NAME

rcmd, rresvport, ruserok – routines for returning a stream to a remote command

SYNOPSIS

cc [*flag* …] *file* … **–lsocket -lnsl** [*library* …]

int rcmd(char ***ahost*, **unsigned short** *inport*, **const char** **luser*, **const char** **ruser*,
    const char **cmd*, **int** **fd2p);*

int rresvport(int **port);*

int ruserok(const char **rhost*, **int** *suser*, **const char** **ruser*, **const char** **luser);*

MT-LEVEL

Unsafe

DESCRIPTION

rcmd( ) is a routine used by the super-user to execute a command on a remote machine using an authentication scheme based on reserved port numbers. **rresvport( )** is a routine which returns a descriptor to a socket with an address in the privileged port space. **ruserok( )** is a routine used by servers to authenticate clients requesting service with **rcmd**. All three functions are present in the same file and are used by the **in.rshd**(1M) server (among others).

rcmd( ) looks up the host **ahost* using **gethostbyname**(3N), returning –1 if the host does not exist. Otherwise **ahost* is set to the standard name of the host and a connection is established to a server residing at the well-known Internet port *inport*.

If the connection succeeds, a socket in the Internet domain of type **SOCK_STREAM** is returned to the caller, and given to the remote command as its standard input (file descriptor 0) and standard output (file descriptor 1). If *fd2p* is non-zero, then an auxiliary channel to a control process will be set up, and a descriptor for it will be placed in **fd2p*. The control process will return diagnostic output from the command (file descriptor 2) on this channel, and will also accept bytes on this channel as signal numbers, to be forwarded to the process group of the command. If *fd2p* is 0, then the standard error (file descriptor 2) of the remote command will be made the same as its standard output and no provision is made for sending arbitrary signals to the remote process, although you may be able to get its attention by using out-of-band data.

The protocol is described in detail in **in.rshd**(1M).

The **rresvport( )** routine is used to obtain a socket bound to a privileged port number. This socket is suitable for use by **rcmd( )** and several other routines. Privileged Internet ports are those in the range 1 to 1023. Only the super-user is allowed to bind a socket to a privileged port number. The application must pass in *port*, which must be in the range 512 to 1023. The system first tries to bind to that port number. If it fails, it then tries to bind to port numbers less than *port* until either it succeeds or port number 512 is reached.

ruserok( ) takes a remote host's name, as returned by a **gethostbyaddr( )** (see **gethostbyname**(3N)) routine, two user names and a flag indicating whether the local user's name is that of the super-user. It then checks the files **/etc/hosts.equiv** and possibly **.rhosts** in the local user's home directory to see if the request for service is allowed. **0** is returned if the machine name is listed in the **/etc/hosts.equiv** file, or the host and remote user name are found in the **.rhosts** file; otherwise **ruserok( )** returns **–1**. If the super-user flag is **1**, the checking of the **/etc/hosts.equiv** file is bypassed.

RETURN VALUES

rcmd( ) returns a valid socket descriptor on success. It returns **–1** on error and prints a diagnostic message on the standard error.

rresvport( ) returns a valid, bound socket descriptor on success. It returns **–1** on error with the global value **errno** set according to the reason for failure.

FILES

/etc/hosts.equiv	system trusted hosts and users
˜/.rhosts	user's trusted hosts and users

SEE ALSO

rlogin(1), **rsh**(1), **in.rexecd**(1M), **in.rshd**(1M), **intro**(2), **gethostbyname**(3N), **rexec**(3N)

NOTES

The error code **EAGAIN** is overloaded to mean "All network ports in use."

These interfaces are unsafe in multithreaded applications. Unsafe interfaces should be called only from the main thread.