the necessary files. After running it, look closely at the files it generates to ensure that all information is correct for your site.

4. Ensure that `syslog` is logging `LOG_DAEMON` messages at the `LOG_DEBUG` level. Check /etc/syslog.conf—`named` error messages are probably directed to /usr/adm/log/daemonlog. If /etc/syslog.conf does not contain an entry for these messages, configure `syslog` to log them. *Managing ConvexOS: Configuration Guide* contains detailed information on configuring the /etc/syslog.conf file.

5. At this point, you are ready to install the configuration files and start up the name server. Enter the following commands (you must be running as `root` to complete this step):

```
make install
/usr/etc/named
```

If your system is running the `share` scheduler, the last command will be

```
slxqt network -x /usr/etc/named
```

6. Use the `ps` command to verify that `named` is running.

7. Check the `syslog` file for an entry indicating that `named` was restarted.

8. Run the script, /usr/etc/named.reload, to cause the server to reload its database. You should see a message in the `syslog` file indicating that the server has been reloaded.

9. Send the name server a `SIGINT` signal to cause it to dump its cache and database to the file, /usr/tmp/named_dump.db. Verify that this file contains data that resembles configuration data the named.hosts file.

10. Try using the `ftp` command to communicate with a host you know is up and running and exists in your /etc/hosts file. Because the name server is not active yet, you should have no problem using `ftp`. If when you execute `ftp` to a known host, you get the initial message, "Connected to <hostname>," you can be reasonably sure of the connection.

11. Activate name server usage by creating the file, /etc/use_nameserver. Use the `touch` command as follows:

    `touch /etc/use_nameserver`

12. Try using the `ftp` command again. The initial message should now read similar to "Connected to <hostname.domain>."

13. To further verify that `named` is working, run the utility, `nslookup`, with no options. To do so, enter

    `/usr/etc/nslookup`

    Try a few nslookup commands, as follows:

    —Enter a few names of known hosts, one at a time. Make sure to try a few hosts that are in the /etc/hosts file and a few that are not. In response, you should receive the fully-qualified host name and its internet address.

    —Try using the `finger` command. The output displayed should be the same as if you had used `rlogin` on the host, then executed `finger`.

    —Enter a command similar to

    `ls -d` *acme.com* > *output_file*

    to dump the database to *output_file*. This file should resemble the database dumped in step 9, above.

    — Exit `nslookup` by typing **CTRL-D**.

14. Try using `/usr/etc/ping` to test the connections to several different hosts, both with the standard name and the fully-qualified name.

15. Send the server a `SIGIOT` signal to cause it to dump statistics to the file, /usr/tmp/named.stats.

16. Set up your rc.local file to start `named` whenever the system is initialized.

Note

You must start named before starting any utilities that depend on its services.

Figure 9-22 shows a partial /etc/rc.local file containing commands to start the name server.

Figure 9-22
Starting named from rc.local

```
# redirect all output to the console
exec > /dev/console 2>&1
PATH=/etc:/usr/etc:/bin:/usr/ucb:/usr/bin:/usr/acme
export PATH

/bin/rm -f /etc/use_nameserver

/bin/hostname pluto.acme.com

if [ -f /bin/domainname ]; then
        /bin/domainname ""      only set if yellow pages is used
fi

if [ -f /etc/ifconfig ]; then
   /etc/ifconfig ex0 `/bin/hostname` up arp netmask 0xffffff00
        #
        # all physical interfaces MUST be configured
        # before lo0 is configured
        #
        /etc/ifconfig lo0 localhost up
fi

#syslogd must be started before any other daemons
if [ -f /usr/etc/syslogd ]; then
        rm -f /dev/log
        /usr/etc/syslogd & echo 'starting system logger'
fi

# start up named now
if [ -f /etc/named.boot ]; then
        touch /etc/use_nameserver
        /usr/etc/named & echo -n ' started named'
fi
.
.
.
(rest of the file)
```

Note

To deactivate the name server, enter

```
rm -f /etc/use_nameserver
```

Figure 9-23 shows the typical contents of a named.boot file for a primary master server.

Figure 9-23
Boot File for a Primary Master Server

```
; boot file for authoritative master name server for
; Berkeley.EDU. There should be one primary entry
; for each SOA record.
;
sortlist 10.0.0.0

directory   /usr/local/domain

; type      domain                  source host/file

primary     Berkeley.EDU            berkeley.zone
primary     32.128.IN-ADDR.ARPA     berkeley.rev
primary     0.0.127.IN-ADDR.ARPA    localhost.rev
cache       .                       root.cache

;
```

Figure 9-24 shows the contents of a short named.hosts file for a primary master server.

Figure 9-24
Hosts File for a Primary Master Server

```
; named.hosts

@           IN   SOA  pluto.acme.com. root.pluto.acme.com. (
                      1.1       ; Serial
                      10800     ; Refresh 3 hours
                      3600      ; Retry 1 hour
                      3600000   ; Expire 1000 hours
                      86400 )   ; Minimum 24 hours
                IN    A         140.50.73.4
                IN    NS        pluto.acme.com.

localhost       IN    A         127.0.0.1
saturn          IN    A         140.50.70.1
                IN    HINFO     machine_room c1e22
cave            IN    CNAME     saturn
antares         IN    A         140.50.70.2
                IN    HINFO     machine_room c1e22
snoopy          IN    A         140.50.70.4
                IN    HINFO     covue_vax_ultrix c2w47-1
solo            IN    A         140.50.70.100
                IN    HINFO     dpz c1e70-r
postmaster      IN    MR        root
```

Figure 9-25 shows the named.rev file that corresponds to the named.hosts file in Figure 9-24.

Figure 9-25
named.rev File for a Primary Master Server

```
; named.rev

@           IN    SOA pluto.acme.com. root.pluto.acme.com. (
                   1.1     ; Serial
                   10800   ; Refresh 3 hours
                   3600    ; Retry 1 hour
                   3600000 ; Expire 1000 hours
                   86400 ) ; Minimum 24 hours
              IN   NS      pluto.acme.com.

1.70          IN   PTR     saturn.acme.com.
2.70          IN   PTR     antares.acme.com.
4.70          IN   PTR     snoopy.acme.com.
100.70        IN   PTR     solo.acme.com.
```

Figure 9-26 shows the contents of the named.local file for `pluto`.

Figure 9-26
named.local File for a Primary Master Server

```
; named.local

@     IN   SOA      pluto.acme.com.  root.pluto.acme.com. (
           1.1      ; Serial
           10800    ; Refresh 3 hours
           3600     ; Retry 1 hour
           3600000  ; Expire 1000 hours
           86400 )  ; Minimum 24 hours
      IN   NS       pluto.acme.com.

1     IN   PTR      localhost.
```

Configuring a Secondary Master Server

Configuring a secondary master server requires fewer steps than configuring a primary master. Basically, all you have to do is set up the boot and data files, and activate the server. Follow the procedure outlined below.

The directory, /usr/lib/conf/bind contains example files you can customize for your system.

1. Create the file, named.boot, on directory /etc. As installed, your system may already have a copy of /etc/named.boot that you can modify. If not, copy the file from /usr/lib/conf/bind to /etc, then customize it for your server configuration.Figure 9-27 shows a typical named.boot file for a secondary master server.

Figure 9-27
Boot File for a Secondary Master Server

```
; named.boot
;
; boot file for secondary name server
; There should be one primary entry for each SOA record.
;
sortlist 10.0.0.0

directory   /usr/local/domain

; type      domain                source host/file          backup file

secondary   Berkeley.EDU          128.32.137.8 128.32.137.3 ucbhosts.bak
secondary   32.128.IN-ADDR.ARPA   128.32.137.8 128.32.137.3 ucbhosts.rev.bak
primary     0.0.127.IN-ADDR.ARPA  localhost.rev
cache       .                     root.cache
```

2. Create the directory you specified in the boot file if it does not already exist.
3. Copy the sample data files, named.local and root.cache, from /usr/lib/conf/bind to the directory you specified in the boot file.
4. Modify the named.local file you copied into the data directory, changing the host name to the name of the host on which you are configuring the server. For example, the named.local file for the local host, mars, is as shown in Figure 9-28.

Figure 9-28
named.local File for a Secondary Master Server

```
; named.local

@       IN  SOA     mars.acme.com.   root.mars.acme.com. (
            1.1     ; Serial
            10800   ; Refresh 3 hours
            3600    ; Retry 1 hour
            3600000 ; Expire 1000 hours
            86400 ) ; Minimum 24 hours
        IN  NS      mars.acme.com.
```

5. Activate name server usage by creating the file, /etc/use_nameserver. Use the `touch` command as follows:

```
touch /etc/use_nameserver
```

6. Modify the rc.local file to start named when the system is booted, as shown in Figure 9-22.

Note

To deactivate the name server, enter

```
rm -f /etc/use_nameserver
```

Configuring a Caching-Only Server

To configure a caching-only server, follow the procedure outlined below.

1. The directory, /usr/lib/conf/bind contains example files you can customize for your system.

   Create the file, named.boot, in directory /etc. As installed, your system may already have a copy of /etc/named.boot that you can modify. If not, copy the file from /usr/lib/conf/bind to /etc, then customize it for your server configuration. Figure 9-29 shows a typical named.boot file for a caching-only server.

Figure 9-29
Boot File for a Caching-Only Server

```
; named.boot
;
; boot file for caching-only name server
;
directory   /usr/local/domain

; type      domain                source host/file           backup file

cache .                           root.cache
primary     0.0.127.IN-ADDR.ARPA  named.local
secondary   acme.com              128.32.137.8 128.32.137.3  hosts.bak
secondary   32.128.IN-ADDR.ARPA   128.32.137.8 128.32.137.3  hosts.rev.bak
forwarders                        128.32.137.8 128.32.137.3
```

2. Complete configuration by following steps 2-6 in the "Configuring a Secondary Master Server" section.

Note

To deactivate the name server, enter

```
rm -f /etc/use_nameserver
```

Configuring a Resolving-Only Server

To configure a resolving-only server, follow the procedure outlined below.

1. Create the file, resolv.conf, on directory /etc. As installed, your system may already have a copy of /etc/resolv.conf that you can modify. If not, create it and add lines like those shown in Figure 9-30.

Figure 9-30
resolv.conf File for a Resolving-Only Server

```
# /etc/resolv.conf
#
# designate the alternate name servers
#
nameserver   128.109.178.1
nameserver   129.109.193.1
nameserver   128.109.130.3
```

2. Activate name server usage by creating the file, /etc/use_nameserver. Use the `touch` command as follows:

```
touch /etc/use_nameserver
```

3. Complete configuration by modifying rc.local as described in the "Configuring a Primary Master Server" section.

Note

To deactivate the name server, enter

```
rm -f /etc/use_nameserver
```

Adding Hosts to the Name Server Database

To add hosts to the database for a primary master server, follow the steps outlined below:

1. Edit the named.hosts file as follows:

   For each host, add the host name, all of its network addresses, its host information record, and common aliases for it. The `WKS` records list optional well-known services.

   This is the entry for the host, `calder`.

```
calder     IN   A       128.32.140.1
           IN   A       128.32.129.3
           IN   WKS     128.32.0.12   tcp telnet ftp smtp echo
           IN   WKS     128.32.0.12   udp echo time tftp
           IN   HINFO   jones
ucbcalder IN   CNAME   calder
```

   For the machine you are adding,

   —Replace `calder` with the new host name.

   —Replace `128.32.140.1` with the internet address of the new machine.

   —If there are multiple addresses for the host, add the other address lines.

2. Edit the named.rev file as follows:

   For each address assigned to a machine, enter the address in reverse notation. For example, the lines for `calder` are as follows:

```
12.0    IN        PTR   calder.acme.com.
3.129   IN        PTR   calder.acme.com.
```

Note

You must include the final period on the host name.

3. Increment the serial numbers on both the above files. If you use `rcs`, arrange for this to happen automatically when changes are checked in.
4. Reload name server data by running the script, /usr/etc/named.reload.

When you have completed the steps above, the name server should be configured and running properly.

Configuring Serial Line Internet Protocol (SLIP)

10

The *Serial Line Internet Protocol* (SLIP) provides point-to-point communication over asynchronous serial lines. SLIP is commonly used on dedicated serial lines and dial-up lines. By using SLIP, you can achieve moderate network performance without the need for high-speed network hardware. Most standard network applications use SLIP transparently (though slowly) as they would a LAN.

This chapter describes the steps necessary to configure SLIP. Perform the procedure described here only if you intend to run TCP/IP over serial lines.

Prerequisites

Before performing the procedure below, you must establish a point-to-point connection between two hosts. A connection may be established via an autodial modem, a dedicated line, or by manually dialing the destination host.

TCP/IP protocols view SLIP as simply a network interface. As with a network interface, you must configure SLIP before running network utilities, such as `ftp`, over it.

Use the procedure below to configure the local end of the point-to-point connection. The other end of the serial line must also be connected to a SLIP implementation.

Configuring SLIP

To configure SLIP, perform the following tasks:

1. Identify source and destination network interfaces.
2. Attach a tty line to the source network interface.

Identifying Interfaces

Add an entry to the host database for the interface at each end of the connection. To do so, update the /etc/hosts file and, optionally, configure the name server to provide the address.

Figure 10-1 on the following page shows an example of an /etc/hosts file with hosts defined for SLIP.

Figure 10-1
Sample /etc/hosts File for Use with SLIP

```
# Host Database
#
127.1    localhost
#
# Serial Line Interface
#
100.50.0.1 jupiter-slip# local end of connection
#
140.55.0.10sirius-slip# remote end of connection
#
```

Attaching tty Lines

After you have associated a host name with each end of the SLIP connection, assign a tty to the local end by using `slattach`.

Note

You must ensure that `getty` is not running for the tty you have chosen. Turn off `getty` for that tty by entering `off ttyXX`.

Only the superuser, `root`, can use `slattach`. The syntax of the command is:

`/etc/slattach` *ttyname src-name dst-name* [*baudrate*]

where:

ttyname	A string in the form `ttyXX` or `/dev/ttyXX`.
src-name	Source (local) host name as defined in the host database.
dst-name	Destination (remote) host name as defined in the host database.
baudrate	Speed of the connection. The default is 9600.

For host names defined in the /etc/hosts file in Figure 10-1, the `slattach` command would be

```
/etc/slattach jupiter-slip sirius-slip 19200
```

`slattach` has additional options. Refer to the `slattach`(8) man page for more details.

When you run `slattach`, the system responds with a message similar to the following:

```
slattach: unit sl0 assigned to /dev/tty02
```

At this point, you should be able to use the link the same as any other network connection.

To detach the tty from the interface when you are through, kill the `slattach` process.

Testing and Troubleshooting the Configured Network

11

Once you have properly configured the network and its supporting software , you should experience few serious problems. Often, you find that the problems reported are largely the result of user error. A user trying to connect to a nonexistent host, for example, may reach the conclusion that `ftp` is broken when the connection times out. Or, the user may decide the network is down when trying to connect to a host that is not running. Most other problems—runaway jobs, missing daemons, lack of kernel memory, bad connections, unplugged transceivers—can be located by using the procedures discussed in this chapter.

If you have installed an UltraNet network, you can use all the functions provided by `netstat` to troubleshoot the internet interface. `netstat` does not report the status of native UltraNet networks, however, so you must use the `unetcf stat` command or the `tsock` network exerciser to isolate communication failures on that interface. Because the two interfaces share a common path through much of the software, as well as a common adapter and device-driver, you can often locate the problem by using the "General Troubleshooting Procedure" outlined below. If the internet interface checks out, it is then time to use `tsock` as described in the section, "Using tsock."

Testing the Configured Network

The simplest way to test the network is to use `ping`. `ping` tests connections with other hosts by sending them a particular type of datagram, called an ICMP (Internet Control Message Protocol) ECHO. These datagrams work well for testing connections because they are simply bounced back by the other host once they are received.

To test a connection to another host, start the transmission stream by invoking `ping`, and then check to make sure that packets are being returned. If at least 95 percent of the packets are being returned, you know that the network is functioning properly.

Use `ping` as follows:

`/usr/etc/ping` *remote_host_name*

where *remote_host_name* is the name of the remote machine with which you want to communicate. For example, to check whether the connection between `veeger` and `jupiter` is viable, use `ping` on `veeger` as follows:

```
/usr/etc/ping jupiter
```

Sample output is shown in Figure 11-1.

Figure 11-1
Sample ping Output

```
PING jupiter: 56 data bytes
64 bytes from 140.50.0.1: icmp_seq=0. time=40. ms
64 bytes from 140.50.0.1: icmp_seq=1. time=40. ms
64 bytes from 140.50.0.1: icmp_seq=2. time=20. ms
64 bytes from 140.50.0.1: icmp_seq=3. time=20. ms
64 bytes from 140.50.0.1: icmp_seq=4. time=20. ms
64 bytes from 140.50.0.1: icmp_seq=5. time=30. ms
64 bytes from 140.50.0.1: icmp_seq=6. time=30. ms
64 bytes from 140.50.0.1: icmp_seq=7. time=30. ms
----veeger-il PING Statistics----
8 packets transmitted, 8 packets received, 0% packet loss
round-trip (ms) min/avg/max = 20/29/40
```

In this example, eight packets were received from network address 140.50.0.1 with transmission times ranging from 20 to 40 milliseconds. No packets were lost; therefore, you can assume that the link between the hosts is viable. If no packets had been returned, or if there had been a high rate of packet loss, you would have been correct in assuming a problem in the line between hosts. Return-trip-time statistics included in the output allow you to determine whether transmission times are prohibitive.

General Troubleshooting Procedure

Problems detected by `ping` should be dealt with according to the procedure outlined below.

1. First, check running processes with the `ps` command. You may notice a network process at the top of the list. If so, that process is probably a runaway flooding the network with "renegade" packets. During a crisis of this sort, users may experience problems ranging from sluggish response to a lack of memory buffers for network programs. As a check, you may want to run `syspic` to see if the CPU is being consumed by the runaway process. Of course, when a runaway process is identified, the solution is obvious and simple: kill the process, then use `netstat` to make sure that the network is again running normally.

2. As a second step, run `netstat`. Because the options available enable you to look at all aspects of the network from memory use to I/O throughput, `netstat` is your best way to gain information about how the network is running. (Instructions for using `netstat` are included in the next section, "Using netstat.")

3. If `netstat` provides clues but no solutions, run `ping` to isolate problems. If you cannot successfully reach another host by using `ping`, you can be confident that there is a problem in the transmission media or a software problem on either the remote host or the local host.

4. If you suspect software problems on either the local host or the remote host,
   — Verify contents of the /etc/hosts, /etc/networks, and /etc/rc.local files.
   — Check /etc/rc.local to make sure that all the daemons needed are present.
   — Make sure `inetd` is running and verify the contents of the /etc/inetd.conf file.
   — If you have installed an UltraNet network, ensure that `unetd` is running and verify the contents of the /etc/unetd.conf file.

5. If you are able to isolate problems to a particular daemon or protocol, `trpt` (explained below) may be useful. `trpt` is used primarily for socket-level debugging, the process of tracing packets as they move through various layers of the network. Although `trpt` is normally used for debugging individual network programs, it may prove useful if you are experienced with socket-level debugging.

If you are troubleshooting an Ethernet or HYPERchannel network and you still cannot pinpoint the problem, call the CONVEX Technical Assistance Center (TAC). If you are troubleshooting an UltraNet network, and have completed the above procedures without finding a problem with the internet interface, try using `tsock` as described at the end of this chapter. Call the CONVEX TAC if you still need assistance.

Using netstat

`netstat` displays statistical "snapshots" of various aspects of network operation. These snapshots enable you to monitor network use and efficiency during periods of normal operation, and to quickly track down problems when the network is malfunctioning. A few of the most useful `netstat` options are discussed here. For information about other `netstat` options, refer to the `netstat`(1C) man pages.

When invoked with no options, `netstat` displays the status of current network transmissions. Network addresses are shown as host names. To use `netstat`, enter

```
netstat
```

Output is similar to Figure 11-2.

Figure 11-2
Sample `netstat` Output

```
Active internet connections
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
udp        0      0  toto2.53               *.*
udp        0      0  localhost.53           *.*
Active UNIX domain sockets
Address  Type   Recv-Q Send-Q    Vnode     Conn     Refs  NextrefAddr
 ff6040c dgram       0      0   279378        0  ffc6c0c        0/dev/log
 ff5ee0c dgram       0      0        0  ff6060c        0        0
 ffc3a0c dgram       0      0        0  ff6060c        0  ff5ec0c
 ffd300c stream      0      0   27f030        0        0        0/dev/printer
 ff6080c stream      0      0   27daa0        0        0        0/etc/cronsock
 ff5f60c dgram       0      0        0  ff6060c        0  ffc380c
```

Displaying Status of Autoconfigured Interfaces

`netstat -i` is useful for quick checks on network status and for checking the fate of outgoing `ping` packets. Status information is displayed for all network interfaces autoconfigured at boot time. To use this option, enter

```
netstat -i
```

Typical output is shown in Figure 11-3.

Figure 11-3
Sample `netstat -i` Output

```
Name  Mtu   Network     Address    Ipkts   Ierrs    Opkts    Oerrs    Collis    Queue
ex0   1500  acme-net    veeger     1520512  0       1327049    0       293         0
lo0   1536  loopback-n  localhost  434254   0       434254     0       0           0
```

The output displayed in Figure 11-3 shows interfaces that are up (`acme-net` and `loopback-n`), the name and addresses of the interfaces, and the maximum size of packets that can be transmitted across each network (listed under the column headed by "Mtu").

`netstat -i` displays statistics on the number of incoming packets (*Ipkts*), incoming packet errors (*Ierrs*), outgoing packets (*Opkts*), outgoing packet errors (*Oerrs*), and "collisions" on the network (*Collis*). (Collisions occur when two computers try to transmit packets across the network at exactly the same instant.) *Queue* indicates the output queue lengths.

Information about incoming and outgoing packets enables you to judge the status of the network at a glance. Furthermore, if you suspect trouble with a particular remote host, you can use `netstat -i` to watch the network in real time. Use the following procedure.

1. Run `ping` to start a packet stream directed toward the host in question. For example:

   ```
   /usr/etc/ping obewan > & /dev/null &
   ```

2. Start `netstat -i` with an appended "update" argument. For example:

   ```
   netstat -i 3
   ```

   The appended argument causes `netstat` to update the statistics and redisplay them every *n* seconds. (In this example, the display is updated every three seconds.)

3. Check to see if packets are being sent and received by the local host.

Displaying Status of all Sockets

`netstat -a` displays the state of all sockets, including those used by server processes. Enter:

```
netstat -a
```

Output is similar to Figure 11-4.

Figure 11-4
Sample `netstat -a` Output

```
Active connections (including servers)
Proto  Recv-Q  Send-Q Local Address   Foreign Address  (state)
udp    0       0      *.1080         *.*
tcp    0       0      *.*            *.*              CLOSED
tcp    0       0      convex.48473   convex.batch     TIME_WAIT
tcp    0       0      convex.48472   convex.batch     TIME_WAIT
tcp    0       1024   convex.48470   vaxip.35         FIN_WAIT_1
tcp    0       0      convex.login   medea.1021       TIME_WAIT
tcp    0       0      convex.1010    altair.login     ESTABLISHED
tcp    0       0      convex.login   slam.1019        ESTABLISHED
tcp    0       0      *.smtp         *.*              LISTEN
udp    0       0      *.talk         *.*
%
```

The example in Figure 11-4 shows an output line with a large number in the `Send-Q` field, which indicates that a communication problem is preventing `netstat` from transmitting its data queue. This is the type of problem to look for when using `netstat`.

Displaying Addresses Numerically

`netstat -n` displays the status of current network transmissions. Internet addresses are displayed instead of host names. The -n option may be used with any display format. Enter:

```
netstat -n
```

Typical output is displayed in Figure 11-5.

Figure 11-5
Sample `netstat -n` Output

```
Active connections
Proto  Recv-Q  Send-Q Local Address    Foreign Address (state)
tcp    0       0      100.0.28.2.1324 100.0.1.13.25   TIME_WAIT
tcp    0       0      100.0.28.2.1021 100.0.1.13.513  ESTABLISHED
tcp    0       0      100.0.28.2.513  100.0.1.13.1022 FIN_WAIT_2
tcp    0       0      100.0.28.2.513  100.0.1.13.1023 FIN_WAIT_2
tcp    0       0      100.0.28.2.1022 100.0.1.13.513  ESTABLISHED
tcp    0       0      100.0.28.2.1014 100.0.1.13.1022 ESTABLISHED
```

The output shown in Figure 11-5 enables you to determine the status of the current network connections. The displayed fields show you, respectively, which protocols are being used, and whether there is any data in either the send or the receive queues.

The `state` codes displayed show the state of the protocol when `netstat` was run.

Display Memory Management Statistics

`netstat -m` displays information about how memory buffers inside the kernel are being used. Typical output is shown in Figure 11-6.

Figure 11-6
Sample `netstat -m` Output

```
188/216 mbufs in use:
        16 mbufs allocated to data
        53 mbufs allocated to socket structures
        74 mbufs allocated to protocol control blocks
        39 mbufs allocated to routing table entries
        4 mbufs allocated to socket names and addresses
        2 mbufs allocated to interface addresses
8/96 mapped pages in use
0 interface pages allocated
492 Kbytes allocated to network (25% in use)
0 requests for memory denied
0 requests for memory delayed
0 calls to protocol drain routines
```

Statistics are rendered in `mbufs`, which are the kernel memory buffers used for storing network packets. The most important thing to notice about this display is the amount of memory in use. If this number exceeds about 95%, try to locate the user processes that are holding all the `mbufs`, and determine why the processes have not relinquished the buffers.

Display Status for Selected Protocol

`netstat -p` displays information about a particular protocol. Enter:

`netstat -p` *protocol*

where *protocol* is either the protocol's well-known name or an alias specified in /etc/protocols. The two most common protocols used by CONVEX Internet Services are `tcp` and `udp`.

Typical output is shown in Figure 11-7.

Figure 11-7
Sample `netstat -p` Output

```
udp:
        0 incomplete headers
        0 bad data length fields
        0 bad checksums
```

```
tcp:
       121620 packets sent
              107368 data packets (42636080 bytes)
              89 data packets (70288 bytes) retransmitted
              11773 ack-only packets (9975 delayed)
              35 URG only packets
              181 window probe packets
              934 window update packets
              1240 control packets
       122571 packets received
              81129 acks (for 42638015 bytes)
              822 duplicate acks
              0 acks for unsent data
              73961 packets (4985917 bytes) received in-sequence
              20 completely duplicate packets (9 bytes)
              3 packets with some dup. data (3 bytes duped)
              616 out-of-order packets (91927 bytes)
              32 packets (0 bytes) of data after window
              0 window probes
              398 window update packets
              7 packets received after close
              0 discarded for bad checksums
              0 discarded for bad header offset fields
              0 discarded because packet too short
       427 connection requests
       423 connection accepts
       840 connections established (including accepts)
       931 connections closed (including 2 drops)
       10 embryonic connections dropped
       79366 segments updated rtt (of 79870 attempts)
       101 retransmit timeouts
              0 connections dropped by rexmit timeout
       4 persist timeouts
       27 keepalive timeouts
              11 keepalive probes sent
              7 connections dropped by keepalive
```

Display Status For Each Protocol

`netstat -s` displays statistics about protocols in use. Typical output is displayed in Figure 11-8.

Figure 11-8
Sample `netstat -s` Output

```
udp:
        0 incomplete headers
        0 bad data length fields
        0 bad checksums
tcp:
        3829520 packets sent
                3311538 data packets (411856679 bytes)
                7525 data packets (978124 bytes) retransmitted
                441589 ack-only packets (400152 delayed)
                556 URG only packets
                8560 window probe packets
                32017 window update packets
                27733 control packets
        4141707 packets received
                2946885 acks (for 411913907 bytes)
                17068 duplicate acks
                0 acks for unsent data
                3016894 packets (195998159 bytes) in-sequence
                11352 completely duplicate packets (393728 bytes)
                199 packets with some dup. data (14662 bytes duped)
                18199 out-of-order packets (8292794 bytes)
                1996 packets (0 bytes) of data after window
                0 window probes
                16186 window update packets
                108 packets received after close
                0 discarded for bad checksums
                0 discarded for bad header offset fields
                0 discarded because packet too short
        9533 connection requests
        8345 connection accepts
        17100 connections established (including accepts)
        19269 connections closed (including 125 drops)
        821 embryonic connections dropped
        2923732 segments updated rtt (of 2940335 attempts)
        8810 retransmit timeouts
                19 connections dropped by rexmit timeout
        122 persist timeouts
        4258 keepalive timeouts
                1026 keepalive probes sent
                111 connections dropped by keepalive
icmp:
        2592 calls to icmp_error
        0 errors not generated 'cuz old message was icmp
        Output histogram:
                echo reply: 1627
                destination unreachable: 2592
        0 messages with bad code fields
        0 messages < minimum length
        0 bad checksums
        0 messages with bad length
```

Figure 11-8
Sample `netstat -s` Output (continued)

```
        Input histogram:
                echo reply: 201
                destination unreachable: 5383
                echo: 1627
        1627 message responses generated
ip:
        7107025 total packets received
        28 bad header checksums
        0 with size smaller than minimum
        0 with data size < data length
        0 with header length < data size
        0 with data length < header length
        852948 fragments received
        0 fragments dropped (dup or out of space)
        2238 fragments dropped after timeout
        0 packets forwarded
        3 packets not forwardable
        0 redirects sent
```

As you can see, a typical LAN uses many different protocols. Transmission Control Protocol (TCP) is used by the standard utilities for packet transmission. User Datagram Protocol (UDP) is used by `rwhod` for the transmission of user datagrams. Internet Control Message Protocol (ICMP) is the protocol used by `ping`. Internet Protocol (IP) is the underlying protocol layer associated with TCP, UDP, ICMP, and other protocols.

As you look over the output from `netstat -s`, check the checksums to see how many transmissions have had errors. (Checksums are validity checks on the packets transmitted.) If the percentage of bad checksums is high, you more than likely have problems with the LAN cabling, transceiver, or connections. Unless the percentage is high, though, don't worry—networks are designed to handle errors.

The `sysgen` parameter, `udpcksum`, enables checksumming of User Datagram Protocol (UDP) datagrams. UDP checksumming results in additional overhead, because each transmitted and received UDP datagram is checksummed when the parameter is turned on.

If you are running Serial Line Internet Protocol (SLIP), you must enable checksumming.

Socket-Level Debugging

`trpt` is a protocol-tracing daemon typically used to debug network programs. Although several options are available, the *-p* option, which prints the addresses of protocol control blocks, is the most useful for debugging the network itself. `trpt` should be used under the following conditions:

- When you have tracked problems to a particular daemon, and want a closer look at how the daemon is working.
- When you want to check various network daemons as you bring up the network.

You use a similar procedure for both cases. Assume that you have noticed problems with `ftp`. First, make sure that no other users are using `ftp`, then shut down the `ftp` daemon by using this procedure.

1. Log in as `superuser`.
2. Modify the /etc/inetd.conf file to replace the line similar to

   ```
   ftp  stream  tcp  nowait  root  /usr/etc/in.ftpd  ftpd
   ```

   with

   ```
   ftp  stream  tcp  nowait  root  /usr/etc/in.ftpd  ftpd -d
   find inetd; kill -Hup inetd
   ```

Note

Remember to change the entry in /etc/inetd.conf back to the original line when you finish debugging.

Begin an `ftp` to `localhost`, log in, and start a file transfer; then immediately stop the transfer using CTRL-Z as follows:

```
% ftp localhost
Connected to localhost.
220 convex FTP server (Version 4.98 Mon Mar 23 21:27:10 CST
1987) ready.
Name (localhost:): watson
Password (localhost:watson): comehere
331 Password required for watson.
230 User watson logged in.
ftp> get program.c
CTRL-Z (Hit before the transfer is complete)
```

Then issue the command:

```
netstat -A
```

Output similar to that shown in Figure 11-9 is produced.

Figure 11-9
Sample `netstat -A` Output

```
Active connections
PCB       Proto  Recv-Q   Send-Q   Local Address      Foreign Address      (state)
ff58a0c   tcp    0        0        acme-il.login      veeger-ex.1016       ESTABLISHED
ff5360c   tcp    0        0        convex.login       convex.1018          ESTABLISHED
ff4be0c   tcp    0        0        convex.1018        convex.ftpd          ESTABLISHED
ff60e0c   tcp    0        0        convex.login       convex.1019          ESTABLISHED
ff55a0c   tcp    0        0        convex.1019        convex.login         ESTABLISHED
ff5ae0c   tcp    0        0        acme-il.login      veeger-ex.1018       ESTABLISHED
ff5aa0c   tcp    0        0        acme-il.login      veeger-ex.1019       ESTABLISHED
ff52c0c   tcp    0        1        acme-il.login      veeger-ex.1020       ESTABLISHED
```

Now run `trpt -p` with the address of the protocol control block in question:

```
trpt -p ff4be0c
```

This produces a listing of the trace records associated with the protocol control block in question. Sample output is shown in Figure 11-10.

Figure 11-10
Sample `trpt -p` Output

```
8061d98c:
405 CLOSED:user ATTACH -> CLOSED
405 LISTEN:input 877901@0(win=2048)<URG> -> SYN_RCVD
405 SYN_RCVD:output [877941..877945)@877902(win=2048)<URG> -> SYN_RCVD
406 SYN_RCVD:input 877902@877942(win=2048)<ACK> -> ESTABLISHED
406 ESTABLISHED:user ACCEPT -> ESTABLISHED
410 ESTABLISHED:user CONTROL -> ESTABLISHED
414 ESTABLISHED:output [877942..87798c)@877902(win=2048)<ACK> -> ESTABLISHED
414 ESTABLISHED:user SEND -> ESTABLISHED
416 ESTABLISHED:input 877902@87798c(win=1974)<ACK> -> ESTABLISHED
418 ESTABLISHED:user CONTROL -> ESTABLISHED
912 ESTABLISHED:user PROTOSEND -> ESTABLISHED
912 ESTABLISHED:input 877902@87798c(win=2048)<ACK> -> ESTABLISHED
```

Although correct interpretation of these trace records depends on familiarity with protocol-level debugging, they may be useful to you or to someone on your staff. As you look at the output, notice that after an initial `LISTEN`, protocol connections were opened, or `ESTABLISHED`, without problem.

Using tsock

The UltraNet network exerciser, `tsock`, tests network connections by sending data over the network. It enables you to isolate communication problems to a particular network adapter. The most useful of its functions, the ability to send patterns from one host to another, is discussed here. For information about its other functions, refer to the `tsock`(1C) man pages.

To use `tsock`, you establish one host as the receiver and another as the transmitter. The receiver listens for an incoming connection; the transmitter initiates a connection and repeatedly sends either a default or a user-defined pattern, or the contents of a file. For example, to test the connection between hosts `scorpio` and `altair`, enter

```
tsock -r scorpio >testdata
```

on host `altair`, and

```
tsock -t altair <testdata
```

on host `scorpio`. If the connection is sound, the data in file testdata on host `altair` will transfer to file testdata on host `scorpio`.

To exercise network communications by repeatedly sending and receiving a test pattern, enter

```
tsock -r -s -l10k
```

on host `altair`, and

```
tsock -t -s -n100 -l10k altair
```

on host `scorpio`. A 10K buffer filled with an internally-generated test pattern will be transmitted 100 times.

The following `tsock` commands cause a 1K buffer of data to be sent across the network 10 times. Enter

```
tsock -r -s -l1K
```

on host `altair`, and

```
tsock -t -s -l1K -n10 -m1k altair
```

on host `scorpio`.

Figure 11-11 shows the output from `tsock`.

Figure 11-11
Typical `tsock` Output

```
(On host altair)

     tsock-r: buflen=1024
     tsock-r: socket tsock-r: accept
     tsock-r: transfer begin: Fri Jul 7 10:46:39 1989
     tsock-r: transfer end:  Fri Jul 7 10:46:39 1989
     tsock-r: 0.0user 0.0sys 0:00real 25% 0i+0d 50maxrss 1+0pf 11+0csw
     tsock-r: 10240 bytes in 0.010000 CPU seconds = 1000.000000 KB/cpu sec
     tsock-r: 10240 bytes in 0.049366 real seconds = 202.568569 KB/sec


(On host scorpio)

     tsock-t: nbuf=10, buflen=1024, host 'altair'
     tsock-t: socket
     tsock-t: connect
     tsock-t: transfer begin: Fri Jul 7 10:46:39 1989
     tsock-t: XXXXXXXXXX
     tsock-t: transfer end:  Fri Jul 7 10:46:39 1989
     tsock-t: 0.0user 0.0sys 0:00real 9% 0i+0d 22maxrss 0+0pf 11+0csw
     tsock-t: 10240 bytes in 0.010000 CPU seconds = 1000.000000 KB/cpu sec
     tsock-t: 10240 bytes in 0.119658 real seconds = 83.571512 KB/sec
```

If you cannot isolate the problem with `tsock`, call the CONVEX Technical Assistance Center (TAC) for help.

If you can isolate the problem to a particular adapter, you can prevent the system from hanging by deleting the malfunctioning adapter with the `unetcf adel` command. To use it, enter

`/etc/unetcf adel` [*adapter-id*]

If you omit *adapter-id*, all adapters are deleted.

Network System Files

This appendix describes network configuration and management files, some of which require periodic maintenance.

Each entry includes a file description, format, sample entries, and sources of further information. Where applicable, specific maintenance instructions are provided. You can read more about these files in the *ConvexOS Programmer's Reference*.

NAME

/etc/ftpusers—Database of user names denied access to `ftp`.

DESCRIPTION

The /etc/ftpusers file contains a list of user names to which the system denies access to the `ftp` utility. `ftpd`, the `ftp` daemon, checks this file each time it receives an `ftp` request. If the user making the request is listed in this file, `ftpd` denies the request.

The /etc/ftpusers file should be created or augmented if you intend to use an anonymous `ftp` account. An anonymous `ftp` account allows users to transfer files between machines without having to use a password. There are security risks, however, if you allow access to the account by users logged in as `uucp` or `root`, or if you allow logins by accounts with nonstandard shells and no passwords (for example, `who` or `finger`). To avoid such security risks, add these types of names to the /etc/ftpusers file.

EXAMPLE

```
adams
bin
buchanan
finger
grant
johnson
lincoln
root
uucp
who
```

Each user name must be on a line by itself, followed by **RETURN**. Be careful not to leave any white space on the line with the name.

PROGRAMS

`ftp`(1C)

SEE ALSO

`ftpd`(8C)

NAME

/etc/gateways—Gateway routing database.

DESCRIPTION

The network routing daemon, `routed`, supports the notion of passive and active gateways. When `routed` starts, it reads the /etc/gateways file to find gateways that may not be identified using the standard method (`SIOGIFCONF ioctl`). Gateways specified in this manner should be marked `passive` if they are not expected to exchange routing information, while gateways marked `active` should be willing to exchange routing information (that is, they should have a routing daemon running on the machine). Passive gateways are maintained in the routing tables forever, and information regarding their existence is included in any routing information transmitted. Active gateways are treated equally to network interfaces. Routing information is distributed to the gateway and if no routing information is received for a specific period of time, the route is deleted.

The format for entries in the /etc/gateways file is:

{ **net** | **host** } *name1* **gateway** *name2* **metric** *value* { **passive** | **active** }

where:

net **host**	Keywords that indicate if the route is to a network or to a specific host.
name1	Name of the destination network or host. This may be a symbolic name located in /etc/networks or /etc/hosts, or an internet address specified in dot notation.
gateway *name2*	Name or address of the gateway to which messages should be forwarded.
metric *value*	Metric indicating the hop count to the destination host or network.
passive **active**	Keywords indicating if the gateway should be treated as passive or active (as described above).

EXAMPLE

```
host      acme1    acme3    metric 3   active
network   mktgnet  acme3    metric 1   active
```

This sample file specifies that routing originating from the host `acme1` should go to the destination host `acme3`, which is three jumps away. There is a `routed` process running on `acme3`. In the second entry, routing from the `mktgnet` network is also to be routed with one jump to `acme3`.

PROGRAMS

`routed`(8C), `ifconfig`(8C)

SEE ALSO

`route`(8C)

NAME

/etc/hosts—Host name database.

DESCRIPTION

The /etc/hosts file contains information regarding known hosts on the DARPA Internet. For each host your network needs access to, a single line containing the official host name, internet address (in dot notation), and aliases should be present.

Blanks or tabs separate the items. A "#" indicates the beginning of a comment; characters after "#" up to the end of the line are ignored by routines that search the file. This file is normally created from the official host database maintained at the DDN Network Information Control (NIC) Center. You may need to make local changes to add unofficial aliases or unknown hosts (for example, your local host name).

Network addresses are specified in the conventional dot notation. Host names may contain any printable character besides a field delimiter, newline, or comment character.

EXAMPLE

```
#
# Berkeley Host Database
#
127.1           localhost
#
# Convex local Ethernet
#
140.60.1.0x82   acmes    south
140.60.0.1      acmee
140.60.0.6      acme1    first
```

This example shows the local portion of the /etc/hosts database. The lines following it in the /etc/hosts file come from NIC distribution tables or from the CONVEX distribution.

The entry, `127.1 localhost`, defines a well-known local host address used for testing, loopbacks, and some daemons.

When an unknown host name is requested, the `arp` name-resolution protocol broadcasts packets requesting identification.

PROGRAMS

`ftp`(1C), `netstat`(1C), `rcp`(1C), `rlogin`(1C), `rsh`(1C), `sendmail`(8), `talk`(1), `telnet`(1C)

SEE ALSO

`gethostent`(3N), `arp`(4P)

NAME

/etc/hosts.equiv—List of trusted hosts or network groups.

DESCRIPTION

The /etc/hosts.equiv file contains a list of trusted hosts or groups. When a user on a listed host makes an `rlogin` or `rsh` request and the user exists in the /etc/passwd file, the system does not prompt the user for a password. In effect, the remote user is "equivalenced" to a local user with the same user name.

Entries in the /etc/hosts.equiv file specify host or group names in one of the following formats:

simple_hostname
+@*group*
-@*group*

When an entry contains a simple host name, all users on that host are trusted if they have an entry in the /etc/passwd file. An entry consisting of +@*group* designates that all hosts in the named group are trusted. Likewise, an entry consisting of -@*group* indicates that all hosts in the specified group are not trusted. An entry consisting of a single "+" indicates that all users on all hosts are trusted.

Only sites that run Yellow Pages (YP) recognize +@*group* and -@*group* entries. Network groups are defined in the /etc/netgroup file. For information about network groups, refer to Chapter 8, "Configuring Network Services," the entry for /etc/netgroup in this appendix, and the `netgroup`(5) man page. (Yellow Pages is an optional product included in the NFS package. For more information, contact your CONVEX sales representative.)

EXAMPLE

```
tiamat
apsu
+@engr
+@mis
-@mktg
```

In this example, users on hosts `apsu` and `tiamat`, and on hosts belonging to network groups `engr` and `mis` are trusted. Users on hosts belonging to network group `mktg` are not trusted; they must enter a password to gain access.

PROGRAMS

`rlogin`(1C), `rsh`(1C)

SEE ALSO

`netgroup`(5), `rhosts`(5)

NAME

/etc/inetd.conf—Internet superserver daemon configuration file.

DESCRIPTION

The /etc/inetd.conf file contains an entry for each network service handled by `inetd`, the internet superserver daemon. `inetd` monitors ports defined in the /etc/services file, and invokes the corresponding server program defined in the /etc/inetd.conf file.

Entries in the /etc/inetd.conf file follow the format:

service_name { **stream** | **dgram** | **raw** } *protocol* { **wait** | **nowait** } *user* *server_program* [*arguments*]

where:

service_name	Name of a valid service in the /etc/services file or, if your site runs NFS, in the /etc/rpc file. See below.
stream **dgram** **raw**	Socket type.
protocol	Name of a valid protocol listed in the /etc/protocols file.
wait **nowait**	Flag indicating whether the server processes all incoming messages on the socket (**wait**), or frees the socket after receiving a message (**nowait**). This parameter applies only to datagram (**dgram**) sockets.
user	User name under which the server runs. This parameter allows the system manger to give servers less permission than `root`.
server_program	Pathname of the program `inetd` executes when it receives a request on the socket. If `inetd` provides the service internally, this field should be **internal**.
arguments	Arguments passed to the server program.

EXAMPLE

```
#
# Internet server configuration database
#
ftp     stream  tcp  nowait  root  /usr/etc/in.ftpd     ftpd
telnet  stream  tcp  nowait  root  /usr/etc/in.telnetd  telnetd
shell   stream  tcp  nowait  root  /etc/in.rshd         rshd
login   stream  tcp  nowait  root  /etc/in.rlogind      rlogind
syslog  dgram   udp  wait    root  /usr/etc/in.syslog   syslog
```

This example shows a small portion of the /etc/inetd.conf file installed with CONVEX Internet Services software.

PROGRAMS

`comsat`(8C), `ftpd`(8C), `inetd`(8C), `rexecd`(8C), `rlogind`(8C), `rshd`(8C), `telnetd`(8C), `crashdump`(8), `syslog`(8), `talkd`(8), `mountd`(8C), `rexd`(8C), `rquotad`(8C), `rstatd`(8C), `rusersd`(8C), `rwalld`(8C), `sprayd`(8C)

SEE ALSO

`rpc`(5), `services`(5)

NAME

/etc/netgroup—Network group definitions.

DESCRIPTION

The /etc/netgroup file defines network-wide groups used for permission checking when performing remote mounts or executing `rlogin` and `rsh` commands. For remote mounts, the system uses the /etc/netgroup file to classify machines; for `rlogin` and `rsh` commands, /etc/netgroup is used to classify users. The format for entries in the /etc/netgroup file is:

groupname (*member1*) [*member2*. . .]

where:

groupname Descriptive name you assign to the group.

members List of members, each specified as three parts enclosed in parentheses. For example,

(*hostname, username, domainname*)

You may leave any of the three fields empty to signify a wild card. For example, the entry

```
everybody (,,)
```

defines a group to which all users belong. If you begin a field with a character other than a letter, digit, or underscore, the field is considered empty. For example, in the entries

```
machines    (aries,-,utd)
people      (-,jones,utd)
```

the `machines` group contains the host `aries` in the `utd` domain, but no users belong to it. The `people` group includes the user `jones` in the `utd` domain, but no hosts belong to it.

The /etc/netgroup file is an optional product included with the NFS package. For more information, contact your CONVEX sales representative.

EXAMPLE

```
world(,,)
engr    (aries,,) (taurus,,) (gemini,,) (cancer,,) \
        (leo,,) (virgo,,) (libra,,) (scorpio,,) \
        (sagittarius,,) (capricorn,,) (aquarius,,) (pisces,,)
doc     (-,smith,) (-,jones,) (-,baker,) (-,white,)
```

In this example, the `world` group includes all users on all machines. The `engr` group includes all users of engineering machines. Four users make up the `doc` group; it contains no hosts.

PROGRAMS

`getnetgrent`(3N), `exports`(5), `makedbm`(8), `ypmake`(8), `ypserv`(8)

SEE ALSO

`hosts.equiv`(5), `rhosts`(5)

NAME

/etc/networks—Network name database.

DESCRIPTION

The /etc/networks file contains information regarding known networks that make up the DARPA Internet. The file contains a single line for each network with the following information:

official_network_name *network_number* [*aliases*]

Blanks or tabs separate the items. A "#" indicates the beginning of a comment; characters after the "#" up to the end of the line are not interpreted by routines that search the file. This file is normally created from the official network database maintained at the DDN Network Information Control (NIC) Center. You may need to make changes to add unofficial aliases or unknown networks (e.g., your local network name).

Network numbers may be specified in dot notation. Network names may contain any printable character other than a field delimiter, newline, or comment character.

EXAMPLE

```
devel-net       192.18.44    dnet
test-net        192.18.45    tnet
mktg-net        192.18.46    mnet
hyperchannel    192.18.47    hairnet
loopback-net    12           localnet
arpanet         10
```

PROGRAMS

`netstat`(1C), `routed`(8C), `route`(8C)

SEE ALSO

`getnetent`(3N)

NAME

/etc/rpc—rpc program number database.

DESCRIPTION

The /etc/rpc file contains user-readable names that can be used in place of rpc program numbers. The format for records in the /etc/rpc file is:

server_name program_number [*aliases*] [*#comment*]

where:

server_name	Name of the command associated with the service.
program_number	Number of the port associated with the service and name of the protocol it uses. These parameters are expressed as a single item with a "/" separating them, as in 512/tcp.
aliases	Any other names you wish to call the server.
#comment	All characters right of the "#" are ignored by routines that use the file.

Blanks or tabs separate the fields.

The /etc/rpc file is an optional product included with the NFS package. For more information, contact your CONVEX sales representative.

EXAMPLE

```
portmapper      100000      portmap sunrpc
rstatd          100001      rstat rup perfmeter
nfs             100003      nfsprog nfsd
ypbind          100007
sprayd          100012      spray
rexd            100017      rex
```

This example shows a small portion of the /etc/rpc file installed as a function of installing NFS software.

PROGRAMS

getrpcent(3N), inetd(8C)

SEE ALSO

services(5)

NAME

/.rhosts, ~/.rhosts—Remote host access files.

DESCRIPTION

The /.rhosts file contains a list of remote host and user names specifying which users on which systems are permitted to use `rlogin` or `rsh` as `root` without supplying a password. When a user on a listed host makes an `rlogin` or `rsh` request and the user exists in the /etc/passwd file, the system does not prompt the user for a password.

Entries in the ~/.rhosts file specify host, user, and group names in one of the following formats:

hostname [*username*]
{ + | - }*@group1* [{ + | - }*@group2*]

@group1 and *@group2* may be either a group defined in the /etc/group file or a network group defined in the /etc/netgroup file.

When an entry in the /.rhosts file contains only a host name, all users on that host are permitted to use `rlogin` or `rsh` without supplying a password, provided they have an entry in the /etc/passwd file. If *username* is specified and the named user exists in the /etc/passwd file, the system permits that user to execute `rlogin` or `rsh` as anyone, that is, the user may specify any name when using the `rlogin -l` flag. Thus, the /.rhosts file entry,

```
gaia jones
```

allows `jones` to log in from `gaia` as anyone, without supplying a password.

An entry in the /.rhosts file that includes +*@group* designates that all users in the group or all users on all hosts in the network group are permitted to use `rlogin` or `rsh` as any user. For example, the entry

```
+@doc +@engr
```

allows any member of the group `doc` to execute `rlogin` or `rsh` commands from any host in the `engr` network group.

Likewise, an entry that includes -*@group* indicates that all users in the group or all users on all hosts in the specified network group must enter a password to use `rlogin` or `rsh`.

Each user's home directory may contain a private .rhosts file. When a user executes an `rlogin` or `rsh` command, the system conceptually concatenates the .rhosts file in the user's home directory onto the /etc/hosts.equiv file before checking to see if the user is permitted to execute `rlogin` or `rsh` commands without supplying a password. If a user is excluded by a "-" entry in the /etc/hosts.equiv file but permitted via an entry in the ~/.rhosts file, the user is not required to enter a password. If the user name is `root`, the system checks only the ~/.rhosts file.

To use different login names on different machines, add entries to your .rhosts file containing your other login names and the names of machines from which you want to access your local account.

An entry in a user's .rhosts file consisting of a single "+" indicates that all users on all hosts may execute `rlogin` or `rsh` commands under that user's name, provided an entry for the requesting user exists in the /etc/passwd file.

Only systems that run Yellow Pages (YP) recognize +@group and -@group entries. Network groups are defined in the /etc/netgroup file. For information on network groups, refer to Chapter 8, "Configuring Network Services," the description of the /etc/netgroup file in this appendix, and the `netgroup`(5) man page. (Yellow Pages is an optional product included in the NFS package. For more information, contact your CONVEX sales representative.)

EXAMPLES

```
#
# /.rhosts file
#
tiamat
apsu
gaia +@engr
aries +@doc
+@engr +@pubs
-@sales
```

In this example,

- All users who have entries in the local /etc/passwd file are permitted to log in from hosts `tiamat` and `apsu`.
- All users of machines in the `engr` network group who have entries in the local /etc/passwd file are permitted to log in from host `gaia`.
- All members of the `doc` group who have entries in the local /etc/passwd file are permitted to log in from the host `aries`.
- Members of the `pubs` group who have entries in the local /etc/passwd file are permitted to log in from any machines in the `engr` network group.

PROGRAMS

`rlogin`(1C), `rsh`(1C)

SEE ALSO

`group`(5), `netgroup`(5)

NAME

/etc/services—Service name database.

DESCRIPTION

The /etc/services file contains information regarding available DARPA Internet services. The format for records in the /etc/services file is:

official_service_name port_number/protocol_name [*aliases*] [*#comment*]

where:

service_name	Name of the command associated with the service.
port_number/ protocol_name	Number of the port associated with the service and name of the protocol it uses. These parameters are expressed as a single item with a "/" separating them, as in `512/tcp`.
aliases	Any other names you wish to call the server.
#comment	All characters right of the "#" are ignored by routines that use the file.

Blanks or tabs separate the items.

EXAMPLE

```
#
# Internet Services
#
netstat 15/tcp
ftp     21/tcp
telnet  23/tcp
smtp    25/tcp    mail
uucp    33/tcp    uucpd
time    37/tcp    timserver
```

This example shows a small portion of the /etc/services file installed with CONVEX Internet Services software.

PROGRAMS

`comsat`(8C), `ftpd`(8C), `inetd`(8C), `rexecd`(8C), `rlogind`(8C), `rshd`(8C), `telnetd`(8C), `crashdump`(8), `syslog`(8), `talkd`(8), `mountd`(8C), `rexd`(8C), `rquotad`(8C), `rstatd`(8C), `rusersd`(8C), `rwalld`(8C), `sprayd`(8C)

SEE ALSO

`rpc`(5), `getservent`(3N)

Reporting Problems

B

This appendix introduces the CONVEX Technical Assistance Center (TAC) and the `contact` utility.

The `contact` utility is an online system for reporting problems to the TAC. To use it, enter **`contact`** at the system prompt and answer the questions as they appear on the screen.

This appendix describes:

- Prerequisites for using `contact`.
- Tips for using `contact`.
- The step-by-step process `contact` takes you through.

Technical Assistance Center

The CONVEX Technical Assistance Center (TAC) is staffed by technical specialists who can address diverse questions and problems that arise in a supercomputing environment. If you have a hardware, software, or documentation question, contact the TAC. This group stands ready to solve such problems.

The contact Utility

The TAC recommends using the `contact` utility to report a hardware, software, or documentation problem. The `contact` utility is an interactive program that helps the TAC track reports and route them to the CONVEX personnel most qualified to fix a problem.

After you invoke `contact`, it prompts you for information about the problem. When you finish your report, `contact` electronically mails it to the TAC. The TAC notifies you within 48 hours that your report has been received.

Use of the `contact` utility requires:

- UNIX-to-UNIX Communication Protocol (UUCP) connection to the TAC.
- Full path name of the program or utility in question.
- Version number of the program or utility in question.

UUCP Connection

Before using `contact`, ask your system administrator if your site has a UUCP connection to the TAC. A UUCP connection allows files to be copied from one UNIX-based system to another. The `uucp` (UNIX-to-UNIX copy) command relies on either a dial-up or hard-wired UUCP communication line.

Finding the Program Path Name

To determine the full path name of the program or utility in question, use the `which` command. Figure B-1 illustrates use of the `which` command to find the full path name of the loader utility.

Figure B-1
Using the `which` Command

```
> which ld
/bin/ld
>
```

In this example, the full path name of the loader is `/bin/ld`.

If you use the C shell (`csh`), you can also use the `whence` command to find the program path name. The `whence` command works like `which`, but faster.

For more information on the `which` command, refer to the `which`(1) man page. You can also use the `info` online information system by entering **`info which`** at the system prompt.

Finding the Program Version Number

To determine the version number of the program or utility in question, use the `vers` command. Figure B-7 illustrates use of the `vers` command to find the version number of the loader (`ld`) utility. Enter **`vers`**, then the path name of the program or utility.

Figure B-2
Using the `vers` Command

```
> vers /bin/ld
/bin/ld: 7.0
>
```

In this example, the loader version number is 7.0.

For more information on the `vers` command, refer to the `vers`(1) man page. You can also use the `info` online information system by entering **`info vers`** at the system prompt.

Using contact

The `contact` utility prompts for the following information:

- ❑ Your name, title, phone number, and corporate name.
- ❑ Name and version of the product.
- ❑ One-line summary of the problem.
- ❑ Detailed description of the problem.
- ❑ Priority of the problem.
- ❑ Instructions on how to reproduce the problem.
- ❑ Comments about the problem.
- ❑ Comments about the documentation relating to the problem.
- ❑ Files to include in the `contact` report.

Following is a step-by-step discussion of these prompts.

Step 1a

To invoke the `contact` utility, enter **`contact`** at the system prompt. The system responds with a welcome message and a series of questions regarding your hardware, software, or documentation question. Figure B-3 illustrates use of the `contact` command.

Figure B-3
Beginning a `contact` Session

```
> contact
Welcome to contact version 0.11 ()

Enter your name, title, phone number, and corporate name (^D to terminate)
>
```

Step 1b

If there is a .contact file in your home directory, `contact` skips the first prompt. (Refer to "Using a .contact File" for more information.) Figure B-4 illustrates the `contact` command and the system response when you have a .contact file in your home directory.

Figure B-4
Beginning a `contact` Session with a .contact File

```
> contact
Welcome to contact version 0.11 ()

Enter the name of the product involved
>
```

Step 2

The `contact` utility prompts for the version number of the product. If you do not know the version number, press **CTRL-Z** to suspend the session.

Use the `which` (or `whence` if you use `csh`) and `vers` commands to find the version number of the product. Use the `fg` command to return to the session, and enter the version number in the form X.X or X.X.X.X.

Step 3

The `contact` utility prompts for a one-line summary of the problem. This summary is the subject header in any further correspondence regarding the problem. Please make this summary as descriptive as possible in one line.

Step 4

The `contact` utility prompts for a detailed description of the problem. Please make this description as complete as possible. Include source code and a stack backtrace when possible. (Refer to the `adb`(1) or `csd`(1) man page for information on obtaining a stack backtrace.) The more information you provide, the quicker the TAC can isolate and solve the problem.

Step 5

The `contact` utility prompts for the priority of the problem. Figure B-5 illustrates this prompt and priority levels from which to choose.

Figure B-5
Specifying Priority
of a Problem

```
Enter a problem priority, based on the following:
1) Critical     - work cannot proceed until the problem is resolved.
2) Serious      - work can proceed around the problem, with difficulty.
3) Necessary    - problem has to be fixed.
4) Annoying     - problem is bothersome.
5) Enhancement  - requested enhancement.
6) Informative  - for informational purposes only.
>
```

Step 6

The `contact` utility prompts for an explanation of how to reproduce the problem. Please include the command syntax, the options you used, and anything else you did to run the program.

Step 7

The `contact` utility prompts for any other pertinent comments. Please include all relevant information.

Step 8

The `contact` utility prompts for suggestions regarding documentation supporting the product. Indicate whether the documentation could be revised to address the problem.

Step 9

The `contact` utility prompts for names of files necessary to reproduce the problem. Figure B-6 illustrates this prompt and sample user response.

Figure B-6
Including Files in a `contact` Report

```
Are there any files that should be included in this report (yes | no)?
> yes
Please enter the names of the files, one to a line (^D to terminate)
> test.f
> ~/subroutines/sub.f
>
```

Note

Tilde-escape sequences are not recognized in responses to this prompt. In `contact`, a tilde in this section indicates your home directory. This convention is based on use of the tilde for expanding file names in `csh`.

If you specify small text files, they are automatically included in the `contact` report. If the files are too large to be included in this report, `contact` gives further instructions on how to submit these files.

To specify a directory, combine directory files into a single file using the `tar` command (refer to the `tar`(1) man page for further information) or enter each file name in the directory on a single line in the `contact` report.

Step 10

The `contact` utility prompts you to review, edit, submit, or abort the report. Figure B-7 illustrates this prompt.

Figure B-7
Prompt to Review, Edit, Submit, or Abort Report

```
Please select one of the following options:
1) Review the problem report.
2) Edit the problem report.
3) Submit the problem report.
4) Abort the problem report.
>
```

Choose the number of the option you want to select. These options let you do the following:

Review Review the text of the `contact` report. You are then prompted again to select an option.

Edit Edit the text of the `contact` report. If you choose to edit the report, `contact` opens your default text editor.

Submit Send the report to the CONVEX TAC. The TAC notifies you within 48 hours that your report has been received. Choosing this option exits the `contact` utility and returns you to the shell.

Abort Save the text of the report in a file named ~/dead.report. Choosing this option exits `contact` and returns you to the shell.

Tips for Using contact

The `contact` utility is interactive and easy to use. This section lists tips to help you use it efficiently. In particular, this section explains how to:

- Use a `.contact` file.
- Abort a `contact` session.
- Resubmit an aborted report.
- Suspend a `contact` session.
- Move within `contact` from one prompt to another.
- Use tilde-escape sequences in the `contact` utility.

Using a .contact File

When you invoke `contact`, it first prompts for your name, title, phone number, and company name. You can, however, create a .contact file to skip this first prompt.

Follow these steps to create a .contact file:

1. Create a .contact file in your home directory.
2. Enter your name, job title, phone number, and company name, each on a new line.

When you invoke `contact`, it automatically includes the .contact file as input for the first prompt and proceeds to the next prompt.

Aborting the Report

To abort a `contact` report, either press the interrupt key (usually **CTRL-C**) or choose the abort option when prompted by the `contact` utility. Using **CTRL-C** to abort does not save the contents of the report. Using the `abort` option saves the contents of the report in a file named ~/dead.report.

Submitting the dead.report File

After you abort a `contact` session, the `contact` utility saves the report in a file named ~/dead.report. Using the `contact` command with the `-r` option automatically merges the contents of the ~/dead.report file into the new `contact` session. Enter

```
contact -r
```

and `contact` finds the ~/dead.report file and merges it into the `contact` report. You can then edit the report. When you end the editing session, `contact` resumes at the final prompt, which asks you to review, edit, submit, or abort the report.

Suspending a Report

Sometimes it is necessary to stop in the middle of a `contact` report and return to the shell (for instance, to suspend the `contact` session to find the program path name or version number). To suspend the `contact` session, press **CTRL-Z**.

To return to the `contact` session, type **`fg`**. Using **CTRL-Z** and the `fg` (foreground) command, you can switch between the `contact` utility and the shell. You cannot, however, use **CTRL-Z** and `fg` to switch back and forth in the Bourne shell (`sh`).

Ending a Response

The `contact` utility prompts for information pertinent to your hardware, software, or documentation question. Some prompts require one-line responses; to move to the next prompt, press **RETURN**. Other prompts require more than a one-line response; to move to the next prompt, press **CTRL-D**.

Tilde-Escape Sequences

The `contact` utility treats input beginning with a tilde (~) as a special sequence. The character following the tilde is considered a request for a special function. You can use the following tilde sequences within `contact`:

`~e`	Start the text editor (defined in the `EDITOR` environment variable).
`~h`	Display a list of available tilde-escape sequences.
`~p`	Print the `contact` report to the terminal screen.
`~r` *filename*	Read the contents of *filename* as a response to the current prompt. Some prompts require only a one-line response. This tilde-escape sequence works only for prompts that allow more than a one-line response.
`~~`	Insert a single tilde as the first character in the line.

Index

A

B

C

D

E

F

G

H

I

L

M

N

P

R

S

T

U

V

W

Y

Z