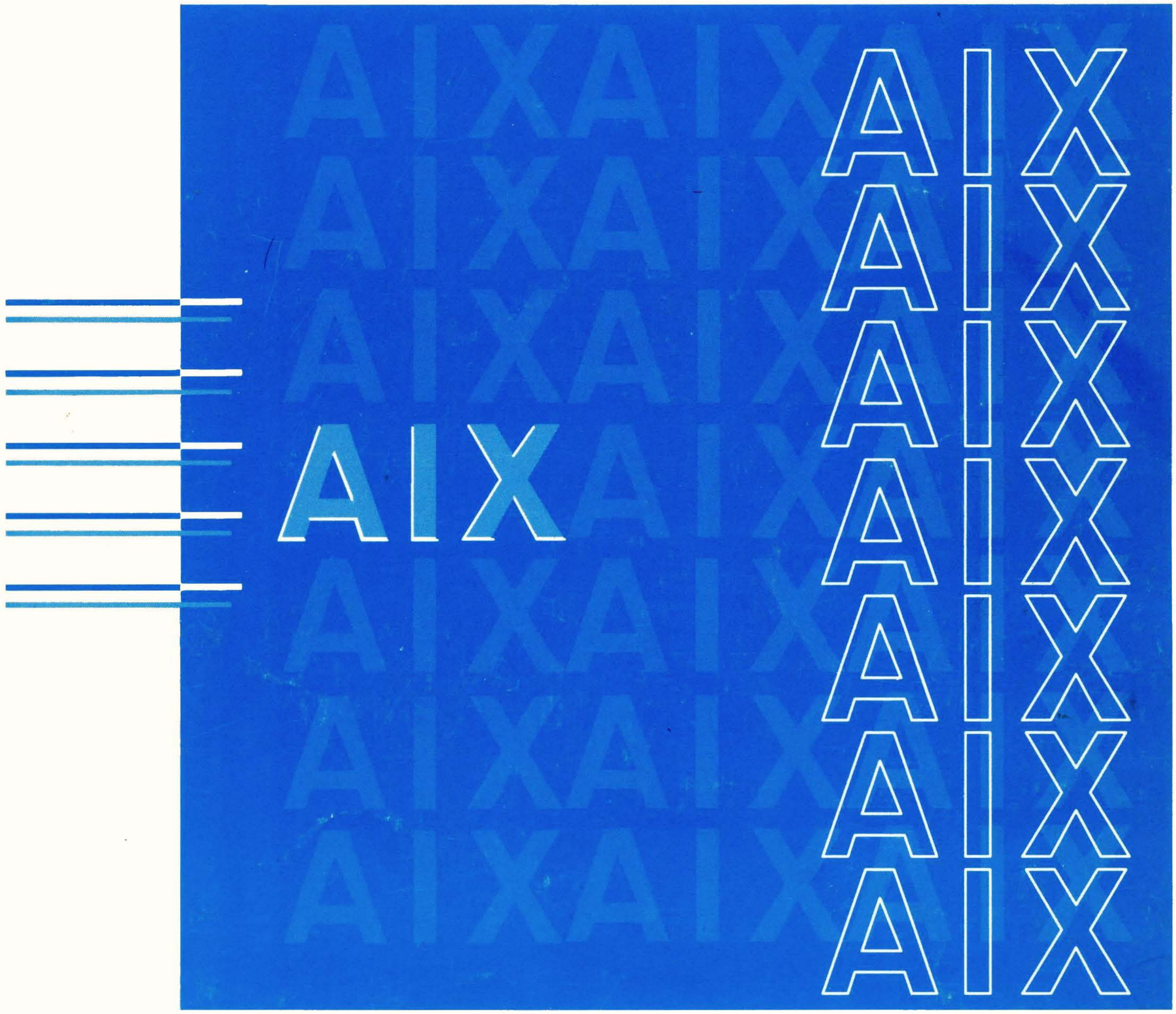
AIX Operating System

TCP/IP User's Guide



IBM

AIX Operating System TCP/IP User's Guide

Version 1.2

IBM

SC23-2309-00

Second Edition (December 1989)

This edition applies to Version 1.2 of the IBM Advanced Interactive Executive for the System/370 (AIX/370), Program Number 5713-AFL, and to Version 1.2 of the IBM Advanced Interactive Executive for the Personal System/2 (AIX PS/2), Program Number 5713-AEQ, and to all subsequent releases until otherwise indicated in new editions or technical newsletters. Changes are made periodically to the information herein; these changes will be reported in technical newsletters or in new editions of this publication.

Portions of the code and documentation were developed at the Electrical Engineering and Computer Sciences Department at the Berkeley Campus of the University of California under the auspices of the Regents of the University of California.

About This Book

This book explains how to use the Transmission Control Protocol/Internet Protocol (TCP/IP) program on the IBM Advanced Interactive Executive AIX Operating System. It provides both guide and reference information on how to:

- Install TCP/IP on your system
- Log in to and run command on remote systems
- Print files on remote systems
- Manage networks
- Send and receive mail and transfer files across the network
- Customize TCP/IP to your own needs.

Who Should Read This Book

This book is written for AIX Operating System users who want to transfer data between an IBM Personal System/2 or a System/370 and another host computer, users who want to use another IBM Personal System/2 or System/370 or host remotely, or users who want to manage networks.

What You Should Know

Programmers who want to develop applications that use the TCP/IP facilities should refer to the *AIX Operating System Technical Reference Manual* for information on the operating system subroutines and system calls that provide access to these facilities.

This book assumes that you are familiar with AIX commands and know how to use them.

If you need information on specific system calls or subroutines, refer to the *AIX Operating System Technical Reference*.

How to Use This Book

This book contains special terms common to TCP/IP. If you are unfamiliar with a term, refer to the glossary in the back of the book. For information on specific AIX commands, refer to the *AIX Operating System Commands Reference*.

If you are familiar with using TCP/IP on another system, you may want to move directly to the task you want to perform. If not, it is suggested that you begin with Chapter 1.

We suggest that you first read Chapter 1 which introduces you to the TCP/IP Program. Then, read Chapter 2 which shows you how to install TCP/IP on your system. By becoming familiar with Chapter 3, 4 and 5 you learn how to use the TCP/IP user commands, server commands and file formats to transfer data, log in to remote systems and manage the network.

AIX is a trademark of the International Business Machines Corporation.

Personal System/2 is a registered trademark and System/370 is a trademark of the International Business Machines Corporation.

Highlighting

- All AIX Operating System commands, options, parameters, names of keys, keywords, and actual file names are in **boldface** type.
- Protocols are in UPPERCASE type.
- Environment variables in **UPPERCASE** boldface type.
- New terms introduced in the text are shown in ***boldface italic***.
- Variable information is in *italic* type.
- Anything users type is in `monospace` type.
- Anything appearing on a display screen that is referred to in a paragraph of text is in `monospace` type.
- If the instruction is set off from the paragraph, it is printed in `monospace` type.

Related Publications

For additional information, you may want to refer to the following publications:

- *AIX/370 Administration Guide*, SC23-2088, describes such administrative tasks as updating the file system, backing up files, and fine-tuning and monitoring the performance of the operating system.
- *AIX Operating System Commands Reference*, SC23-2292, (Vol 1) and SC23-2184 (Vol 2), lists and describes the AIX /370 and AIX PS/2 Operating System commands.
- *AIX/370 Diagnosis Guide*, SC23-2090, describes procedures and tools that can be used to define and categorize symptoms of problems that may occur during daily operation.
- *AIX/370 General Information*, SC23-2062, describes AIX/370 and its position in the AIX family of products. This book also describes AIX/370 functions and capabilities and the product's relationship to the UNIX Operating System.
- *Installing and Customizing the AIX PS/2 Operating System*, SC23-2290, provides step-by-step instructions for installing the AIX PS/2 Operating System and related programs. This book also shows how to customize the operating system to suit the user's specific needs and work environment.
- *Installing and Customizing the AIX/370 Operating System*, SC23-2066, provides step-by-step instructions for installing the AIX/370 Operating System and related programs. This book also shows how to customize the operating system to suit the user's specific needs and work environment.
- *Managing the AIX Operating System*, SC23-2293, describes such system-management tasks as adding and deleting user IDs, creating and mounting file systems, backing up the system, repairing file system damage, and setting up an electronic mail system and other networking facilities.
- *Messages Reference*, SC23-2294, lists messages displayed by the AIX Operating System and explains how to respond to them.

- *AIX/370 Planning Guide*, GC23-2065, describes the functions and capabilities of the AIX/370 Operating System and lists the requirements for all supported hardware and software. This book also includes information to assist with planning for installation and customization of the operating system.
- *AIX Operating System Technical Reference*, SC23-2300 (Vol.1) and SC23-2301 (Vol. 2) describes the system calls and subroutines a programmer uses to write application programs. This book also provides information about the AIX Operating System file system, special files, miscellaneous files, and the writing of device drivers.
- *Using the AIX Operating System*, SC23-2291, shows the beginning user how to use AIX Operating System commands to do such basic tasks as log in and out of the system, display and print files, and set and change passwords. It includes information for intermediate to advanced users about how to use communication and networking facilities and write shell procedures.

Other Publications

For general information about TCP/IP, the following publications are recommended. These publications are distributed by the Network Information Center on behalf of the Defense Communications Agency and Defense Advanced Research Projects Agency (DARPA). The mailing address is:

Network Information Center
SRI International
Menlo Park, CA 92025

- *An Advanced 4.3 BSD Interprocess Communication Tutorial*, S. Leffler, R.S. Fabry, W. N. Joy, P. Lapsley
- *An Ethernet Address Resolution Protocol*, RFC 826, D. Plummer
- *Assigned Numbers*, RFC 1010, J. Reynolds, J. Postel
- *Broadcasting Internet Datagrams*, RFC 919, J. Mogul
- *Domain Names - Concepts and Facilities*, RFC 882, P. Mockapetris
- *Domain Names - Implementation and Specification*, RFC 883, P. Mockapetris
- *File Transfer Protocol*, RFC 959, J. Postel R. Hinden, A. Sheltzer
- *Internet Control Message Protocol*, RFC 792, J. Postel
- *Internet Name Server Protocol*, IEN116, J. Postel
- *Internet Numbers*, RFC 997, J. Reynolds, J. Postel
- *Internet Protocol*, RFC 791, J. Postel
- *Internet Standard Subnetting Procedure*, RFC 950, J. Mogul
- *Name/Finger*, RFC 742, K. Harrenstien
- *Official ARPA-Internet Protocols*, RFC 944, J. Reynolds, J. Postel
- *Name Server Operations Guide for BIND*, Release 4.3, Kevin J. Dunlap
- *Standard for the Format of ARPA-Internet Text Messages*, RFC 822, D. Crocker

---

Ethernet is a registered trademark of Xerox Corporation.

- *Simple Mail Transfer Protocol*, RFC 821, J. Postel
- *Telnet Binary Transmission*, RFC 856, J. Postel, J. Reynolds
- *Telnet Option Specifications*, RFC 855, J. Postel, J. Reynolds
- *Telnet Protocol Specification*, RFC 854, J. Postel, J. Reynolds
- *Telnet Terminal Type Option*, RFC 930, M. Solomon, E. Wimmers
- *Time Protocol*, RFC 868, J. Postel., K. Harrenstien
- *Transmission Control Protocol*, RFC 793, J. Postel
- *Trivial File Transfer Protocol*, RFC 783, K. R. Sollins
- *User Datagram Protocol*, RFC 768, J. Postel

For additional information about the IBM Token-Ring Network, you may want to order the *IBM Token-Ring Architecture Reference*.

Contents

Figures

Chapter 1. Introduction

CONTENTS

About This Chapter

This chapter provides an overview of the AIX Operating System TCP/IP protocols and commands and explains the other features of the program.

The AIX Personal System/2 or System/370 Transmission Control Protocol/Internet Protocol (TCP/IP) program consists primarily of protocols and commands (application programs) that enable the AIX Operating System system to:

- communicate with other AIX Operating System machines with TCP/IP installed
- communicate with host systems that support TCP/IP.

Before You Begin

Before you can use the Transmission Control Protocol/Internet Protocol program (TCP/IP), you must have installed the following software and hardware on *each* system.

Software

- AIX Operating System licensed program
- AIX System/370 TCP/IP Licensed Program Product.

Hardware

- 802.3 adapter for use with Ethernet
- IBM Token-Ring Network adapter
- 8232 LAN Channel Station
- ES/9370 Integrated Ethernet adapters
- ES/9370 Integrated Token-Ring adapters
- Cables and connectors.

Refer to the following publication for information about installing the software:

- *Installing and Customizing the AIX Operating System*

After you install the AIX Operating System on your System/370, see Chapter 2, "TCP/IP Installation and Configuration" for information on installing TCP/IP on your system and configuring it to meet your personal requirements.

Overview

A Brief Look at TCP/IP

TCP/IP includes commands and facilities that allow users to:

- Install TCP/IP on your system
- Customize TCP/IP to your own needs
- Transfer files across the network
- Log in to remote systems
- Run commands on remote systems
- Print files on remote systems
- Manage networks
- Send and receive mail.

IBM is a registered trademark of International Business Machines Corporation.

Personal System/2 is a registered trademark and System/370 is a trademark of the International Business Machines Corporation.

AIX is a trademark of the International Business Machines Corporation.

Programmers who wish to develop applications that use the TCP/IP facilities should refer to the *AIX Operating System Technical Reference Manual* which describes the operating system calls and subroutines that provide access to these facilities.

The Internet Environment

The TCP/IP network protocols include the Internet Protocol (IP) transport layer and higher level protocols that use the Internet address format. TCP/IP provides facilities to make the System/370 system a host that attaches directly to a network.

The Internet environment consists of hosts connected to networks that use packet switching technology (for example, networks that use the 802.3 adapter for use with Ethernet). Networks, in turn, can be interconnected via ***gateways***. Host processes originate and receive network packets. Protocols at different levels in the networks, gateways, and hosts support interprocess communication, providing two-way data flow on logical connections.

Data is transmitted between process ***ports***. Each process may have a number of ports through which it communicates with other processes. Each port provides queues for sending and receiving data. A process may have logical connections to several other processes and, if necessary, can treat each of the connected processes independently.

Protocols

The TCP/IP licensed program provides the following protocols:

- Address Resolution Protocol (ARP)
- Internet Protocol (IP)
- Transmission Control Protocol (TCP)
- User Datagram Protocol (UDP)
- Other Network Protocols:
  - File Transfer Protocol (FTP)
  - Internet Control Message Protocol (ICMP)
  - Remote Command Execution Protocol
  - Remote Printing Protocol
  - Routing Information Protocol (RIP)
  - Telnet Protocol (TELNET)
  - Trivial File Transfer Protocol (TFTP).

Following are brief explanations of these protocols.

Address Resolution Protocol

Address Resolution Protocol (ARP) is a protocol that dynamically maps between Internet addresses and addresses on a local area network (LAN) with broadcast capability. It is used by the interface driver for the LAN hardware and is not directly available to users.

Using the LAN broadcast capability, ARP provides for mapping between the Internet addresses and the LAN addresses. When the Internet and LAN addresses are matched, the information is stored in the address mapping tables in the system.

When a LAN interface driver requests a mapping for an address not in its memory, ARP sends the ARP request packet (broadcast) on the network requesting the address mapping. If a response is provided, the new mapping is stored and any Internet packets pending for that address are transmitted.

Internet Protocol (IP)

Internet Protocol (IP) provides the interface from the higher level host-to-host protocols to the local network protocols. Addressing at this level is usually from host to host.

IP is used by host-to-host protocols (such as TCP/IP) as a basic transport mechanism. The IP is designed for use in interconnected systems of packet-switched computer communication networks. The network-connecting systems are called ***gateways***. IP provides the means for transmitting blocks of data from sources to destinations. Sources and destinations are hosts identified by fixed length addresses. All Internet packets have an IP header to standardize addressing and routing. (The IP header is removed before the packet is sent to the user.) This protocol provides the universal addressing of hosts in the network.

IP does not provide a reliable communications facility for many reasons. It does not provide acknowledgements from the sending host, from the receiving host, or from intermediate hosts. IP does not provide error detection or correction for data, although it does provide header checksums. IP treats each datagram as an independent entity unrelated to any other datagram. It provides only rudimentary flow control (with ICMP packets), and does not perform re-transmissions. A higher level protocol that uses IP must implement its own reliability procedures if it requires reliable communications. AIX System/370 implements the following IP architecture options:

- **End of Option List**
- **No Operation**
- **Loose Source Routing**
- **Strict Source Routing**
- **Stream Identifier**
- **Internet Timestamp**
- **Record Route**
- **Security.**

Security always defaults to unclassified.

IP provides for fragmentation and reassembly of complete messages. Type of Service (TOS) defaults to routine precedence and normal delay, throughput, and reliability.

The assignment of network numbers, internet version numbers, and internet protocol numbers complies with the Assigned Numbers RFC (Request for Comments). IP implements the special address **local host** and special address **loopback** (class A network address 127). The following IP system parameters can be set through the **/etc/rc.tcpip.local** file:

- Interface name
- IP address.

Transmission Control Protocol

Transmission Control Protocol (TCP) is used in the Advanced Research Projects Agency (ARPA) Internet and any network following the U.S. Department of Defense standards for inter-network protocols. This protocol provides a reliable host-to-host protocol between hosts in packet-switched computer communication networks and in interconnected systems of such networks.

All of the TCP architecture options are implemented including:

- **End of Option List**
- **No Operation**
- **Maximum Segment Size.**

The interface to TCP consists of a set of system calls and library subroutines. The system calls invoke operating system kernel functions in a manner similar to standard system calls such as **open**, **read**, or **write**. TCP/IP communicates asynchronously with application programs. TCP operates in a very general environment of interconnected networks. It supports the transmission of 8-bit bytes.

AIX System/370 supports a reasonably large number of potential TCP connections. The exact number depends upon a configuration parameter used when the kernel is built, and can vary with system resource usage. The assignment of TCP ports is in accordance with the Assigned Numbers RFC.

The TCP retransmission timeout value is dynamically determined for each connection, based on round-trip time. The timeout value cannot be set. The TCP system parameter that can be set is **window size**.

User Datagram Protocol

User Datagram Protocol (UDP) allows a datagram mode of packet-switched communication in the environment of an interconnected set of computer networks and is usually used for electronic mail applications. UDP provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction-oriented. It offers no guarantee of delivery and no protection from duplication. Applications that require reliable delivery of streams of data should use the Transmission Control Protocol. UDP is implemented as described in the *User Datagram Protocol*, RFC 768.

The interface to UDP consists of a set of system calls. UDP must be able to determine the source and destination Internet addresses along with the protocol field from the Internet header.

Other Network Protocols

Following are additional protocols used by TCP/IP:

File Transfer Protocol

FTP makes it possible to transfer computer files between hosts and to use foreign hosts indirectly.

Internet Control Message Protocol

In addition to the protocols for transmitting user data, the Internet Control Message Protocol (ICMP) is used to monitor and manage network data.

The ICMP protocol is an integral part of the IP implementation in the kernel. User programs do not normally interact with ICMP. Programs such as **ping** that send and receive ICMP packets must do so through the raw interface.

ICMP messages are sent automatically (from the user's point of view) in several situations, for example:

- When a datagram cannot reach its destination
- When the gateway does not have the buffering capacity to forward a datagram
- When the gateway can direct the host to send traffic on a shorter route.

In addition, the user can generate the following messages:

timestamp	to measure the round trip time
echo	to determine whether the foreign host is available
subnet mask	to indicate whether a host supports sub-networks

The AIX System/370 implementation sends and receives the message types listed below:

echo request	**echo reply**
destination unreachable	**parameter problem**
information request	**information reply**
timestamp request	**timestamp reply**
address mask request	**address mask reply**
source quench	**time exceeded**
redirect message	

Control messages provide feedback about problems in the communication environment. They do not make IP more reliable. There is no guarantee that a datagram will be delivered or that a control message will be returned.

The function provided by the daemon **icmpd** in other implementations of TCP/IP is provided in the kernel in AIX System/370 TCP/IP.

Remote Command Execution Protocol

The Remote Command Execution Protocol allows users to execute commands on remote hosts as if locally connected. The **rexecd** command provides the remote command execution protocol. For information about **rexecd** and the remote command execution, see "rexecd" on page 4-12.

Routing Information Protocol

The routing daemon (**routed**) uses a variant of the Xerox Network System (NS) Routing Information Protocol to maintain current kernel routing table entries. For related information, see "routed" on page 4-16.

Telnet Protocol

The Telnet Protocol (TELNET) provides a standard method to interface terminal devices and terminal-oriented processes to each other. TELNET assumes that TCP is the underlying protocol, provides for full-duplex (concurrently bi-directional) communication and sends data as either 7-bit netascii characters or as 8-bit binary data.

TELNET is commonly used by terminal emulation programs that allow you to log in to a foreign host. However, TELNET can also be used for terminal-to-terminal communication (called ***linking***) and inter-process communication (called ***distributed computation***). TELNET is also used by other protocols (for example, FTP) for establishing a protocol control channel.

TCP/IP implements TELNET in the **tn** user command and the **telnetd** server command. TCP/IP does not provide an Application Programming Interface (API) to TELNET. Refer to "telnet, tn, tn3270" on page 3-66 and "telnetd" on page 4-26 for more information on the **tn** and **telnetd** commands. Refer to RFC 854 (see "Related Publications" on page iv for information on ordering a copy) for additional information on TELNET and RFC 855 (see "Related Publications" on page iv for information on ordering a copy) for additional information on TELNET options.

Trivial File Transfer Protocol

TFTP's only function is to read and write files (or mail) to and from a foreign host. TFTP cannot list directories and it has no provision for user authentication. TFTP passes 8-bit bytes of data.

Internet Router

The Internet router enables a host running TCP/IP to act as a gateway for routing data between separate networks that use either of the following adapters:

- IBM Personal System/ 802.3 Ethernet
- IBM Personal System/2 Token-Ring.

One or two Ethernet adapters and one or two Token-Ring adapters can be installed on the System/370. The adapters (or interfaces) are made known to the system by running the **ifconfig** (interface configuration) program. (The **ifconfig** program should be called from the **/etc/rc.tcpip.local** file for each interface so that the interface is defined in the system routing tables whenever the host is rebooted.)

In addition, an entry for each adapter should be added to the file **/etc/hosts**. Each host has one primary name, but can have multiple entries in **/etc/hosts**. Additional entries in **/etc/hosts** define secondary names for the host.

Figure 1-1 depicts three networks, which are interconnected by two gateway hosts. The networks are:

Network Part of Address	**Hosts**
192.9.200	(hosts 1 through 12 with addresses 192.9.200.1 to 192.9.200.12)
192.9.201	(host5, host13, host14, and host15)
192.9.202	(host15, host16, and host17) Host5 and host 15 are the gateway hosts.

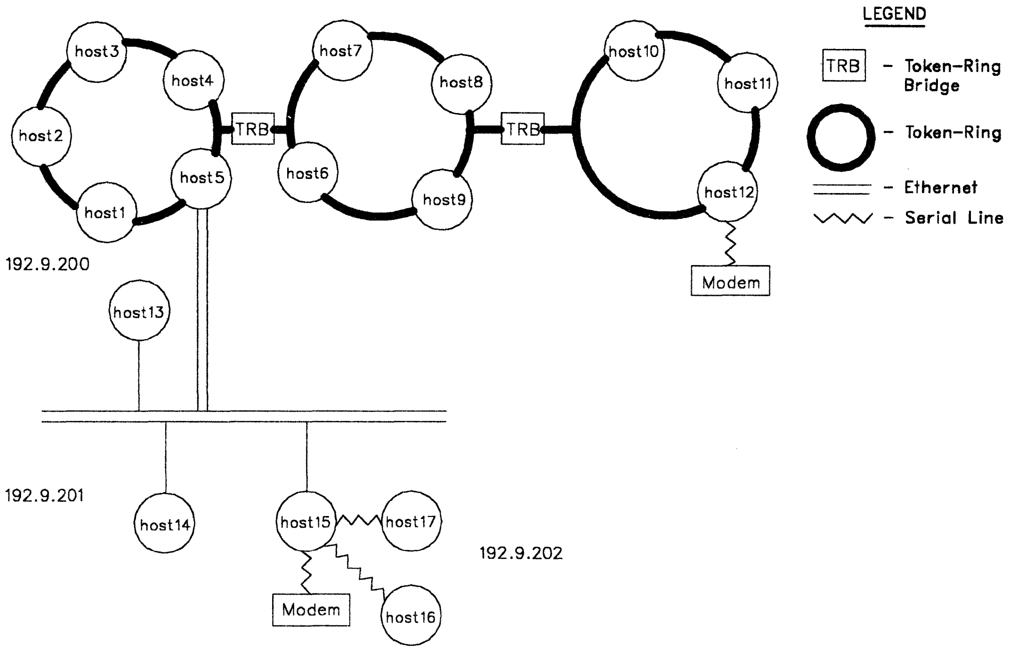


Figure 1-1. Network and Gateway Routing

There are two different types of routes:

NETWORK ROUTES Direct packets to any host on a specific network.

HOST ROUTES Direct packets to a specific Internet host.

Network routes are more general, but host routes may be used to support unusual network topologies, point-to-point links, or to affect system performance.

Routes can be established in three different ways:

- Implicitly, by running **ifconfig** for an interface adapter
- Explicitly, by adding explicit routes to the routing tables, using the **route** command
- Dynamically, by running **routed** (the routing daemon) on the gateway.

The **routed** daemon queries other defined gateway hosts periodically to obtain the information necessary to generate and update routing tables. This daemon uses two files - **/etc/gateways** and **/etc/networks** - to determine which hosts routing information must be exchanged. For more information, see "routed" on page 4-16 and "gateways" on page 5-2.

In Figure 1-1, the gateway host5 contains two interfaces - an Ethernet adapter (for gatewaying to network 201) and a Token-Ring adapter (to gateway to network 200). Host15 contains two interfaces - one Etherent adapter and three serial connections.

A route to a particular host is obtained using the following algorithm:

- Search the route table for a host route. If one is present, use that route.
- Search the route table for a network route. If one is present, use that route.
- If a default route is specified, use that route.

If no route can be found, the error `Network is unreachable` is generated.

If a route cannot be established to a host, the error message `Network is unreachable` is generated locally.

In the configuration of Figure 1-1 on page 1-8, routing can be established for **host1** by simply deciding on a default gateway for the network. To establish **host5** as the default gateway on the network that includes **host5** and **host1**, enter the following **route** command on **host1**: **/etc/route add default host5 (where 0 is the hop count** metric **in this case).** This **route** command establishes **host5** as the exit point, or default gateway, from the network. For more information, see "route" on page 3-56.

On the 201 network, there are two gateways (**host5**, the default gateway, and **host15**). The preferred routing is to the default gateway.

An alternate way to establish routing to the other network from the 202 network is to set up routes explicitly with the following **route** commands:

```
route add 192.9.200 host5   4
route add 192.9.202 host15  5
```

If the locally administered systems are set up to use **routed**, then explicit routing with the **route** command is not necessary.

If the **route** command is used for routing, the routing table must be configured on both networks. For example, for **host1** on network **200** to have access to all hosts on network **201, host1** must issue the **route** command and all hosts on the **201** network must issue the **route** command to access the **200** network. (See Figure 1-1 on page 1-8).

Commands

TCP/IP provides three general types of commands:

- File transfer
- Network management
- Printing, Remote Login, Command Execution and Conversation.

Following are brief explanations of the individual commands in each of these groups.

File Transfer

TCP/IP contains three file transfer commands: the **ftp** command, the **rcp** command and the **tftp** command. The **ftp** command provides more functions and is the preferred command for most file transfer tasks.

ftp The **ftp** command implements the File Transfer Protocol (FTP), which makes it possible to transfer data among hosts and to use foreign hosts indirectly.

The **ftp** command can be used to transfer files between the user and server or between two hosts. The user must request the close of the telnet connection when use of the FTP service is finished.

Some of the commands that can be used with **ftp** are:

get	Retrieves remote files and stores them on the local machine.
ls	Prints a listing of a directory on a remote machine when the remote-directory is specified.
put	Stores local files on a remote machine.

In addition, the **ftp** command provides subcommands for such tasks as changing directories (local and foreign), transferring multiple files in a single request, creating and removing directories, escaping to the shell (performing shell commands locally), and many other functions. For more information, see "ftp" on page 3-6.

rcp The **rcp** command copies and transfers remote files. For more information, see "rcp" on page 3-44.

tftp The **tftp** command implements the Trivial File Transfer Protocol (TFTP). Its only function is to read and write files (or mail) to and from a foreign host. The **tftp** command cannot list directories and it has no provisions for user authentication. The **tftp** command passes 8-bit bytes of data.

The TFTP protocol is designed to be implemented using UDP. Since UDP is implemented using Internet Protocol, packets have an IP header, a UDP header, and a TFTP header. Additionally, the packets may have a header to allow them through the local transport medium.

For related information, see:

- "tftp" on page 3-73
- "tftpd" on page 4-28

Network Management

arp The **arp** command performs address resolution display and control.

finger The **finger** command returns information about users on the specified host.

host The **host** command determines the network address of the specified host machine.

hostid The **hostid** command prints the identifier of the current host in hexadecimal. (This value is the host's Internet address.) The superuser may use the **hostid** command to set the Internet address of the current host.

hostname The **hostname** command shows or sets the name and address of the local host.

ifconfig	The **ifconfig** command is used to assign an address to a network interface or to configure network interface parameters.
netconfig	The **netconfig** command configures network interface parameters.
netstat	The **netstat** command shows local and foreign addresses, routing tables, hardware statistics, and summary of packets transferred.
ping	The **ping** command sends an echo request to a network host to determine whether that host is operational and on the network.
rdist	The **rdist** command maintains identical copies of files over multiple hosts.
route	The **route** command permits you to manually manipulate the routing tables.
ruptime	The **ruptime** command gives a status line like *uptime* for each machine on the local network.
rwho	The **rwho** command produces output similar to **who** for all machines on the local network.
timedc	The **timedc** command is used to determine the difference between the local host machine's time and that of another named host.

Remote Login, Command Execution and Conversation

rexec	The **rexec** command makes it possible to execute commands remotely without maintaining a TELNET session. For additional information, see "rexec" on page 3-51 and "rexecd" on page 4-12.
rlogin	The **rlogin** command connects your terminal on the current local host system **lhost** to the remote host system **rhost**.
rsh	The **rsh** command connects to the specified **host** and executes the specified command.
talk	The **talk** command allows you to converse with another user.
telnet, tn, tn3270	The **telnet** command is a terminal emulation program that allows you to log in to a foreign host. It implements the Telnet Protocol (TELNET), a bi-directional, 8 bit-per-byte communications facility and provides a standard method to interface terminal devices and terminal-oriented processes. It can also be used for terminal-to-terminal communication (called ***linking***) and inter-process communication (called ***distributed computation***). See "telnet, tn, tn3270" on page 3-66 and "telnetd" on page 4-26 for related information.

Security Features

This section describes the security features provided for TCP/IP and some security considerations that are appropriate in a network environment.

The **rexec**, **telnet**, and **ftp** functions provide the following forms of system and data security:

rexec	The **rexec** command provides a secure environment for executing commands on a foreign host. **rexec** first attempts an automatic login: on the local host, **rexec** searches the **$HOME/.netrc** file for the user's ID and password on the foreign host. If the **$HOME/.netrc** file does not exist, the user is prompted for both a login ID and a password.

For security, the permissions on **$HOME/.netrc** must be set to 600 (read and write by owner only). Otherwise, automatic login fails.

The **-n** flag disables the automatic login feature, forcing a query for the user ID and password.

For more information, see "rexec" on page 3-51 and ".netrc" on page 5-12.

telnet The **telnet** function provides a secure environment for logging into a foreign host. The user is prompted for both a login ID and a password. The user's terminal is treated just like a terminal connected directly to the host. That is, access to the terminal is controlled by permission bits. Other users (group and other) do not have read access to the terminal, but they can write messages to it if the owner gives them write permission.

ftp The **ftp** command provides a secure environment for transferring files. **ftp** first attempts an automatic login: on the local host, **ftp** searches the **$HOME/.netrc** file for the user's ID and password on the foreign host. If the **$HOME/.netrc** file does not exist, the user is prompted for a login ID; the current user's login ID is given as a default. If a password is associated with that login ID at the foreign host, the user is prompted for a password.

For security, the permissions on **$HOME/.netrc** must be set to 600 (read and write by owner only). Otherwise, automatic login fails.

The **-n** flag disables the automatic login feature, forcing a query for the user ID (and password, if required).

Data Security and Information Protection

TCP/IP does not encrypt user data transmitted through the network. Therefore, it is suggested that users identify any risk in communication that could result in the disclosure of passwords and other sensitive information, and based on that risk, apply appropriate countermeasures.

The use of this product in a Department of Defense environment may require adherence to DoD 5200.5 and NCSD - 11 for communications security.

Program Customization

This section describes the usual means for customizing TCP/IP to meet your requirements.

/etc/hosts

The **/etc/hosts** file contains the names and associated addresses for hosts on the network. That is, **/etc/hosts** lists the foreign hosts with which a local host can communicate. For more information, see "hosts" on page 5-4.

/etc/hosts.equiv

The **/etc/hosts.equiv** file contains the names of the foreign hosts that are permitted to perform remote operations on a particular host.

/etc/networks

The **/etc/networks** file contains information about known networks that comprise the DARPA network. For more information, see "networks" on page 5-13.

/etc/rc.tcpip

The **/etc/rc.tcpip** file is a shell script that defines which adapters are used for the TCP/IP, initializes the host names and routing tables, and starts the daemons. The **/etc/rc.tcpip** file may be tailored to the requirements of a particular host. For more information, see "rc.tcpip, rc.tcpip.local" on page 5-15 and Chapter 2, "TCP/IP Installation and Configuration."

<LOCAL>/rc.tcpip.local

The /etc/rc.tcpip.local file is stored on the local file system of each site in the TCF cluster. It can be customized by the system administrator to run commands specific to a particular cluster site.

finger

The **finger** command provides information about a specific user. This information is retrieved from the full name and site information fields of the **/etc/passwd** file. The format of the full name is the user's real name (or whatever is entered in the full name field of **/etc/passwd**, not the user's user ID). The site information field is 20 characters long and has the following format (no imbedded blanks, fields separated by commas):

```
office_number,office_phone_extension,home_phone
```

Examples of office extensions are 123 and 89456.

Note: To update the **/etc/passwd** file, use the **chfn** command.

If you want to send additional information to other users who run **finger** on your user ID, you can include the following files in your home directory:

.plan	A file that contains plans. The **.plan** file can contain more than one line.
.project	A file that states what project you are currently working on. The **.project** file can contain only one line.

For additional information about the **finger** command, see "finger" on page 3-4.

$HOME/.netrc

The **$HOME/.netrc** file contains the user ID and password information required by the automatic login features of **rexec** and **ftp**. For more information, see ".netrc" on page 5-12.

Related Information

In this book:
"ifconfig" on page 3-23
"route" on page 3-56
"hostname" on page 3-21
"inetd" on page 4-6

Chapter 2. TCP/IP Installation and Configuration

CONTENTS

About This Chapter

This chapter contains step-by-step instructions on installing TCP/IP on your system.

Installing TCP/IP on the Personal System/2

To install TCP/IP from the distribution diskette on a Personal System/2, the following procedures are required. TCP/IP is automatically installed on the System/370. However, if you ever need to install TCP/IP separately on a System/370, use the **MasterInstall** command. After you've chosen TCP/IP from the menu, follow the PS/2 instructions below.

Important Note: When **installp** prompts you for **fstore** values, do not answer "none." If you do, the necessary TCP/IP files will not be propagated to all the hosts on the TCP cluster.

To install TCP/IP on your Personal System/2:

1. Log into the system as **root**.
2. Insert the distribution diskette into the diskette drive and make sure it locks in place.
3. Type:

   ```
   # installp -d /dev/rfd0h
   ```

   The system responds:

   ```
   000-123  Before you continue, you must make sure there is no other
   activity on the system.  You should have just restarted the system and
   no other terminals should be enabled.  Refer to the AIX System/370
   Message Reference manual for more information.

   Do you want to continue with this command? (y or n)
   ```
4. Type:

   ```
   y
   ```

   The system responds:

   ```
   Please mount volume 1 on /dev/rfd0
   ```
5. Confirm that the distribution diskette is in the diskette drive and press **Enter**.

   The system responds:

   ```
     files restored:  1

   -

   The program "AIX TCP/IP" will be installed

   Do you want to do this? (y/n)
   ```
6. Type

   ```
   y
   ```

   to install TCP/IP. The system responds with the number of files restored and a copyright message. The following message is displayed:

   ```
   Please mount volume 1 on /dev/rfd0
   ... and press Enter to continue.
   ```

7. Confirm that the distribution diskette is in the diskette drive and press **Enter**.

   The system displays the names of the files as they are copied from the distribution diskette. When all the files are copied, the following messages are displayed:

   ```
   files restored:  81

   _
   Program "AIX TCP/IP" is now installed

   flag is -n
   queue pointer is n
   File System Backup Utility
   Done at Fri May 20 14:38:07 1988
   queue pointer is n
   stack pointer is n
   ```

8. At this point, the system displays the following prompts:.

   ```
   You will now be prompted for an internet style address to associate
   with your network interface device; the format of this address is
   "X.X.X.X." where each "X" is a number in the range
   of 0-255.  The address should be obtained from your local network
   administrator.

   Enter Internet address for <sitename>

   where <sitename> is the name of your site.
   ```

9. Enter the internet address for your computer and press **Enter**.

   The internet address looks similar to the following:

   ```
   128.100.12.204
   ```

   The system then displays the following messages:

   ```
   You will need to associate your network address with a Network
   Interface Controller (NIC).  Currently supported are Token-Ring and
   Ethernet interfaces.

   Enter your primary network interface
   T(okenring) E(thernet):
   ```

10. Enter `E` for Ethernet or `T` for Token-ring and press **Enter**.

    After you enter this information, the system displays the information you have entered and then prompts you to commit the information. These messages look similar to the following:

    ```
    You have chosen:
           Network address:  193.255.1.31
           Network interface:  Ethernet

    Do you wish to use these values?
    ```

11. If the information is correct, type `Y` and press **Enter**.

    When you press Enter, you see the following message:

    After this installation has completed, you must run the "devices" command to install the network devices(s) and rebuild the kernel for this site. For more information about installing devices and the "devices" command, see *Installing and Customizing the AIX Operating System,* Chapter 6.

    To use TCP/IP you must use **devices** to set up at least one **pty** device.

    To start **devices**, you must first be logged in as a member of the system group or you must have superuser authority. The # prompt appears when you are logged in as superuser. The $ prompt appears when you are logged in as a system or regular user.

12. Type `devices` and press **Enter**.

    The Device Customizing Commands screen is displayed:

```
                  DEVICE CUSTOMIZING COMMANDS


Devices commands are:

Command    Description

add        Add a new device.
change     Change information about an existing device.
delete     Delete an existing device.
showall    Show general information about all existing devices.
showdev    Show detailed information about an existing device.

To EXIT the devices command, press F3.
To USE a devices command, type the Command and press Enter.

>
```

13. Type a to select the **add** command and press **Enter**.

    A list of device classes is displayed.

```
The following device classes are available.

Device Class     Description

printer     Printer or Plotter
ttydev      Asynchronous Terminal (Any Non-Prntr Device on an Async Port)
tape        Streaming Tape Drive
ptydev      Asynchronous Pseudo-Terminal
adapters    IBM System/370 Adapters
lan         Local Area Network

To CANCEL and return to the list of commands, press F3.
To CHOOSE from the list, type the Device Class and press Enter.

>
```

14. Type ptydev to add a **pty** device.

    The next screen displays two choices.

15. Type ttyp and press **Enter**.

    The following screen is displayed.

```
Name       Description               Current   Possible

ae        Automatic Enable            false  true, false, share, delay
tt        Terminal Type               dumb   vt100, vt100-em
bpc       Bits Per Character          8
nosb      Number of Stop Bits         1      1, 1.5, 2
pt        Parity Type                 none   even, odd, mark, space, none
elevel    Enable Level for Terminal   1,4    one or more of 0-6, a, b, c
logger    Pby Supports Login Shell    false  true, false

To CANCEL and return to the list of commands, press F3.
To CHANGE a current choice, type the name followed by your new choice
   (example: lpi 6) and press Enter.
To ADD this device with the current choices, press Enter.

>
```

16. Press **Enter**.

    The device is added to the system and the following screen is displayed:

```
The device was added to the system.

Device Name:  ttyp0
Device Type:  ttyp
Device Class: ptydev

To return to the list of commands, press Enter.

>
```

    This screen confirms the information about the device added. From this screen, you can return to the list of commands.

17. Press **Enter**

    The list of commands is redisplayed. At this point, you have completed all the steps necessary to add the device. You can now add another **pty** device or exit from the **devices** command. To add another **pty** device, repeat this procedure from Step 13. To exit from the **devices** command,

18. Press **F3.**

    After you have added all the necessary devices, you should reboot your system to enable the new devices.

19. Type `sync;sync;sync;/etc/reboot`

The system reboots and the new devices are enabled.

Problems with Devices

If the rebuild process does not work properly after you exit the **devices** program, the following messages are displayed:

```
Devices session ended.

All device customization has successfully completed, but rebuilding the
kernel failed.  Please refer to Using the AIX Operating System
and Managing the AIX Operating System manual for
instructions on how to generate the new kernel.

#
```

In this case, you must rebuild and enable the kernel yourself. To rebuild the kernel, at the # prompt:

1. Type `cd /usr/sys` and press **Enter**.
2. Type `newkernel` and press **Enter**.

   The kernel is rebuilt. You should then reboot your system to enable the new kernel.

   To reboot your system,
3. Type `sync;sync;sync;reboot` and press **Enter**.

Configuring TCP/IP

Whether you have installed TCP/IP manually on a Personal System/2 as described above or TCP/IP was installed automatically, you should complete the following installation steps:

1. You will need to change either the **/etc/rc.tcpip** file or the **<LOCAL>/rc.tcpip.local** file, depending on the network hardware devices you use. If you use the same network hardware devices on all your hosts, you only need to add the device type to the **/etc/rc.tcpip** file. If you use different network hardware devices on different hosts, you will need to put each host's device type in the host's **<LOCAL>/rc.tcpip.local** file, and delete the device configuration line from the **/etc/rc.tcpip** file. (See step 2 below.)
2. Search for the line containing the string **/etc/ifconfig**.

   This line should look similar to the following:

   ```
   /etc/ifconfig net0 $SITE # default to Pegasus board
   ```

   Note: The `$SITE` variable in this line is for the case where the IP host name is the same as the AIX site name. If these names are not the same, or if the site has more than one IP interface, replace the `$SITE` with the correct IP host name for the interface.
3. Change this line, if necessary, so that it is correct for the type of network you are using. The instructions below will help you to determine what change to make.

The string following **/etc/ifconfig** should read as follows:

1. If you are using Ethernet on a PS/2, this string should be net0. In this case, the line should read:

   ```
   /etc/ifconfig net0 $SITE
   ```
2. If you are using Token-Ring on a PS/2, this string should be tk0:

   ```
   /etc/ifconfig tk0 $SITE
   ```
3. If you are using a 9370 integrated Ethernet adapter on a 370, this string should be ce0:

   ```
   /etc/ifconfig ce0 $SITE
   ```
4. If you are using a 9370 integrated Token-Ring adapter on a 370, this string should be il0:

   ```
   /etc/ifconfig il0 $SITE
   ```

5. If you are using an 8232 LAN channel station Ethernet adapter on a 370, this string should be lcAe0:

   ```
   /etc/ifconfig lcAe0 $SITE
   ```

6. If you are using an 8232 LAN channel station Token-Ring adapter on a 370, this string should be lcAt0:

   ```
   /etc/ifconfig lcAt0 $SITE
   ```

7. If you are using an X.25 adapter on a PS/2, this string should be:

   ```
   /etc/ifconfig  xip0 $SITE
   ```

You might want to change the comment at the end of the line to reflect whether you are using ethernet or token-ring. Once you have made these changes, you should run **devices** to add the network device that you want to use. TCF kernels are distributed with network devices preconfigured and some pty devices are also included. To use TCP/IP, you must set up at least one pty device.

To start **devices**, you must first be logged in as a member of the system group or you must have superuser authority. The # prompt appears when you are logged in as a superuser. The $ prompt appears when you are logged in as a system or regular user.

TCP/IP Routing

By default, TCP/IP uses the file **/etc/hosts** to resolve network addresses. You can add additional network addresses and hostnames to this file to allow your system to access other systems on the network. To add a hostname,

1. Edit the file **/etc/hosts**.
2. Add a line containing your site address and hostname at the end of the list of site adddress and hostnames. The line you add should use the following syntax:

   ```
   XXX.XXX.XXX.XXX <TAB> sitename # comments
   ```

   where XXX.XXX.XXX.XXX is the network address of the system you want to add, *sitename* is the hostname of the system. Because AIX is case-sensitive, you should use lower-case letters for the sitename. The comment field is optional and begins with a #. For example, your entry may look similar to the following:

   ```
   193.255.1.31        bragi#  entry for bragi
   ```

   In addition, you can use the **route** command to manage traffic over your network. Using the **route** command, you can specify the network path, or route, used to access a specific system on the network. In this way, you can route network traffic away from highly loaded or unreliable systems on the network.

   You use the **route** command to add or delete network routes from your system's default routing table. The **route** command has the following format:

   ```
   route (add/delete) <destination> <gateway> <metric>
   ```

   where the <destination> is the system you want to access, <gateway> is the system through which you want to pass to reach the <destination> and <metric> is a positive integer (usually 5), indicating the number of hops to the <destination>. For more information on the **route** command, see "route" on page 3-56.

You can use either host names or network addresses to specify the <destination> and <gateway>. For example, to address **host1** (address 189.33.5.1) through **host2** (address 187.45.6.1), you might type

```
route add host1 host2 5
```

or

```
route add 189.33.5.1  187.45.6.1 5
```

TCP/IP Nameservers

Your system can treat other systems on the network as nameservers. To use another system as a nameserver, create a file called **/etc/resolv.conf** on your system and add a line to the file for the system you want to use as a nameserver. The line you add to this file should use the following format:

```
nameserver <TAB> XXXX.XXXX.XXXX.XXXX
```

where `XXXX.XXXX.XXXX.XXXX` is the network address of the host you want to use as the nameserver.

For more information, see "resolv.conf" on page 5-17.

Configuring the Network

Before you can use TCP/IP, you must customize your installation for the network on which TCP/IP runs. To do this, follow the steps given below. You should note that some of the steps are optional. Optional steps are indicated.

1. Install the hardware required to support the networking communications facilities. (At present, the AIX supports the Ungerman-Bass Ethernet adapter, and the IBM Token-Ring adapter.)

   If you have a PS/2, you can add up to 2 Ethernet adapters, up to 2 Token-Ring adapters, or 1 of each type. If you install two of the same network hardware devices in a PS/2, the device in the lower numbered slot will always be assigned 0 as its unit number, e.g., **net0**. The device in the higher numbered slot will always have 1 as its unit number. So, if you initially install one network hardware device in your PS/2, and think you may eventually add another of the same kind, put the initial device in the lowest numbered slot possible so its unit number won't change if you add another network hardware device of the same kind.

   If you have a 370, you can add up to eight 8232 LAN channel station adapters, eight 9370 integrated Ethernet adapters, and eight 9370 integrated Token-Ring adapters. Note, however, that most hardware configurations are more restrictive.

2. Use the **devices** command to add the adapters.

3. Use the **devices** command to add PTYs if your system is to be available to other users for remote login (as in **telnet** sessions). Select the appropriate terminal type for each PTY just as you would for a TTY). If that PTY is to be used for **telnet** (remote login to this host), set the values of **logger** and **automatic enable** to FALSE. (**automatic enable** causes the **penable** command to be run for that device at system start, making it available for use by **telnet**.)

4. Customize the **/etc/hosts** file according to the information under "hosts" (See "hosts" on page 5-4).

5. Delete from **/etc/rc.tcpip** any **/etc/ifconfig** lines that do not apply to every host in your network, such as network hardware devices. If you use more than one kind of network hardware device in your network, add the following line to each host's **< LOCAL > /rc.tcpip.local** file:

   ```
   /etc/ifconfig dev# $SITE
   ```

   dev# is the network hardware device name (see table) followed by a unit number, such as `net0`. If all of the hosts on your network use the same kind of network hardware device, you only need to put the **/etc/ifconfig** *dev#* **$SITE** line in the /etc/rc.tcpip file. (See "rc.tcpip, rc.tcpip.local" on page 5-15 for additional information.)

Device Name	**Manufacturer and Product**
lo	Software loopback
net	Ungerman Bass Ethernet for the PS/2
tk	IBM Token-Ring for the PS/2
ceti	Ethernet device (9370 only)
ilans	Token-Ring (9370 only)
vctc	virtual channel-to-channel
lc [A|B] [e|t]	Lan Channel Station (8232) (either Ethernet or Token-Ring)
xip	X25 interface for the PS/2

Steps 6 through 9 are optional:

6. Customize the **/etc/hosts.equiv** file according to the information under "hosts.equiv" on page 5-6.

7. Run **/etc/routed** on all hosts on the network that function as gateways. The **routed** command can also be run on non-gateway machines. To run **/etc/routed** automatically when the system is started, remove the comment symbol from the beginning of the **/etc/routed** line in the **/etc/rc.tcpip** file.

8. The **routed** command refers to the **/etc/gateways** and the **/etc/networks** files, if they exist, for a broader view of the networks.

9. Customize the **/etc/networks** file according to the information under "networks" on page 5-13.

   Note: The name of the host running must be used as the parameter to **/bin/hostname** in **/etc/rc.tcpip**.

10. Customize the **/etc/gateways** file according to the information under "gateways" on page 5-2.

11. If the host is to act as a nameserver, remove the comment symbol (#) from the lines that start the name daemon (**/etc/named**) in the **/etc/rc.tcpip** file. You must also edit the file **/etc/named.boot** so it contains network specifications. (Refer to "named" on page 4-8.) Note that on the distributed system, the name date base file referred to as **named.db** in the **named** description will be called **domainhosts**. This data base file must be edited to contain the appropriate names for the server's data base. (See "named" on page 4-8)

12. If the host is to act as a timeserver, remove the comment symbol (#) from the lines that start the time daemon (**/etc/timed**) in the **/etc/rc.tcpip** file.

13. In **/etc/rc**, remove the comment symbol (#) from the line that starts the **inetd** daemon process. This causes the network to be configured and the daemon started when the system is rebooted.

Chapter 3. User Commands

CONTENTS

About This Chapter

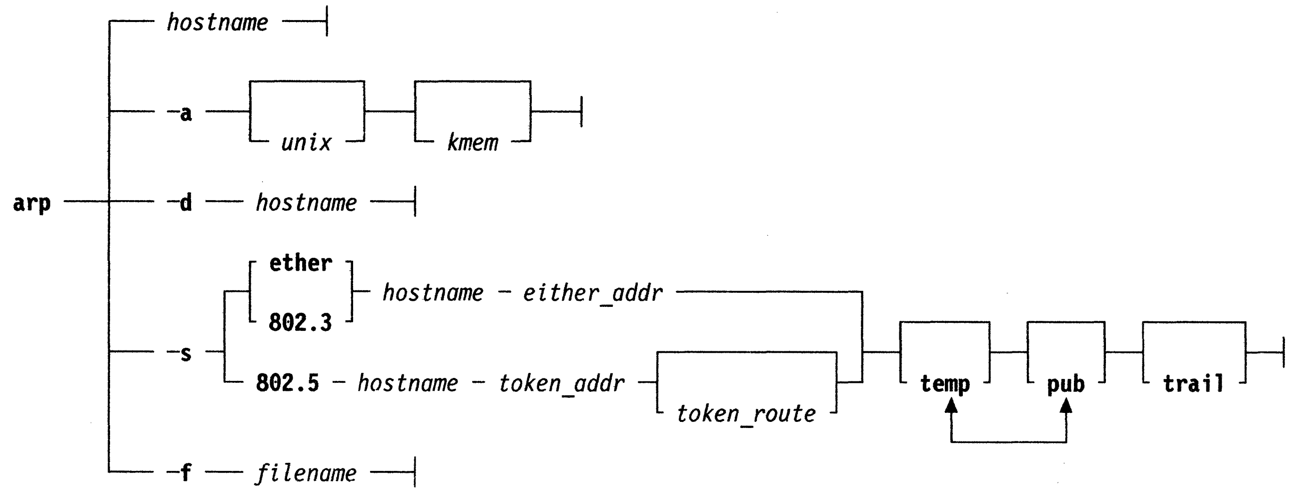
This chapter describes the user commands that are part of the TCP/IP program for the AIX Operating System. The commands are organized alphabetically.

arp

Purpose

Performs address resolution display and control.

Syntax



Description

The **arp** command displays and modifies the Internet-to-Ethernet address translation tables used by the Address Resolution Protocol (ARP). With no flags, the program displays the current **arp** entry for *hostname*, as specified in **/etc/hosts**. The host may be specified by name or by number using Internet dot notation. For more information on Address Resolution Protocol, see "Address Resolution Protocol" on page 1-3.

Flags

The **arp** command options are:

-a Displays all of the current **arp** entries by reading the table from the file, **kmem**, (**default /dev/kmem**) based on the kernel file **unix** (**default /unix**).

-d Deletes an entry for the host called *hostname* for the superuser.

-f Causes the file *filename* to be read and multiple entries to be set in the **arp** tables. Entries in the file should be in the following form:

```
hostname ether_addr [temp] [pub][trail]
```

The Ethernet address is given as 6 hex bytes separated by colons. The entry is permanent unless the word **temp** is given in the commands. If the word **pub** is given, the entry is published. The word **trail** indicates that trailer encapsulations can also be sent to this host.

-s Creates an **arp** entry for the host called *hostname* with the Ethernet address, *ether-addr* or the Token-Ring address *token-addr*. The Ethernet or Token-Ring address is given as 6 hex bytes separated by colons. The entry is permanent unless the word **temp** is given in the commands. If the word **pub** is given, the entry is published. The word **trail** indicates that trailer encapsulations can also be sent to this host. The optional *token-route* parameter gives the Token-Ring route to the host in a multiple Token-Ring configuration. The *token-route* is given as one to nine hex numbers separated by colons. The hex numbers may be of up to 4 digits each.

This means that the host accepts data packets of the following format:

```
[fixed_header] [packet_data] [trailer][additional_header]
```

By moving the bulk of the network headers to the end of the packet and preceding them with a trailer needed to help the protocol driver to find the additional headers, you can improve performance by arranging to move the data packets to page-aligned boundaries on the host machine.

Examples

```
# arp -a
draco (193.255.2.15) at 0:dd:0:fb:84:0
coins (193.255.2.7) at 0:dd:0:39:a:0
sabik (193.255.2.26) at 0:dd:1:2:0:cc
electra (193.255.2.18) at 0:dd:1:2:0:cd
cups (193.255.2.29) at 0:dd:0:b:69:0
frodo (193.255.2.2) at 2:60:8c:0:34:63
swords (193.255.2.4) at 0:dd:0:84:b8:0
wheels (193.255.2.24) at 0:dd:1:1:3a:bb
merak (193.255.2.16) at 0:dd:1:2:0:cb
spica (193.255.2.8) at 0:dd:1:1:3a:c3
#

# arp swords
swords (193.255.2.4) at 0:dd:0:84:b8:0
#


# arp nonesuch
nonesuch (193.255.1.4) -- no entry
#
```

Related Information

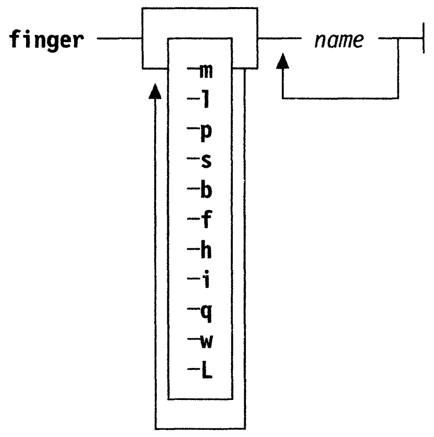
In this book: "ifconfig" on page 3-23
inet, refer to the *AIX Operating System Technical Reference*

finger

Purpose

Shows user information.

Syntax



Description

By default, the **finger** command lists the login name, full name, terminal name and write status (as a * before the terminal name if write permission is denied), site, idle time, login time, office location and phone number, if known, for each current AIX user on the TCF cluster. Idle time is minutes if it is a single integer, hours and minutes if a (**:**) is present, or days and hours if a **d** is present.

A longer format also exists and is used by the **finger** command whenever a list of people's names is given. Account names as well as first and last names of users are accepted. This format is multi-line and includes all the information described above as well as the user's home directory and login shell, any plan which the person has placed in the file **.plan** in their home directory, and the project on which they are working from the file **.project** also in the home directory.

The **chfn** command allows you to change the information the **finger** command displays about you.

The **finger** command can be used to look up users on a remote machine. The format is to specify the user as **user@host.** If the user name is left off, the standard format listing is provided on the remote machine.

Flags

The **finger** command options include:

-b	Gives a brief output (directory and shell not printed).
-f	Suppresses header on short format output.
-h	Suppresses printing of the **.project** files.

-i	Prints out idle time using short format. Short format is used even if the **-l** option is used.
-l	Forces long output format.
-L	On short format output, only lists user logged in to the local.
-m	Matches arguments only on user name.
-p	Suppresses printing of the **.plan** files.
-q	Gives brief output in short format (prints login, tty, and date of log in).
-s	Forces short output format.
-w	Suppresses real name when **-s** is used with **-w**.

Examples

```
$ finger
Login      Name           TTY Site   Idle    When           Office
jefff    Jeff Fields       C0 host1    25 Fri 07:54
tsmith   Tim Smith          1 host1       Thu 16:37  3330-213     x363
tim      Tim Burt           2 host2    12 Thu 09:20
jon      Jon Buerge        p1 host3    1d Wed 08:00
$

$ finger tim
Login name: tim                         In real life: Tim Burt
Directory: /u2/tim                      Shell: /bin/csh
On since Jun 16 09:20:13 on tty3 at host2.  12 minutes Idle Time
No Plan.
$ finger tsmith
Login name: tsmith                      In real life: Tim Smith
Office: 3330-213,  x363                 Home phone: 207-1041
Directory: /u3/tsmith                   Shell: /bin/csh
On since Jun 16 16:37:15 on tty1 at host1.
Project: TCP/IP command examples
Plan:  AIX System/370 documentation.
$
```

Files

/etc/utmp	who file
/etc/passwd	for users names, offices
~/.plan	plans
~/.project	projects -- only the first line of the **.project** file is printed.

Related Information

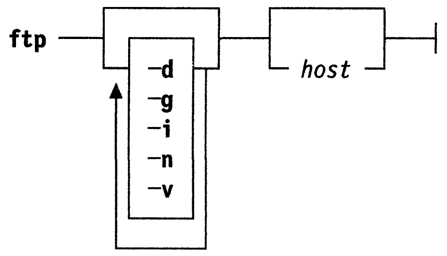
In this book: "finger" on page 3-4
who, refer to the *AIX Operating System Commands Reference*
chfn, refer to the *AIX Operating System Commands Reference*

ftp

Purpose

Transfers files between local and remote host.

Syntax



Description

The **ftp** command is the user interface to the ARPANET standard File Transfer Protocol. The program allows a user to transfer files to and from a remote network site.

Flags

The **ftp** command options are:

-d Enables debugging.

-g Disables file name globbing.

-i Shuts off interactive prompting during multiple file transfers.

-n Restrains **ftp** from attempting auto login upon initial connection. If auto login is enabled, **ftp** checks the **.netrc** file (see "The .netrc File" on page 3-13) in the user's home directory for an entry describing an account on the remote machine. If no entry exists, **ftp** prompts for the remote machine login name (default is the user identity on the local machine), and, if necessary, prompts for a password and an account with which to log in.

-v (verbose on)
Forces **ftp** to show all responses from the remote server, as well as report on data transfer statistics. By default, verbose is set to "on." To set verbose off, use verbose sub-command.

Commands

The client host, which **ftp** is to communicate with, may be specified on the command line. If this is done, **ftp** immediately attempts to establish a connection to an FTP server on that host. Otherwise, **ftp** enters its command interpreter and awaits instructions from the user. When **ftp** is awaiting commands from the user, the prompt `ftp>` is provided to the user. The following commands are recognized by **ftp**:

! [*command* [*args*]]
Invokes an interactive shell on the local machine. If there are arguments, the first is taken to be a command to execute directly, with the rest of the arguments as its arguments.

$ *macro-name* [*args*]	Executes the macro *macro-name* that was defined with the **macdef** command. Arguments are passed to the macro unglobbed.
account [*passwd*]	Supplies a supplemental password required by a remote system for access to resources once a login has been successfully completed. If no argument is included, the user is prompted for an account password in a non-echoing input mode.
append *local-file* [*remote-file*]	Appends a local file to a file on the remote machine. If *remote-file* is left unspecified, the local file name is used in naming the remote file after being altered by any **ntrans** or **nmap** setting. File transfer uses the current settings for **type, format, mode,** and **structure**.
ascii	Sets the file transfer type to ascii. This is the default type.
bell	Arranges for a bell to sound after each file transfer command is completed.
binary	Sets the file transfer type to support binary image transfer. Binary image transfer should be used for all binary files.
bye	Terminates the FTP session with the remote server and exits **ftp**. An end of file also terminates the session and exits **ftp**.
case	Toggles remote computer file name case mapping during **mget** commands. When **case** is on (default is off), remote computer file names with all letters in upper case are written in the local directory with the letters mapped to lower case.
cd <*dir*>	Changes the remote machine working directory to *dir* as indicated.
cdup	Changes the remote machine working directory to the parent of the current remote machine working directory.
close	Terminates the **ftp** session with the remote server, and returns to the command interpreter. Any defined macros are erased.
cr	Toggles carriage return stripping during ascii-type file retrieval. Records are denoted by a carriage return/linefeed sequence during ascii-type file transfer. When **cr** is on (the default), carriage returns are stripped from this sequence to conform with the AIX single line feed record delimiter. Records on non-AIX System/370 remote systems may contain single linefeeds. When an ascii-type transfer is made, these linefeeds may be distinguished from a record delimiter only when **cr** is off.
delete *remote-file*	Deletes the file *remote-file* on the remote machine.
debug [*debug-value*]	Toggles the debugging mode. If an optional *debug-value* is specified, it is used to set the debugging level. When debugging is on, **ftp** prints each command sent to the remote machine, preceded by the string (-->). If the *debug-value* is 0 debugging is off. If the *debug-value* is a positive integer, debugging is on.

dir [*remote-directory*] [*local-file*]
Prints a listing of the directory contents in the directory, *remote-directory* and, optionally, placing the output in *local-file*. If no directory is specified, the current working directory on the remote machine is used. If no local file is specified or *local-file* is **(-)**, output is displayed on the terminal.

disconnect
Is a synonym for **close**.

form format
Sets the file transfer **form** to **format**. The only supported format is non-print. Only for use with ascii and ebcdic.

get *remote-file* [*local-file*]
Retrieves the *remote-file* and stores it on the local machine. If the local file name is not specified, it is given the same name it has on the remote machine, subject to alteration by the current **case**, **ntrans**, and **nmap** settings. The current settings for **type, form, mode, and structure** are used while transferring the file.

glob
Toggles file name expansion for **mdelete**, **mget** and **mput**. If globbing is turned off with **glob**, the file name arguments are taken literally and not expanded. Globbing for **mput** is done as in **csh**. For **mdelete** and **mget**, each remote file name is expanded separately on the remote machine and the lists are not merged. Expansion of a directory name is likely to be different from expansion of the name of an ordinary file. The exact result depends on the foreign operating system and **ftp** server, and can be previewed by entering

```
mls remotefilename -
```

Note: **mget** and **mput** are not meant to transfer entire directory subtrees of files. This can be done by transferring a **tar**, refer to *AIX Operating System Commands Reference*, archive of the subtree (in binary mode).

hash
Toggles hash-sign (#) printing for each data block transferred. The size of a data block is 4096 bytes.

help [*command*]
Prints an informative message about the meaning of *command*. If no argument is given, **ftp** prints a list of the known commands.

lcd [*directory*]
Changes the working directory on the local machine. If no directory is specified, the user's home directory is used.

ls [*remote-directory* [*local-file*]
Prints an abbreviated listing of the contents of a directory on the remote machine. If remote-directory is left unspecified, the current working directory is used on the remote machine. If no local file is specified, or if *local-file* is **-**, the output is sent to the terminal.

macdef *macro-name*	Defines a macro. Subsequent lines are stored as the macro *macro-name*. A null line (consecutive new line characters in a file or carriage returns from the terminal) terminates macro input mode. There is a limit of 16 macros and 4096 total characters in all defined macros. Macros remain defined until a **close** command is executed. The macro processor interprets **$** and \ as special characters. A **$** followed by a number (or numbers) is replaced by the corresponding argument on the macro invocation command line. A **$** followed by an **i** signals the macro processor that the executing macro is to be looped. On the first pass, **$i** is replaced by the first argument on the macro invocation command line. On the second pass, it is replaced by the second argument, and so on. A \ followed by any character is replaced by that character. Use the \ to prevent special treatment of the **$**.
mdelete [*remote-files*]	Deletes the *remote-files* on the remote machine.
mdir *remote-files local-file*	Like *dir*, except multiple remote files may be specified. If interactive prompting is on, **ftp** prompts the user to verify that the last argument is indeed the target local file for receiving *mdir* output.
mget *remote-files*	Expands the *remote-files* on the remote machine and supplies a **get** for each file name thus produced. See **glob** for details on the file name expansion. Resulting file names are then processed according to **case, ntrans,** and **nmap** settings. Files are transferred into the local working directory, which can be changed with **lcd directory**. New local directories can be created with **! mkdir** *directory*.
mkdir *directory-name*	Makes a directory on the remote machine.
mls *remote-files local-file*	Similar to **ls**, except multiple remote files may be specified. If interactive prompting is on, **ftp** prompts the user to verify that the last argument is indeed the target local file for receiving **mls** output.
mode [*mode-name*]	Sets the file transfer **mode** to *mode-name*. The only supported mode is 'stream' mode.
mput *local-files*	Expands wild cards in the list of local files given as arguments and supplies a **put** for each file in the resulting list. See **glob** for details of file name expansion. Resulting file names are then processed according to **ntrans** and **nmap** settings.
nmap [*inpattern* [*outpattern*]	Sets or unsets the file name mapping mechanism. If no arguments are specified, the file name mapping mechanism is unset. If arguments are specified, remote file names are mapped during **mput** commands and **put** commands issued without a specified remote target file name. If arguments are specified, local file names are mapped during **mget** commands and **get** commands issued without a specified local target file name. This command is useful when connecting to a non-AIX System/370 remote computer with different file naming conventions or practices.

The mapping follows the pattern set by *inpattern* and *outpattern*. *Inpattern* is a template for incoming file names (which may have already been processed according to the **ntrans** and **case** settings). Variable templating is accomplished by including the sequences *$1*, *$2*, ..., *$9* in *inpattern*. Use \ to prevent this special treatment of the **$** character. All other characters are treated literally, and are used to determine the **nmap** *inpattern* variable values. For example, given *inpattern* *$1.$2* and the remote file name **mydata.data**, *$1* has the value **mydata**, and *$2* has the value **data**.

The *outpattern* determines the resulting mapped file name. The sequences *$1*, *$2*, ..., *$9* are replaced by any value resulting from the *inpattern* template. The sequence *$0* is replaced by the original file name. Additionally, the sequence [*seq1*,*seq2*] is replaced by *seq1* if *seq1* is not a null string. Otherwise, it is replaced by *seq2*. For example, the command **nmap** *$1.$2.$3* [*$1*,*$2*]. [*$2*,*file*] would yield the output file name **myfile.data** for input file names **myfile.data** and **myfile.data.old**, **myfile.file** for the input file name **myfile**, and **myfile.myfile** for the input file name **.myfile**. Spaces may be included in *outpattern*, as in the example:

```
nmap $1 |sed "s/  *$//" > $1 .
```

Use the \ character to prevent special treatment of the **$**, [,], and , characters.

ntrans [*inchars*[*outchars*]
: Sets or unsets the file name character translation mechanism. If no arguments are specified, the file name character translation mechanism is unset. If arguments are specified, characters in remote file names are translated during **mput** commands and **put** commands issued without a specified remote target file name. If arguments are specified, characters in local file names are translated during **mget** commands and **get** commands issued without a specified local target file name. This command is useful when connecting to a non-AIX System/370 remote computer with different file naming conventions or practices. Characters in a file name matching a character in **inchars** are replaced with the corresponding character in **outchars**. If the character's position in **inchars** is longer than the length of **outchars**, the character is deleted from the file name.

open *host* [*port*]
: Establishes a connection to the specified *host* FTP server. An optional port number may be supplied, in which case, **ftp** attempts to contact an FTP server at that port. If the **auto login** option is on (default), **ftp** also attempts to automatically log the user in to the FTP server.

prompt
: Toggles interactive prompting. Interactive prompting occurs during multiple file transfers to allow the user to selectively retrieve or store files. If prompting is turned off (default is on), any **mget** or **mput** transfers all files, and any **mdelete** deletes all files.

proxy *ftp-command*
: Executes an **ftp** command on a secondary control connection. This command allows simultaneous connection to two remote **ftp** servers for transferring files between the two servers. The first **proxy** command should be an **open**, to establish the secondary control connection. Enter the command **proxy ?** to see other **ftp** commands executable on the secondary connection.

  The following commands behave differently when prefaced by **proxy** :

open	Does not define new macros during the auto login process.
close	Does not erase existing macro definitions.
get, mget	Transfers files from the host on the primary control connection to the host on the secondary control connection.

put, mput, append
: Transfers files from the host on the secondary control connection to the host on the primary control connection. Third party file transfers depend upon support of the FTP protocol **PASV** (passive) command by the server on the secondary control connection.

put *local-file* [*remote-file*]
: Stores a local file on the remote machine. If *remote-file* is left unspecified, the local file name is used after processing according to any **ntrans** or **nmap** settings in naming the remote file. File transfer uses the current settings for **type, format, mode,** and **structure**.

pwd
: Prints the name of the current working directory on the remote machine.

quit
: Exit **ftp**.

quote *arg1 arg2* **...**
: The arguments specified are sent, verbatim, to the remote FTP server.

recv *remote-file* [*local-file*]
: Is a synonym for **get**.

remotehelp [*command-name*]
: Requests help from the remote **ftp** server. If a *command-name* is specified, it is also supplied to the server.

rename [*from*] [*to*]
: Renames the file **from** on the remote machine, to the file **to**.

reset
: Clears reply queue. This command re-synchronizes command/reply sequencing with the remote **ftp** server. Resynchronization may be necessary following a violation of the FTP protocol by the remote server.

rmdir *directory-name*
: Deletes a directory on the remote machine.

runique
: Toggles storing of files on the local system with unique file names. If a file already exists with a name equal to the target local file name for a **get** or **mget** command, a *.1* is appended to the name. If the resulting name matches another existing file, a *.2* is appended to the original name. If this process continues up to *.99*, an error message is printed, and the

	transfer does not take place. The generated unique file name is reported. **Note:** **runique** does not affect local files generated from a shell command (see below). The default value is off.
send *local-file* [*remote-file*]	Is a synonym for **put**.
sendport	Toggles the use of **port** commands. By default, **ftp** sends a **port** command before each data transfer.
status	Shows the current status of **ftp** .
struct [*struct-name*]	Sets the file transfer **structure** to *struct-name.* The only supported structure is file.
sunique	Toggles the storing of files on remote machine under unique file names. Remote FTP server must support FTP protocol **STOU** command for successful completion. The remote server reports unique name. Default value is off.
tenex	Sets the file transfer type to that needed to talk to TENEX machines.
trace	Toggles packet tracing.
type [*type-name*]	Sets the file transfer **type** to *type-name.* If no type is specified, the current type is printed. The default type is network ascii; the other types available are image and ebcdic.
user *user-name* [*password*] [*account*]	Identifies the user to the remote FTP server. If the password is not specified and the server requires it, **ftp** prompts the user for it (after disabling local echo). If an account field is not specified and the FTP server requires it, the user is prompted for it. If an account field is specified, an account command is relayed to the remote server after the login sequence is completed if the remote server did not require it for logging in. Unless **ftp** is invoked with auto login disabled, this process is done automatically on initial connection to the FTP server.
verbose	Toggles verbose mode.
? [*command*]	Is a synonym for **help**.

Command arguments which have embedded spaces may be quoted with quote (") marks.

Aborting a File Transfer

To abort a file transfer, use the terminal **Ctrl-C** (INTERRUPT) key. Sending transfers are immediately halted. Receiving transfers are halted by sending a FTP protocol **ABOR** command to the remote server, and discarding any further data received. The speed at which this is accomplished depends upon the remote server's support for **ABOR** processing. If the remote server does not support the **ABOR** command, an `ftp>` prompt does not appear until the remote server has completed sending the requested file.

The terminal **INTERRUPT** key sequence is ignored when **ftp** has completed any local processing and is awaiting a reply from the remote server. A long delay in this mode may result from the **ABOR** processing described above, or from unexpected behavior by the remote server, including violations of the FTP protocol. If the delay results from unexpected remote server behavior, the local **ftp** program must be killed by hand.

File Naming Conventions

Files specified as arguments to **ftp** commands are processed according to the following rules.

- If the file name **-** is specified, the **stdin** (for reading) or **stdout** (for writing) is used.
- If the first character of the file name is **|**, the remainder of the argument is interpreted as a shell command. **ftp** then forks a shell, using **popen**, (see *AIX Operating System Technical Reference*) with the argument supplied, and reads (writes) from the **stdout** (stdin). If the shell command includes spaces, the argument must be quoted; such as "**| ls -lt**". A particularly useful example of this mechanism is (dir | more).
- Failing the above checks, if globbing is enabled local file names are expanded according to the rules used in the C shell. If the **ftp** command expects a single local file (such as **put**), only the first file name generated by the globbing operation is used.
- For **mget** commands and **get** commands with unspecified local file names, the local file name is the remote file name, which may be altered by a **case, ntrans**, or **nmap** setting. The resulting file name may then be altered if **runique** is on.
- For **mput** commands and **put** commands with unspecified remote file names, the remote file name is the local file name, which may be altered by a **ntrans** or **nmap** setting. The resulting file name may then be altered by the remote server if **sunique** is on.

Parameters

The FTP protocol specifies many parameters which may affect a file transfer. The **type** may be one of ascii, binary, EBCDIC, and local byte size. FTP supports the ascii and image types of file transfer, plus local byte size 8 for tenex mode transfers.

FTP supports only the default values for the remaining file transfer parameters: *mode*, *form*, and *struct*.

The .netrc File

The **.netrc** file contains login and initialization information used by the auto login process. It resides in the user's home directory. The following tokens are recognized. They may be separated by spaces, tabs, or new lines:

machine *name*	Identifies a remote machine name. The auto login process searches the **.netrc** file for a **machine** token that matches the remote machine specified on the **ftp** command line or as an **open** command argument. Once a match is made, the subsequent **.netrc** tokens are processed, stopping when the end of file is reached or another **machine** token is encountered.
login *name*	Identifies a user on the remote machine. If this token is present, the auto login process initiates a login using the specified name.
password *string*	Supplies a password. If this token is present, the auto login process supplies the specified string if the remote server requires a password as part of the login process.

Note: If this token is present in the **.netrc** file, **ftp** stops the auto login process if the **.netrc** is readable by anyone besides the user.

account *string* Supplies an additional account password. If this token is present, the auto login process supplies the specified string if the remote server requires an additional account password, or the auto login process initiates an ACCT command if it does not.

macdef *name* Defines a macro. This token functions like the **ftp macdef** command functions. A macro is defined with the specified name. Its contents begin with the next **.netrc** line and continues until a null line (consecutive new-line characters) is encountered. If a macro named **init** is defined, it is automatically executed as the last step in the auto login process.

Examples

- The user `smith` is logged in on `host1`. This example shows how `smith` can log in on the foreign host `host2`, check the current working directory on `host2`, list the contents of the working directory, and then transfer the file `testfile` to `tmp.testfile`:

```
$ ftp host2
connected to host2.
220 host2 FTP Server ready.
Name (host2:smith):
331 Passwd required for smith
Password:
230 User smith logged in

ftp> binary
200 Type set to I

ftp> pwd
550 "/u/smith" is current directory
ftp> ls
200 PORT command successful.
150 Opening data connection for /bin/ls (192.9.200.1,1026) (0 bytes)
pri
testfile
226 Transfer complete.
26 bytes received in 0.12 seconds (0.22 Kbytes/s)

ftp> get testfile tmp.testfile
200 PORT command successful.
150 Opening data connection for testfile (192.9.200.1,1029) (1201 bytes)
226 Transfer complete.

local:tmp.testfile remote:testfile
1201 bytes received in 0 seconds (1.2 Kbytes/s)

ftp> quit
221 Goodbye.
$ _
```

- The user smith is logged in on host1. This example shows how smith can log in as the user smith on the foreign host host2.:

```
$ ftp host2
connected to host2
220 host2 FTP Server ready.
Name (host2:smith):
331 Passwd required for smith
Password:
230 User smith logged in
ftp>
```

- The user fred makes a typing error and tries to log in as the user miths:

```
$ ftp host2
connected to host2
220 test FTP Server ready.
Name (test:fred): miths
530 User miths access denied
ftp> user smith
331 Passwd required for smith
Password:
230 User smith logged in
ftp>
```

- The user fred issues the **ftp** command without specifying a host name:

```
$ ftp
ftp> open host1
connected to host1
220 host1 FTP Server ready.
Name (host1:fred):
331 Passwd required for fred
Password:
230 User fred logged in
ftp>
```

Files

.netrc	Is the auto login file in the current or home directory.

Related Information

In this book: "ftpd" on page 4-3

host

Purpose

Resolves a host name into an Internet address.

Syntax

```
host — hostname —|
```

Description

The **host** command determines the Internet address of the specified host. If **host** finds the address, **host** displays it in both dotted decimal and aliased name formats.

If the local host is using the DOMAIN protocol, the local or remote nameserver database is queried before searching the local **/etc/hosts** file.

Parameters

hostname
: Can be either a unique host name or a well-known host name like **nameserver**, **printserver**, or **timeserver**.

Examples

To display the address of host2:

```
$ host host2
host2 is 192.100.13.5  Aliases:   host2.abc, host2.xyz
$ _
```

Files

/etc/hosts	Defines the name-to-address map table.
/etc/resolv.conf	Defines the nameserver.

Related Information

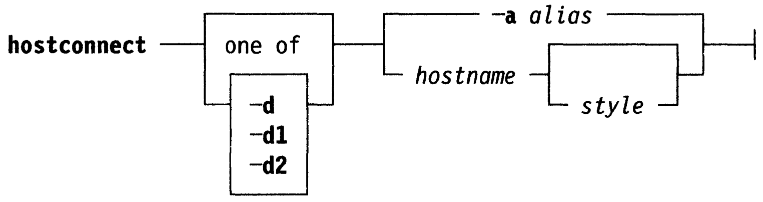
In this book: "named" on page 4-8.

hostconnect

Purpose

Creates a connection between an AIX system and an IBM host

Syntax



Description

The **hostconnect** command is an AIX command which creates a connection to *hostname*, an IBM VM host. You can then run CMS commands on *hostname* by using **onhost**. It can do an automatic login using the **alias** option. **hostconnect** requires a binary TELNET session with the host and identifies itself as an IBM-3278-2. After a successful login, **hostconnect** goes into the background to listen for **onhost** commands or full-screen emulation. This means that you can now use all the usual AIX commands and, in addition, you can use the **onhost** command to run CMS commands in the IBM VM host you specified with hostname.

When you log out of the host system, **hostconnect** terminates. You can also terminate **hostconnect** by using "**kill** *n*," where *n* is its process id as reported by the **ps** command. Do *not* use "**kill -9**"! If you do, **hostconnect** does not get a chance to close the host connection. Your host userid will remain active and, under some circumstances, you may not be able to login again.

You can follow the progress of **hostconnect** by decoding the character string which it produces. The characters L, P, I, E, and S show the login, password, initialize, execution preparation, and sequence complete stages. Also present are equal signs to indicate minor state changes. If the string does not end in S, try repeating the **hostconnect** command.

Flags

-a *alias*	This option selects a hostname, userid, and password entry stored in the file named **onhost.***alias*. This file must reside in the current or home directory and, because the file contains passwords, **hostconnect** sets 600 permission so that only the owner may read and write this file.
-d	The **-d** option turns on debugging. **-d** sends debug output to the terminal. The **-d1** option sends debug output to the file **hostcon.debug**. The **-d2** option is the same as **-d1** except that it writes more information.

onhost.alias

The first line of the **alias** file specifies your local userid and password to be used by **onhost** for FTP file transfer from the host. A full-screen 3270 terminal emulator is also specified on this line. **tn** and **tn3270** are two such emulators.

Successive lines, one per alias, contain the following tokens.

alias This field labels the entry. It is used to create a file named **hostcon.alias** which contains the socket name used by **onhost**. The file **hostcon.noalias** is created if the **alias** option is not used.

hostname This is the host system name specified as an Internet domain name or address. When you are connecting to VM/CMS and have specified **hostname** as LDSF, **hostconnect** makes a connection using the VM Logical Device Support facility. Otherwise, **hostconnect** uses the standard TCP/IP socket interface.

style The style of the host system is defined as follows: CMS is of style 2 and unknown hosts are of style 6. If no style is specified, it defaults to style 0 which defaults to CMS.

In addition, if 1 is added to make the style odd, a connection is made to the host system but the automatic login is suspended. This feature is required to allow **hostconnect** to work with arbitrary host systems. The automatic login is nice, but it only works in limited cases. An AIX shell script driving **onhost** can be used to automatically connect to most hosts.

Automatic login is not attempted for unknown style hosts. When **hostconnect** is run and automatic login is suspended, it makes a connection which must be completed by the use of **onhost**. After completing the login, execute the host command, **ONHOSTLD**, to initialize the host environment for subsequent use by **onhost**.

Subsequent tokens are passed to the host system one at a time to complete an automatic login to the host system. Additional tokens are requested from the user by **hostconnect** if they are required.

Here is a sample **onhost.alias** file.

```
AIXuserid AIXpasswd tn
pvm paloalto 1
oak 129.33.192.128 2 root fungus
```

A template for **onhost.alias** file exists in the **/usr/lib/onhost** directory.

Notes

Before using **hostconnect** for the first time, make sure that you have a valid userid and password for the host system you want to use. In addition, some programs used by **hostconnect** and **onhost** should be present on the host. See Appendix B, "Onhost Installation" for instructions on how to install these programs.

The **hostconnect** command on the VM host, expects a VM READ in the standard status area of the VM host's virtual screen whenever a command ends or a program needs input on the VM host. If you change the setting of autoread or change the VM READ status area on the VM host, then **hostconnect** or **onhost** may not work correctly.

Related Information

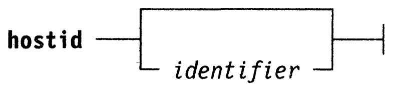
In this book: "onhost" on page 3-37
Appendix B, "Onhost Installation" on page B-1

hostid

Purpose

Sets or prints identifier of current host system.

Syntax



Description

The **hostid** command prints the identifier of the current host in hexadecimal. This numeric value is expected to be unique across all hosts and is commonly set to the host's Internet address. The superuser can set the **hostid** by giving a hexadecimal argument or the hostname. This is usually done in a startup script, for example, `/etc/rc.` If the host is on two networks, set the **hostid** to one of the ID addresses.

Examples

The **hostid** command returns the network address of the current host as a 32-bit hexadecimal number.

```
$ /etc/hostid
0x491464c0
```

Related Information

gethostid, **sethostid**, refer to *AIX Operating System Technical Reference*

hostname

Purpose

Sets or prints name of current host system.

Syntax

```
                      ┌──────┐
              ┌───────┤      ├───────┐
hostname ─────┤       └─ -v ─┘       ├───┤
              └─── name_of_host ─────┘
```

Description

The **hostname** command displays the name of the current host. If you have superuser authority, you can change the host name and address by supplying the *name-of-host* parameter. This is usually done in the startup script **/etc/rc**. If no parameter is supplied, the **hostname** command prints the name. The **hostname** command does not modify the **/etc/hosts** file.

A gateway host (a host that connects independent networks) has multiple interfaces, each with a different name and address. The **hostname** command sets the primary name of the gateway host. References to gateway hosts can be by primary or secondary name.

Flags

-v Verbose output that includes node and internet address.

Parameters

name_of_host
Must be supplied in the **/etc/hosts** file to set the host name and the Internet address of this host. The *name_of_host* is a character string of up to 24 characters. Superuser authority is required.

Examples

- To display the name and address of the local host :

```
$ hostname
host1
$ -
```

- To change the name of the local host :

```
# hostname west
# -
```

Files

/etc/hosts

Related Information

gethostname, **sethostname**, refer to *AIX Operating System Technical Reference*

ifconfig

Purpose

Configures network interface parameters.

Syntax

```
ifconfig — interface ─┬────────────────┬─┬─────────┬─┬────────────┬─┤
                      └ address_family ┘ └ address ┘ └ parameters ┘
```

Description

The **ifconfig** command is used to assign an address to a network interface or configure network interface parameters. The **ifconfig** command must be used at boot time to define the network address of each interface present on a machine. It may also be used at a later time to redefine an interface's address or other operating parameters. The *interface* parameter is a string of the form **name unit**, such as **tk0**.

To configure a point-to-point interface (e.g., vctc), use the following format, where dev# is the point-to-point interface name followed by a unit number, such as vctc0: ifconfig dev# localhost remotehost. For the DARPA Internet family, the *address* is either a host name present in the host name data base, **hosts**, or a DARPA Internet address expressed in the Internet standard dot notation.

The **ifconfig** command serves a similar purpose to the **netconfig** command.

The address family default is DARPA Internet. Currently, AIX does not support other address families.

Parameters

The following parameters may be set with **ifconfig**:

up
: Marks an interface **up**. This may be used to enable an interface after an **ifconfig down**. It happens automatically when setting the first address on an interface. If the interface is present when previously marked down, the hardware is re-initialized.

down
: Marks an interface **down**. When an interface is marked down, the system does not attempt to transmit messages through that interface. If possible, the interface is reset to disable reception as well. This action does not automatically disable routes using the interface.

arp
: Enables the use of the Address Resolution Protocol in mapping between network level addresses and link level addresses (default). **Arp** is currently implemented for mapping between Internet addresses and Ethernet addresses or IBM token-ring addresses.

-arp
: Disables the use of the Address Resolution Protocol.

metric *n*
: Sets the routing metric of the interface to *n*, (not 0). The routing metric is used by the routing protocol ("route" on page 3-56). Higher metrics have the effect of making a route less favorable. Metrics are counted as additional hops to the destination network or host.

debug
: Enables driver dependent debugging code. Usually, this turns on extra console error logging.

-debug
: Disables driver dependent debugging code.

netmask *mask*
: (net only) Specifies how much of the address to reserve for subdividing networks into sub-networks. The *mask* includes the network part of the local address and the subnet part, which is taken from the host field of the address. The mask can be specified as a single hexadecimal number with a leading 0x, with a dot notation Internet address or with a pseudo-network name listed in the network table "networks" on page 5-13. The mask contains 1's for the bit positions in the 32-bit address which are to be used for the network and subnet parts, and 0's for the host part. The mask should contain at least the standard network portion, and the subnet field should be contiguous with the network portion.

dstaddr
: Specifies the address of the correspondent on the other end of a point to point link.

broadcast
: (In net only) Specifies the address to use to represent broadcasts to the network. The default broadcast address is the address with a host part of all 1's.

ieee
: Enables the **ieee** 802 protocol on an Ethernet device.

-ieee
: Disables the **ieee** 802 protocol on an Ethernet device.

trailers
: Request the use of a trailer link level encapsulation when sending (default). If a network interface supports *trailers*, the system will, when possible, encapsulate outgoing messages in a manner which minimizes the number of memory-to-memory copy operations performed by the receiver. On networks that support the Address Resolution Protocol ("arp" on page 3-2; currently, only 10M bytes Ethernet), this flag indicates that the system should request that only systems use *trailers* when sending to this host. Similarly, *trailer* encapsulations will be sent to other hosts that have made such requests currently used by Internet protocols only.

-trailers
: Disables the use of a *trailer* link level encapsulation.

The **ifconfig** command displays the current configuration for a network interface when no optional parameters are supplied. If a protocol family is specified, **ifconfig** reports only the details specific to that protocol family.

Only the superuser may modify the configuration of a network interface.

Note: **ifconfig** *interface address* sets the interface **up**.

Examples

Configure the interface net0 on the host called host2.

```
# /etc/ifconfig net0 host2
#
```

Messages

Messages are provided indicating the specified interface does not exist, the requested address is unknown, or the user is not privileged and tried to alter an interface's configuration.

Related Information

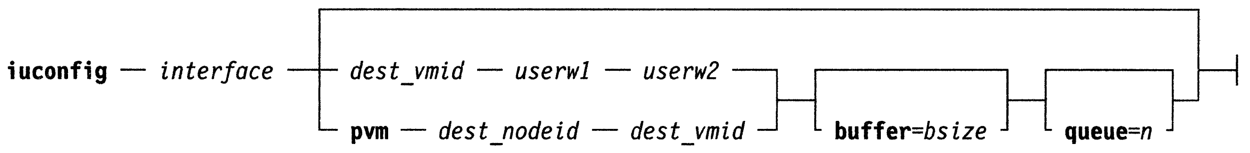
In this book:
"netstat" on page 3-32
"rc.tcpip, rc.tcpip.local" on page 5-15
"netconfig" on page 3-30

iuconfig

Purpose

Configure network interface parameters for iucv connections.

Syntax



Description

The **iuconfig** command is used to configure connections between AIX/370 machines. The connection can be made either through a direct Inter-User Communication Vehicle (iucv) connection, or through the VM/Pass-Through Facility (pvm).

When using a direct iucv connection, *vmid* must be the VM user-ID of the destination AIX/370 guest. The two user data words may be any string as long as they are same and are the same as the user data words specified on the remote AIX/370 local host. The user data words are used to distinguish the path. Each of *vmid, userwl,* and *userw2* may be up to 8 characters in length.

When connecting through pvm facility, you must supply the node-id of the destination system, and the VM user-ID of the destination AIX/370 guest.

Since pvm supports 32768 bytes only when the target pvm node is running VM/SP Release 4 or above and PVM 4.2004 PLC 4445 or above, the default maximum transmission unit for network iucv interface is 4096 bytes. This default can be overridden by the buffer = bsize option described below.

The following parameters may be set by iuconfig:

buffer = bsize	Set the maximum transmission unit for the interface to bsize kilobytes. The default value for *n* is 2 (double buffering).
queue = *n*	Use single (queue = 1), double (queue = 2), or triple (queue = 3) buffering. The default value for *n* is 2 (double buffering).

The **iuconfig** command, when used alone with an interface name, displays the status of the iucv connection.

The **ifconfig** or **netconfig** command must be used to assign an IP address to the network iucv interface after the **iuconfig** command is used to make an iucv connection.

The **iuconfig** command may be placed in the script **/local/local.init.dir/Singl2multi** to establish an iucv connection automatically when the system boots.

Diagnostics

Messages indicating that the specified interface does not exist, the requested address is unknown, or that a non-privileged user tried to alter an interface's configuration are provided.

Related Information

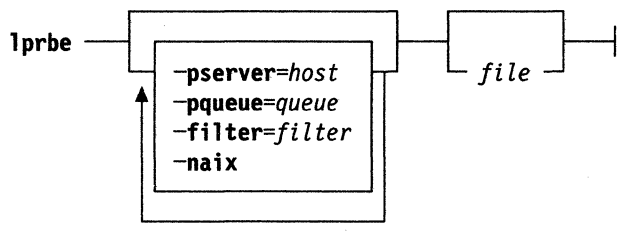
In this book:
"ifconfig" on page 3-23
"netconfig" on page 3-30

lprbe

Purpose

Sends a file to a print server.

Syntax



Description

The **lprbe** command is a backend program that sends print requests to a remote print server (the **lpd** program and printer on a foreign host). The **lprbe** command is normally called by the **qdaemon** command after you have queued a file with the **print** command. The flags and parameters that you enter with **print** are passed to **lprbe**, and it is **lprbe** that determines where and how the remote print job is done. The **lprbe** command supports the AIX **piobe** and **print** flags, as well as a set of filters that may exist on non-AIX systems. For more information on **piobe** and **print**, see *AIX PS/2 Operating System Command Reference.*

If a client requests a filter that is not listed in the print server's **/etc/filters** file, the print daemon, **lpd**, send the file directly to the printer without processing (just as if the file were sent by the **cat** command.

Flags

-pserver = *host*
Specifies which host is to receive the print request. If **-pserver** = is not specified, **lprbe** sends the print request to the host specified by a **printserver** entry in the **/etc/hosts** file or to the remote print server identified in the **/etc/qconfig** file.

-pqueue = *queuename*
Specifies a particular remote printing queue that is to receive the print request.

-filter = *filter*
Specifies a *filter* that is either a user-defined program (with or without its own flags) that pipes its output to **print**, or one of the following flags (which indicate that the files to be printed are not standard text files). If *filter* contains embedded blanks, it must be enclosed in double quotes (" "). These flags generate a control file that is compatible with non-AIX systems; if they are sent to an AIX system, they are ignored.

-c	Handles files that contain data produced by **cifplot.**
-d	Handles files that contain Tex data.

-f	Uses a filter that interprets the first character of each line as a standard FORTRAN carriage control character.
-g	Handles files that contain standard plot data as produced by the **plot** routines.
-l	Uses a filter that allows control characters to be printed and suppresses page breaks.
-n	Handles files that contain **ditroff** data.
-p	Uses **pr** to format the files. The results are equivalent to those obtained with the **print** command.
-r	Handles files that contain FORTRAN carriage control characters.
-t	Handles files that contain **troff** data.
-v	Handles files that contain a raster image for devices like the Benson Varian.

-naix	Generates a print control file for a non-AIX system.
file	Names one or more files to be printed.

Examples

1. To print the file `testcase` on a foreign AIX host (when `rp0` is the name of the local host queue that handles outbound print requests):

   ```
   $ print rp0 testcase
   $
   ```

2. To print on a foreign host (`host1`) other than the default **printserver**:

   ```
   $ print rp0 -pserver=host1 testcase
   $
   ```

3. To select a specific print queue (`lp1`) on a foreign host:

   ```
   $ print rp0 -pqueue=lp1 testcase
   $
   ```

4. To perform preprocessing on the foreign host (by the **pr** command) before printing:

   ```
   $ print rp0 -filter=00pr -w60 i500 testcase
   $
   ```

   This filter generates the following command on the foreign host:

   ```
   pr -w60 -i5 testcase ] print
   ```

5. To print on a non-AIX foreign host (that is configured as the default **printserver**):

   ```
   $ print rp0 -naix testcase
   $
   ```

Files

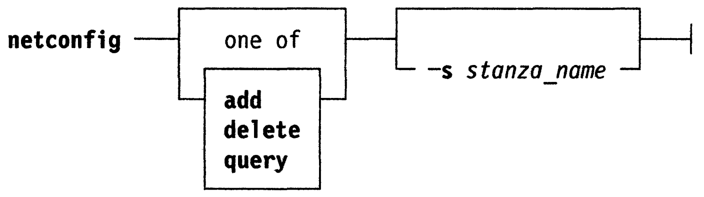
/etc/qconfig	Defines the queueing system configuration.

netconfig

Purpose

Configures network interface parameters.

Syntax



Description

The **netconfig** command is used to assign an address to a network interface or configure network interface parameters. This command may be used at boot time to establish the network address of each interface present on the machines. Superuser authority is required to use this command to add or delete. Any user can use this command to query. The network interface parameters are in the file **/etc/net**. The **netconfig** command serves a similar purpose to the **ifconfig** command.

Flags

The **netconfig** command option is:

-s *stanza_name*	Indicates that only the specified stanza, *stanza_name*, as defined in the file **/etc/net**, is to be processed.

Parameters

add
Adds the interface specified by **-s** *stanza_name*. If **-s** *stanza_name* is not specified, **netconfig** processes all stanzas in the **/etc/net** file and adds those interfaces.

delete
Deletes the interface specified by **-s** *stanza_name*. If **-s** *stanza_name* is not specified, **netconfig** processes all stanzas in the **/etc/net** file and deletes those interfaces.

query
Returns the state of all interfaces defined by stanzas in **/etc/net**. If **-s** *stanza_name* is specified, **netconfig** returns the state of the interface specified by **-s** *stanza_name*.

Examples

- To add a specified interface, `net0` that is defined in **/etc/net**:

  ```
  # netconfig add -s net0
  # _
  ```

- To delete all interfaces defined in **/etc/net**:

  ```
  # netconfig delete
  # _
  ```

- To query all interfaces that are defined in **/etc/net**:

```
$ netconfig query
```

net0: Internet Address: 130.200.8.10 inetlen: 1492 hardware type: local area net subnet mask: ffff0000 remote inetlen: 576 Metric: 3

```
This interface is currently available
```

tk0: Internet Address: 195.223.5.10 inetlen: 2010 hardware type: local area net subnet mask: ffffff00 remote inetlen: 576 broadcast: all rings metric: 6

```
This interface is currently available
```

lo0: Internet Address: 127.0.0.1 inetlen: 1536 hardware type: local area net

```
subnet mask:  ff000000   remote inetlen:  576
```

```
This interface is currently available
```

Files

/etc/net	Network Interface definition for TCP.
/etc/system	System device configurations.

Related Information

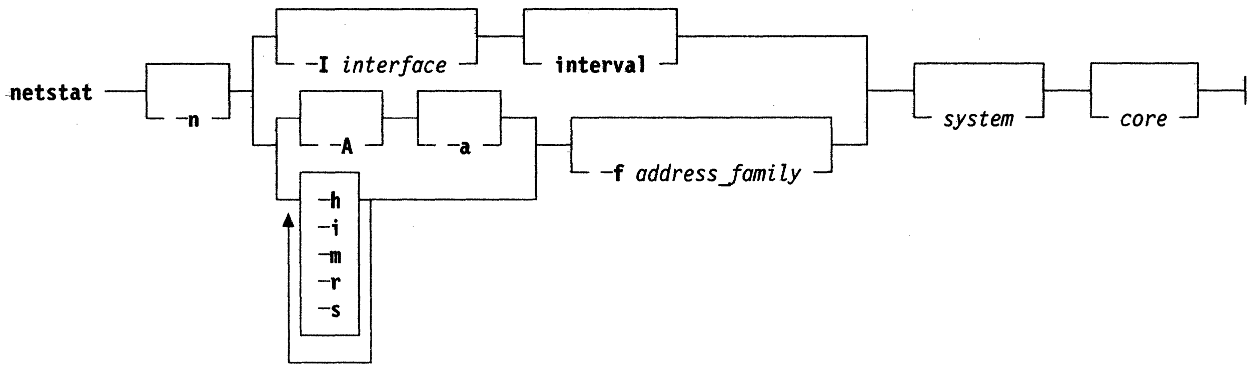
In this book:
"ifconfig" on page 3-23
".netrc" on page 5-12

netstat

Purpose

Shows network status.

Syntax



Description

The **netstat** command symbolically displays the contents of various network-related data structures. There are a number of output formats, depending on the options for the information presented. With the **-a** option, the command displays a list of active sockets for each protocol. With the **-h**, **-i**, **-m**, **-r**, and **-s** options, the command presents the contents of one of the other network data structures according to the option selected. If **-I** interface options is used with an **interval** specified, **netstat** continuously displays the information regarding packet traffic on the configured network interfaces.

Flags

The **netstat** command options are:

-A	Shows the address of any protocol control blocks associated with sockets used for debugging, with the default display.
-a	Shows the state of all sockets with the default display. Normally sockets used by server processes are not shown. Used only with the **-A** option.
-h	Prints the host table associated with the Arpanet IMP if one is attached to the system and support for it has been added to the AIX kernel.
-i	Shows the state of interfaces which have been auto-configured (interfaces statically configured into a system, but not located at boot time are not shown).
-m	Shows statistics recorded by the memory management routines (the network manages a private pool of memory buffers).
-n	Shows network addresses as numbers (normally **netstat** interprets addresses and attempts to display them symbolically). This option may be used with any of the display formats.

-r Shows the routing tables. When **-s** is also present, shows routing statistics instead.

-s Shows per-protocol statistics.

-t Used only with the **-i** flag to print interface timer information. You can specify a watchdog function for each network interface that will run when the watchdog timer is decremented to zero. The watchdog timer is decremented once per second.
Only looks at UNIX-domain sockets.

-f *address_family*
Limits statistics or address control block reports to those of the specified address family. The following address families are recognized:

inet for **AF_INET**

ns for **AF_NS**

UNIX for **AF_UNIX**

-I *interface*
Shows information only about this interface. Used with an **interval** as described below.

The arguments, *system* and *core*, allow substitutes for the defaults **/unix** and **/dev/kmem**.

The default display, for active sockets, shows the local and remote addresses, sends and receives queue sizes (in bytes), protocol, and the internal state of the protocol. Address formats are of the form **host.port** or **network.port** if a socket's address specifies a network but no specific host address. When known, the host and network addresses are displayed symbolically according to the data bases **/etc/hosts** and **/etc/networks**, respectively. If a symbolic name for an address is unknown, or if the **-n** option is specified, the address is printed numerically, according to the address family. For more information regarding the Internet dot format, refer to **inet** in the *AIX Operating System Technical Reference*. Unspecified or wildcard addresses and ports appear as *.

The interface display provides a table of cumulative statistics regarding packets transferred, errors, and collisions. The network addresses of the interface and the maximum transmission unit **mtu** are also displayed.

The routing table display indicates the available routes and their status. Each route consists of a destination host or network and a gateway to use in forwarding packets. The flags field shows the state of the route (**U** if **UP**), whether the route is to a gateway (**G**), and whether the route was created dynamically by a redirect (**D**). Direct routes are created for each interface attached to the local host. The gateway field for such entries shows the address of the outgoing interface. The **refcnt** field gives the current number of active uses of the route. Connection oriented protocols normally hold on to a single route for the duration of a connection while connectionless protocols obtain a route while sending to the same destination. The use field provides a count of the number of packets sent using that route. The interface entry indicates the network interface utilized for the route.

When **netstat** is invoked with an interval argument, it displays a running count of statistics related to network interfaces. This display consists of a column for the primary interface (the first interface found during autoconfiguration) and a column summarizing information for all interfaces. The primary interface may be replaced with another interface with the **-I** option. The first line of each screen of information contains a summary since the system was last rebooted. Subsequent lines of output show values accumulated over the preceding interval.

Examples

```
$ netstat
Active Internet connections
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp        0      0  host1.1052              host1.ftp              ESTABLISHED
tcp        0      0  host1.1051              host1.telnet           ESTABLISHED
tcp        0      0  host1.1050              host1.telnet           ESTABLISHED

$ netstat -A
Active Internet connections
PCB      Proto Recv-Q Send-Q  Local Address      Foreign Address    (state)
f1043b0c tcp        0      0  host1.1052         host1.ftp         ESTABLISHED
f1037f8c tcp        0      0  host1.1051         host1.telnet      ESTABLISHED
f103818c tcp        0      0  host1.1050         host1.telnet      ESTABLISHED

$ netstat -a
Active Internet connections (including servers)
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp        0      0  host1.1052             host1.ftp              ESTABLISHED
tcp        0      0  host1.1051             host1.telnet           ESTABLISHED
tcp        0      0  host1.1050             host1.telnet           ESTABLISHED
tcp        0      0  *.sunrpc               *.*                    LISTEN
tcp        0      0  *.time                 *.*                    LISTEN
tcp        0      0  *.daytime              *.*                    LISTEN
tcp        0      0  *.chargen              *.*                    LISTEN
tcp        0      0  *.discard              *.*                    LISTEN
tcp        0      0  *.echo                 *.*                    LISTEN
tcp        0      0  *.smtp                 *.*                    LISTEN
tcp        0      0  *.finger               *.*                    LISTEN
tcp        0      0  *.exec                 *.*                    LISTEN
tcp        0      0  *.login                *.*                    LISTEN
tcp        0      0  *.telnet               *.*                    LISTEN
tcp        0      0  *.ftp                  *.*                    LISTEN
udp        0      0  *.1031                 *.*
udp        0      0  *.2049                 *.*
udp        0      0  *.who                  *.*
udp        0      0  *.sunrpc               *.*
udp        0      0  *.time                 *.*
udp        0      0  *.daytime              *.*
udp        0      0  *.chargen              *.*
udp        0      0  *.discard              *.*
udp        0      0  *.echo                 *.*
udp        0      0  *.talk                 *.*
udp        0      0  *.tftp                 *.*

$ netstat -i
Name  Mtu   Network     Address      Ipkts   Ierrs Opkts   Oerrs Collis
tk0   1500  192.100.20  host1        1698272 0     507349  0     0
lo0   1536  loopback-n  localhost    13098   0     13098   0     0
```

```
$ netstat -s -r
routing:
 0 bad routing redirects
 0 dynamically created routes
 0 new gateways due to redirects
 5 destinations found unreachable
 0 uses of a wildcard route
$ netstat -s
ip:
 591797 total packets received
 0 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
 12654 fragments received
 0 fragments dropped (dup or out of space)
 1 fragment dropped after timeout
 0 packets forwarded
 68856 packets not forwardable
 0 redirects sent
icmp:
      10336 calls to icmp_error
 0 errors not generated 'cuz old message was icmp
 Output histogram:
  echo reply: 9
  destination unreachable: 10336
 0 messages with bad code fields
 0 messages < minimum length
 0 bad checksums
 0 messages with bad length
 Input histogram:
  destination unreachable: 9281
  echo: 9
 9 message responses generated
tcp:
 0 incomplete headers
 0 bad checksums
 0 bad header offset fields
udp:
 0 incomplete headers
 0 bad data length fields
 0 bad checksums

$ netstat -m
77/160 mbufs in use:
 3 mbufs allocated to packet headers
 26 mbufs allocated to socket structures
 41 mbufs allocated to protocol control blocks
 5 mbufs allocated to routing table entries
 2 mbufs allocated to interface addresses
0/36 mapped pages in use
164 Kbytes allocated to network (5% in use)
0 requests for memory denied

$ netstat -n
Active Internet connections
Proto Recv-Q Send-Q  Local Address          Foreign Address        (state)
tcp        0      0  193.255.2.4.1052       193.255.1.4.21         ESTABLISHED
tcp        0      0  193.255.2.4.1051        193.255.1.5.23          ESTABLISHED
tcp        0      0  193.255.2.4.1050        193.255.1.7.23          ESTABLISHED

$ netstat -f inet
Active Internet connections
Proto Recv-Q Send-Q  Local Address           Foreign Address         (state)
tcp        0      0  host1.1082              host1.telnet            ESTABLISHED
tcp        0      0  host1.1052              host1.ftp               ESTABLISHED
$
```

Related Information

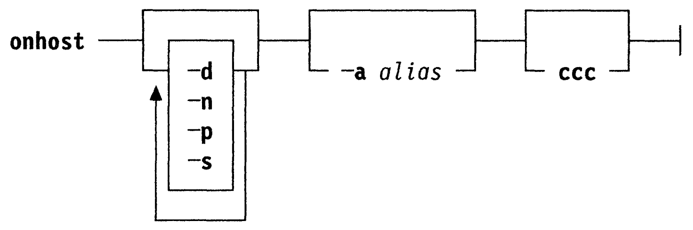
In this book:
"hosts" on page 5-4
"networks" on page 5-13
"protocols" on page 5-14
"rc.tcpip, rc.tcpip.local" on page 5-15

onhost

Purpose

Execute CMS commands on an IBM host

Syntax



Description

The **onhost** command uses the host connection created by the **hostconnect** command to execute commands on the host system. **onhost** behaves in one of three modes.

- *command line* is not present. This signals full-screen mode. The **onhost** command sets up full-screen emulation using an unmodified IBM 3270 terminal emulator such as **tn** or **tn3270**. (See "hostconnect" on page 3-17, **onhost.alias**, to specify the emulator.)
- *command line* is rewritten and passed to the host for execution. This process is called special command mode. It is controlled by **onhost.profil**, described below. Host input and output can be redirected, as usual. When the host command completes, a distinguished end-of-command line signals **onhost** to terminate with the exit code set by the host.
- Additional input is simply passed to the host. This is normal line mode. An exception is made if the line begins with a ∧(carat) and one of these characters, `1234567890-=abez`, which produce the 3270 actions pf1..pf12, pa1, pa2, enter, and clear, so that accidental execution of a full screen application can be controlled in line mode.

Note: The shell scans the command line before it is passed to onhost, so you must escape any shell meta-characters. The simplest way to do this is to put the command in quotes.

```
onhost "ls cms:*exec.a"
```

will list all files of type exec in your CMS A-disk directory, whereas

```
onhost ls cms:*exec.a
```

will typically report a `No match` error from the shell.

Flags

-a [*alias*]	The **-a** *alias* option uses the host connection made by **hostconnect -a** *alias*. It is not necessary to include the alias on every **onhost** command; the last alias used is assumed. You can display the current alias by specifying **onhost -a**.
-d	This option turns on debugging to study the code or track down a problem. The **-d** option sends debug output to the terminal. The **-d1** option sends debug output to the file **onhost.debug**.
-n	This option forces normal line mode treatment for ccc.
-p	This option displays a prompt before all host input. Special command mode is indicated by >>, normal line mode by >.
-s	This option shows the host return code.

Full Screen Mode

If **onhost** is invoked without any command, it sets up full-screen mode by informing **hostconnect**, then it executes **tn** or **tn3270**, for example, with the socket address of **hostconnect**, and disappears. The emulator behaves normally. But when you quit emulation, using **control-t**, **c** for **tn**, for example, **hostconnect** maintains the host connection for later **onhost** use. You must log out of the host to drop the connection.

Special Command Mode

onhost simulates some aspects of certain AIX commands. Actually, the IBM host portion of **onhost** does the simulation. The local **onhost** code simply looks up the command line token in **onhost.profil**, where the entry determines what is sent to the host. In most cases, a command such as **ls** is replaced by the string `onhost ls`, then sent to the host. You could do the same thing in full-screen mode, for example, by entering `onhost ls`. Enter `onhost ?` for a quick reference display.

The simulation does not attempt to provide AIX facilities on CMS, it simply helps you to do **cat**, **ls**, and **rm**, for example, without having to learn CMS. Most of the commands are simulated by a program named **onhost exec**. This program is written in REXX, an interpretive language, and can be easily modified to extend the scope and number of the commands which it simulates.

Special Command Descriptions

Several of the following special commands use host file names. These names are distinguished by the **cms:** prefix for CMS files. A file in CMS has a name and a type, each no longer than eight characters. A disk directory in CMS is identified by a single letter, called the mode when alone or suffixed with a single digit. So a CMS file with file name *fn* and file type *ft* in the disk directory *fm* is specified by **cms:***fn.ft.fm*.

onhost CAT

Displays a CMS file on standard output. For example,

```
onhost cat cms:fn.ft.fm
```

Only one file name may appear and there are no options. Pipe the result to "cat -n" for the -n and similar cat options. Files with more than eighty characters per line may contain lines broken at column 80.

Use the standard CMS copy command to catenate files into the single file, *fn ft fm,* for example,

```
onhost copy fn1 ft1 fm1 fn2 ft2 fm2 ... fn ft fm (replace
```

onhost CP

Copies files between AIX and the host. These commands copy a file from AIX to CMS, from CMS to AIX, and from CMS to CMS. The files are copied between systems using FTP; see the file transfer section, below, for details.

```
onhost cp test1.pascal cms:test1.pascal.a
onhost cp cms:output.data.a output.data
onhost cp cms:test1.pascal.a cms:oldtest1.pascal.a
```

The first file is the source. The second file is replaced and carries the current timestamp.

onhost DATE

Displays the date and time for the local time zone.

onhost DF

Displays the amount of free space on all accessed CMS disks. The free space on the CMS disk accessed as mode s is displayed by

```
onhost df s
```

onhost HEAD

Displays a file on standard output. The command

```
onhost head -n cms:fn.ft.fm
```

displays the first *n* lines of the CMS file. Only one file name can appear. If **-n** is omitted, then the default value of 10 is used. Files with more than eighty characters per line may contain lines broken at column 80.

onhost LS

Lists the contents of a CMS disk directory. The contents of the CMS disk accessed as mode x is displayed by

```
onhost ls ../x
```

The -l option gives a long form of listing. The -t option orders the result according to time, with the newest file at the head of the list. The -r option reverses the order. The options can be combined, for example, -lrt. Files may be listed which match patterns allowed by the CMS listfile command. Briefly, an asterisk represents any number of characters, and a percent represents any single character. For example, all files of type FORTRAN in any directory with names beginning with **a** and ending with **z**

```
onhost "ls -l cms:a*z.fortran.*"
```

The C-shell type of pattern matching using [ ] and the C-Shell use of { } is not supported. The format and main results are similar to those of the AIX ls command but some of the results differ. However, the command does show what files exist, whether they can be read or written, and their (approximate) size and date.

onhost RM

Removes a CMS file.

onhost TAIL

Displays the end of a file in a manner similar to onhost head.

onhost WHO	Lists users of the host system running with terminals connected. The userid and terminal is displayed. The command will not show how long a terminal has been logged in as this information is not available. `onhost who am i` will show your userid and terminal.
onhost WRITE	Writes a message to a user of the host system. `onhost write user message text` sends the message text to the user. Unlike AIX, it does not warn the recipient or wait for you to type the message.

onhost.profil

onhost.profil controls simple rewriting of **onhost** special commands. It is found in one of the directories specified in your path. There is a default **onhost.profil** in one of the standard directories. This file has entries of the form:

www sss

where *www* is a word and *sss* is a string. If you enter **onhost** *www xxx* and the word, *www*, is found in **onhost.profil**, then *www* is replaced by the string *sss*, and *sss xxx* is sent to the host. If *www* is not found, then **onhost** *www xxx* is sent to the host.

File Transfer

The **ftp** command, at the host, is used for file transfer. The CMS console is spooled to a file named **cms:onhostcp.ftptrace.a** during file transfer. The CMS **onhostcp** exec checks this file to determine if the file was copied. If the **onhost cp** command returns a non-zero return code, you can **cat** this trace file for details.

The **onhost cp** command is not different than any other special command. However, the address of the local system, the userid, and the password must be supplied to the host system so **ftp** can connect to the local system. This information is needed on every file transfer request. The token, $FTPAUPW, found in **onhost.profil**, is replaced by the AIX address, userid, password, and working directory before the special command goes to the host.

Trouble

It is difficult for **onhost** to properly handle the many situations which can arise in host communication. If a host command takes too long (taking communication delays into account) or behaves poorly, then enter **control-c** (or your escape character) to interrupt **onhost**. If **onhost** is waiting for your input, it will terminate. If **onhost** is waiting for the host to read, the interrupt will force a line to the host, or if no line is present, you will be prompted for a line or full-screen mode.

The host may have the console spooled with no output, as indicated by no display of entered lines; enter `sp cons stop` to clear this condition. If you can not log out of the host system, then kill **hostconnect**. Do not use kill -9! (See "hostconnect" on page 3-17 for additional information.)

Related Information

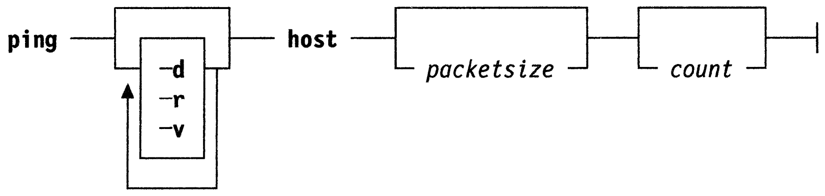
In this book:
"hostconnect" on page 3-17
Appendix B, "Onhost Installation" on page B-1

ping

Purpose

Sends ICMP ECHO_REQUEST packets to network hosts.

Syntax



Description

The DARPA Internet is a large and complex aggregation of network hardware, connected together by gateways. Tracking a single-point hardware or software failure can often be difficult. The **ping** command utilizes the ICMP protocol's mandatory ECHO_REQUEST datagram to elicit an ICMP ECHO_RESPONSE from a host or gateway.

ECHO_REQUEST datagrams (**pings**) have an IP and ICMP header, followed by a **struct timeval**, and then an arbitrary number of pad bytes used to fill out the packet. The default datagram length is 64 bytes, but this may be changed using the command line option.

When using **ping** for fault isolation, it should first be run on the local host to verify that the local network interface is up and running. Then, hosts and gateways further and further away should be pinged. The **ping** command sends one datagram per second and prints one line of output for every ECHO_RESPONSE returned. By default, **ping** sends one packet. If an optional count is given, only that number of requests are sent. Round-trip times and packet loss statistics are computed. When all responses have been received or the program times out (with a count specified), or if the program is terminated with a SIGINT, a brief summary is displayed.

This program is intended for use in network testing, measurement and management. It should be used primarily for manual fault isolation.

Flags

The **ping** command options are:

-d Sets the SO-Debug flag on the socket that **ping** is using. This enables console output of debugging messages from the TCP/IP routines in the kernel. Although this flag has no direct effect on **ping's** output, it does affect console output from debugging routines in the kernel.

-r Bypasses the normal routing tables and sends directly to a host on an attached network. If the host is not on a directly attached network, an error is returned. This option can be used to ping a local host through an interface that has no route through it (for example, after the interface was dropped by routed.)

-v Verbose output. All ICMP packets other than ECHO_RESPONSE that are via the raw socket are listed (not only those packets related **ping**).

Examples

```
$ /etc/ping host1
PING host1: 56 data bytes
----host1 PING Statistics----
packet transmitted, packet received, 0% packet loss
round-trip (ms)  min/avg/max = 26/32/55
$
```

Related Information

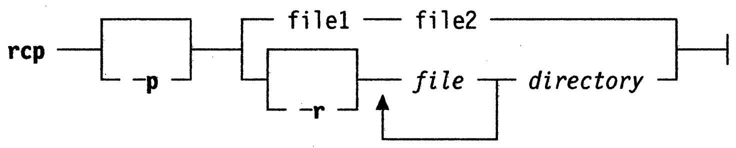
In this book:
"netstat" on page 3-32
"ifconfig" on page 3-23

rcp

Purpose

Copies and transfers remote files.

Syntax



Description

The **rcp** command copies files between machines. Generally used to access files outside your cluster. Each file or directory argument is either a remote file name of the form *rhost:path*, or a local file name (containing no **:** characters, or a **/** before any **:**'s.).

If *path* is not a full path name, it is interpreted relative to your login directory on *rhost*. A *path* on a remote host may be quoted (using \, ", or ') so that the metacharacters are interpreted remotely.

The **rcp** command does not prompt for passwords; your current local user name must exist on *rhost* and allow remote command execution via **rsh**. Also, an *.rhosts* file must exist in your home directory on rhost. It must contain the name of your local host.

This command can handle remote-to-remote copies, where neither the source nor the target files are on the machine where the **rcp** command is expected. A remote host name may be specified in the **user@host** format, where the user can be different from the UID of the process executing the command. The destination host name may also take the form **rhost.rname** to support destination machines that are running 4.2BSD versions of **rcp**.

Note: The **rcp** command doesn't detect all cases where the target of a copy might be a file where only a directory should be legal. The command is confused by any output generated by commands in a **.login**, **.profile**, or **.cshrc** file on the remote host.

Flags

The **rcp** command options are:

-p Causes the **rcp** command to attempt to preserve (duplicate) in its copies the modification times and modes of the source files, ignoring the **umask**.

-r Copies each subtree rooted, if any of the source files are directories. By default, the mode and owner of *file2* are preserved if it already existed. Otherwise, the mode of the source file modified by the **umask** on the destination host that is used.

Examples

1. Remote-copy the file **myfile.1** from the local host to the remote host named **remote_host**, and name the file **myfile.2** on the remote host. Then copy the file **myfile.2** from the remote host back to the local host.

   ```
   $ rcp myfile.1 remote_host:myfile.2
   $ rcp remote_host:myfile.2 myfile.3
   ```

2. A user John logged into **host6** can execute the following commands, provided that he has appropriate access privileges:

   ```
   $ rcp mary@host7:oldfile fred@host8:newfile
   $ rcp mary@host7:oldfile vax4_2bsd.fred:newfile
   ```

Related Information

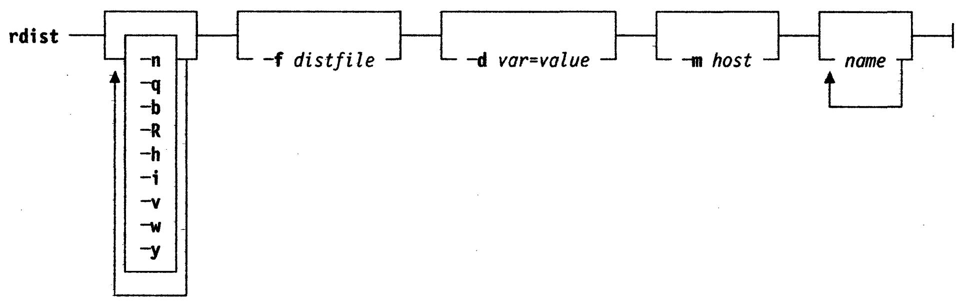
In this book:
"ftp" on page 3-6
"rsh, remsh" on page 3-58
"rlogin" on page 3-54
".rhosts" on page 5-16
"hosts.equiv" on page 5-6
cp, refer to *AIX Operating System Commands Reference*

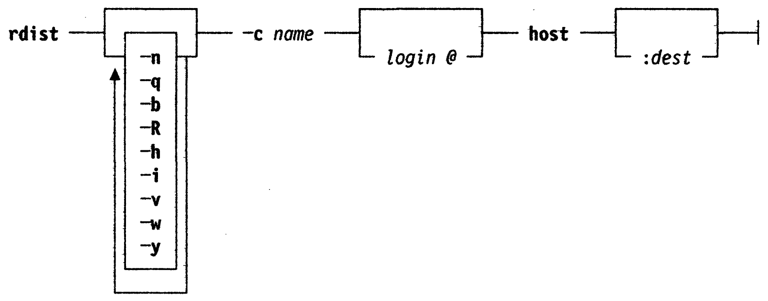
rdist

Purpose

Maintains identical copies of files over multiple hosts.

Syntax





Description

The **rdist** command maintains identical copies of files over multiple hosts. It preserves the owner, group, mode, and mtime of files if possible and can update programs that are executing. The **rdist** command aborts when using files which have a negative **mtime** (file's time of last modification; see **statx** in the *AIX Operating System Technical Reference*). This command reads commands from *distfile* to direct the updating of files and directories. Source files must reside on the local host where **rdist** is executed. If *distfile* is **-**, the standard input is used. If no **-f** option is present, the program looks in the current directory first for a file called *distfile*, then a file called *Distfile* to use as the input. If no names are specified on the command line, **rdist** updates all of the files and directories listed in *distfile*. Otherwise, the argument is taken to be the name of a file to be updated or the label of a command to execute. If label and file names conflict, it is assumed to be a label. These may be used together to update specific files using specific commands. There is no simple way to have a special command executed after all files in a directory have been updated.

Note: When rdist encounters a hidden directory on the local host, it creates or updates a flat file on the destination host. The flat file will contain one of the selectable components of the hidden directory.

Flags

The **rdist** command options are:

-b Binary comparison. Performs a binary comparison and updates files if they differ rather than comparing dates and sizes.

-c *name* Forces **rdist** to interpret the remaining arguments as a small **distfile**. The equivalent distfile is as follows.

```
(name ... ) -> [login @]host
install [dest] ;
```

-d *var = value*
Defines **var** to have value. The **-d** option is used to define or override variable definitions in the **distfile**. Value can be the empty string, one name, or a list of names surrounded by parentheses and separated by tabs and spaces.

-D Displays debugging information.

-f *distfile* Specifies the path name for *distfile*. If no **-f** option is present, it looks in the current directory first for **distfile**, then **Distfile** to use as the input.

-h Follows symbolic links. Copies the file that the link points to rather than the link itself.

-i Ignores unresolved links. **rdist** normally tries to maintain the link structure of files being transferred and warns the user if all the links cannot be found.

-m *host* Limits the machines which are to be updated. Multiple **-m** arguments can be given to limit updates to a subset of the hosts listed the **distfile**.

-n Prints the commands without executing them. This option is useful for debugging **distfile**.

-q Quiets mode. Files that are being modified are normally printed on standard output. The **-q** option suppresses this.

-R Removes extraneous files. If a directory is being updated, any files that exist on the remote host that do not exist in the master directory are removed. This is useful for maintaining truly identical copies of directories.

-v Verifies that the files are up to date on all the hosts. Any files that are out of date are displayed but no files are changed nor any mail sent.

-w Whole mode. The whole file name is appended to the destination directory name. Normally, only the last component of a name is used when renaming files. This preserves the directory structure of the files being copied instead of flattening the directory structure. For example, renaming a list of files such as (**dirl/fl dir2/f2**) to **dir3** would create files **dir3/dirl/fl** and **dir3/dir2/f2** instead of **dir3/fl** and **dir3/f2**.

-y Younger mode. Files are normally updated if their **mtime** and size disagree. The **-y** option causes **rdist** not to update files that are younger than the master copy. This can be used to prevent newer copies on other hosts from being replaced. A warning message is printed for files which are newer than the master copy.

Distfile contains a sequence of entries that specifies the files to be copied, the destination hosts, and what operations the user needs to perform to update the files. Each entry has one of the following formats.

```
<variable name>'='<name list>
[ label: ] <source list> '→'<destination list> <command list>
[ label: ] <source list>'::'<time_stamp file> <command list>
```

The first format is used for defining variables. The second format is used for distributing files to other hosts. The third format is used for making lists of files that have been changed since some given date. The source list specifies a list of files and directories on the local host which are to be used as the master copy for distribution. The destination list is the list of hosts to which these files are to be copied. Each file in the source list is added to a list of changes if the file is out of date on the host which is being updated (second format) or the file is newer than the time stamp file (third format).

Labels are optional. They are used to identify a command for partial updates.

New lines, tabs, and blanks are only used as separators and are otherwise ignored. Comments begin with # and end with a new line.

Variables to be expanded begin with **$** followed by one character or a name enclosed in curly braces (see the examples at the end). Variable expansion only works for name lists.

The source and destination lists have the following format:

```
<name>
or
( <zero or more names separated by white-space> )
```

The shell meta-characters **[**, **]**, **{**, **}**, *****, and **?** are recognized and expanded (on the local host only) in the same way as the C shell. They can be escaped with a backslash \. The ~ character is also expanded in the same way as the C shell but is expanded separately on the local and destination hosts. When the **-w** option is used with a file name that begins with ~, everything except the home directory is appended to the destination name. File names which do not begin with **/** or ~ use the destination user's home directory as the root directory for the rest of the file name.

Parameters

The command list consists of zero or more commands of the following format.

```
install <options> opt_dest_name ;
notify <name list> ;
except <name list> ;
except_pat <pattern list> ;
special <name list> string ;
```

install
Copies out-of-date files and directories. Each source file is copied to each host in the destination list. Directories are recursively copied in the same way. *opt_dest_name* is an optional parameter to rename files. If no **install** command appears in the command list or the destination name is not specified, the source file name is used. Directories in the path name are created if they do not exist on the remote host. To help prevent disasters, a non-empty directory on a target host is never replaced with a regular file or a symbolic link. However, under the **-R** option a non-empty directory is removed if the corresponding file name is completely absent on the master host. The options are **-R**, **-h**, **-i**, **-v**, **-w**, **-y**, and **-b** and have the same semantics as options on the command line except they only apply to the files in the source list. The login name used on the destination

host is the same as the local host unless the destination name is of the format *login@host*.

notify
Mails the list of updated files (and any errors that may have occurred) to the listed names. If no @ appears in the name, the destination host is appended to the name (*name1@host, name2@host*, ...).

except
Updates all of the files in the source list except for the files listed in namelist. This is usually used to copy everything in a directory except certain files.

except_pat
Like the **except** command except that **pattern list** is a list of regular expressions (see **ed** in the *AIX Operating System Command Reference* for details). If one of the patterns matches some string within a file name, that file is ignored. Note that since \ is a quote character, it must be doubled to become part of the regular expression. Variables are expanded in pattern list but not shell file pattern matching characters. To include a **$**, it must be escaped with \.

special
Specifies **sh** (see *AIX Operating System Commands Reference*) commands that are to be executed on the remote host after the file in name list is updated or installed. If the name list is omitted then the shell commands are executed for every file updated or installed. The shell variable *FILE* is set to the current file name before executing the commands in *string*. *String* starts and ends with " and can cross multiple lines in **distfile**. Multiple commands to the shell should be separated by ;. Commands are executed in the user's home directory on the host being updated. The **special** command can be used to rebuild private data-bases after a program has been updated.

The following is a short example:

```
HOSTS = (matisse root@arpa)

FILES = ( /bin/lib/usr/bin/usr/games
        /usr/include/{*.h,{stand,sys,vax*,pascal,machine}/*.h}
        /usr/lib/usr/man/man? /usr/ucb/usr/local/rdist )

EXLIB = ( Mail.rc aliases aliases.dir aliases.pag crontab dshrc
        sendmail.cf sendmail.fc sendmail.hf sendmail.st uucp vfont)

$ {FILES} → $ {HOSTS}
       install -R ;
       except /usr/lib/$/{EXLIB} ;
       except /usr/games/lib ;
       special /usr/lib/sendmail "/usr/lib/sendmail - bz";

srcs:
/usr/src/bin → arpa
         except_pat ( \\.o\$ /SCCS     lash.$ ) ;

IMAGEN = (ips dviimp catdvi)

imagen:
/usr/local/$(IMAGEN) → arpa
          install /usr/local/lib ;
          notify ralph ;

$ {FILES}   stamp.cory
                 notify root@cory ;
```

Examples

To print update commands without executing them:

```
$ rdist -n -f mydistfile
$ _
```

To limit update to host2 and host3:

```
$ rdist -f mydistfile -m host2 -m host3
$ _
```

To display a list of out-of-date files:

```
$ rdist -v -f mydistfile
$ _
```

Messages

A complaint about mismatch of **rdist** version numbers may really stem from some problem with starting your shell, such as you are in too many groups.

Files

distfile	input command file
/tmp/rdist*	temporary file for update lists

Related Information

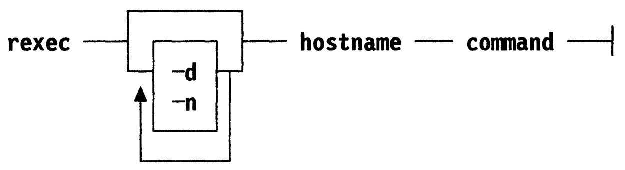
sh, **csh**, refer to *AIX Operating System Command Reference*
stat, refer to *AIX Operating System Technical Reference*

rexec

Purpose

Executes commands one at a time on a foreign host in a secure environment.

Syntax



Description

The **rexec** command executes a command on the specified foreign host. The user is prompted for a valid user ID and password (if applicable) for the foreign host. The **rexec** command sends and receives data over a TCP connection.

The **rexec** command also allows you to run interactive commands remotely, provided that they do not require a full screen display. In interactive processing, **rexec** sends all characters typed at the local keyboard to the foreign host until the EOF character is sent. If the shell running on the foreign host is **csh**, **rexec** sends **SIGINT** and **SIGQUIT** to the foreign host.

For each user who concurrently runs commands on a host with **rexec**, the host must have one PTY configured. That is, if three users on *host1* are using **rexec** to run commands on *host2* concurrently, *host2* must have three PTYs configured. In the PTY definitions, the ***auto enable*** and ***logger*** keywords must be set to ***false***.

Flags

The **rexec** command options are:

-d	Is an optional flag that provides a debug service. The **-d** flag causes **rexec** to print debug statements to the file **rexec.log** in the current directory.
-n	Is an optional flag that disables an automatic login feature. By default **rexec** searches the **$HOME/.netrc** file for the user's ID and password on the foreign host.

Parameters

hostname
Is a required parameter that specifies the name of the host where the command is to be executed. **hostname** can be in octal, dotted decimal, or name (alphanumeric) form.

command
A required parameter that specifies the command to be executed on the foreign host. If **command** (the command name, together with flags or parameters) contains embedded blanks, it may be enclosed in double quotes (").

Examples

1. To execute the **date** command on foreign host,**host1**:

```
$ rexec host1 date
Name (host1:tom):  tom
Password (host1:tom):
[date command output]
$ _
```

2. To list the directory of user tom on the foreign host, **host1**:

```
$ rexec host1 "li -l /u/tom"
Name (host1:tom):  tom
Password (host1:tom):
[listing of tom's directory on foreign host]
$ _
```

3. To list the /tmp directory of the foreign host, **host1**:

```
$ rexec host1 "li /tmp"
Name (host1:tom):  tom
Password (host1:tom):
[listing of /tmp on foreign host]
$
```

4. To start a shell (**csh**) on the foreign host, **host1** enter:

```
$ rexec host1 csh
Name (host1:tom):  tom
Password (host1:tom):
%
:
[EOF to end the foreign shell.]
```

5. To use the automatic login feature for the user ID tom on the foreign host, **host1**, create a file called **.netrc** in the user's home directory on **host1** using the following format:

   ```
      machine hostname login userid password password
   ```

   where:
   hostname is the name of the host on which the *userid* exists, **host1** in this example. *userid* is the user ID on the host, tom in this example. *password* is the password for the *userid.*

   For example,

   ```
   machine host1 login tom password tomspswd
   ```

   Now user tom will not be queried for his user ID and password when he executes a command such as:

   ```
   $ rexec host1 date
   ```

For more information on the format of the .netrc file, see "The .netrc File" on page 3-13.

Files

/etc/hosts	Name to address map table
./rexec.log	Debugging statements
$HOME/.netrc	User IDs and passwords for remote login.

Related Information

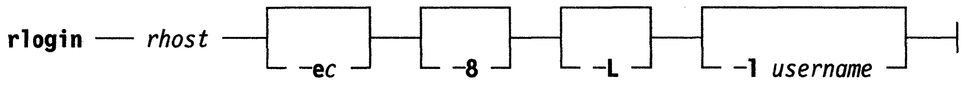
In this book:
"rexecd" on page 4-12
".rhosts" on page 5-16
"hosts.equiv" on page 5-6
".netrc" on page 5-12

rlogin

Purpose

Connects your terminal on the current local host system **lhost** to the remote host system **rhost**.

Syntax



Description

The **rlogin** command connects your terminal on the current local host system **lhost** to the remote host system **rhost**, which is not normally a member of the TCF cluster.

Each host has a file **/etc/hosts.equiv** which contains a list of rhost's with which it shares account names. When any user except root uses **rlogin** to log in as the same user on an equivalent host, you don't need to give a password. Each user including root may also have a private equivalence list in a file **.rhosts** in his directory. Each line in this file should contain an **rhost** and a username separated by a space, giving additional cases where logins without passwords are to be permitted. If the originating user is not equivalent to the remote user, then a login and password will be prompted for on the remote machine as in **login**. To avoid some security problems, the **.rhosts** file must be owned by either the remote user or root.

The remote terminal type is the same as your local terminal type (as given in your environment **term** variable). The terminal or window size is also copied to the remote system if the server supports the option, and changes in size are reflected as well. All echoing takes place at the remote site. The **rlogin** session is transparent except for delays. Flow control via **Ctrl S** and **Ctrl Q** and flushing of input and output on interrupts are handled properly. The optional argument **-8** allows an eight-bit input data path at all times. Otherwise, parity bits are stripped except when the remote side's stop and start characters are other than **Ctrl S/Ctrl Q**. The argument **-L** allows the **rlogin** session to be run in ***litout*** mode (terminal literal output mode; see **termio** in the *AIX Operating System Technical Reference*). A line of the form ~ disconnects from the remote host, where ~ is the escape character. Similarly, the line ~∧**Z** (where **Ctrl Z**, is the suspend character) suspends the **rlogin** session. Substitution of the delayed-suspend character (normally **Ctrl Y**) for the suspend character delays the send portion of the **rlogin** session, but allows output from the remote system. A different escape character may be specified by the **-e** option. There is no space separating this option flag and the argument character.

Flags

The **rlogin** command options are:

-e*c*	Changes the escape character. Substitutes the character you choose for *c*.
-8	Allows an 8-bit data path at all times. Otherwise, unless the start and stop characters on the foreign host are not **Ctrl-S** and **Ctrl-Q**, the **rlogin** command uses a 7-bit data path and parity bits are stripped.
-l	Changes the remote user name to the one you specify. Otherwise, your local user name is used at the foreign host.
-L	Allows the **rlogin** session to be run in ***litout*** mode.

Examples

The following example assumes that:

- An *.rhosts* file exits in the users directory on *host2*.
- The prompt on *host2* has been customized to be [host2].

First, perform a remote login to the remote host named **host2**, on which you have a login. No password is required, and the remote hosts prompts with [host2/]$.

```
$ rlogin host2
[host2]$
```

In the second example, try to log in as user bwalker using the **-l** option. You are asked for a password. If you enter the correct one, you are logged in. If not, as in the example, you get an error message and **rlogin** does not succeed.

```
$ rlogin host2 -l bwalker
Password: iforget
You entered a login name or password that is not valid.
Connection closed.
$
```

Related Information

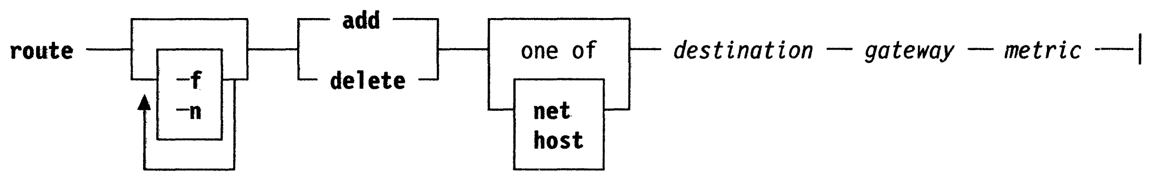
In this book:
"rsh, remsh" on page 3-58
".rhosts" on page 5-16
"hosts.equiv" on page 5-6

route

Purpose

Manually manipulates the routing tables.

Syntax



Description

The **route** command is used to manually manipulate the network routing tables. It normally is not needed, as the system routing table management daemon, **routed,** tends to this task.

The **route** command accepts two commands: **add**, to add a route, and **delete**, to delete a route. In addition the **route** command accepts the following parameters:

destination
Is the destination host or network.

gateway
Is the next-hop gateway to which packets should be addressed.

metric
Is a count indicating the number of hops to the *destination.*

The metric is required for **add** commands. It must be zero if the destination is on a directly-attached network, and nonzero if the route utilizes one or more gateways. If adding a route with metric 0, the gateway given is the address of this host on the common network, indicating the interface to be used for transmission. Routes to a particular host are distinguished from those to a network by interpreting the Internet address associated with *destination.*

The optional keywords **net** and **host** force the *destination* to be interpreted as a network or a host, respectively. Otherwise, if the *destination* has a local address part of INADDR_ANY, or if the *destination* is the symbolic name of a network, then the route is assumed to be to a network. It is presumed to be a route to a host. If the route is to a *destination* connected via a gateway, the **metric** should be greater than 0. All symbolic names specified for a *destination* or *gateway* are looked up first as a host name using **gethostbyname**. If this lookup fails, **getnetbyname** is then used to interpret the name as that of a network.

The **route** command uses a raw socket and the SIOCADDRT and SIOCDELRT **ioctl**'s to do its work. As such, only the superuser may modify the routing tables.

Flags

The **route** command options are:

-f Flushes the routing tables of all gateway entries. If this is used in conjunction with one of the commands described above, the tables are flushed prior to the command's application.

-n Prevents attempts to print host and network names symbolically when reporting actions.

Examples

```
# /etc/route add newnet thishost 3
```

```
# /etc/route delete newnet thishost
```

Messages

add [**host | network**] *destination gateway*

Explanation: The specified route is being added to the tables. The values printed are from the routing table entry supplied in the **ioctl** call. If the gateway address used was not the primary address of the gateway (the first one returned by **gethostbyname**), the gateway address is printed numerically as well as symbolically.

delete [**host | network**] *destination gateway*

Explanation: As above, but when deleting an entry.

Flushing routing tables: *destination gateway*

Explanation: When the **-f** flag is specified, each routing table entry deleted is indicated with a message of this form.

network is unreachable

Explanation: An attempt to add a route failed because the gateway listed was not on a directly-connected network. The next-hop gateway must be given.

not in table

Explanation: A **delete** operation was attempted for an entry which wasn't present in the tables.

routing table overflow

Explanation: An **add** operation was attempted, but the system was low on resources and was unable to allocate memory to create the new entry.

Related Information

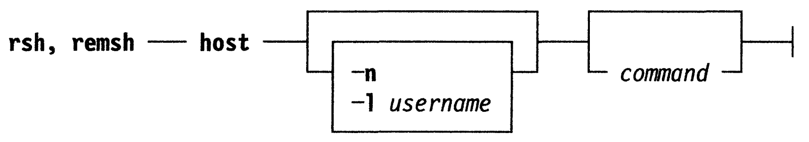
In this book: "routed" on page 4-16

rsh, remsh

Purpose

Connects to the specified **host** and executes the specified command.

Syntax



Description

The **rsh** command connects to the specified *host*, and executes the specified command. This command copies its standard input to the remote command, the standard output of the remote command to its standard output, and the standard error of the remote command to its standard error. Interrupt, quit and terminate signals are propagated to the remote command. The **rsh** command normally terminates when the remote command terminates.

The remote username used is the same as your local username, unless you specify a different remote name with the **-l** options. This remote name must be equivalent (in the sense of **rlogin**, see "rlogin" on page 3-54,) to the originating account. No provision is made for specifying a password with a command.

If you omit *command*, then instead of executing a single command, you log in on the remote host using the **rlogin** command.

Shell metacharacters which are not quoted are interpreted on the local machine, while quoted metacharacters are interpreted on the remote machine. Thus the command

```
rsh otherhost cat remotefile >> localfile
```

appends the remote file **remotefile** to the local file **localfile**, while

```
rsh otherhost cat remotefile ">>" otherremotefile
```

appends **remotefile** to **otherremotefile**.

Host names are given in the file **/etc/hosts**. Each host has one standard name (the first name given in the file), which is rather long and clearly defined. Optionally, the host has one or more nicknames.

If you are using C shell and put a **rsh** command in the background without redirecting its input away from the terminal, it blocks even if no reads are posted by the remote command. If no input is desired, you should redirect the input of **rsh** to **/dev/null** using the **-n** option.

Flags

-n	Redirects the input of **rsh** to **/dev/null** if you desire no input.
-l	Specifies a different remote name. If **-l** is not used, the remote username used is the same as your local username.

Examples

To compile a file on a remote host:

```
$ rsh host2 "cc -o hello hello.c"
$ _
```

To read a remote file:

```
$ rsh host2 "cat /tmp/log"
This is the output from the file /tmp/log on
host2
.
.
.
$ _
```

Files

/etc/hosts

Related Information

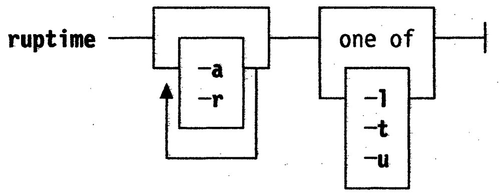
In this book:
"rlogin" on page 3-54
"hosts.equiv" on page 5-6
".rhosts" on page 5-16

ruptime

Purpose

Gives a status line like **uptime** for each machine on the local network.

Syntax



Description

The **ruptime** command gives a status line like **uptime** for each machine on the local network. These are formed from packets broadcast by each host on the network once a minute.

Machines, for which no status report has been received for 11 minutes, are shown as being down.

Users idle an hour or more are not counted unless the **-a** flag is given.

Normally, the listing is sorted by host name.

Flags

The **ruptime** command options are:

-a	Allows output to be produced on status of users last known to be logged into a machine but idle for an hour or more.
-l	Specifies sorting by load average.
-r	Reverses the sort order.
-t	Specifies sorting by uptime.
-u	Specifies sorting by number of users.

Examples

```
$ ruptime
host1        up  1+15:16,    2 users, load  0.22,  0.26,  0.30
host2        up  3+18:45,    0 users, load  0.37,  0.13,  0.11
host3      down  2+00:56
host4        up  1+00:54,    2 users, load  0.04,  0.22,  0.18
host5        up  2+22:38,    0 users, load  0.11,  0.14,  0.18
host6        up     0:33,    1 user,  load  0.47,  0.51,  0.45
host7        up  1+20:05,    7 users, load  0.57,  0.60,  0.54
host8        up  2+22:39,    5 users, load  0.29,  0.27,  0.36
host9        up     8:07,    0 users, load  0.12,  0.13,  0.17
```

Files

/usr/spool/rwho/whod.* data files

Related Information

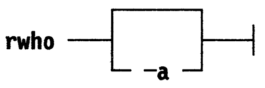
In this book: “rwho” on page 3-62

rwho

Purpose

Produces output similar to **who** for all machines on the local network.

Syntax



Description

The **rwho** command produces output similar to **who**, but for all machines on the local network that are running the **rwhod** daemon. (See page 4-22.) An **rwho** daemon must be running on a host in order for **rwho** to receive information about the users on that host. If no report has been received from a machine for 5 minutes then **rwho** assumes the machine is down, and does not report users last known to be logged into that machine.

If a user hasn't typed to the system for a minute or more, then **rwho** reports this idle time. If a user hasn't typed to the system for an hour or more, then the user is omitted from the output of **rwho** unless the **-a** flag is given.

Flags

The **rwho** command option is:

-a Allows output to be produced on status of users last known to be logged into that machine but idle for an hour or more.

Examples

The following example assumes that host1 - host7 are running the **rwhod** daemon.

```
$ rwho
arw      host1:tty0      Jan 23 16:11 :54
dang     host1:ttyp0     Oct 20 06:18 :52
eve      host2:console   Jul 27 07:20 :06
joel     host1:tty1      Jun  3 23:07 :30
oleg     host4:console   Aug 18 12:01 :40
root     host6:console   Jan  1 08:51 :12
root     host3:console   Jan  1 08:51 :36
root     host5:console   Jan  1 08:51 :32
root     host7:console   Jan  1 08:51 :11
timabel  host1:tty3      Jan 13 05:31 :09
wu       host1:console   Oct  4 19:44 :26
$
```

Files

/usr/spool/rwho/whod.* information about other machines

Related Information

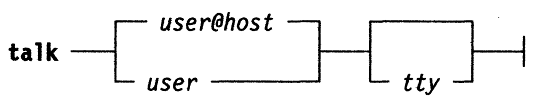
In this book:
"ruptime" on page 3-60
"rwhod" on page 4-22

talk

Purpose

Converse with another user.

Syntax



Description

The **talk** command allows two users to type simultaneously into windows displayed on each other's terminals. To initiate a conversation, a user executes the **talk** command specifying the second user's account name. When the second user is logged onto the same host or onto a host in the same TCF cluster as the first user, only the user name need be specified—the *host* and *tty* specifications may be omitted.

If the second user is known to be on a specific foreign host, either inside or outside of the TCF cluster, the name of the host may be specified in one of the following ways:

user@host

host!user

host.user

host:user

If you specify the *host!user* form from the **csh** shell, the '!' (exclamation point) needs to be quoted; otherwise, the **csh** shell will try to interpret the (!) as a magic character. One way of quoting the **csh** shell is to specify *host\!user*. When the second user is logged onto more than one terminal, the **talk** command selects one of the terminals according to the following rules. If a specific terminal is specified with the tty parameter, that terminal is selected. Otherwise, the **talk** command selects the first login session it finds on the specified host. If no host is specified, **talk** selects the first login session it finds for that user, looking first on the local host, then on other sites within the TCF cluster.

When the first user initiates the conversation, a message is sent to the second user, inviting a conversation. Once the invitation is received, the **talk** command displays two windows on the first user's terminal and displays progress messages until the second user responds to the invitation.

If the second user wants to have the conversation, the second user also executes **talk** from any terminal and specifies the first user's account name and host name, if appropriate. If the second user accepts the invitation, **talk** displays two windows on each user's terminal. One window displays what is typed by the local user. The other displays what is typed by the remote user. To end the conversation, either user can press **Cntl-C** (INTERRUPT) key and the connection is closed.

If the second user does not want to permit **talk** invitations, that user should issue the **mesg n** command. For more information on using the **mesg** command, see *AIX Operating System Commands Reference.*

Note: The **talk** command uses the talk 4.3BSD protocol, which is not compatible with 4.2BSD versions of **talk**.

Examples

1. If john at **host1** wants to talk to fred, who is logged in on **host2**, john enters:

   ```
   $ talk fred@host2
   ```

   The following message is displayed on fred's terminal:

   ```
   Message from TalkDaemon@host1 at 15:16...
   talk: connection requested by john@host1.
   talk: respond with: talk john@host1
   ```

   To accept the invitation, fred enters:

   ```
   $ talk john@host1
   ```

2. To talk to fred, only if he is logged in on the console at **host2**, enter:

   ```
   $ talk fred@host2 console
   ```

Related Information

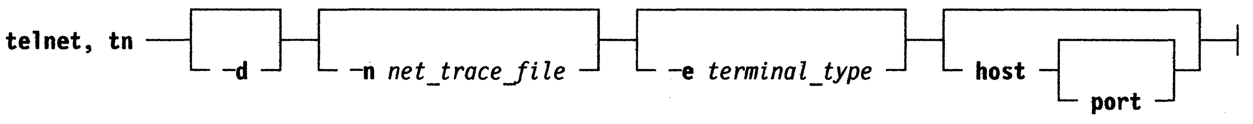
In this book: "talkd" on page 4-24.
mesg, refer to *AIX Operating System Command Reference*

telnet, tn, tn3270

Purpose

Provides the TELNET interface for logging in to a foreign host.

Syntax



Description

The **telnet** command implements the TELNET protocol, which allows remote login to other hosts. It uses the Transmission Control Protocol (TCP) to communicate with other hosts in the network. The **tn** command is identical to the **telnet** command, and in the following discussion, **telnet** means either the **telnet** command or the **tn** command. The **tn3270** command is used to connect to an IBM main frame system and is equivalent to running **telnet -e3270**.

The **telnet** command operates in two different modes: ***command mode*** and ***input mode***. When issued without arguments, **telnet** enters command mode, as indicated by the `telnet>` prompt. In this mode, the subcommands listed under "Parameters" on page 3-68 can be executed.

If the **telnet** command is issued with arguments, it performs an **open** subcommand with those arguments, then enters input mode. The type of input mode is either ***character-at-a-time*** or ***line-by-line***, depending on what the remote system supports.

In character-at-a-time mode, most text typed is immediately sent to the remote host for processing. In line-by-line mode, all text is echoed locally and completed lines are sent to the remote host. The local echo character is used to shut the local echo off and turn it back on. Its initial value is **Ctrl-E**.

In either input mode, if **toggle localchars** is true (see "Parameters" on page 3-68), the user's **QUIT**, **INTR**, and **FLUSH** characters are trapped locally and sent as TELNET sequences to the remote host. The **toggle autoflush** and **toggle autosynch** subcommands cause this action to flush subsequent output to the terminal until the remote host acknowledges the TELNET sequence and to flush previous terminal input (in the case of **QUIT** and **INTR**).

To enter **telnet** command mode while connected to a remote host, type the TELNET escape key sequence. The default escape sequence is **Ctrl-T**.

You may use screen-oriented programs on a host with which you are communicating through **telnet**. The **telnet** command will pass the name of your local terminal type through to the remote host when you first connect to it. If the remote host supports *terminal-negotiation* and does not recognize the name of your terminal, **telnet** will attempt to negotiate a mutually acceptable terminal type with the remote host. If you are using an HFT terminal (such as the System/370 console), **telnet** can emulate either a DEC VT100 terminal or an IBM 3270 terminal. If your local terminal is not an HFT, **telnet** usually can emulate an IBM 3270 terminal. If you can use the **vi** command (see the *AIX System/370 Text Formatting Guide* and the *AIX Operating System Commands Reference* for information on the **vi** command) on your terminal, then **telnet** can emulate a 3270 using your terminal. If none of the terminals **telnet** can emulate are acceptable to the remote host, the original terminal type will be used. You may specify a terminal-type to be emulated. If you do so, **telnet** will not negotiate for terminal type with the remote host. You can bypass terminal emulation and negotiation altogether by asking **telnet** to emulate terminal "none."

To override the terminal negotiation from the console, use the **EMULATE** environment variable or the **-e** option. To determine whether terminal-type negotiation is performed, the following list describes the order of the **telnet** command processing:

- The **-e** command line flag. (No negotiation)
- The **EMULATE** environment variable. (No negotiation)
- If the first two items are not present, terminal-type negotiation occurs automatically.

If the client and the server negotiate to use a 3270 data stream, the keyboard mapping to be used is determined by the following precedence:

$HOME/.3270keys	User's 3270 keyboard mapping.
/etc/3270.keys	Default 3270 keyboard mapping.

The **telnet** command supports these 3270 terminal types: 3277-1, 3278-1, 3278-2, 3278-3, 3278-4, and 3278-5. If you are using the **telnet** command in 3270 mode on a color display, the colors and fields are displayed the same as those of an actual 3279 display.

Restrictions

The mouse cannot be used as an input device with the **telnet** command.

The **telnet** command does not support the APL data stream.

VT100 is a trademark of Digital Equipment Corporation.

Environment Variables

The following environment variables can be used with the **telnet** command:

EMULATE	Specifies type of terminal for **telnet** to emulate. If given the **-e** flag overrides this variable. The value of EMULATE may be set to **vt100**, but **telnet** will only try to emulate a **vt100** if you are using an HFT terminal. If the type to emulate is 3270, **telnet** will do so on any terminal (as long as that terminal can be used with **vi**).
TNESC	Specifies an alternate TELNET escape character, other than the default **Ctrl-T**. You can use **TNESC** to change the **telnet** escape sequence. To change the sequence, set **TNESC** to the octal value of the character you want to use and export **TNESC**. Refer to the *AIX Operating System Technical Reference* for a table that maps octal values to their ascii equivalents.

For more information on using environment variables, see the **env** command in the *AIX Operating System Commands Reference.*

Flags

The **telnet, tn** command options are:

-d	Turns debugging mode on.
-e *terminaltype*	Overrides terminal-type negotiation. Possible values for terminal type are **vt100**, **3270** and **none**.
-n *net_trace_file*	Records network trace information in the file specified by *net_trace_file*.

Parameters

For each of the subcommands listed below, you only need to type enough letters to uniquely identify the command. (For example, **q** is sufficient for the **quit** command.) This is also true for the arguments to **emulate**, **display**, **mode**, **set**, and **toggle**.

The subcommands for **telnet** and **tn** are:

? [*command*]
Requests help on **telnet**. Without arguments, the **telnet** command prints a help summary. If a command is specified, the **telnet** command prints help information for just that command.

close
Closes the TELNET connection. If **telnet** was started with a hostname on the command line, **close** exits **telnet**. Otherwise, **close** returns to command mode.

display [*argument*]
Displays all of the **set** and **toggle** values if no argument is specified. Otherwise, lists only those values that matches the argument.

emulate *terminaltype*
Overrides terminal type negotiation with the specified *terminaltype*. Possible choices are:

?	- Prints help information.
3270	- Emulates a 3270 terminal.
none	- Specifies no emulation.
vt100	- Emulates a DEC VT100 terminal (only on hft).

All output received from the remote host is processed by the specified emulator. The initial terminal type to emulate can be specified through the **EMULATE** environment variable or the **-e** option to the **telnet** command.

mode *type*
Specifies the current input mode. When *type* is **line**, the mode is line by line. When *type* is **character**, the mode is character at a time. Permission is requested from the remote host before entering the requested mode, and if the remote host supports it, the new mode is entered.

open *host [port]*
Opens a connection to the specified **host**. The host specification can be either a host name or an Internet address in dotted decimal form. If no *port* is given, the **telnet** command attempts to contact a TELNET server at the default port.

quit
Closes a TELNET connection and exists **telnet**. An **END-OF-FILE** in command mode also closes the connection and exits.

send *arguments*
Sends one or more arguments (special character sequences) to the remote host. Multiple arguments are separated by spaces.

The following arguments can be used:

?	- Prints help information for the **send** command.
ao	- Sends the TELNET AO (Abort Output) sequence, which causes the remote host to flush all output from the remote system to the local terminal.
ayt	- Sends the TELNET AYT (Are You There) sequence, to which the remote system can respond.
brk	- Sends the TELNET BRK (Break) sequence.
ec	- Sends the TELNET EC (Erase Character) sequence, which causes the remote host to erase the last character entered.
el	- Sends the TELNET EL (Erase Line) sequence, which causes the remote system to erase the line currently being entered.
escape	- Sends the current **telnet** escape character (**Ctrl-T** by default).
ga	- Sends the TELNET GA (Go Ahead) sequence.
ip	- Sends the TELNET IP (Interrupt Process) sequence, which causes the remote system to cancel the currently running process.
nop	- Sends the TELNET NOP (No Operation) sequence.
synch	- Sends the TELNET SYNC

set *variable value*
Sets a TELNET *variable* to the specified value. The special value **off** cancels the function associated with the variable name entered. The **display** command can be used to query the current setting of each variable. The variables that can be specified are:

echo	- Toggles between local echo of entered characters and suppressing local echo. Local echo is used for normal processing, while suppressing the echo is used for entering text that should not appear on the display, such as passwords. This variable can only be used in line-by-line mode.

EOF - Defines the **END-OF-FILE** character for **telnet**. When **telnet** is in line-by-line mode, entering the **EOF** character as the first character on a line sends the character to the remote host. The initial value for the EOF character is the terminal's **EOF** character.

erase - Defines the erase character for **telnet**. When **telnet** is in character-at-a-time mode and **localchars** is true, typing the erase character sends the TELNET EC sequence to the remote host. The initial value for the erase character is the terminal's **erase** character.

escape - Specifies the **telnet** escape character, which puts **telnet** into command mode when connected to a remote host. This character can also be specified in octal in the **TNESC** environment variable.

flushoutput
- Defines the flush character for **telnet**. When **localchars** is true, typing the flushoutput character sends the TELNET AO sequence to the remote host. The initial value for the flush character is **Ctrl-O**. If the remote host is running &ix., **flushoutput**, unlike the other special characters defined by **set**, only works in **localchars** mode since it has **no termio** equivalent.

interrupt - Defines the interrupt character for **telnet**. When **localchars** is true, typing the interrupt character sends the TELNET IP sequence to the remote host. The initial value for the interrupt character is the terminal's **interrupt** character.

kill - Defines the kill character for **telnet**. When **telnet** is in character-at-a-time mode and **localchars** is true, typing the kill character sends the TELNET EL sequence to the remote host. The initial value for the kill character is the terminal's **kill** character.

quit - Defines the quit character for **telnet**. When **localchars** is true, typing the quit character sends the TELNET BRK sequence to the remote host. The initial value for the quit character is the terminal's **quit** character.

status
Shows the status of **telnet**, including the current mode and the currently connected remote host.

toggle *arguments*
Toggles one or more arguments that control how **telnet** responds to events. Possible values are **true** and **false**. Multiple arguments are separated by spaces. The **display** command can be used to query the current setting of each argument.

The following arguments can be used:

? - Displays valid arguments to **toggle**.

autoflush - If **autoflush** and **localchars** are both true, then when the **AO**, **INTR**, and **QUIT** characters are recognized and transformed into TELNET sequences, **telnet** does not display any data on the user's terminal until the remote system acknowledges (with a TELNET **timing mark** option) that it has processed those TELNET sequences. The initial value of **autoflush** is true if the terminal has not done an **stty noflsh** and false if it has.

autosynch	- If **autosynch** and **localchars** are both true, then typing the **INTR** or **QUIT** character sends that character's TELNET sequence, followed by the TELNET SYNC sequence. This procedure causes the remote host to discard all previously typed input until both of the TELNET sequences have been read and acted upon. The initial value of this toggle is false.
crmod	- Toggles carriage return mode. When set to true, most carriage return characters received from the remote host are mapped into a carriage return followed by a line feed. This mode does not affect the characters typed by the user, only those received from the remote host. This mode is useful when the remote host sends only a carriage return and not a line feed. The initial value of this toggle is false.
debug	- Toggles debugging at the socket level. This argument can only be entered by a user with superuser privileges. The initial value of this toggle is false.
localchars	- Determines the handling of TELNET special characters. When this value is true, the **ERASE**, **FLUSH**, **INTERRUPT**, **KILL**, and **QUIT** characters are recognized locally and transformed into the appropriate TELNET control sequences (**EC**, **AO**, **IP**, **BRK**, and **EL**, respectively). The initial value of **localchars** is true in line-by-line mode and false in character-at-a-time mode.
netdata	- Toggles the display of all network data (in hexadecimal format). The data is written to standard output unless a *net_trace_file* is specified with the **-n** flag on the **telnet** command line. The initial value of this toggle is false.
options	- Toggles the display of some internal TELNET processing options. The initial value of this toggle is false.
z	- Suspends this **TELNET** session. If the user's shell does not support job control (assumed to be true if no **stty susp** character has been set), then this command will open a sub-shell on the local host. The shell started is the one specified by the **SHELL** environment variable. When the shell is exited, **telnet** returns to its previous mode.

Examples

1. To log in to **host1** and do terminal type negotiation:

```
$ tn host1
Trying...
Connected to host1.
Escape character is '^T'.


AIX telnet (host1)

login: _
```

2. To log in to **host1** as a vt100 (no terminal type negotiation):

   ```
   $ EMULATE=vt100; export EMULATE
   $ tn host1
   Trying...
   Connected to host1.
   Escape character is '^T'.


   AIX telnet (host1)

   login: _

   $ tn -e vt100 host1
   Trying...
   Connected to host1.
   Escape character is '^T'.


   AIX telnet (host1)

   login: _
   ```

Files

$HOME/.3270keys	Defines user's 3270 keyboard mapping.
/etc/3270.keys	Default 3270 keyboard mapping.

Related Information

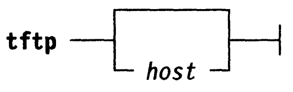
In this book: "telnetd" on page 4-26.
pty device driver, refer to *AIX Operating System Technical Reference*.

tftp

Purpose

Trivial file transfer program.

Syntax



Description

The **tftp** command is the user interface to the Internet TFTP (Trivial File Transfer Protocol), which allows users to transfer files to and from a remote machine. The remote *host* may be specified on the command line, in which case, **tftp** uses *host* as the default host for future transfers (see the **connect** command below).

Note: Different implementations of the **tftp** server impose different user access restrictions on file transfers, for example, **get** and **put**. These restrictions are determined by the remote host. See "tftpd" on page 4-28 for restrictions on connecting to an AIX server for **tftp**.

Commands

Once **tftp** is running, it issues the prompt **tftp>** and recognizes the following commands:

connect *host-name* [*port*]	Set the **host** (and optionally *port*) for transfers. Note that the TFTP protocol, unlike the FTP protocol, does not maintain connections between transfers. Thus, the **connect** command does not actually create a connection, but merely remembers what host is to be used for transfers. You do not have to use the **connect** command. The remote host can be specified as part of the **get** or **put** commands.
mode[ascii|binary]	Set the mode for transfers. Transfer mode may be one of **ascii** or **binary**. The default is **ascii**.
put *filename* **put** *localfile remotefile* **put** *file1 file2...fileN remote-directory*	Put a file or set of files to the specified remote file or directory. The destination can be in one of two forms: a file name on the remote host, if the host has already been specified, or a string of the forms *host:filename* to specify both a host and file name at the same time. If the latter form is used, the hostname specified becomes the default for future transfers. If the remote-directory form is used, the remote host is assumed to be an AIX system.

get *filename*

get *remotename localname*

get *file1 file2...fileN*	Get a file or set of files from the specified sources. Source can be in one of two forms: a file name on the remote host, if the host has already been specified, or a string of the form *host:filename* to specify both a host and file name at the same time. If the latter form is used, the last hostname specified becomes the default for future transfers.
quit	Exit **tftp**. An end of file also exits.
verbose	Toggle verbose mode.
trace	Toggle packet tracing.
status	Show current status.
rexmt *retransmission-timeout*	Set the per packet retransmission timeout in seconds.
timeout *total-transmission-timeout*	Set the total transmission timeout in seconds.
ascii	Shorthand for **mode ascii**
binary	Shorthand for **mode binary**
? [*command-name* ...]	Print help information.

Examples

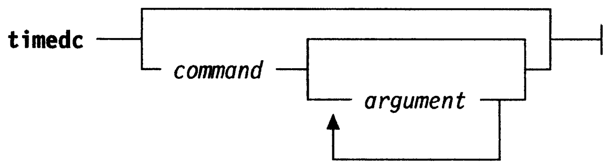
```
$ tftp host1
tftp> ?
Commands may be abbreviated.  Commands are:

connect    connect to remote tftp
mode       set file transfer mode
put        send file
get        receive file
quit       exit tftp
verbose    toggle verbose mode
trace      toggle packet tracing
status     show current status
binary     set mode to octet
ascii      set mode to netascii
rexmt      set per-packet retransmission timeout
timeout    set total retransmission timeout
?          print help information

tftp> quit
$
```

timedc

Purpose

Controls the operation of the **timed** program.

Syntax



Description

The **timedc** command is used to control the operation of the **timed** program. It may be used to:

- measure the differences between machines' clocks
- find the location where the master time server is running
- enable or disable tracing of messages received by **timed**
- perform various debugging actions.

Without any arguments, **timedc** prompts for commands from the standard input. If arguments are supplied, **timedc** interprets the first argument as a command and the remaining arguments as parameters to the command. The standard input may be redirected causing **timedc** to read commands from a file. Commands may be abbreviated. Recognized commands are:

? [*command* ...]	Prints a short description of each command specified in the argument list or if no arguments are given, a list of the recognized commands.
help	Prints a short description of the specified command(s), or if none are named, a list of the recognized commands.
clockdiff *host* ...	Computes the differences between the clock of the host machine and the clocks of the machines given as arguments.
election *host*	Causes the election timers of the time servers on the named hosts to expire. This is used to test the functions which cause a new master to be elected.
msite	Prints the hostname of the current master time server.
quit	Exits from **timedc**.
trace (*on* | *off*)	Enables or disables the tracing of incoming messages to **timed** in the file **/usr/adm/timed.log**.

Examples

To list commands:

```
$ timedc help
```

Commands may be abbreviated. Commands are:

```
clockdiff       help     quit      ?
election        msite    trace
```

To measure a clock difference against 1 other host:

```
$ timedc clockdiff host2
time on host2 is 53 ms. ahead of time on host1
```

To measure clock differences of all sites in the cluster:

```
$timedc clockdiff `ptn`
time on host1 is   ms. ahead of time on host1
time on host2 is 53 ms. ahead of time on host1
time on host3 is 28 ms. behind time on host1
```

To turn timed tracing on:

```
$ timedc trace on
timedc tracing enabled
```

Interactive use:

```
$ timedc
timedc> clockdiff host2
time on host2 is 27 ms. behind time on host1
timedc> trace off
timed tracing disabled
timedc> quit
```

Files

/usr/adm/timed.masterlog	log file for master **timed**.
/usr/adm/timed.log	file that logs messages from **timed** and **timedc**.

Related Information

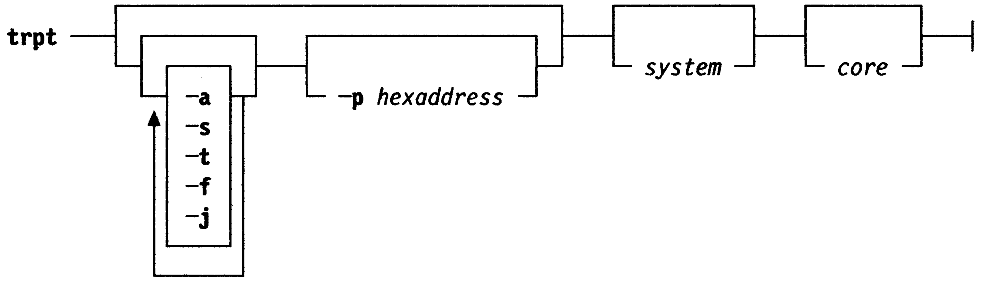
In this book: "timed" on page 4-29
date, refer to the *AIX Operating System Commands Reference*
adjtime, refer to the *AIX Operating System Technical Reference*

trpt

Purpose

Transliterates protocol trace.

Syntax



Description

The **trpt** interrogates the buffer of TCP trace records created when a socket is marked for debugging (see **setsockopt** in the *AIX Operating System Technical Reference.*) and prints a readable description of these records. When no options are supplied, **trpt** prints all the trace records found in the system grouped according to TCP connection protocol control block (PCF). The following options may be used to alter this behavior.

Flags

-a	Prints the values of the source and destination addresses for each packet recorded, in addition to the normal output.
-s	Prints a detailed description of the packet sequencing information, in addition to the normal output.
-t	Prints the values for all timers at each point in the trace.
-f	Follows the trace as it occurs, waiting a short time for additional records each time the end of the log is reached.
-j	Gives a list of the protocol control block addresses for which there are trace records.
-p	Shows only trace records associated with the protocol control block, the address of which follows.

The recommended use of the **trpt** is as follows:

- Isolates the problem and enables debugging on the socket(s) involved in the connection.
- Finds the address of the protocol control blocks associated with the sockets using the **netstat -a** option. For more information, see "netstat" on page 3-32.

- Run the **trpt** command with the **-p** option, supplying the associated protocol control block addresses. The **-f** option can be used to follow the trace log once the trace is located. If there are many sockets using the debugging option, the **-j** option may be useful in checking to see if any trace is located. If there are many sockets using the debugging option, the **-j** option may be useful in checking to see if any trace records are present for the socket in question.

If debugging is being performed on a system or core file other than the default, the last two arguments may be used to supplant the defaults.

Messages

no namelist

Explanation: When the system image doesn't contain the proper symbols to find the trace buffer.

Files

/unix
/dev/kmem

Related Information

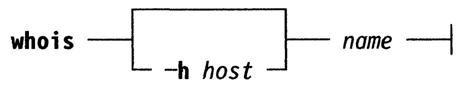
In this book: "netstat" on page 3-32
setsockopt, refer to the *AIX Operating System Technical Reference.*
trsp, refer to the *AIX Operating System Commands Reference*

whois

Purpose

DARPA Internet user name directory service.

Syntax



Description

Whois help produces a message similar to the following:

```
Please enter a name or a handle ("ident") such as Smith or
SRI-NIC.  Starting with a period forces a name only search; starting
with exclamation point forces handle only.  Examples:

Smith          [  looks for name or handle Smith  ]
!SRI-NIC       [  looks for handle SRI-NIC only ]
.Smith, John   [  looks for name JOHN SMITH only ]
```

Adding " . . ." to the argument will match anything from that point, such as "ZU" will match ZUL, ZUM, etc.

To have the entire membership list of a group or organization, if you are asking about a group or organization, shown with the record, use an asterisk character (*) directly preceding the given argument. If there are a lot of members this will take a long time. You may, of course, use an exclamation point and asterisk or a period and asterisk together.

Note: **whois** is only useful with an Internet directory service. AIX is not currently distributed with an Internet directory service.

Flags

-h *host*	Connects to the specified host to find an Internet Directory Service. If no host is specified, the host sri-nic.arpa is assumed.

Chapter 4. Server Commands

CONTENTS

About This Chapter

This chapter describes the server commands that are part of the TCP/IP program for the AIX Operating System. Server commands provide support for user commands (see Chapter 3, "User Commands"). Server commands run either at system start by entries in the **/etc/rc.tcpip** file or as needed by the **inetd** command (see "inetd" on page 4-6).

The server commands are listed alphabetically within the chapter.

fingerd

Purpose

Provides the server function for the **finger** command.

Syntax

```
/etc/fingerd ──┤
```

Description

The **fingerd** command is a simple protocol based on *Name/Finger*, RFC742 (see "Related Publications" on page iv for additional information) that provides an interface to the **finger** program at several network sites. The program is supposed to return an easily readable status report. There is no required format and the protocol consists mostly of specifying a single command line.

The **fingerd** command listens for **tcp** requests at port 79. Once connected, it reads a single command line terminated by a **<CR> <LF>** which is passed to the **finger** command. The **fingerd** command closes its connections as soon as the output is completed.

If the line is null (just a **<CR> <LF>** is sent), the **finger** command returns a default report that lists all users logged in to the system at that moment.

If a user name is specified (**eric<CR> <LF>**), the response given contains more extensive information for only that particular user, whether logged in or not. Allowable names on the command line include both login names and user names. If a name is not clearly defined, all possible derivations are returned.

Related Information

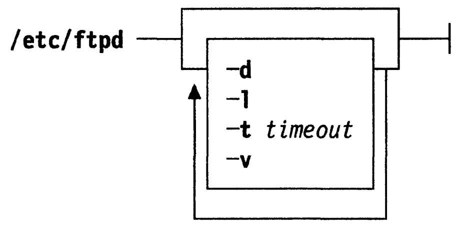
In this book: "finger" on page 3-4

ftpd

Purpose

Provides the server function for the FTP protocol.

Syntax



Description

The **ftpd** command is the DARPA Internet File Transfer Protocol server process. The server uses the TCP protocol and listens at the port specified in the FTP service specification (see "services" on page 5-19). The **ftpd** command is usually started by **inetd**, a daemon, which listens on the **ftp** port for requests.

Flags

The **ftpd** command options are:

-d	Writes debugging information to the syslog.
-l	Logs each FTP session in the syslog.
-t	Sets the inactivity timeout period to **timeout**
-v	Equivalent to **-d**.

The FTP server timeouts an inactive session after 15 minutes. If the **-t** option is specified, the inactivity timeout period is set to **timeout**.

The FTP server currently supports the following FTP requests. Case is not distinguished.

Request	Description
ABOR	Abort previous command
ACCT	Specify account (ignored)
ALLO	Allocate storage (vacuously)
APPE	Append to a file
CDUP	Change to parent of current working directory
CWD	Change working directory
DELE	Delete a file
HELP	Give help information
LIST	Give list files in a directory (**ls -lg**)
MKD	Make a directory
MODE	Specify data transfer **mode**
NLST	Give name list of files in directory (**ls**)
NOOP	Do nothing
PASS	Specify password

PASV	Prepare for server-to-server transfer
PORT	Specify data connection port
PWD	Print the current working directory
QUIT	Terminate session
RETR	Retrieve a file
RMD	Remove a directory
RNFR	Specify rename-from file name
RNTO	Specify rename-to file name
STOR	Store a file
STOU	Store a file with a unique name
STRU	Specify data transfer **structure**
TYPE	Specify data transfer **type**
USER	Specify user name
XCUP	Change to parent of current working directory
XCWD	Change working directory
XMKD	Make a directory
XPWD	Print the current working directory
XRMD	Remove a directory

The FTP server stops an active file transfer only when the **abor** command is preceded by a TELNET Interrupt Process (IP) signal and a TELNET SYNCH signal in the command TELNET stream, as described in J. Postel and J. Reynolds etal *File Transfer Protocol* (see "Related Publications" on page iv).

This command interprets file names according to the globbing conventions used by the C shell. This allows users to utilize the metacharacters * ? [{ }].

The **ftpd** command verification users according to the following rules:

1. The user name must be in the password data base, **/etc/passwd**, and not have a null password. In this case, a password must be provided by the client before any file operations may be performed.
2. The user name must not appear in the file **/etc/ftpusers**.

   Note: This file must be created by the system administrator.
3. The user must have a standard shell.
4. If the user name is anonymous or FTP, an anonymous FTP account must be present in the password file (user FTP). In this case, the user is allowed to log in by specifying any password (by convention this is given as the client host's name).

In the last case, **ftpd** takes special measures to restrict the client's access privileges. The server performs a **chroot** system call, see *AIX Operating System Technical Reference* for additional information, to the home directory of the FTP user. In order that system security is not breached, it is recommended that the FTP subtree be constructed with care. The following rules are recommended.

ftp	Make the home directory owned by FTP and unwritable by anyone.
ftp/bin	Make this directory owned by the superuser and unwritable by anyone. The program **ls**, refer to *AIX Operating System Command Reference*, must be present to support the list commands. This program should have mode 111.

ftp/etc	Make this directory owned by the superuser and unwritable by anyone. The files **passwd** and **group** (see *AIX Operating System Technical Reference*) must be present for the **ls** command to work properly. These files should be mode 444.
ftp/pub	Make this directory mode 777 and owned by FTP. Users should then place files, which are to be accessible via the anonymous account, in this directory.

Related Information

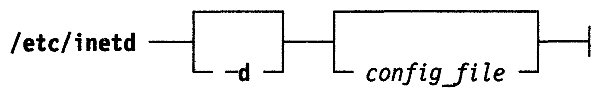
In this book: "ftp" on page 3-6

inetd

Purpose

Provides **inetd** socket management.

Syntax



Description

The **inetd** command should be run at boot time by **/etc/rc.tcpip**. It then listens for connections on certain internet sockets. When a connection is found on one of its sockets, it decides what service the socket corresponds to and invokes a program to service the request. After the program is finished, it continues to listen on the socket. Essentially, **inetd** is a master daemon which invokes service daemons as needed, thus reducing the system load.

There must be an entry for each field of the configuration file, with entries for each field separated by a tab or a space. Comments are denoted by a "#" at the beginning of a line. The fields of the configuration file are as follows:

service name
socket type
protocol
wait/nowait
user
server program
server program arguments

The **service name** entry is the name of the valid service in the file **/etc/services/**. For internal services (discussed below), the service name *must* be the official name of the service (that is, the first entry in **/etc/services**).

The **socket type** should be of *stream*, *dgram*, *raw*, *rdm*, or *seqpacket* depending on whether the socket is a stream, datagram, raw, reliably delivered message, or sequenced packet socket.

The **protocol** must be a valid protocol as given in **/etc/protocols**. Examples might be *tcp* or *udp*.

The **wait/nowait** entry is applicable to datagram sockets only (other sockets should have a *nowait* entry in this space). If a datagram server connects to its peer, freeing the socket so **inetd** can receive further messages on the socket, it is said to be *multi-threaded* server, and should use the *nowait* entry. For datagram servers which process all incoming datagrams on a socket and eventually time out, the server is said to be *single-threaded* and should use the *wait* entry. *Comsat* (*biff*) and *talk* are both examples of the latter type of datagram server. **Tftpd** is an exception; it is a datagram server that establishes pseudo-connections. It must be listed as *wait* in order to avoid a race; the server reads the first packet, creates a new socket, and then forks and exits to allow **inetd** to check for new service requests to spawn new servers.

The **user** entry should contain the user name of the user as whom the server should run. This allows for servers to be given less permission than root. The **server program** entry should contain the path name of the program which is to be executed by **inetd** when a request is found on its socket. If **inetd** provides this service internally, this entry should be *internal.*

The arguments to the server program should be just as they normally are, starting with **argv[0]**, which is the name of the program. If the service is provided internally, the work *internal* should take the place of this entry.

The **inetd** command provides several trivial services internally by use of routines within itself. These services are echo, discard, chargen (character generator), daytime (human readable time), and time (machine readable time, in the form of the number of seconds since midnight, January 1, 1900). All of these services are **TCP** based.

The **inetd** command rereads its configuration file when it receives a hangup signal, SIGHUP. Services may be added, deleted or modified when the configuration file is reread.

Flags

-d Causes debugging information to be printed, in a single self-explanatory format, on inetd's standard error output each time the daemon services a request from the network.

Related Information

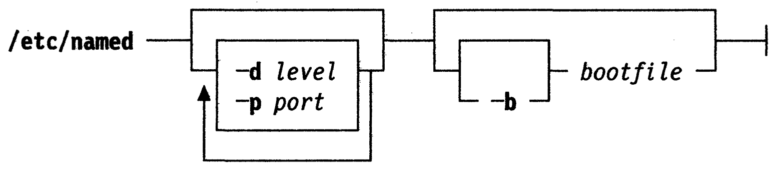
In this book:
"inetd.conf" on page 5-7
"ftpd" on page 4-3
"lpd" on page 4-11
"rexecd" on page 4-12
"telnetd" on page 4-26
"tftpd" on page 4-28
"talkd" on page 4-24
"rshd" on page 4-19
"rlogind" on page 4-14
"fingerd" on page 4-2

named

Purpose

Provides the server function for the Domain Name Protocol.

Syntax



Description

The **named** command is the Internet domain name server (refer to *Domain Names - Implementation and Specification*, RFC883 for more details, see "Related Publications" on page iv for information on obtaining a copy). Without any arguments, the **named** command reads the default boot file **/etc/named.boot**, reads any initial data and listens for queries.

Flags

The **named** command options are:

-b	Specifies that the next argument is the name of the boot file. If the boot file name is the last argument on the command line, the **-b** may be omitted.
-d	Prints debugging information. A number after the **-d** determines the level of messages printed.
-p	Uses a different port number. The default is the standard port number as listed in **/etc/services** (see "rc.tcpip, rc.tcpip.local" on page 5-15).

Examples

Any additional argument is taken as the name of the boot file. The boot file contains information about where the name server is to get its initial data. The following is a short example:

```
;
;        boot file for name server
;
; type      domain          source file or host
;
domain      server1
primary     server1         named.db
secondary   cc.server1      10.2.0.78 128.32.0.10
cache                       named.ca
```

- The first line specifies that serverl is the domain for which the server is authoritative.
- The second line states that the file `named.db` contains authoritative data for the domain serverl. The file `named.db` contains data in the master file format described in RFC883 except that all domain names are relative to the origin. In this case, serverl (see below for a more detailed description).
- The third line specifies that all authoritative data under `cc.serverl` is to be transferred from the name server at 10.2.0.78. If the transfer fails, it tries 128.32.0.10 and continues trying the addresses, up to 10, listed on this line. The secondary copy is also authoritative for the specified domain.
- The fourth line specifies data in `named.ca` is to be placed in the cache (well known data such as locations of root domain servers). The file **named.ca** is in the same format as `named.db`.

The master file, `named.db` in the example above, consists of entries such as:

```
$INCLUDE <filename>
$ORIGIN <domain>
<domain> <opt_ttl> <opt_class> <type> <resource_record_data>
```

where domain is . for root, @ for the current origin, or a standard domain name. If domain is a standard domain name that does not end with ., the current origin is appended to the domain. Domain names ending with . are unmodified. The **opt_ttl** field is an optional integer number for the time-to-live field. It defaults to zero. The **opt_class** field is the object address type; currently only one type is supported, *IN*, for objects connected to the DARPA Internet. The **type** field is one of the following tokens; the data expected in the **resource_record_data** field is in parentheses.

A	A host address (dotted quad)
NS	An authoritative name server (domain)
MX	A mail exchanger (domain)
CNAME	The canonical name for an alias (domain)
SOA	Marks the start of a zone of authority (5 numbers (see RFC883))
MB	A mailbox domain name (domain)
MG	A mail group member (domain)
MR	A mail rename domain name (domain)
NULL	A null resource record (no format or data)
WKS	A well know service description (not implemented yet)
PTR	A domain name pointer (domain)
HINFO	Host information (cpu_type OS_type)
MINFO	Mailbox or mail list information (request_domain error_domain)

Note: The following signals have the specified effect when sent to the server process using the **kill** command.

SIGHUP	Causes server to read **/etc/named.boot** and reload database.
SIGINT	Dumps current data base and cache to **/tmp/named_dump.db**
SIGUSR1	Turns on debugging; each SIGUSR1 increments debug level.
SIGUSR2	Turns off debugging completely.

Files

/etc/named.boot	Name server configuration boot file
/etc/named.pid	The process id
/tmp/named.run	Debug output
/tmp/named_dump.db	Dump of the name servers database

Related Information

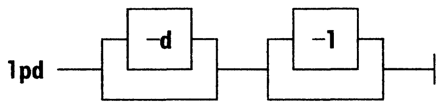
In this book: "resolv.conf" on page 5-17
kill, **gethostbyname**, refer to *AIX Operating System Technical Reference*
signal, refer to *AIX Operating System Technical Reference*

lpd

Purpose

Provides the server function for the **qdaemon** command.

Syntax



Description

The **lpd** daemon allows AIX systems to accept print jobs from remote computers that use the **lpr/lpd** printing system. If a remote system using **lpd** is configured to route print jobs to an AIX system, the **lpd** daemon can receive print jobs from the remote system and route them to **qdaemon** for printing. The AIX **lpd** is not intended for use with other AIX systems or within a single system or cluster. See the **print** and **qdaemon** pages in the *AIX Operating System Commands Reference* for information on printing within a single system or between two AIX system.

Flags

The **lpd** daemon options are:

-d	Turns on debugging. Debugging information will be written to **/tmp/lpd.out**.
-l	Causes **lpd** to log information using **syslog**.

Related Information

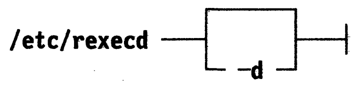
print, **qdaemon** and **syslogd**, refer to *AIX Operating System Commands Reference*
syslog, refer to *AIX Operating System Technical Reference*

rexecd

Purpose

Provides the server function for the **rexec** command.

Syntax



Description

The **rexecd** command is the server for the **rexec** routine. The server provides remote execution facilities with authentication based on user names and passwords.

The **rexecd** command listens for service requests at the port indicated in the service specification; see "services" on page 5-19. When a service request is received the following protocol is initiated:

1. The server reads characters from the socket up to a null **(0)** byte. The resultant string is interpreted as an ASCII number, base 10.
2. If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine.
3. A null terminated user name of at most 16 characters is retrieved on the initial socket.
4. A null terminated, unencrypted password of at most 16 characters is retrieved on the initial socket.
5. A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
6. The **rexecd** command then validates the user, as is done at login time and, if the verification is successful, changes to the user's home directory, and establishes the user and group protections of the user. If any of these steps fail, the connection is broken with a diagnostic message returned.
7. A null byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by the **rexecd** command.

Note: The verification procedure used assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an open environment.

Messages

All diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 7 above upon successful completion of all the steps prior to the command execution).

Flags

-d Sends debugging messages to the **syslogd**.

username too long

Explanation: The name is longer than 16 characters.

password too long

Explanation: The password is longer than 16 characters.

command too long

Explanation: The command line passed exceeds the size of the argument list (as configured into the system).

Login incorrect

Explanation: No password file entry for the user name existed.

Password incorrect

Explanation: The wrong password was supplied.

No remote directory

Explanation: The **chdir** command to the home directory failed.

Try again

Explanation: A **fork** by the server failed.

Related Information

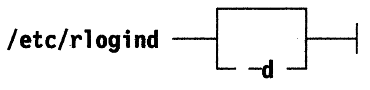
In this book:
"rexec" on page 3-51
"services" on page 5-19

rlogind

Purpose

Provides the server function for the **rlogin** command.

Syntax



Description

The **rlogind** command is the server for the **rlogin** program. The server provides a remote login facility with verification based on privileged port numbers from trusted hosts.

This command listens for service requests at the port indicated in the login service specification, see "rc.tcpip, rc.tcpip.local" on page 5-15. When a service request is received, the following protocol is initiated:

1. The server checks the client's source port. If the port is not in the range 0-1023, the server breaks the connection.
2. The server checks the client's source address and requests the corresponding host name (refer to **gethostbyaddr**, in the *AIX Operating System Technical Reference*, "hosts" on page 5-4 and "named" on page 4-8). If the hostname cannot be determined, the dot-notation representation of the host address is used.

Once the source port and address have been checked, the **rlogind** command allocates a pseudo-terminal (refer to **pty** in the *AIX Operating System Technical Reference*) and manipulates file descriptors so the slave half of the pseudo-terminal becomes the **stdin**, **stdout**, and **stderr** for a login process. The login process is an instance of the login program. The login process then proceeds with the verification process as described in "rshd" on page 4-19, but if automatic authentication fails, it re-prompts the user to log in as one finds on a standard terminal line.

The parent of the login process manipulates the master side of the pseduo-terminal, operating as an intermediary between the login process and the client instance of the **rlogin** program. In normal operation, the packet protocol described in **pty** is invoked to provide **Ctrl S/Ctrl Q** type facilities and propagate interrupt signals to the remote programs. The login process propagates the client terminal's baud rate and terminal type, as found in the environment variable, **TERM**. The screen or window size of the terminal is requested from the client, and window size changes from the client are propagated to the pseudo-terminal.

Note: The verification procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an open environment.

Flags

-d	Sends debugging messages to the **syslogd.**

Messages

All diagnostic messages are returned on the connection associated with the **stderr**, after which any network connections are closed. An error is indicated by a leading byte with a value of 1.

Related Information

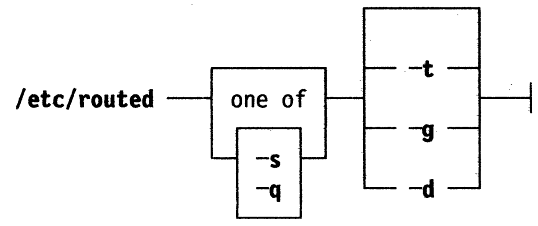
In this book: "rlogin" on page 3-54
login, refer to *AIX Operating System Command Reference*
pty, refer to *AIX Operating System Technical Reference*
syslogd, refer to *AIX Operating System Technical Reference*

routed

Purpose

Manages network routing tables.

Syntax



Description

The **routed** command is invoked at boot time to manage the network routing tables. This routing daemon uses a variant of the Xerox NS Routing Information Protocol (RIP) to maintain current kernel routing table entries. It uses a generalized protocol capable of use with multiple address types, but is currently used only for Internet routing within a cluster of networks.

In normal operation **routed** listens on the UDP socket for the **route** service (see "rc.tcpip, rc.tcpip.local" on page 5-15) for routing information packets. If the host is an internetwork router, it periodically supplies copies of its routing tables to any directly connected hosts and networks.

When **routed** is started, it uses the SIOCGIFCONF **ioctl** to find those directly connected interfaces defined in the system configuration and marked up (the software loopback interface is ignored). If multiple interfaces are present, it is assumed that the host forwards packets between networks. The **routed** command then transmits a request packet on each interface (using a broadcast packet if the interface supports it) and enters a loop, listening for request and response packets from other hosts.

When a request packet is received, the **routed** command formulates a reply based on the information maintained in its internal tables. The response packet generated contains a list of known routes, each marked with a hop count metric (a count of 16, or greater, is considered infinite). The metric associated with each route returned provides a metric relative to the sender.

Defined in *Internet Transport Protocols*, XSIS 028112, Xerox System Integration Standard.

Response packets received by the **routed** command are used to update the routing tables if one of the following conditions is satisfied:

1. No routing table entry exists for the destination network or host, and the metric indicates the destination is reachable (the hop count is not infinite).
2. The source host of the packet is the same as the router in the existing routing table entry. That is, updated information is being received from the very internetwork router through which packets for the destination are being routed.
3. The existing entry in the routing table has not been updated for some time (defined to be 90 seconds), and the route is at least as cost effective as the current route.
4. The new route describes a shorter route to the destination than the one currently stored in the routing tables. The metric of the new route is compared against the one stored in the table.

When an update is applied, the **routed** command records the change in its internal tables and updates the kernel routing table. The change is reflected in the next response packet sent.

In addition to processing incoming packets, the **routed** command periodically checks the routing table entries. If an entry has not been updated for 3 minutes, the entry's metric is set to infinity and marked for deletion. Deletions are delayed an additional 60 seconds to insure the invalidation is propagated throughout the local internet.

Hosts acting as internetwork routers automatically supply their routing tables every 30 seconds to all directly connected hosts and networks. The response is sent to the broadcast address on nets capable of that function, to the destination address on point-to-point links, and to the router's own address on other networks. The normal routing tables are bypassed when sending automatic responses. The reception of responses on each network is used to determine that the network and interface are functioning correctly. If no response is received on an interface, another route may be chosen to route around the interface, or the route may be dropped if no alternative is available.

In addition to the facilities described above, **routed** supports the notion of distant **passive** and **active** gateways. When **routed** is started up, it reads the file **/etc/gateways** to find gateways which may not be located using only information from the 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 (such as, they should have a **routed** process 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 the same as network interfaces. Routing information is distributed to the gateway and if no routing information is received for a period of the time, the associated route is deleted. External gateways are also passive, but are not placed in the kernel routing table nor are they included in routing updates. The function of external entries is to inform **routed** that another routing process will install such a route, and that alternate routes to that destination should not be installed. Such entries are only required when both routers may learn of routes to the same destination.

Internetwork routers that are directly attached to the ARPANET should use the External Gateway Protocol (EGP) to gather routing information rather than using a stationary routing table of passive gateways. EGP is required in order to provide routes for local networks to the rest of the Internet system.

Flags

The **routed** command options are:

-d	Enables additional debugging information to be logged, such as bad packets received.
-g	Offers a route to the default destination. This is typically used on a gateway to the Internet, or on a gateway that uses another routing protocol whose routes are not reported to other local routers.
-s	Forces the **routed** command to supply routing information whether it is acting as an internetwork router or not. This is the default if multiple network interfaces are present, or if a point-to-point link is in use.
-q	Never supplies routing information (opposite to the **-s** option).
-t	Prints all packets sent or received on the standard output. In addition, the **routed** command does not separate itself from the controlling terminal, so interrupts from the keyboard kills the process.

Any other argument supplied is interpreted as the name of file in which **routed**'s actions should be logged. This log contains information about any changes to the routing tables and, if not tracing all packets, a history of recent messages sent and received which are related to the changed route.

Files

/etc/gateways	Routes through distant and external gateways.
/etc/networks	Contains the network name data base.

Related Information

Internet Transport Protocols, XSIS 028112, Xerox System Integration Standard.

In this book: "route" on page 3-56

rshd

Purpose

Provides the server function for remote command execution.

Syntax

```
/etc/rshd ────┤
```

Description

The **rshd** command is the server for the **rcmd** routine and, consequently, for the **rsh** program. The server provides remote execution facilities with verification based on privileged port numbers from trusted hosts.

The **rshd** command listens for service requests at the port indicated in the **cmd** service specification (see "services" on page 5-19). When a service request is received, the following protocol is initiated:

1. The server checks the client's source port. If the port is not in the range 0-1023, the server breaks the connection.
2. The server reads characters from the socket up to a null (\0) byte. The resultant string is interpreted as an ASCII number, base 10.
3. If the number received in step 1 is non-zero, it is interpreted as the port number of a secondary stream to be used for the **stderr**. A second connection is then created to the specified port on the client's machine. The source port of this second connection is also in the range 0-1023.
4. The server checks the client's source address and requests the corresponding host name (refer to **gethostbyaddr** in the *AIX Operating System Technical Reference*, "hosts" on page 5-4 and "named" on page 4-8). If the hostname cannot be determined, the dot-notation representation of the host address is used.
5. A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as the user identity on the client's machine.
6. A null terminated user name of at most 16 characters is retrieved on the initial socket. This user name is interpreted as a user identity to use on the server's machine.
7. A null terminated command to be passed to a shell is retrieved on the initial socket. The length of the command is limited by the upper bound on the size of the system's argument list.
8. The **rshd** command then validates the user according to the following steps.
   - The local (server-end) user name is looked up in the password file and a **chdir** is performed to the user's home directory. If the lookup fails, the connection is terminated.

- If the user is not the superuser, (user id 0), the file **/etc/hosts.equiv** is consulted for a list of hosts considered equivalent. If the client's host name is present in this file, the verification is considered successful. If the lookup fails, or the user is the superuser, then the file **.rhosts** in the home directory of the remote user is checked for the machine name and identity of the user on the client's machine.
- If this lookup fails, the connection is terminated.

9. A null byte is returned on the initial socket and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by **rshd**.

Note: The verification procedure used here assumes the integrity of each client machine and the connecting medium. This is insecure, but is useful in an open environment.

Messages

All diagnostic messages are returned on the initial socket, after which any network connections are closed. An error is indicated by a leading byte with a value of 1 (0 is returned in step 9 above upon successful completion of all the steps prior to the execution of the login shell).

locuser too long

Explanation: The name of the user on the client's machine is longer than 16 characters.

remuser too long

Explanation: The name of the user on the remote machine is longer than 16 characters.

command too long

Explanation: The command line passed exceeds the size of the argument list (as configured into the system).

login incorrect

Explanation: No password file entry for the user name existed.

no remote directory

Explanation: The **chdir** command to the home directory failed.

permission denied

Explanation: The verification procedure described above failed.

can't make pipe

Explanation: The pipe needed for the **stderr** wasn't created.

Try again

Explanation: A **fork** by the server failed.

Files

/etc/hosts.equiv
/etc/services
$HOME/.rhost

Related Information

In this book: "rsh, remsh" on page 3-58
gethostbyaddr, refer to *AIX Operating System Technical Reference*

rwhod

Purpose

Provides the server function which maintains the database used by the **rwho** and **ruptime** programs.

Syntax

```
/etc/rwhod ──┤
```

Description

The **rwhod** command is the server which maintains the database used by the **rwho** and **ruptime** programs. Its operation is predicated on the ability to broadcast messages on a network.

The **rwhod** command operates as both a producer and consumer of status information. As a producer of information it periodically queries the state of the system and constructs status messages which are broadcast on a network. As a consumer of information, it listens for other **rwhod** servers' status messages, validating them, then recording them in a collection of files located in the directory **/usr/spool/rwho**.

The server transmits and receives messages at the port indicated in the **rwho** service specification (see "rc.tcpip, rc.tcpip.local" on page 5-15). The messages sent and received, are of the form:

```
struct    outmp {
      char out_line[8];             /* tty name */
      char out_name[8];             /* user id */
      long out_time;        /* time on */
};

struct    whod {
      char wd_vers;
      char wd_type;
      char wd_fill[2];
      int  wd_sendtime;
      int  wd_recvtime;
      char wd_hostname[32];
      int  wd_loadav[3];
      int  wd_boottime;
     struct    whoent {
           struct      outmp we_utmp;
           int   we_idle;
      } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order prior to transmission. The load averages are as calculated and represent load averages over the 5, 10, and 15 minute intervals prior to a server's transmission; they are multiplied by 100 for representation in an integer. The host name included is that returned by the **gethostname** (see *AIX Operating System Technical Reference*) system call, with any trailing domain name omitted. The array at the end of the message contains information about the users logged in to the sending machine. This information includes the contents of the **utmp** (see *AIX Operating System Technical Reference*) entry for each non-idle terminal line and a value indicating the time in seconds since a character was last received on the terminal line.

Messages received by the **rwho** server are discarded unless they originated at an **rwho** server's port. In addition, if the host's name, as specified in the message, contains any unprintable ASCII characters, the message is discarded. Valid messages received by **rwhod** are placed in files named **whod** hostname in the directory **/usr/spool/rwho**. These files contain only the most recent message in the format described above.

Status messages are generated approximately once every 3 minutes. The **rwhod** command performs an **nlist** (see *AIX Operating System Technical Reference*) on **/unix** every 5 minutes to guard against the possibility that this file is not the system image currently operating.

Related Information

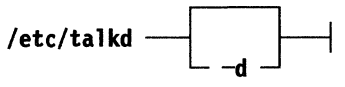
In this book:
"rwho" on page 3-62
"ruptime" on page 3-60

talkd

Purpose

Provides the server function for the **talk** command.

Syntax



Description

The **talkd** command is the server that notifies a user (the ***callee***) that someone else (the ***caller***) wants to initiate a conversation and sets up the conversation if the callee accepts the invitation. The caller initiates the conversation by executing the **talk** command specifying the callee. The callee accepts the invitation by executing the **talk** command specifying the caller.

The **talkd** command listens at the socket defined in the **/etc/services** file with an entry beginning with `ntalk`. This server is normally started by the **inetd** server. When **talkd** receives a LOOK_UP request from a local or remote **talk** process, **talkd** scans its internal invitation table for an entry matching the client with a caller.

If such a table entry exists, the client is the callee and **talkd** returns the appropriate rendezvous address to the **talk** process for the callee. The callee process then establishes a stream connection with the caller process.

If no such entry exists, the client is the caller and the client sends an **ANNOUNCE** request. When **talkd** receives the **ANNOUNCE** request, **talkd** broadcasts an invitation on the console of the foreign host where callee is logged in unless the caller specifies a particular TTY. At approximately 1 minute intervals, **talkd** rebroadcasts the invitation until either the invitation is answered by the callee or the call is canceled by the caller.

Note: **talkd** uses the talk 4.3BSD protocol, which is not compatible with 4.3BSD versions of **talk**.

Flags

-d	Displays debugging information.

Files

/etc/services	Defines Internet socket assignments.
/etc/utmp	Contains data users currently logged in.

Related Information

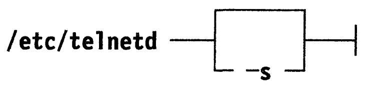
In this book:
"talk" on page 3-64
"inetd" on page 4-6
"rc.tcpip, rc.tcpip.local" on page 5-15

telnetd

Purpose

Provides the server function for the telnet protocol.

Syntax



Description

The **telnetd** command is a server that supports the DARPA standard telnet virtual terminal protocol. **Telnetd** is started by the **inetd** daemon when a request for a **telnet** connection has been received in the **/etc/services** file.

When a TELNET session is started, **telnetd** sends telnet options to the client (remote) host to indicate an ability to do **remote echo** of characters, to **suppress go ahead**, and to receive **terminal-type** information.

If the client host agrees, **telnetd** requests its terminal type. On receipt, **telnetd** checks whether the indicated type is supported on the local system. If not, it again requests a terminal type. This terminal type negotiation continues until the remote client sends an acceptable terminal type or until the client sends the same type twice in a row, indicating that it has no other types available.

The **telnetd** server supports the following telnet options:

- Binary
- Echo/no echo
- Suppress go ahead
- Timing mark.

The **telnetd** command also recognizes the following options for the remote client:

- Binary
- Suppress go ahead
- Terminal type.

Flags

-s	Silent flag. The **-s** flag causes **telnetd** to suppress the banner that is usually provided to remote users. This flag is used whenever banner information, which includes the hostname, should not be made available to remote users.

Files

/etc/inetd.conf	Configures the **inetd** daemon.

Related Information

In this book: "telnet, tn, tn3270" on page 3-66
pty, refer to the *AIX Operating System Technical Reference*

tftpd

Purpose

Provides the server function for the Trivial File Transfer Protocol.

Syntax

```
/etc/tftpd ──┤
```

Description

The **tftpd** command is a server which supports the DARPA Trivial File Transfer Protocol. The **tftp** server operates at the port indicated in the **tftp** service description (see "services" on page 5-19). The server is normally started by **inetd**, (see "inetd" on page 4-6).

The use of **tftp** does not require an account or password on the remote system. Due to the lack of verification information, **tftpd** allows only publicly readable files to be accessed. Files may be written only if they already exist and are publicly writable. Files on the server must be specified with full path names. Since no account information is provided, there is no way to identify a user's home directory. Note that this extends the concept of public to include all users on all hosts that can be reached through the network. This may be appropriate on all systems, and its implications should be considered before enabling **tftp** service. The server should have the user ID with the lowest possible privilege.

Note: If the server site is not running AIX, consult that machine's **tftpd** documentation for information on restrictions for that site.

Related Information

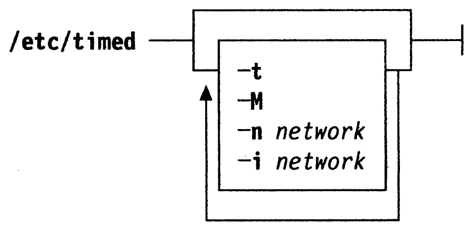
In this book:
"inetd" on page 4-6
"services" on page 5-19
"tftp" on page 3-73

timed

Purpose

Provides network time service.

Syntax



Description

The **timed** server command is the time server daemon and is normally invoked at boot time from the **rc.tcpip** file. It coordinates the host's time with the time of other machines in a local area network running the **timed** program. These time servers slow down the clocks of some machines and speed up the clocks of others to bring them to the average network time. The average network time is computed from measurements of clock differences using the ICMP time stamp request message.

The service provided by the **timed** command is based on a master-slave scheme. When **timed** is started on a machine, it asks the master for the network time and sets the host's clock to that time. After that, it accepts synchronization messages periodically sent by the master and calls **adjtime** (for information on **adjtime**, refer to *AIX Operating System Technical Reference*) to perform the needed corrections on the host's clock.

To select a host as a master **timed** server, the file **/local/timedmaster** must exist on the host. Typing `touch /local/timedmaster` will create the file.

It also communicates with **date** (refer to *AIX Operating System Commands Reference*) in order to set the date globally, and with **timedc**, a timed control program. If the machine running the master crashes, then the slaves elect a new master from among slaves running with the **-M** flag. A **timed** running without the **-M** flag remains a slave. The **-t** flag enables **timed** to trace the messages it receives in the file **/usr/adm/timed.log**. Tracing can be turned on or off by the program **timedc**. **timed** normally checks for a master time server on each network to which it is connected, except as modified by the options described here.

It requests synchronization service from the first master server located. If permitted by the **-M** flag, it provides synchronization service on any attached networks on which no current master server was detected. Such a server propagates the time computed by the top-level master. The **-n** flag, followed by the name of a network which the host is connected to (see "networks" on page 5-13), overrides the default choice of the network addresses created by the program. Each time the **-n** flag appears, that network name is added to a list of valid networks. All other networks are ignored. The **-i** flag, followed by the name of a network to which the host is connected (see "networks" on page 5-13), overrides the default choice of the network addresses made by the program. Each time the **-i** flag appears, that network name is added to a list of networks to ignore. All other networks are used by the time daemon. The **-n** and **-i** flags are meaningless if used together.

Flags

The **timed** command options are:

-i	Overrides the default choice of network addresses created by the program when followed by the name of a network to which the host is connected. Each time the **-i** flag appears, that network name is added to a list of networks to ignore.
-M	Creates a new master from among slaves, if the machine running the master crashes. A **timed** running without the **-M** remains a slave.
-n	Overrides the default choice of the network addresses created by the program when followed by the name of a network which the host is connected.
-t	Enables **timed** to trace the messages it receives in the file **/usr/adm/timed.log**.

Files

/usr/adm/timed.log	tracing file for **timed**
/timedmaster	identifies master **timed** server

Related Information

In this book: "timedc" on page 3-75
date, refer to *AIX Operating System Technical Reference*
adjtime, refer to *AIX Operating System Technical Reference*
gettimeofday, refer to *AIX Operating System Technical Reference*

Chapter 5. File Formats

CONTENTS

About This Chapter

This chapter describes the TCP/IP files you may need to monitor or modify. The files are listed alphabetically within the chapter.

gateways

Purpose

Defines and maintains routing information.

Synopsis

/etc/gateways

Description

The **/etc/gateways** file identifies gateways for the **routed** command. Ordinarily, the **routed** command queries the network, building routing tables from routing information transmitted by other hosts that are directly connected to the network. However, there may be gateways that the **routed** command cannot identify through its queries (***distant*** gateways). Such gateways should be identified in **/etc/gateways**, which **routed** reads when it starts.

The general format of an entry in **/etc/gateways** is:

```
destination name1 gateway name2 metric value type
```

Following is a brief description of each element in a **/etc/gateways** file entry:

destination	A keyword that indicates whether the route is to a network or to a specific host. The two possible keywords are **net** and **host**.
name1	The name associated with destination, name1, can be either a symbolic name (as used in **/etc/hosts** or **/etc/networks**) or an Internet address specified in dotted decimal format.
gateway	Indicator that the following string identifies the gateway host.
name2	Name or address of the gateway host to which messages should be forwarded.
metric	Indicator that the next string represents the hop count to the destination host or network.
value	The hop count.
type	A keyword that indicates whether the gateway should be treated as active or passive. The two possible keywords are **active** and **passive**. An active gateway is treated like a network interface (that is, it is expected to exchange routing information and if it does not do so for a period of time, the route associated with it is deleted). A passive gateway is not expected to exchange routing information. Information about it is maintained in the routing tables indefinitely and is included in any routing information that is transmitted.

Examples

Following are four sample **/etc/gateways** entries:

1. A route to a network, net2, through the gateway, host4. The hop count metric to net2 is 4 and the gateway is treated as passive.

   ```
   net net2 gateway host4 metric 4 passive
   ```

2. A route like the one in the previous example except that it is to a specific host (rather than to a network):

   ```
   host host2 gateway host4 metric 4 passive
   ```

3. A route to a specific host, host10, through the gateway, 192.9.201.5. The hop count metric to host10 is 9 and the gateway is treated as active.

   ```
   host host10 gateway 192.9.201.5 metric 9 active
   ```

4. A route like the one in the previous example except that the gateway is treated as passive (rather than active):

   ```
   host host10 gateway 192.9.201.5 metric 9 passive
   ```

Files

/usr/lpp/tcpip/samples/gateways

Related Information

In this book: "routed" on page 4-16

hosts

Purpose

Contains information regarding the known hosts on the network.

Synopsis

/etc/hosts

Description

The **hosts** file contains information regarding the known hosts on the network. A single line should be present, for each host, including the following information:

internet_address
official_host_name
aliases (optional)

Items are separated by any number of blanks or tab characters. A # indicates the beginning of a comment; characters up to the end of the line are not interpreted by routines which search the file.

When using the nameserver **named**, this file provides a backup when the nameserver is not running. For the nameserver, it is suggested that only a few addresses be filed. These include addresses for the local interfaces that the **ifconfig** command needs at boot time and a few machines on the local network.

This file may be created from the official host data base maintained at the Network Information Control Center (NIC), though local changes may be required to bring it up-to-date regarding unofficial aliases or unknown hosts. As the data base maintained at NIC is incomplete, use of the nameserver is recommended for sites on the DARPA Internet.

Network addresses are specified in the conventional (.) notation using the **inet_addr** routine. Host names may contain any printable character other than a field delimiter, new line, or comment character.

Examples

```
127.0.0.1              localhost
193.255.2.1            fafnir
193.255.2.2            frodo
193.255.2.3            atlas
193.255.2.4            swords
193.255.2.5            hardy
193.255.2.6            sirius
193.255.2.7            coins
193.255.2.8            spica
193.255.2.9            smeagol
193.255.2.10           antares
193.255.10.10          antares.tkr
193.255.2.11           darryl
193.255.2.12           polaris
```

Files

/etc/hosts

Related Information

In this book:
"ifconfig" on page 3-23
"named" on page 4-8
gethostbyname, refer to the *AIX Operating System Technical Reference*
inet_addr, **inet_network**, **inet_ntoa**, **inet_makeaddr**, **inet_lnaof**, **inet_netof**; refer to the *AIX Operating System Technical Reference*

hosts.equiv

Purpose

Defines foreign hosts that are permitted to execute commands.

Synopsis

/etc/hosts.equiv

Description

The **/etc/hosts.equiv** file of a particular host defines from which foreign hosts users other than root can execute commands or login without a password. The format of the **/etc/hosts.equiv** file is a simple list of host names.

Examples

Following are sample entries in an **/etc/hosts.equiv** file:

```
fafnir
frodo
atlas
swords
deneb
sirius
coins
```

Files

/etc/hosts.equiv

Related Information

In this book: "rlogin" on page 3-54

inetd.conf

Purpose

Configures the **inetd** daemon.

Synopsis

/etc/inetd.conf

Description

The **inetd.conf** file is used to configure the **inetd** daemon.

Each new line-delimited entry in the **inetd.conf** file contains seven fields with entries for each field separated by a tab or a space. Comments are denoted by a # at the beginning of a line. There must be an entry for each field. The fields of the configuration file are as follows:

service-name
socket-type
protocol
wait/nowait
user
server-program
server-program-arguments

The **service name** entry is the name of a valid service in the file **/etc/services**. For internal services, discussed below, the service name *must* be the official name of the service (that is, the first entry in **/etc/services**).

The *socket-type* should be one of **stream**, **dgram**, **raw**, **rdm**, or **seqpacket**, depending on whether the socket is a stream, datagram, raw, reliably delivered message, or sequenced packet socket.

The protocol must be a valid protocol as in **/etc/protocols**. Examples might be TCP or UDP.

The **wait/nowait** entry is applicable to datagram sockets only (other sockets should have a nowait entry in this space). If a datagram server connects to its peer, freeing the socket so **inetd** can receive further messages on the socket, it is said to be a multi-threaded server, and should use the **nowait** entry. For datagram servers which process all incoming datagrams on a socket and eventually time out, the server is said to be single-threaded and should use a **wait** entry. **biff** and **talk** are both examples of the latter type of datagram server. **tftpd** is an exception. It is a datagram server that establishes pseudo-connections. It must be listed as **wait** in order to avoid a race. The server reads the packet, creates a new socket, and then forks and exits to allow **inetd** to check for new service requests to spawn new servers.

The **user** entry should contain the name of the user to whom the server should run. This allows for servers to be given less permission than root.

The **server** program entry should contain the path name of the program which is to be executed by **inetd** when a request is found on its socket. If **inetd** provides this service internally, this entry should be internal.

The arguments to the server program should be just as they normally are, starting with argv[0], which is a pointer to the name of the program.

Examples

```
ftp      stream  tcp    nowait   root    /etc/ftpd        ftpd
telnet   stream  tcp    nowait   root    /etc/telnetd     telnetd
shell    stream  tcp    nowait   root    /etc/rshd        rshd
login    stream  tcp    nowait   root    /etc/rlogind     rlogind
exec     stream  tcp    nowait   root    /etc/rexecd      rexecd
smtp     stream  tcp    nowait   root    /usr/lib/sendmail sendmail -bn
tftp      dgram   udp    wait     nobody  /etc/tftpd       tftpd
comsat   dgram   udp    wait     root    /etc/comsat      comsat
talk     dgram   udp    wait     root    /etc/talkd       talkd
ntalk     dgram   udp    wait     root    /etc/talkd       talkd
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
```

Files

/etc/inetd.conf

Related Information

In this book:
"inetd" on page 4-6
"rc.tcpip, rc.tcpip.local" on page 5-15
"protocols" on page 5-14
"tftpd" on page 4-28

net

Purpose

Defines network interface characteristics for TCP/IP for use with the **netconfig** command.

Synopsis

/etc/net

Description

The **/etc/net** file contains the keyword associated with each network interface that TCP/IP can use and a stanza that describes the characteristics for the network interface. There is one stanza entry for each network interface defined for use with TCP/IP.

The format of network interface stanza in **/etc/net/** is:

```
sys_stanza_name:     dstaddr =
                     inetlen =
                     localbroadcast =
                     netaddr =
                     protocol =
                     r_inetlen =
                     subnetmask =
```

The information contained in an adapter stanza is:

sys_stanza_name:
: Starting in column one, the name of the network interface defined in the **/etc/system** file. (The **/etc/system** file is defined using the **devices** command.)

dstaddr
: The IP network address for the destination host of a point-to-point connection, used for serial line interfaces. The address must be specified in dotted decimal notation. This keyword is required for a point-to-point interface.

inetlen
: The maximum IP packet length for transmission to this network interface. The default is the maximum packet length defined by the system. The specified value cannot exceed the default value.

localbroadcast
: An optional keyword application to Token-Ring devices only. Used to indicate if machines on the same Token-Ring Network, but different rings, can communicate. The values can be:

- **true** - Only machines on the same ring can communicate.
- **false** - Machines on same network but different rings can communicate.

If not specified, the default value is **true**.

netaddr The IP network address to be used for this adapter. The address must be specified in dotted decimal notation. This is a required keyword for all interfaces.

protocol A value used to specify whether an 802.3 interface is being described. The value 802.3 indicates an 802.3 interface, while the value standard indicates Ethernet headers. If this keyword is present, its default value is standard.

r_inetlen An optional, maximum IP packet length for remote transmission (transmission outside of the local network). If not specified, default is 576 bytes. When specified, this value cannot exceed the value of **inetlen.**

subnetmask
An optional, hexadecimal mask that defines a sub-network within a local address. A mask is specified only for the local address portion of an Internet address. If **subnetmask** is not set, subnetworks are not used. If subnetworks are to be used, all hosts on the network must have **subnetmask** set.

The following are sample entries in the **/etc/net** file:

1. The following is a stanza for a network adapter, net0, using an IP address of `192.100.10.1`. In this example, which uses a class C IP address, the four high order bits of the local address represent the subnet number, while the four low order bits refer to the local host on the subnetwork. The maximum size of packets transmitted outside the local network is 586 bytes. Any IP datagram can be sent or received through all interfaces for this host. IP headers for this host do not contain the IP security option.

   ```
   net0

        netaddr = 192.100.10.1
        inetlen = 1492
        subnetmask = FFFFFF00
        r_inetlen = 586
   ```

2. The following is a stanza for an IBM Token-Ring Adapter using an IP address of `128.114.100.14`. This interface can communicate with all rings on its Token-Ring network. Because **r_inetlen** is not specified, the maximum size of packets transmitted outside the local network is 576 bytes.

   ```
   tk0

        netaddr = 128.114.100.1
        inetlen = 2010
        localbroadcast = false
   ```

3. Use the following **/etc/net** stanzas to establish a serial line interface on two hosts that are connected by a serial cable. These stanzas establish a point-to-point connection.

   a. On the first host, **engineer**, add the following stanza:

      ```
      tty1:

           netaddr = 192.9.201.3
           dstaddr = 192.9.254.7
      ```

b. On the second host, **market**, add the following stanza:

```
ttyl:

     netaddr = 192.9.254.7
     dstaddr = 192.9.201.3
```

Files

/etc/net	Defines network interfaces for TCP/IP.

Related Information

In this book: "netconfig" on page 3-30

.netrc

Purpose

Contains information used by the **rexec** and the **ftp (xftp)** commands for automatic login.

Synopsis

$HOME/.netrc

Description

The **.netrc** file contains the user ID and password information required for the automatic log in features of the **rexec** and the **xftp** commands. It is a hidden file in the user's home directory, and its permissions must be set to 600 (read and write by owner only).

The format of an entry in **.netrc** is:

```
machine hostname login userid password password
```

where:

hostname is the name of the host on which the user ID exists.

userid is the user ID on that host.

password is the password for that user ID.

```
machine hostl login tom password toms-pass
```

The **/usr/lpp/tcpip/samples/netrc** file is a sample **.netrc** file. Use the following procedure to create a **.netrc** file based on the sample:

1. Copy **/usr/lpp/tcpip/samples/netrc** to your home directory.
2. Edit **netrc** to supply the appropriate *hostname*, *userid*, and *password.*
3. Set the permissions on **netrc** to 600.
4. Rename the file **.netrc** (the initial **.**, or dot, causes the file to be hidden).

```
machine hostl login tom password toms-pass
```

Note: Because your **.netrc** file may contain the root passwords of other systems (in addition to your userid and password), it is especially important to take security measures, such as not leaving your terminal unattended when you are logged in.

Files

$HOME/.netrc

/usr/lpp/tcpip/samples/netrc

Related Information

In this book:
"rexec" on page 3-51
"ftp" on page 3-6

networks

Purpose

Contains information regarding the known networks which comprise the DARPA Internet.

Synopsis

/etc/networks

Description

The **networks** file contains information regarding the known networks which comprise the DARPA Internet. For each network, a single line should be present with the following information:

official-network-name
network-number
aliases (optional)

Items are separated by any number of blanks and tab characters. A # indicates the beginning of a comment. Characters, up to the end of the line, are not interpreted by routines which search the file. This file is normally created from the official network data base maintained at the Network Information Control Center (NIC). Local changes may be required to bring it up-to-date regarding unofficial aliases and unknown networks.

Network number may be specified in dotted decimal (.) notation. Network names may contain any printable character other than a field delimiter, new line, or comment character.

Examples

```
# official net name     net number    alias (if any, eg. by type)
loopback-net            127           software-loopback-net
ne-region               128.1         classb.net1
atl-region              128.2         classb.net2
se-region               128.3         classb.net3
plant1-net1             192.0.1       classc.net0.sub1
plant1-net2             192.0.2       classc.net0.sub2
lab-net                 192.5.1       classc.net5
process-control-test    193.255.99    classc.net255
```

Files

/etc/networks

Related Information

getnetent, refer to *AIX Operating System Technical Reference*
inet, refer to *AIX Operating System Technical Reference*

protocols

Purpose

Contains information regarding the known protocols used in the DARPA Internet.

Synopsis

/etc/protocols

Description

The **protocols** file contains information regarding the known protocols used in the DARPA Internet. For each protocol, a single line should be present with the following information:

official-protocol-name
protocol-number
aliases

Items are separated by any number of blanks or tab characters. A # indicates the beginning of a comment. Characters up to the end of the line are not interpreted by routines which search the file.

Protocol names may contain any printable character other than a field delimiter, new line, or comment character.

Examples

```
ip         0         IP      # internet protocol, pseudo protocol number
icmp       1         ICMP    # internet control message protocol
ggp        3         GGP     # gateway-gateway protocol
tcp        6         TCP     # transmission control protocol
egp        8         EGP     # exterior gateway protocol
udp        17        UDP     # user datagram protocol
```

Files

/etc/protocols

Related Information

getprotoent, refer to *AIX Operating System Technical Reference*

rc.tcpip, rc.tcpip.local

Purpose

Sets up host names and addresses, initializes routes, and starts the TCP/IP daemons.

Synopsis

/etc/rc.tcpip, **/[LOCAL]/rc.tcpip.local**

Description

The **rc.tcpip** file contains the commands necessary to run TCP/IP. The file **rc.tcpip** contains commands that run on every site in a TCF cluster. The file **rc.tcpip.local** is stored in the <LOCAL> file system in a TCF cluster and can be customized by the system administrator to run commands specific to a particular cluster site. The **ifconfig** command defines the interfaces (adapter cards) to be configured. The **hostname** command defines the host name and internet address. The **route** command sets the routing tables statically. The remaining entries execute the TCP/IP daemons.

The daemon, **inetd**, acts as a super daemon for many of the basic TCP/IP services. When requests for those services arrive, **inetd** runs the appropriate daemon. The remaining daemons not supported by **inetd** are:

named
routed
timed
rwhod

These daemons are not started by the **inetd** command because they generally do not run on all hosts.

If TCP/IP is to be initialized when the system is started, the following line must be in the system startup script file:

```
sh /etc/rc.tcpip
```

Files

/etc/rc.tcpip
/[LOCAL]/rc.tcpip.local

Related Information

In this book:
"hostname" on page 3-21
"inetd" on page 4-6
"route" on page 3-56
"Configuring TCP/IP" on page 2-7
"TCP/IP Nameservers" on page 2-9

.rhosts

Purpose

Defines privileged remote hosts and users.

Synopsis

$HOME/.rhosts

Description

Each AIX user may have a **.rhosts** file in his home directory. The contents of this file specify, by hostname and username, the users of remote hosts who are allowed to execute commands through the **rlogin** and **rsh** programs (including **rcp**) on the local machine without supplying a password. For example, if there are two AIX machines with TCP/IP connected through a network, having hostnames, respectively, of **pearl** and **opal**, a user named **joe** on **pearl** could permit a user named **frank** on **opal** to **rlogin** to his (**joe's**) account without supplying **joe's** password by placing a line of the form:

```
opal      frank
```

in the file **.rhosts** file consisting of a hostname followed by a username, separated by a TAB character. Hostnames may be qualified with domain names, e.g. **opal.ibm.avs.com.** Only one hostname and one username may appear on each line. Multiple lines may appear in the file. If root is to have a **.rhosts** file, it should be placed in the root directory. (It is not recommended that root have a **.rhosts** file because this is a serious security risk.)

Files

/usr/lpp/tcpip/samples/rhosts

Related Information

In this book:
"rcp" on page 3-44
"rlogin" on page 3-54
"rsh, remsh" on page 3-58

resolv.conf

Purpose

Contains information that is read by the resolver routines the first time they are invoked by a process.

Synopsis

/etc/resolv.conf

Description

The **resolv.conf** file contains information that is read by the **resolver** routines the first time they are invoked by a process. The file contains a list of name value pairs that provide various types of **resolver** information.

On a normally configured system, this file should not be necessary. The only nameserver to be queried is on the local machine. The domain name is retrieved from the system.

The different configuration options are:

nameserver	followed by the Internet address (in dot notation) of a nameserver that the **resolver** should query. At least one nameserver should be listed. Up to MAXNS (currently 3), nameservers may be listed. In that case, the resolver library queries try them in the order listed. If no nameserver entries are present, the default is to use the nameserver on the local machine. (The algorithm used is to try a nameserver, and if the query times out, try the next, until you run out of nameservers, then repeat trying all the nameservers until a maximum number of reentries are made).
domain	followed by a domain name, that is the default domain to append to names that do not have a dot in them. If no domain entries are present, the domain returned by **gethostname** is used (everything after the first (.). Finally, if the host name does not contain a domain part, the root domain is assumed.

The name value pair must appear on a single line, and the keyword (such as nameserver) must start the line. The value follows the keyword, separated by white space.

Examples

```
domain dan.ibm.com
nameserver 192.100.10.1
```

Files

/etc/resolv.conf

Related Information

In this book: "named" on page 4-8
gethostbyname, refer to *AIX Operating System Technical Reference*
resolver, refer to *AIX Operating System Technical Reference*

services

Purpose

Contains information regarding the known services available in the DARPA Internet.

Synopsis

/etc/services

Description

The **services** file contains information regarding the known services available in the DARPA Internet. For each service, a single line should be present with the following information:

official-service-name
port-number
protocol-name
aliases (optional)

Items are separated by any number of blanks or tab characters. The port number and protocol name are considered a single item. A / is used to separate the port and protocol (such as 512/tcp). A # indicates the beginning of a comment. Characters up to the end of the line are not interpreted by routines which search the file.

Service names may contain any printable character other than a field delimiter, new line, or comment character.

Examples

```
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
qotd          17/tcp     quote
chargen       19/tcp     ttytst source
chargen       19/udp     ttytst source
ftp           21/tcp
telnet        23/tcp
smtp          25/tcp     mail
time          37/tcp     timserver
time          37/udp     timserver
rlp           39/udp     resource       #resource location
name          42/tcp     nameserver
whois         43/tcp     nicname
domain        53/tcp     nameserver     #name-domain server
domain        53/udp     nameserver     #name-domain server
mtp           57/tcp                    #deprecated
hostnames    101/tcp     hostname       #usually from sri-nic
```

Files

/etc/services

Related Information

In this book:
"ifconfig" on page 3-23
"hostname" on page 3-21
"inetd" on page 4-6
getservent, refer to *AIX Operating System Technical Reference*

.3270keys

Purpose

Defines a user keyboard mapping for TELNET (3270).

Synopsis

$HOME/.3270keys

Description

The **.3270keys** file contains your customized key-binding profile, which exists as a hidden file (the name begins with a ., or a dot) in your home directory.

TELNET (3270) looks first in your home directory for **.3270keys**. If one is present, TELNET (3270) uses it to bind keys to functions; otherwise, TELNET (3270) uses the standard profile, **/etc/3270.keys**.Since these two profiles are mutually exclusive, the profile you create must specify all the bindings you want, rather than only those for which you would like an alternate.

To create a **$HOME/.3270keys** file, you may use as models either the **/etc/3270keys.hft** file or the **/etc/3270keys.dflt** file (the former is the default key-binding profile for PS2/HFT terminals and the latter is the default key-binding profile for non-System/370 terminals).

Key-Sequences

A key-sequence is a sequence of characters. It can be delimited optionally by single (') or double quotes ("). If the key-sequence contains white space, carriage returns, or keywords, such as **bind** or **load**, then the key sequence must be delimited by quote marks.

∧ Indicates that the following character is to be converted to a control character.

Note: Converting the following character to a control character is done by masking out all but the low 5 bits of the character, therefore, ∧? (077) and ∧_(0137) both yield 037.

~ Indicates that the following character is to be converted into a character beyond the ASCII set; that is, the resulting character has the eighth bit turned on; for example, C (0103) yields 0303.

\e The escape key (033).

Note: Keep in mind that you cannot map a 3270 function to the **ESC** key alone. You can specify the **ESC** key only in combinations with another key.

\b Blank

\t Tab

\r Carriage return

\n Line feed

Key-Binding Grammar

The following is a BNF description of the 3270-key-binding grammar.

listing → /* empty set */

|*listing statement*

|*listing conditional statement*

statements →
bind *3270-function* key sequence.

|bind *3270-attribute color*

|load *filename*

|print *string*

|quit

|exit

conditions →
string \ condition or condition

conditional statements →
if *condition*

|else

3270 functions →
backspace|backtab|character|clear

|delete|down|dup|enter|eraseof|eraseinput

|fieldmark|home|illegal|insertmode|left

|pa1|pa2||pa3

|penselect

|pf1|pf2|pf3|pf4|pf5

|pf6|pf7|pf8|pf9|pf10

|pf11|pf12|pf13|pf14|pf15 |pf16|pf17|pf18|pf19|pf20
|pf21|pf22|pf23|pf24|pf25 |reset|return|right|sysreq
|tab|up|wordnext|wordprev

3270 attributes →
background

|high_prot|high_unprot

|low_prot|low_unprot

color → black|blue|brown|cyan|green

|magenta|red|white|brightblack

|brightblue|brightbrown|brightcyan

|brightgreen|brightmagenta|brightred

|brightwhite

Statements

bind *3270 function key sequence*
Binds a 3270 terminal function to a sequence of key strokes.

bind *3270 attribute color*
Binds a 3270 field attribute to a color.

load *string* Tells the parser to read in the file name specified by *string* for additional listings. *String* can be delimited optionally by single or double quotes. If the string is identical to a keyword, then *string* must be delimited by quote marks.

print *string*
Prints the string to **stdout**. This command is useful for debugging listings.

exit Unconditionally ends the parse process. No further input is parsed, TELNET (3270) begins remote login.

quit Unconditionally ends the parse process. No further input is parsed, TELNET (3270) begins remote login.

Conditions

Conditions are evaluated by comparing the condition string against the terminal setting. The terminal setting is specified by your environ- ment variable *TERM*. Thus the following listing loads **/etc/3270keys.hft** if *TERM* is set to **hft** or **hft-c**; otherwise, it loads **/etc/3270keys.dft**:

```
if hft or hft-c - load /etc/3270keys.hft
else            - load /etc/3270keys.dflt
```

Conditional Statements

As indicated by the grammar, only one statement may follow an **if** or **else** conditional expression. Also, conditional expressions may not be nested. Due to this limitation of the language, bindings for specific terminals or families of terminals should be stored in separate files.

For example, the following correctly gives either hft or default key bindings:

```
if hft or hft-c
        load /etc/3270keys.hft
else
        load /etc/3270keys.dflt
```

The following, however, will bind functions **pf2** through **pf10** to **escN** regardless of the terminal type:

```
if hft or hft-c
        load /etc/3270keys.hft
else
        bind pf1 "\1"
        bind pf2 "\2"
        .
        .
        .
        bind pf9 "\9"
        bind pf10 "\0"
```

Since it may be desirable to support key bindings for a variety of families of terminals, and since conditional expressions are limited, testing for a default may not be straightforward. The following example, for instance, will not work, since conditional expressions cannot be nested.

```
if hft or hft-c
        load /etc/3270keys.hft
else if hds
        load /etc/3270keys.hds
else if avt
        load /etc/3270keys.avt
else
        load /etc/3270keys.dflt
```

This limitation in the language can be overcome, by using the **exit** or **quit** statement as an unconditional break. The following example loads the correct file for all terminals:

```
if hft or hft-c
        load /etc/3270keys.hft
if hft or hft-c
        quit
if hds
        load /etc/3270keys.hds
if hds
        quit
if avt
        load /etc/3270keys.avt
if avt
        quit

load /etc/3270keys.dflt
```

3270 Functions

Binding a key sequence to an illegal character causes an alarm to sound whenever that sequence is entered.

By default, all standard characters are bound to characters. Binding a key to a character causes that key to be treated as a character. A character should not be bound to a sequence of more than one character.

Penselect is not supported. Binding a key sequence to **penselect** will cause an error message to be printed to **stderr** whenever that key sequence is entered.

3270 Attribute

The following description of 3270 attributes is an informal one, since attributes are used in different ways by different VM applications.

background
background field

high_prot High intensity protected attribute. Attribute of CMS XEDIT status field for status.

high_unprot
High intensity unprotected attribute. Attribute of CMS FILELIST data field.

low_prot Low intensity protected attribute. Attribute of CP commands output.

Colors

Notice that **brightbrown** is yellow for most terminals. Some bright colors may not be supported by your terminal.

Files

/etc/3270.keys	Standard key binding profile.
/etc/3270keys.hft	Key-bindings for PS2/HFT terminals.
/etc/3270keys.dflt	Default key bindings.

Related Information

In this book: "telnet, tn, tn3270" on page 3-66

Appendix A. TCP/IP Addressing

The Internet Protocol (IP) uses a two-part, 32-bit ***address field***. The first part of the address field contains the network address. The second part contains the local address. There are three different types of address fields (class A, B, or C), depending upon how the bits are allocated.

Figure A-1 represents a class A address. It has a 7-bit network number and a 24-bit local address. The highest-order bit is set to 0. There are 128 possible class A networks.

Figure A-1. Class A Address

0	1 2 3 4 5 6 7	1 2 3 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
0	Network	Local Address

Figure A-2 represents a class B address. It has a 14-bit network number and a 16-bit local address. The highest-order bits are set to 1 and 0. There are 16,384 possible class B networks.

Figure A-2. Class B Address

0 1	1 2 3 4 5 6 7 8 9 0 1 2 3 4 5	2 3 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
1 0	Network	Local Address

Figure A-3 represents a class C address. It has a 21-bit network number and an 8-bit local address. The three highest-order bits are set to 1, 1, and 0. There are 2,097,152 possible class C networks. In a class C address, a 0 in the last (local address) field is a wild card; that is, the address 192.9.200.0 addresses all hosts on network 192.9.200.

Figure A-3. Class C Address

0 1 2	1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3	3 4 5 6 7 8 9 0 1
1 1 0	Network	Local Address

Notes:

1. Class C addresses starting at 192 are recommended for use with TCP/IP.
2. There is a set of reserved network addresses (for example, for access to DARPA).
3. No addresses are allowed with the highest-order bits set to 1-1-1. These addresses (sometimes called class D) are multicast addresses and are currently not supported.

An address cannot have highest-order bits set to 1-1-1.

IP or ARP Local Headers

A 2-byte ***type field*** in the local header of a packet distinguishes IP addresses from ARP addresses. The type numbers in hexadecimal are:

Figure A-4. IP and ARP Type Numbers

Protocol	Type Number
IP	0800
ARP	0806

Ethernet Adapter: The header for the Ethernet Adapter is composed of three fields, as Figure A-5 shows:

Figure A-5. Header for Ethernet Adapter Local Address

Header		
DA	SA	Type
6 bytes	6 bytes	2 bytes

The header fields are:

DA	Destination Address
SA	Source Address
Type	Type Number (IP or ARP)

Host Names

Each host on the network has a unique name and Internet address. Names are a maximum of 64 characters, using the letters a-z, the digits 0-9, and the characters '-' and '.'. It is suggested that names be restricted to a length of 24 characters for compatibility.

Names and addresses are associated with each other by entries in the **/etc/hosts** file. For more information about entries in **/etc/hosts**, see "hosts" on page 5-4.

The **hostname** command must be run to identify the local host to IP. **hostname** takes the name of the host to be identified as a parameter. The **hostname** command is typically run at system start by **rc**. All host names and addresses to be set by **hostname** must appear in the **/etc/hosts** file. For more information, see "hostname" on page 3-21.

Routes

A **route** defines a path for sending information. With the **route** command, you can add and delete the routes defined for a particular host or network.

Routes can specify:

- The host or other gateway on the network that is the default gateway for sending information to a host on another network
- The gateway from a particular host on one network to a different network
- The path from a particular host on one network to a particular host on a different network.

Routes are defined in a routing table, which holds the routing definitions. The route table is a dynamic structure maintained in the kernel. It is updated by the **route** command or the routing daemon.

The routing daemon (**routed**), which can be run on a gateway host, queries other defined gateway hosts periodically for the information necessary to generate, update, and maintain routing tables. **routed** uses two files, **/etc/gateways** and **/etc/networks**, to determine with which hosts to exchange routing information. For more information, see "routed" on page 4-16, "gateways" on page 5-2, and "networks" on page 5-13.

TCP Addressing

TCP provides a set of 16-bit port numbers within each host. Each host generates port numbers independently and, therefore, it is possible for port identifiers not to be unique. To create an identifier that is unique throughout all networks, TCP concatenates the port number with the IP address, producing a ***socket***. A connection is fully specified by the pair of sockets it joins.

Sub-Networks

The sub-network capability of TCP/IP makes it possible to divide a single network into multiple logical networks (***subnets***). For example, an organization can have a single Internet network address that is known to users outside the organization, yet configure its network internally into departmental subnets. Fewer Internet network addresses are required while local routing capabilities are enhanced.

As is explained under Appendix A, "TCP/IP Addressing" on page A-1, a standard Internet address field has two parts, a network address and a local address. To make subnets possible, the local address part of an Internet address is divided into a subnet number (or ***mask***) and a host number, for example:

network_number subnet_number host_number

where:

```
network_number is Internet address for the network.
subnet_number is a field of a constant width for a
given network.
host_number is a field that is at least one bit
wide.
```

If the width of the `subnet_number` field is zero, the network is not organized into subnets, and addressing to the network is done with the Internet network address (`network_number`).

As an example, if 8 bits of the host address are to be reserved for the subnetwork number, only 254 subnets are possible, since subnet numbers 0 and 255 (all 1's) are discouraged (to avoid confusion about broadcast addresses).

The presence of local subnetworks is made known to the system when the network interface is configured by using the **netmask** option of the **ifconfig** command. For more information on this command see "ifconfig" on page 3-23. A **network** mask bit pattern is specified to define that part of the Internet address that is considered as the network address. That is, the mask value has bits set for the standard network address part, and extends the network address part into the host address by setting additional mask bits. Mask bits extending the network address do not have to be contiguous, but it is considered good practice to make them do so, and to use only the highest order bits of the host number address space for the subnet address.

For example, a Class B network has 16 of the 32 Internet address bits reserved for the network address, and 16 for the host number. If an **ifconfig** command is issued as follows:

```
/etc/ifconfig tk0 128.32.1.7 netmask 0xffffff00
```

then 8 additional bits (for a total of 24) are reserved for the network address, and only 8 remain for the host number, for the **tk0** interface. In this example, the interface's address is **128.32.1.7** (host 7 on network 128.32.1). A host system 'm' on subnetwork 'n' would then have an address of the form **128.32.n.m.**.

Broadcast Messages

TCP/IP can send data to all hosts on a local network, or to all hosts on all connected networks. Such transmissions are called ***broadcast messages***. For example, the routing daemon (**routed**) uses broadcast messages to maintain routing tables.

For data to be broadcast to all hosts on connected networks, the destination address in the IP header is set to 255.255.255.255. For data to be broadcast to all hosts on a specific network, the IP address is set to all 1's in the host field. For example, 128.32.3.255 would broadcast to all hosts on the 3 subnet, assuming 3 has been set up as a subnet of IP network 128.32 with the netmask option of the ifconfig command. To broadcast to all hosts on all the subnets, set the entire host field, as defined by the IP class specification, to all 1's (i.e. without regard to subnet masks). So, using the previous example, to broadcast to hosts on all the subnets, the address would be 128.32.255.255.

Assigned Numbers

For compatibility with the general network environment, numbers are assigned for ports, the version, and protocols. AIX System/370 TCP/IP complies with the assigned numbers and parameters in RFC 997. The following three sections explain the assigned numbers.

Port Numbers

Ports are used in TCP to name the ends of logical connections that carry long term conversations. For the purpose of providing services to unknown callers, a service contact port is defined. This list specifies the port used by the server process as its contact port. The contact port is sometimes called the ***well-known*** port.

The assigned ports use a small portion of the possible port numbers. On an assigned port, all but the low-order 8 bits are cleared to 0 (zero). The following table specifies the low-order 8 bits of the assigned port numbers:

Port Number (decimal)	Keyword	Description
21	FTP	File Transfer
23	TELNET	Telnet
37	TIME	Time
42	NAMESERVER	Host Name Server
69	TFTP	Trivial File Transfer
79	FINGER	Finger

Version Numbers

The Internet Protocol (IP) includes a 4-bit field to identify the version of the general inter-network protocol in use. Following is the assigned version number:

Version Number (decimal)	**Keyword**	**Version**
4	IP	Internet Protocol

Protocol Numbers

The Internet Protocol (IP) includes an 8-bit field, called ***protocol***, which identifies the next level protocol. Following are the assigned protocol numbers:

Protocol Number (decimal)	**Keyword**	**Protocol**
1	ICMP	Internet Control Message
6	TCP	Transmission Control
17	UDP	User Datagram

Appendix B. Onhost Installation

onhost consists of two parts:

- that part which executes on AIX
- that part which executes on VM/CMS.

The AIX component is installed as part of the TCP/IP product. The VM/ CMS component is provided in the **/usr/lib/onhost** directory and requires further user intervention to install. This appendix describes how to install the VM/CMS component.

The VM/CMS component contains the following four files:

onhost.cms	contains CMS REXX code for most **onhost** commands
onhoste.cms	generates the host end-of-command flag line
onhostcp.cms	implements **cp** command, uses FTP
onhostld.cms	initializes the host for use by AIX **onhost**

VM/CMS Installation

The following instructions cover the installation of the VM/CMS component:

1. Move the **/usr/lib/onhost/*.cms** files to CMS and rename them as ***.exec** files. The files are REXX code which use VM TCP/IP services. They should be placed on a normal CMS disk which is in the search list for users of **onhost**. TCPMAINT 592 is a convenient disk.

   FTP, from the CMS system, is an easy way to move the file from AIX to CMS. Log in to CMS, access TCPMAINT 592, and enter `FTP`. Follow the FTP prompts and enter the AIX system name, userid, and password. Then CD to the AIX **/usr/lib/onhost** directory. Enter `MGET *.cms`, then QUIT after the file transfers have taken place. Use the CMS rename command to change the CMS file type from CMS to EXEC. For example, **ren * cms a = exec a**.

2. Try out the **onhost** code on CMS. Enter `ONHOSTLD` and expect to see

   ```
   RetCode:00000000; OnHost End of Command Flag
   ```

   followed by the CMS ready message. This program is used to initialize the CMS environment for subsequent use by the AIX **onhost** command. This command, **ONHOSTLD**, is sent to CMS as part of the AIX **hostconnect** automatic login sequence.

   Now try one of the AIX-like commands such as `ONHOST DATE` and expect to see the date and time displayed in AIX format followed by the end of command flag and CMS ready message.

   ```
   Tue Aug 30 15:15:35 PDT 1988
   RetCode:00000000; OnHost End of Command Flag
   ```

3. To make **onhost** available to other CMS users, move the EXEC files to a convenient disk, such as TCPMAINT 592. The VM directory entry for each user of onhost should *not* contain the AUTOCR option on the IPL statement. The **onhost** disk should be accessed in the PROFILE EXEC. By the way, the PROFILE EXEC must not remove lines from the CMS stack when it is executed. If it does, then ONHOSTLD will not be executed during login by **hostconnect**.

Make sure that you can do the following test on each CMS userid to be used by AIX onhost.

1. First connect to the VM system and clear the screen.
2. Enter LOGIN userid when a `CP READ` appears.
3. After `ENTER PASSWORD` appears, enter your password.
4. If `CP READ` appears, then enter `IPL CMS` (The VM directory IPL statement eliminates this step and is recommended.).
5. When `VM READ` appears, enter `ONHOSTLD`. After your profile, if any, is executed, you must see the distinctive end of command flag line and `VM READ`.

This completes installation of the VM/CMS **onhost** component.

Glossary

This glossary contains a list of some of the common terms that you may read or hear when working with data communications. The terms are described only as they relate to data communications.

A

access. The manner in which files or data sets are referred to by the computer.

adapter. See *communications adapter.*

address field. The part of a packet containing addressing information.

addressing. (1) The way that the sending or control station selects the station to which it is sending data. (2) A means of identifying storage locations.

American National Standard Code for Information Interchange (ASCII). The code developed by ANSI for information interchange among data processing systems, data communications systems, and associated equipment. The ASCII character set consists of 7-bit control characters and symbolic characters.

American National Standards Institute (ANSI). An organization sponsored by the Computer and Business Equipment Manufacturers Association for establishing voluntary industry standards.

argument. A parameter passed between a calling program and a called program.

B

bandwidth. The difference, in hertz, between the two limiting frequencies of a band. impressed on the transmission medium. At any point on the medium, only one information signal at a time is present.

baud. A unit of signaling speed equal to the number of discrete conditions or signal events per second.

bit rate. The speed at which serialized data is transmitted, usually expressed in bits per second.

block. A group of records that is recorded, processed, or sent as a unit.

bridge. In the connection of local loops, channels, or rings, the equipment and techniques used to match circuits and facilitate accurate data transmission.

broadband. Transmission media and techniques that use a broad frequency range, divided into sub-bands of narrower frequency.

byte. String consisting of a certain number of bits (usually eight) treated as a unit, and representing a character.

C

cable. The physical media for transmitting signals; includes copper conductors and optical fibers.

cache. A high-speed buffer storage that contains frequently accessed instructions and data. It is used to reduce access time.

cancel. To end a task before it is completed.

carrier. A continuous frequency that can be modulated with a second (information-carrying) signal.

carrier sense multiple access with collision detection (CSMA/CD). The generic term for a class of medium access procedures that (1) allows multiple stations to access the medium at will without explicit prior coordination, (2) avoids contention by way of carrier sense and deference, and (3) resolves contention by way of collision detection and transmission.

channel. A path along which data passes.

character. A letter, digit, or other symbol.

class A address. An IP address in which the first byte is the network address and the following three bytes indicate the host. If the first number of an IP address is less than 128, the address is class A.

class B address. An IP address in which the first and second bytes are the network address and the following two bytes indicate the host. If the first number of the address is between 128 and 191, inclusive, it is class B.

class C address. An IP address in which the first, second, and third bytes are the network address and the last byte indicates the host. If the first number of the address is between 192 and 223, inclusive, then the address is class C.

client. On a network, the computer requesting services or data from another computer.

clock. A device that generates periodic signals used for synchronization.

coaxial cable. A cable consisting of one conductor, usually a small copper tube or wire, within and insulated from another conductor of larger diameter, usually copper tubing or copper braid.

collision. A condition caused by multiple overlapping transmissions on the medium, which results in garbled data.

communication channel. An electrical path that facilitates transmission of information from one location to another.

communications. See *data communications.*

communications adapter. A hardware feature that enables a computer or device to become a part of a data communications network.

communications line. The line over which data communications takes place; for example, a telephone line.

configuration. The group of machines, devices, and programs that make up a data processing system.

confirmation. A transmission by a receiver that permits a sender to continue.

console. A part of a computer used for communications between the operator or maintenance engineer and the computer.

console display. A display that can be requested only at the system console. From a console display an operator can display, send, and reply to messages and use all control commands.

contention. A condition on a communications channel when two stations attempt to use the same channel simultaneously.

contention resolution. The process of resolving contention (medium access control conflicts) according to a defined algorithm.

control block. A storage area used by a program to hold control information.

control character. A character, occurring in a particular context, that initiates, modifies, or stops any operation that affects the recording, processing, transmission, or interpretation of data (such as carriage return, font change, and end of transmission).

CSMA/CD. See *carrier sense multiple access with collision detection (CSMA/CD).*

current host. See *local host.*

D

daemon. A background process that is usually begun at system start which runs continuously and performs a function required by other processes.

data circuit. Associated transmit and receive lines that provide a means of two-way data communications.

data communications. The transmission of data between computers and/or remote devices (usually over a long distance).

data link. The equipment and rules (protocols) used for sending and receiving data.

data stream. All information (data and control information) transmitted over a data link.

digital data. Data represented by on and off conditions called bits.

display station. A device that includes a keyboard from which an operator can send information to the system and a display screen on which an operator can see the information sent to or received from the computer.

distortion. An undesirable change in a data communications signal.

dotted decimal. A common notation for Internet host addresses, which divides the 32-bit address into four 8-bit fields. The value of each field is specified as a decimal number and the fields are separated by periods (for example, 010.002.000.052, or 10.2.0.52).

E

echo. A reflected signal on a communications channel.

emulation. Imitation; for example, the imitation of a computer or device.

enable. In interactive communications, to load and start a subsystem.

F

foreign host. Any host on the network except the one at which a particular operator is working; sometimes called *remote host.*

G

gateway. An entity operating above the link layer, which translates, when required, the interface and protocol used by one network into those used by another distinct network.

glob. Like *echo* but no '\' escapes are recognized and words are delimited by null characters in the output. Useful for programs which wish to use the shell to explain a list of words.

H

home directory. The directory a user accesses when he or she logs in. The user can create and delete files and directories to organize his or her work.

hop count. In the IBM Token-Ring Network, the number of bridges through which a frame passes on the way to its destination.

host. (1) The primary or controlling computer in the communications network. (2) A computer attached to a network hostnames are set up in **/etc/hosts**.

I

interface. A common boundary, but not of internal connections.

interrupt. To take an action at a receiving station that causes the sending station to end a transmission.

IP. Internet Protocol

IP address. Internet Protocol address. Each host has at least one four-byte IP Address. The network part of the address is assigned by the Stanford Research Institute - National Information Center (SRI-NIC)

L

LAN. See *local area network.*

local. Pertaining to a device, file or system that is accessed directly from your system without the use of a communications line. Contrast with *remote.*

local area network. A network in which communications are limited to a moderate-sized geographic area (1 to 10 km) such as a single office building, warehouse, or campus. A local network depends upon a communications medium capable of moderate to high data rate, and normally operates with a consistently low error rate.

local host. The host on the network at which a particular operator is working; sometimes called *current host.*

M

medium (media). The material in or on which data may be represented (for example, twisted pairs, coaxial cables, and optical fibers).

memory. Program-addressable storage from which instructions and other data can be loaded directly into registers for subsequent execution or processing.

message. A communication sent from a person or program to another person or program.

N

network. A collection of data processing products connected by communication lines for information exchange between stations.

network adapter. Circuitry that allows devices using a directly attached network to communicate with the system.

network management. The conceptual control element of a data station that interfaces with all of the layers of that data station and is responsible for the resetting and setting of control parameters, obtaining reports of error conditions, and determining if the station should be connected to or disconnected from the medium.

node. See *host.*

P

packet. The data of one transaction between a host and its network. A packet usually contains a network header, followed by one or more headers used by high level protocols, followed by data blocks.

path. In a network, any route between any two nodes.

physical layer (or level). The lowest layer of network design as specified by the ISO Open System Interconnection (OSI) reference model. This layer is responsible for interfacing with the medium, detecting and generating signals on the medium, and converting and processing signals received from the medium and from the data link layer.

parameter. A variable that is given a constant value for a specified application.

port. (1) An access point for data entry or exit. (2) An entrance to or exit from a network.

positional parameter. A parameter that must appear in a specified location relative to other positional parameters.

profile. Data that describes the significant characteristics of a user, a group of users, or one or more computer resources.

program. A sequence of instructions suitable for processing by a computer. Processing may include the use of an assembler, a compiler, an interpreter, or a translator to prepare the program for execution, as well as to execute it.

prompt. A displayed symbol or message that requests input from the user or gives operational information; for example, the standard UNIX prompt is $. The user must respond to the prompt in order to proceed.

protocol. Rules for transferring data.

process. A program in execution.

R

remote. Pertaining to a device, file, or system that is accessed by your system through a communications line. Contrast with *local*. Synonym for link-attached.

remote host. See *foreign host*.

retransmit. To repeat the transmission of a message or segment of a message.

retry. To resend a transmission that did not achieve the desired or intended result; usually follows a timeout.

root. Another name for superuser.

root directory. The top level directory in a file system. It may contain subdirectories. Synonymous with system directory.

route. A path defined for sending data across a network.

route table. A structure in memory that describes, for the computer, all of the routes that are currently defined.

S

server. On a network, the computer that contains the data to be accessed.

session. The logical connection by which a host program or device can communicate with a program or device at a remote location.

socket. (1) A unique host identifier created by the concatenation of a port identifier with an IP address. (2) A port identifier.

status. The state of affairs or the condition of a station that determines its ability to enter into exchanges of control or information.

T

TCP. Transmission Control Protocol.

telecommunications. Transmitting signals over long distance.

teleprocessing. Processing data that is received from or transmitted to a remote location via communication channels.

terminal. A device, usually equipped with a keyboard and a display device, capable of sending and receiving information over a communications line.

timeout. Measurement of time interval allotted for certain events to occur (such as a response to polling or other controls) before corrective (recovery) action is taken.

transfer. To send data to one place and to receive data at another place.

transmission control characters. Special characters that are included in a message to control communication over a data link. For example, the sending station and the receiving station use transmission control characters to exchange information; the receiving station uses transmission control characters to indicate errors in data it receives.

transparent. In communications, pertaining to transmissions that have no possibility of interference with data link control, regardless of format or content. Transparent transmissions are unrecognized by data link controls.

U

UDP. User Datagram Protocol. A protocol that transmits messages from IP to process on the computer.

W

well-known host name. A conventional name associated with an IP address on a particular network (for example, **nameserver** and **timeserver**.

well-known port. A conventional port assignment used by hosts that support the same protocols, whether or not the hosts are on the same network.

wide area network. A network that provides data communication capability in geographic areas larger than those serviced by local area networks.

work station. A device that lets people transmit information to or receive information from a computer; for example, a display station or printer.

Index

IBM

Reader's Comment Form

TCP/IP User's Guide **SC23-2309-00**

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department C7D
36 Apple Ridge Road
Danbury, CT 06810

Fold

ld

Cut or Fold Along Line

Fold and Tape

Please Do Not Staple

ld and Tape

AIX

IBM

Reader's Comment Form

TCP/IP User's Guide **SC23-2309-00**

This publication is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: For prompt resolution to questions regarding set up, operation, program support, and new program literature, contact the authorized IBM dealer in your area.

Is there anything you especially like or dislike about the organization, presentation, or writing in this publication? Helpful comments include general usefulness; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number: Comment:

What is your occupation? ______________________________

If you wish a reply, give your name and address: ______________________________

____

____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department C7D
36 Apple Ridge Road
Danbury, CT 06810

Fold Fold

Cut or Fold Along Line

Fold and Tape Please Do Not Staple Fold and Tape

AIX

IBM

Reader's Comment Form

TCP/IP User's Guide **SC23-2309-00**

This publication is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: For prompt resolution to questions regarding set up, operation, program support, and new program literature, contact the authorized IBM dealer in your area.

Is there anything you especially like or dislike about the organization, presentation, or writing in this publication? Helpful comments include general usefulness; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number: Comment:

What is your occupation? ____________________

If you wish a reply, give your name and address: ____________________

____

____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments.)

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

International Business Machines Corporation
Department C7D
36 Apple Ridge Road
Danbury, CT 06810

Fold

Fold

Cut or Fold Along Line

Fold and Tape

Please Do Not Staple

Fold and Tape

International Business
Machines Corporation
36 Apple Ridge Road
Danbury, CT 06810

Printed in the
United States of America

SC23-2309-00

IBM®

SC23-2309-00

71F0369