Installation-Manual
For TCP/IP
Running On ICC

Documentation-No.: G0920244-0687

Best.-Nr.: G0920244-0687
Author's initials: nb

Trademarks :

MUNIX,CADMUS	for PCS
DEC, PDP	for DEC
UNIX	for Bell Laboratories
ETHERNET	for XEROX Co.

Copyright 1987 by
PCS GmbH, Pfälzer-Wald-Strasse 36, D-8000 München 90, tel. (089) 68004-0

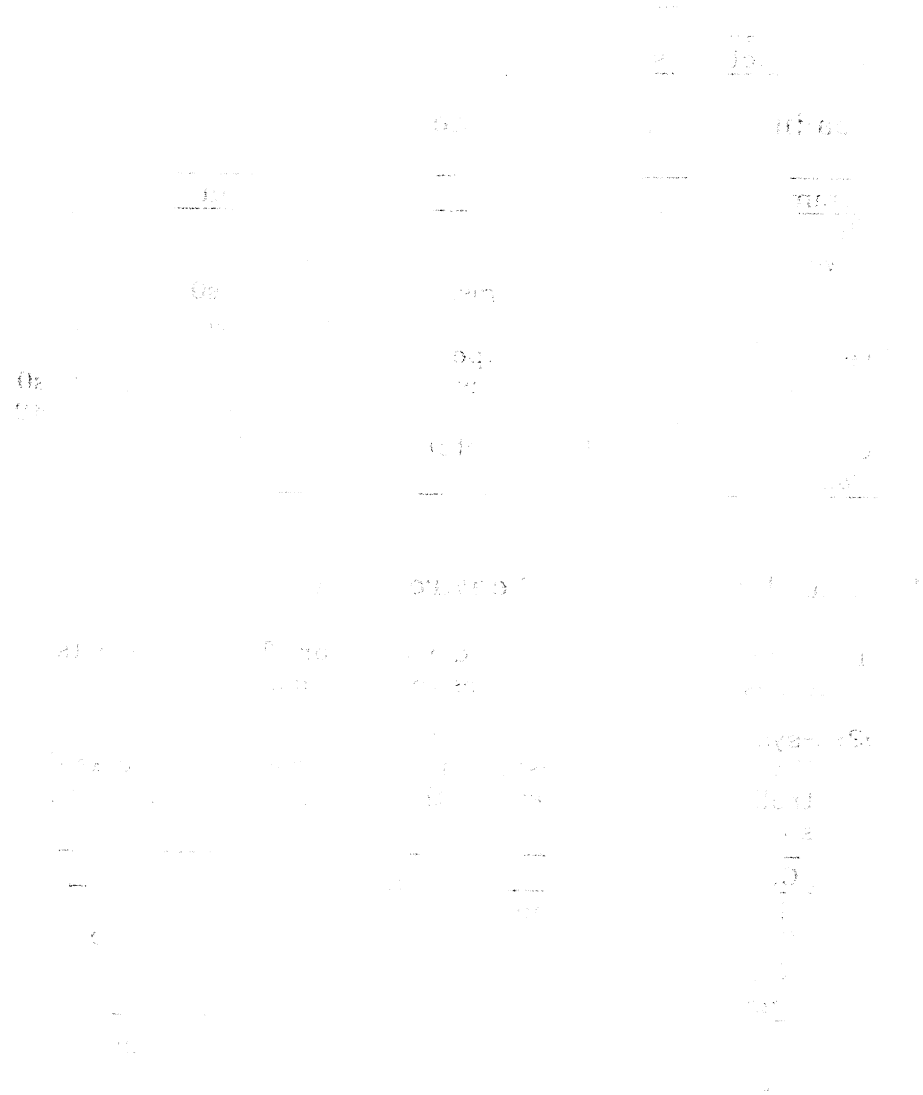
The information contained herein is the property of PCS and shall neither be reproduced in whole or in part without PCS's prior written approval nor be implied to grant any license to make, use or sell equipment manufactured herewith.

PCS reserves the right to make changes without notice in the specifications and materials contained herein and shall not be responsible for any damages (including consequential) caused by reliance on the materials presented.

Table of Contents

* * * * *

This paper was printed by PCS on the Laser-Beam-Printer LBP 68000.



1. Introduction

This paper describes the installation procedure for TCP/IP-software running on ICC 0. It covers the MUNIX-Releases m32-1.2 (32-Bit systems) and V.2/07 (16-Bit systems).

2. Software installation

2.1. Login root

Comment	User-Input
(after login:)	**root**
Enter root-password	**<root-password>**

2.2. Reading files from streamer tape

Comment	User-Input
Change to root-directory	**cd /**
rewind streamer-tape	
99xx and **96xx**-systems:	**</dev/ris0**
92xx-systems:	**</dev/rst0**
read files from streamer-tape	
99xx and **96xx**-systems	**cpio -ivmdS </dev/ris0**
92xx-systems	**cpio -ivmdS </dev/rst0**
Change to just created directory */usr/adm/tcpip*	**cd /usr/adm/tcpip**

2.3. Installing the TCP/IP environment

This section installs the devices for TCP/IP, starts the server processes and adds some lines to */etc/rc* etc.

92xx-systems only

If you have a 92xx-system which has an ICC used as Ethernet controller, there are several things you must do before installing the software.

Comment	User Input
(If **not** already done)	**cd /dev; make ICC0**
(If **not** already done)	**/etc/iccdown -i -b -f /dev/icc0 /icckernel**
Now return to */usr/adm/tcpip*	**cd /usr/adm/tcpip**

To download the software to the ICC every time you start the CADMUS-system you can delete the **#** before the */etc/iccdown*-command in */etc/rc*.

Comment	User Input
To complete the installation just type	**instcp**

First the configuration of */unix* is checked. It is assumed that */unix* is the currently running system. **instcp** outputs **OK** if */unix* is configured for TCP/IP, **NOT OK** otherwise.

In the first case please continue with the next chapter. All you need to install TCP/IP software is now done. If you did not have an Ethernet connection before you will need a cable to connect the plug (Ethernet) in your system to an Ethernet transceiver.

In the latter case you have to configure a new */unix*.

Comment	User Input
To do that you have to change to */usr/sys*.	**cd /usr/sys**
Now save your current configuration files	**sav old**
and create new ones,	**newconf**
answering the question for	...
TCPIP with	**y**
all other questions just as for your current configuration.	...
Now compare your new configuration file with the old one	**diff conf.h conf.h.old**

There must be a difference in at most two lines (two lines in V.2/07, the first line only in m32-1.2). Otherwise your new *conf.h* is different in more than the including of TCP/IP ! The two different lines start with

#define ICC_TCPIP 1

and

#define ARPA_ICC 0

Comment	User Input
Now create a new version of the kernel (MUNIX).	**make**
Save your old */unix*	**mv /unix /ounix**
Install your new one	
on 32-bit systems (/3 and /4 CPU's):	**mv /newunix /unix**
on 16-bit systems (/1 and /2 CPU's):	**kernel.run**
and, after informing active users and their logout, reboot your system.	
on 32-bit systems (/3 and /4 CPU's):	**sync; sync; sysboot**
on 16-bit systems (/1 and /2 CPU's):	**/etc/init s**
and	**sync; sync; rem_init**
After successful booting	
login:	**root**
Enter root password	**<root password>**
Now start again at chapter 2.3 after giving the command	**cd /usr/adm/tcpip**

2.4. Additional actions

2.4.1. Setting the environment for the administrator

All the programs used for the administration of TCP/IP on your CADMUS-System are located in */usr/adm/tcpip*, so it may be useful to add this directory to the PATH variable in */.profile* (which is the profile for the user *root*).

Don't forget to **export** the **PATH** (if using *sh(1)*).

2.4.2. Updating the host database

If this is the first time you are installing TCP/IP on your CADMUS system you have to add several lines to your host database which is located in */etc/hosts*.

For every host you want to communicate with there must be an entry containing his IP-address, his name (unique around the net), and perhaps several alias names (all in one line).

There must also be an entry for your CADMUS system. The host name for this entry must be the node name, i.e. the name you get by typing **uname -n**, which must NOT be **aunix**. This happens if TCP/IP is installed in *Single User Mode* during a new installation of a MUNIX Release running */aunix*.

Since there is no entry *localhost* in our */etc/hosts* you can simply distribute one */etc/hosts* file to all CADMUS-systems which have the TCP/IP-software running on **ICC** and want to have a connection with

each other.

Caution: To change your IP-address you have to edit the entry for your host in */etc/hosts* and to reboot the system.

2.4.3. Allowing remote logins

To allow remote login to your system via *telnet* or *rlogin* there must be a *getty process* on at least one *ptyp*?.

The number of getty's on ptyp? reflects directly the number of remote logins possible. (Caution: The ptyp? will be used for remote login by MUNIX/NET and DECnet).

You may allow several remote logins by adding some lines to */etc/inittab*. An example of an entry in */etc/inittab* (for *ptyp0*) is:

p0:2:respawn: /etc/getty ttyp0 9600 none

For every *ptyp*? you have to add a line like the above.

Our server processes (*telnetd* and *rlogind*) try to open one of the sixteen pty ports
/dev/ptyp0, */dev/ptyp1*, ... ,*/dev/ptyp9*, */dev/ptypa*, ... , */dev/ptypf*
on demand.

If errors are displayed on the console (e.g. getty: can't open ...) check whether all ptyp? are existent in */dev* and whether the PTY-ports are configured in current */unix*. The last check is done by typing:
fgrep PTY /usr/sys/conf.h
Now at least two lines containing PTY must occur on the display starting with
#define PTY 1
and
#define NPTY <number of PTY-ports>

If these two lines do not show, */unix* is not configured for PTY's. You have to change the configuration prior to allow remote logins.

3. Testing the installation

3.1. Local tests

After complete installation you should now test the installation. There are several test programs in */usr/adm/tcpip/test* for local testing. The frame is called *tcpip.test*. It starts two continously running tests, one testing **rcp** (UNIX remote copy) and one testing **ftp** (ARPA remote copy). These two tests use the servers **rshd** (for rcp) and **ftpd** (for ftp) which must run on your system. In addition the test for *ftp* requires a certain user (which is **visi** per default) and a certain password for this user (which is **visi** also per default). If you want to change the username and password, edit the lines starting with
USERNAME=...
and

PASSWORD=...
in */usr/adm/tcpip/test/dauer.ftp*. These tests build a protocol of their work in /tmp/ftp_proto and /tmp/rcp_proto.

In addition you can call the user programs with your host as argument (e.g. rlogin, telnet). These are difficult to test without human intervention because they are remote login programs which require an active user.

3.2. Remote tests

Call some of the user programs with a remote host as argument and try to establish a connection to it.

If that fails, there are several reasons for this behaviour.

1. There is no entry for the remote host in your */etc/hosts*.
2. The IP-address of the remote host is not the IP-address in */etc/hosts*. Then the connection will be timed out.
3. Your host is not known by the remote host, i.e. the host database of the remote host contains no entry for your host. (In most UNIX-installations the host database is in */etc/hosts*.)
4. You have no connection to your Ethernet (cable ?).
5. Your Ethernet doesn't work at all, i.e. there is a faulty hardware installation on the net.

The last two errors can be determined by starting the program **/usr/adm/tcpip/getstat** to request some statistics gathered on the ICC controller board. The output can easily be interpreted and the appropriate actions can be taken (c.f. *getstat(8)*).

4. Actions to be taken before system update

If you get a **new release of the MUNIX-Operating System**, it would be helpful to save several files, because they contain important information. The new re-installation of TCP/IP is much more easier if you save:

Filename	Comment
/etc/hosts	host database for TCP/IP
/etc/hosts.equiv	mapping of users on other hosts to this host
/.rhosts	mapping of accounting of users on other hosts to this host

NAME

telnet – user interface to the TELNET protocol

SYNOPSIS

telnet [–**v**] [*host*_[*port*]]

DESCRIPTION

telnet communicates with another host using the TELNET protocol. If *telnet* is invoked without arguments, it prompts for the host to which a connection is desired. If it is invoked with arguments, it attempts to establish a connection to the host given on its command line. In either case, if no port number is specified, *telnet* attempts to contact a TELNET server at the default port (port 23). The host specification can be either a host name [see *hosts(5x)*] or an Internet address (in "dot notation").

If the *–v* flag is given, all TELNET option negotiations are displayed. Options sent by *telnet* are displayed as "SENT," while options received from the TELNET server are displayed as "RCVD."

Once a connection is established, *telnet* enters input mode. In this mode, typed text is sent to the remote host. This can be done in one of two ways, at the remote host's discretion. If the remote host volunteers to perform character echo, each character typed is sent to the remote host uninspected. Otherwise, local line editing is performed and data are not sent until a complete line has been entered. In either case, the line is terminated with the standard TELNET end-of-line sequence <cr><lf> when it is sent. Similarly, in this mode incoming data are assumed to be in chunks of lines and the <cr><lf> sequence is translated into the local end-of-line marker.

Once a connection has been established, *telnet* enters command mode when the *telnet* escape character is typed. Initially, this character is '^]', but the user can change the escape character, as described below. In command mode, *telnet* accepts and executes the commands listed below. These commands are processed locally and are not seen by the remote host. Only enough characters to uniquely identify the command need be typed.

escape [*escape-char***]**
Set the *telnet* escape character. Single character control characters can be used, for example "control-X" is "^X".

quit Close any open TELNET session and exit *telnet*.

status Show the current status of *telnet*. This includes the peer and debugging state and the current escape character.

open [host-name]
Connect to a site given in *host-name* or in the next input line after the (to)-prompt.

close Close current connection and exit *telnet*.

options
Toggle viewing of options processing (this processing only occurs during connection establishment).

? [*command***]**
Get help. With no arguments, *telnet* prints a help summary. If a

command is specified, *telnet* prints the available information on it.

There are two ways to leave *telnet*. The first is to input the escape-character and to enter **quit**, the second is to input *Ctl-Z*.

BUGS

The process structure used by TELNET reflects a fundamental limitation of many versions of UNIX. This severely restricts the number and kind of TELNET options that can be supported.

NAME

telnetd – DARPA TELNET protocol server

DESCRIPTION

The TELNET server listens on the standard TCP port number for TELNET (23). When a TELNET session starts, it sends the TELNET WILL ECHO option to the client side, indicating a willingness to "remotely echo" characters. Thereafter, it responds with a WILL when requested by the client to DO one of the following options:

ECHO
SUPPRESS-GO-AHEAD
TRANSMIT-BINARY

and returns a DONT in response to requests for any other options. ECHO option semantics are implemented by *pty(4)*.

FILES

/dev/ptyp[0-9a-f] pty device nodes
/dev/ttyp[0-9a-f] pty device nodes

SEE ALSO

telnet(1x), pty(4), rlogind(8x), init_tcp(8x).

NAME

arp – address resolution display and control

SYNOPSIS

arp GET *hostname*
arp KILL *hostname*
arp SET *hostname ether_addr*
arp REQ *hostname*

DESCRIPTION

The *arp* program displays and modifies the Internet-to-Ethernet address translation tables used by the address resolution protocol [*arp(4x)*].

With the **GET** flag, the program displays the current ARP entry for *hostname* by requesting the entry from the ICC board.

With the **KILL** flag, a user can delete an entry for the host called *hostname*.

The **SET** flag creates an ARP entry for the host called *hostname* with the Ethernet address *ether_addr*. The Ethernet address *ether_addr* is given as six bytes. Each of the six bytes may be specified in octal, decimal or hexadecimal, and must be separated by dots. In this case, octal numbers start with 0, decimal numbers start with 1 to 9, and hexadecimal numbers start with 0x or 0X.

The **REQ** flag initiates an ARP-Request to the given *hostname* and stores the (possible received) answer in the current ARP-table on the ICC-controller-board. This entry is shown in an subsequent call of *arp* with the GET flag followed by the *hostname*.

DIAGNOSTICS

The ARP command will complain of errors in its input format, of non-super users attempting to change the table, of references to unknown hosts, and of various system-detected errors.

FILES

/dev/arpa/admin The device-driver for administration of ARP.

SEE ALSO

arp(4x)

NAME

arp – Address Resolution Protocol

DESCRIPTION

ARP is a protocol used to dynamically map between DARPA Internet and Ethernet addresses on a local area network. It is used by the Ethernet interface drivers and is not directly accessible by users.

ARP caches Internet-Ethernet address mappings. When an interface requests a mapping for an unknown address, ARP drops that message and broadcasts a message on the associated network requesting the address mapping. If a response is provided, the new mapping is cached and used with any subsequent messages to that address. Until a response is obtained, ARP discards any messages to the unknown address. ARP itself is not Internet or Ethernet specific, but this implementation of it is.

ARP watches passively for hosts impersonating the local host (that is, a host that responds to an ARP mapping request for the local host's address).

Only super user may alter or display the contents of the ARP table.

FILES

/dev/arpa/admin	The device-driver for administration of ARP.

SEE ALSO

arp(1x)

NAME

ftp – file transfer program

SYNOPSIS

ftp [**–v**] [**–d**] [**–i**] [**–n**] [**–g**] [*host*]

DESCRIPTION

ftp is the user interface to the ARPANET standard file transfer protocol. It can transfer files to and from a remote network site.

A remote host can be specified in the command line. If specified, *ftp* immediately attempts to connect to a FTP server on that host; otherwise, *ftp* enters its command interpreter and awaits instructions from the user. When *ftp* is awaiting commands, the prompt "ftp>" is displayed. *ftp* recognizes the following commands:

! [*cmd* **[***args***]]**
Invoke a shell on the local machine (no *cmd* given) or execution of *cmd* (followed by several *args*) under control of a shell.

append *local_file* [*remote_file*]
Append a local file to a remote file If *remote_file* is not specified, *local_file* copies (and appends) under its original name. The current settings for *type*, *format*, *mode*, and *struct* are used during file transfers.

ascii Set the file transfer *type* to network ASCII (the default).

bell Sound a bell after each file transfer command completes (toggle).

binary
Set the file transfer *type* to support binary image transfer. This is strongly recommended for transfer of binary-files (e.g. executable programs).

bye Terminate the ftp session with the remote server, exit *ftp* command mode, and return to the operating system.

cd *remote_directory*
Change the current working directory on the remote host to *remote_directory*.

close Terminate the FTP session with the remote server but remain in the *ftp* command interpreter.

debug [*debug_value*]
Toggle debugging mode. When *debug_value* is nonzero, debugging is on and *ftp* displays each command sent to the remote host. (Caution: Your remote password shows on the screen during the *user*-command if debugging is enabled). Default: No debugging.

delete *remote_file 1*
Deletes one file on the remote host.

dir [*remote_directory*] [*local_file*]
List the contents of *remote_directory* on the remote host. If *remote_directory* is not specified, the current working directory on the remote host is listed in *local_file*. If *local_file* is not specified, the *remote_directory* listing is shown on the terminal.

form *format*
Set the file transfer *form* to *format*. The default format is "file." This is the only format currently supported.

get *remote_file* [*local_file*]
Copies *remote_file* from the remote host to the local system. If *remote_file* is omitted, *ftp* prompts for the name. If *local_file* is not specified, *remote_file* retains its original name when copied to the local machine. If a - (hyphen) is specified for *local_file*, the file is displayed on the terminal. The current settings for *type*, *format*, *mode*, and *structure* are used during file transfers.

glob Toggle file name globbing. When on (default), each local file or pathname is processed for *csh(1)* metacharacters. These characters include *, ?, [,], ~, {, and } . Remote files specified in multiple item commands (for example, *mput*) are globbed by the remote server. When off, all files and pathnames are treated literally. Remote files specified in multiple-item commands (such as *mget* and *mput*) are globbed by the remote server. When globbing is off, all files and pathnames are treated literally.

hash Toggle printing a hash sign (#) for each data block (1024 bytes) transferred. Default: Off.

help [**command**]
Lists all *ftp* commands. If *command* is specified, on-line help documentation for that command is provided.

lcd [*directory*]
Changes the current working directory on the local system to *local_directory*. If *local_directory* is not specified, the user's home directory becomes the current working directory.

ls [*remote_directory*] [*local_file*]
Lists the abbreviated contents of a directory on the remote host in *local_file*. If *remote_directory* is not specified, the contents of the current working directory are listed. If *local_file* is not specified, the *remote_directory* listing is shown on the terminal.

mdelete *remote_file1* [*remote_file2* ...]
Deletes one or more files on the remote host. If globbing is enabled, *remote_files* is first expanded via *ls*.

mdir *remote_files local_file*
Places directory information for a listing of one or more files from the remote host in *local_file*. The last file in the file list is treated as *local_file*.

mget *remote_files*
Places one or more files from the remote host in the current working directory on the local host. If globbing is enabled, remote filenames are expanded via *ls*.

mkdir *remote_directory*
Creates *remote_directory* on the remote host. The directory must not already exist.

mls *remote_files local_file*
Place abbreviated directory information for one or more files on the remote host into *local_file*. The last file in the file list is treated as *local_file*.

mode [*mode_name*]
Set the file transfer *mode* to *mode_name*. The default mode is "stream" mode. This is the only mode currently supported.

mput *local_files*
Copy one or more files from the local host to the current working directory on the remote host.

open *host* [*port*]
Establishes a connection to the specified *host_name* FTP server. If *port* is specified *ftp* attempts to contact an FTP server at that port. The default *port* for *ftp* is 21. If the *auto_login* option is on (default), *ftp* also attempts to automatically log the user into the FTP server (see FTPD).

prompt
Toggles interactive prompting. This allows the user to selectively retrieve or store files during multiple file transfers. When off, any *mget* or *mput* transfers all files. Default: On.

put *local_file* [*remote_file*]
Copies a file from the local system to the remote system. If *remote_file* is not specified, *local_file* is copied under its original name. The current settings for *type*, *format*, *mode*, and *structure* are used during file transfers.

pwd Display the name of the current working directory of the remote host.

quote *ftp-command*
Quote sends the arbitrary *ftp-command* to the FTP server, if connected to one.

quit Terminate the *ftp* session with the remote server, exit *ftp* command mode, and return to the operating system.

recv *remote_file* [*local_file*]
Same as the *get* command.

remotehelp [*command_name*]
Display help documentation from the remote FTP server. If *command_name* is specified, help for that command is provided.

rename *remote_file1 remote_file2*
Rename *remote_file1* to *remote_file2*.

rmdir *remote_directory*
Delete the *remote_directory*. The remote directory must be empty before it can be deleted.

send *local_file* [*remote_file*]
Same as the *put* command.

sendport
Toggles the use of port commands. When *sendport* is on (default),

ftp attempts to use the port specified by the client when establishing a connection for each data transfer. If this fails, *ftp* uses the default data port (port 21). When *sendport* is off, *ftp* uses the default port. This is useful for certain FTP implementations that ignore port commands, but incorrectly indicate that they have been accepted.

status Display the current state of all *ftp* options.

struct [*struct_name*]
Set the file transfer structure to *struct_name*. The default structure is "stream." This is the only structure currently supported.

type [*type_name*]
Set the file transfer type to *type_name*. If *type_name* is not specified, the current type is printed. Network ASCII is the default type (which transmits 8-bit data with the most significant bit set to zero). The other type is image (binary).

user *user_name* [*password*] [*account*]
Identifies the user to the remote FTP server. If *password* is not specified and one is required by the server, *ftp* prompts the user for one (after disabling local echo). If *account* is not specified and one is required by the server, *ftp* prompts the user. Unless *ftp* is invoked with *auto_login* disabled, this process occurs when first connecting to the FTP server.

verbose
Toggle verbose mode. When on (default when *ftp is used interactively), all responses from the FTP server are displayed, and statistics on the effiency of the file transfer are reported.*

? [*command*]
Same as the *help* command.

Embedded spaces in command arguments can be quoted with quotation marks (").

FILE NAMING CONVENTIONS

Files that are arguments to *ftp* commands are processed as follows:

- If the filename "-" is specified, the *stdin* (reading) or *stdout* (writing) is used.
- If globbing is enabled, local file names are expanded according to the rules used by the local command interpreter. For UNIX, this is *csh (1)*; c.f. *glob*.

FILE TRANSFER PARAMETERS

The FTP specification lists many parameters that may affect a file transfer. The *type* can be "ascii," "image" (binary), "ebcdic," and "local byte size" (PDP-10's and PDP-20's). *ftp* supports the ascii and image types of file transfer.

Caution: The image type of transfer must be used if binary files are transmitted. Otherwise errors occur, i.e. not all bytes may be transfered and the MSB is always zero.

ftp supports only the defaults for the remaining file transfer parameters: *mode*, *form*, and *struct*.

OPTIONS

Options can be specified in the command line or to the command interpreter.

The *-v* option shows all responses from the remote server and reports data transfer statistics. This is the default if ***ftp*** is used interactively.

The *-i* option turns off interactive prompting during multiple file transfers.

The *-d* option enables debugging.

The *-n* option disables autologin. Instead you have to use the ***ftp*** subcommand *user* to log in.

The *-g* option disables filename globbing.

BUGS

Many FTP server implementations do not support experimental operations, such as ***pwd***. Aborting a file transfer does not work correctly; if attempted, you will likely have to leave the local ***ftp***. Building up a new connection is necessary.

NAME

ftpd – DARPA Internet file transfer protocol server

SYNOPSIS

ftpd [–t*timeout* **]**

DESCRIPTION

ftpd is the DARPA Internet file transfer protocol server process. It uses the TCP protocol and listens at the port specified in the "ftp" service specification; see *services(5)*.

The FTP server times out an inactive session after 60 seconds if no *timeout* is specified. If the *–ttimeout* option is specified, the inactivity timeout period is *timeout* seconds.

The FTP server supports the following FTP requests (Internet RFC 765); they can be entered in upper- or lowercase.

Request	Description
ACCT	specify account (ignored)
ALLO	allocate storage (vacuously)
APPE	append to a file
CUP	change to parent directory
CWD	change working directory
DELE	delete a file
HELP	get help
LIST	list directory files ("ls -lg")
MKD	make a directory
MODE	specify data transfer *mode*
NLST	list file names ("ls")
NOOP	do nothing
PASS	specify password
PORT	specify data connection port
PWD	print the current working directory
QUIT	end 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
STRU	specify data transfer *struct*
TYPE	specify data transfer *type*
USER	specify user name
XCUP	change to parent directory
XCWD	change working directory
XMKD	make a directory
XPWD	print the current working directory
XRMD	remove a directory

The remaining FTP Internet RFC 765 requests are recognized, but are not currently implemented.

ftpd interprets file names according to the "globbing" conventions of the normal command interpreter for the host system. For UNIX systems this is the *csh(1)*. For example, on UNIX systems, the metacharacters *, ?, [,], {, }, and ~ can be used.

ftpd authenticates users according to the normal login procedures for the host system.

SEE ALSO

ftp(1x)

BUGS

Autologout (after timeout) does not work.

Commands cannot be aborted.

NAME

getstat – board statistics

SYNOPSIS

getstat [**n**]

DESCRIPTION

getstat gets statistics from the ICC front-end processor board and show it periodically on a VT100-like screen (*getstat* uses the clear-screen function). About every 5 seconds a new statistic is gathered and displayed. This time may be changed to *n* seconds if this parameter is given.

There are individual statistics for every protocol-level with the following headers.

Level/Name	Header
Ethernet	LANCE
ARP-Protocol	ARP
IP-Protocol	IP
TCP-Protocol	TCP
UDP-Protocol	UDP
Host-Comm.	Level 5

The different values for each protocol level are e.g. number of send and received frames, invalid frames, etc.

Only a super-user can activate *getstat*.

OPTIONS

n set a time interval other than 5 seconds (default) for gathering and displaying the new statistic.

NAME

hosts – host name database

DESCRIPTION

The Hosts file contains information on the known hosts on the DARPA Internet. For each host a single line should be present with the following information (each parameter is separated by a blank):

Internet_address host_name [*alias*] ...

Internet–address is a four-byte address. Each byte is in decimal or hexadecimal notation and is separated by a dot.

host_name is the name of the system associated with this Internet address. This is an arbitrary string, preferably long enough to keep hosts unique on the network. Typically some "name space" is chosen, such as alabama, alaska, arkansas ... or washington, adams, jefferson ...

alias is another name for the same system. Typically, this is a shorter name (al, ak, ar ... or george, john, tom) and may not be unique across the network (but of course unique in your */etc/hosts*). A single host can have more than one *alias*. For example, the host "alaska" could have the following address and alias:

129.0.9.5 alaska ak

Each system's own entry in the Hosts file **must** contain the systemname (as returned by *uname -n*) because it is needed during initialization by *init_tcp(8)*.

FILES

/etc/hosts

SEE ALSO

init_tcp(8), remsh(1), rcp(1), ftp(1), telnet(1), rlogin(1).

NAME

hosts.equiv – host names for shared accounts

DESCRIPTION

The *hosts.equiv* file contains information on hosts sharing account access permission. Each line in this file lists the unique name for each host sharing the local host's account names. A local system named "alaska" with the */etc/hosts* file

192.0.9.5 alaska ak
192.0.9.8 alabama al
192.0.9.2 arkansas ak

could have a *hosts.equiv* file

alabama
arkansas

to allow users on "alabama" or "arkansas" to execute *remsh* or *rcp*.

FILES

/etc/hosts.equiv

SEE ALSO

hosts(5), remsh(1), rcp(1).

NAME

init_tcp – initialize and configure TCP/IP protocol module

SYNOPSIS

init_tcp [–d] [–h *host* **] [–e** *ENET_addr* **] [–r] [–i** *INET_addr* **]**

DESCRIPTION

init_tcp initializes the TCP/IP protocol module running on the ICC board.

If no options are specified, *init_tcp* configures TCP/IP with the following defaults:

- Debug messages are not printed during configuration.
- The */etc/hosts* is checked for a host with the name of the local system (as returned by *uname -n*); this name is used for the host's Internet address.
- The Ethernet address supplied by the MUNIX-System is used.

OPTIONS

–d If given, *init_tcp* prints debugging messages during configuration.

–h *host*
Supplies an explicit host name. *Init_tcp* tries to look up a host by this name in the */etc/hosts* file and uses the Internet address therein.

–e ENET_addr
Supplies an Ethernet address, which is used instead of the MUNIX-System's Ethernet address (supplied by PCS). This should be specified in the customary Ethernet format, as six hexadecimal numbers separated by hyphens (-). For example, 08-00-27-00-2A-56. This option is without any effect if the TCP/IP protocol module runs on the System ICC (ICC 0).

–r Specifies a forced reset of the ICC and the device-drivers. Returns EBUSY, if running on the System-ICC (ICC 0).

–i INET_addr
Supplies an Internet address, which is used instead of the IP address found for the entry *"systemname"* in */etc/hosts*. The *INET_addr* must be given in "standard-dot-format", which is 4 bytes separated by dot. For example, 192.12.128.119 or 0xc0.0x0c.0x80.0x77. Both represent the same IP address.
Only one of the options *-h* or *-i* may be given.

FILES

/dev/arpa/admin	device driver for administration of TCP/IP
/etc/hosts	host name/Internet address database

SEE ALSO

hosts(5x)

NAME

rcp – remote file copy

SYNOPSIS

rcp *file1 file2*
rcp [**–r**] *file ... directory*

DESCRIPTION

rcp copies files between machines. Each *file* or *directory* argument is either a remote filename of the form *remote_host:path*, *remote_host.user:file*, or a local file name containing no colons or having a slash (/) before any colons.

If *–r* is specified and any of the source files are directories, *rcp* copies each subtree rooted at that name. Here the destination must be a directory.

If *path* is not a full pathname, it is interpreted relative to your login directory on *remote_host*. A *path* on a remote host can be quoted (using \ , ", or ') to interpret metacharacters remotely.

rcp does not prompt for passwords. The current local username must exist on *remote_host* and allow remote command execution via *remsh(1x)*.

rcp handles third-party copies, where neither source nor target files are on the current machine. Host names may also take the form *remote_host:user_name* to use a different user account rather than the current user name on the remote host.

SEE ALSO

remsh(1), rlogin(1)

BUGS

rcp does not detect all cases where the target of a copy is a file but only a directory is legal. For instance,

rcp <system>:<path>/* <path>/<filename>

copies only the last file matching the first path name onto the second filename.

NAME

remsh – remote shell

SYNOPSIS

remsh *host* [**–n**] [**–l** *user_name*] [*command*]

DESCRIPTION

remsh connects your terminal to the specified *host* and executes the specified command. *remsh* normally copies its standard input to the remote command, the standard output of the remote command to *remsh*'s standard output, and the standard error of the remote command to *remsh*'s standard error. Interrupt, quit, and terminate signals are passed to the remote command. *remsh* normally terminates when the remote command does. It returns immediately to the local host if the *–n* option is used, without redirecting its standard input to the remote command.

The remote username is your local username, unless a different remote name is specified with the *–l* option. Usernames on the two systems must be equivalent (as specified by */etc/hosts.equiv* or *.rhosts*); a password cannot be used here. The *–l* option is equivalent to using a *.rhosts* file entry.

If *command* is specified, *remsh* executes it and returns to the local host. If *command* is not specified, *rlogin(1x)* logs the user in on the remote host.

Unquoted shell metacharacters are interpreted on the local machine; quoted metacharacters are interpreted on the remote machine. Thus the command

remsh otherhost cat remotefile >> localfile

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

remsh otherhost cat remotefile ">>" otherremotefile

appends *remotefile* to *otherremotefile*.

Host names are in the file */etc/hosts*. Each host has one unambiguous standard name (the first name given in the file) and may optionally have one or more nicknames.

FILES

/etc/hosts	hostname to network address mapping
~/.rhosts	

SEE ALSO

rlogin(1x)

BUGS

Interactive commands [such as *rogue(6)* or *vi(1)*] will not run. To use interactive commands, use *rlogin(1x)*.

NAME

rhosts – host and user names for shared accounts

DESCRIPTION

The *rhosts* file contains information on users and hosts sharing account access permission. The *.rhosts* file is in the home directory of the user granting access permission. Each line in this file lists the names of the remote host from which the user can access the system and the name of the user. A user could allow the following users on the remote system "alaska" to access the local system through his or her account:

```
alaska  joe
alaska  samantha
alaska  xerxes
```

The remote hostname and the username must be separated by one blank only (no tab's).

SEE ALSO

remsh(1x), rcp(1x).

FILES

~/.rhosts

NAME

rlogin – remote login

SYNOPSIS

rlogin *remote_host* [**–ec**] [**–l** *username*] [**–8**]

DESCRIPTION

rlogin connects your terminal on the current host system to a login listener on the remote host system *remote_host*. (The remote host name must be a standard name or the alias in */etc/hosts*.)

Some rlogin-servers have an auto-login feature. To use this feature, a frame containing the remote username is transmitted to the server. Your local username is used as the remote username unless the option *-l* is present. In that case the given *username* is used as remote username instead of your local username. The remote machine will prompt for password only, if the auto-login function is implemented there. Otherwise the remote machine prompts for a login plus a password as in *login(1)*.

All echoing occurs at the remote site; the remote login is transparent except for delays. Flow control via CTRL-S and CTRL-Q, and I/O flushing on interrupts are handled properly. A line of the form " ~ . " disconnects your terminal from the remote host, where " ~ " is the escape character. A different escape character can be chosen with the *–e* option. *Rlogin* allows a (local) shell-escape by typing " ~ ! " or (local) command execution via " ~ command ".

The option *-8* enables transmission of 8-bit data, otherwise all data is transmitted with the 8th bit set to zero (default).

There are two ways to leave *rlogin*. The first (and recommended) is to type the disconnect sequence (" ~ . "), the second (but bad way) is to type *Ctl-Z*.

BUGS

The disconnect sequence (" ~ . ") may not only cut the connection to one host but all connections which have been established (e.g. multiple remote login's on several hosts) even those connections not using *rlogin* (e.g. *cu(1C)*, *ulogin(1U)* etc.).

You can avoid this in most cases, if you use the *-e*-Option to define an escape-character other than " ~ ".

SEE ALSO

remsh(1x), hosts(5x)

FILES

/etc/hosts host name to network address mapping

NAME

rlogind – remote login server

DESCRIPTION

The rlogin server listens for *rlogin(1x)* clients connecting on the Internet TCP login port (513). When a rlogin session starts, it waits to receive a NULL byte from the client and then returns a NULL byte. The server then parses out, and throws away, three NULL-terminated strings normally intended to implement autologin (e.g. the -l option of *rlogin*). That means that auto-login is not implemented by *rlogind* so *rlogind* will prompt for a login and a password as in *login(1)*. Subsequently, it passes data in both directions without interpretation.

DIAGNOSTICS

If *rlogind* returns **Login incorrect.** to a user on a remote machine, there is no entry for the remote host in the local */etc/hosts*.

FILES

/dev/ptyp[0-9a-f] pty device nodes
/dev/ttyp[0-9a-f] pty device nodes

SEE ALSO

telnetd(8x), rlogin(1x), pty(4)

NAME

rshd – remote shell server

SYNOPSIS

rshd

DESCRIPTION

rshd is the server for the *remsh (1x)* and *rcp (1x)* programs. It provides remote execution facilities with authentication based on privileged port numbers.

rshd listens for service requests at the "cmd" port (ID = 514). When a service request is received, the following protocol is initiated:

1. The server checks the client's source port. If the port number is not in the range 0 to 1023, the server aborts the connection.
2. The server reads characters up to a null (\0) byte from the socket. The resultant string is interpreted as a decimal number.
3. A nonzero port number (Step 1) is the port number of a secondary stream to be used for the *stderr*. A second connection is made to the specified port on the client machine. The source port of this second connection is also in the range 0 to 1023.
4. The server checks the client source address. If the address is associated with an unknown host [see *hosts (5x)*], the server aborts the connection.
5. The initial socket retrieves a null-terminated user name (16 characters maximum). This is interpreted as the user identity on the server machine.
6. The initial socket retrieves a null-terminated user name (16 characters maximum). This is interpreted as the user on the client machine.
7. The initial socket retrieves a null-terminated command (to a shell). The maximum command length is the upper bound of the system argument list.
8. *rshd* then validates the user according to the following steps:
   a. The remote user name is looked up in the password file and a *chdir* to the user's home directory occurs. If either the lookup or *chdir* fail, the connection is terminated.
   b. If the user is not the super user (user ID 0), *rshd* looks for "equivalent" hosts in the file */etc/hosts.equiv*. If the client host name is listed, it has been authenticated.
   c. If the lookup fails, or if the user is the super user, the file *.rhosts* in the remote user's home directory is checked for the client machine name and user identity. If this lookup fails, the connection is terminated.
9. A null byte is returned on the *stderr* connection and the command line is passed to the normal login shell of the user. The shell inherits the network connections established by *rshd*.

DIAGNOSTICS

All diagnostic messages are returned on the *stderr* connection; then any

network connections are closed. An error is indicated by a 1 in the leading byte (Step 9 returns to a 0).

locuser too long
: The user name on the client machine is more than 16 characters.

remuser too long
: The user name on the remote machine is more than 16 characters.

command too long
: The command line passed exceeds the (system-configured) argument list size.

Hostname for your address unknown.
: No entry in the host name database exists for the client machine.

Login incorrect.
: No password exists for this user name.

No remote directory.
: Could not *chdir* to the home directory.

Permission denied.
: The authentication procedure described above failed.

Can't make pipe.
: The pipe needed for the *stderr* wasn't created.

Try again.
: The server did not *fork*.

/bin/sh: ...
: The user's login shell could not be started.

SEE ALSO

rcp(1x), remsh(1x)

BUGS

This authentication procedure assumes the integrity of each client machine and the connecting medium. This is insecure, but useful in an "open" environment.

NAME

ruptime – show host status of local machines

SYNOPSIS

ruptime [**–a**]

DESCRIPTION

ruptime gives a status line for each machine on the local network. The status lines are formed from packets broadcast by each host on the network once a minute. They include a count of the number of users on a system and an indication of the load on each system, if available.

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

The *rwhod(8X)* issues the broadcast packets. So if this demon is not running, all machines (including the local one) will report that this node is down.

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

FILES

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

SEE ALSO

rwho(1x), rwhod(8x)

BUGS

Not all systems keep load statistics that are usable by *rwhod(8x)*.

NAME

rwho – who is logged in on local machines

SYNOPSIS

rwho [**–a**] [**–u**]

DESCRIPTION

The *rwho* command produces output similar to *who*, but for all UNIX machines on the local network. If no report has been received from a machine for five minutes *rwho* assumes the machine is down and does not report users last known to be logged into that machine.

If a user has not typed to the system for one hour or more, the user is omitted from the output of *rwho* unless the *–a* flag is given.

Normally, information on all hosts with entries in */usr/spool/rwho* is printed.

If the *–u* flag is given, the output is sorted by username. Otherwise, it is sorted by hostname.

FILES

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

SEE ALSO

ruptime(1x), rwhod(8x)

BUGS

This is unwieldy when the number of machines on the local net is large.

NAME

rwhod – system status server

SYNOPSIS

rwhod [*kernel_name*]

DESCRIPTION

rwhod is the server that maintains the database used by the *rwho(1x)* and *ruptime(1x)* programs. Its operation is predicated on the ability to *broadcast* messages on a network.

rwhod 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 that are broadcast on a network. As a consumer of information, it listens for the status messages of other *rwhod* servers, validating them and then recording them in a collection of files located in the directory */usr/spool/rwho*.

The *rwho* server transmits and receives messages at the port *IPPORT_WHOSERVER*, which has the value 513. The messages sent and received are of the following 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];
      long wd_sendtime;
      long wd_recvtime;
      char wd_hostname[32];
      long wd_loadav[3];
      long wd_boottime;
      struct          whoent {
            struct    outmp we_utmp;
            long      we_idle;
      } wd_we[1024 / sizeof (struct whoent)];
};
```

All fields are converted to network byte order prior to transmission. If load averages are available, they represent the average number of processes in the run queue over the past 5, 10, and 15 minutes. The host name included is that returned by the *uname(2)* system call. 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(5)* entry for each nonidle terminal line and a value indicating the time since a character was last received on the terminal line.

Messages received by the *rwho* server are discarded unless they originated at a *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 60 seconds. Systemwide statistics are refreshed periodically. Every 10 minutes *rwhod* performs an *nlist(3)* on *kernel_name* (*/unix* by default) and inspects various global variables in */dev/kmem*. The repeated use of *nlist* guards against the possibility that this file is not the system image currently operating.

FILES

/usr/spool/rwho/*	Information about remote systems
/unix	Systemname (default)
/dev/kmem	Systemimage

SEE ALSO

rwho(1x), ruptime(1x)

BUGS

Should relay status information between networks. People often interpret the server's dying as a machine going down. Load averages are not available on all systems, in which case the values supplied are zero and are not shown.