NAME

rcmd, **rresvport**, **iruserok**, **ruserok**, **rcmd_af**, **rresvport_af**, **iruserok_sa** - routines for returning a stream to a remote command

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <unistd.h>

int
rcmd(*char **ahost, int inport, const char *locuser, const char *remuser, const char *cmd, int *fd2p*);

int
rresvport(*int *port*);

int
iruserok(*u_long raddr, int superuser, const char *ruser, const char *luser*);

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

int
rcmd_af(*char **ahost, int inport, const char *locuser, const char *remuser, const char *cmd, int *fd2p, int af*);

int
rresvport_af(*int *port, int af*);

int
iruserok_sa(*const void *addr, int addrlen, int superuser, const char *ruser, const char *luser*);

DESCRIPTION

The **rcmd**() function is used by the super-user to execute a command on a remote machine using an authentication scheme based on reserved port numbers. The **rresvport**() function returns a descriptor to a socket with an address in the privileged port space. The **ruserok**() function is used by servers to authenticate clients requesting service with **rcmd**(). All three functions are present in the same file and are used by the rshd(8) server (among others).

The **rcmd**() function looks up the host **ahost* using gethostbyname(3), 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 stdin and stdout. 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 (unit 2) on this channel, and will also accept bytes on this channel as being UNIX signal numbers, to be forwarded to the process group of the command. If *fd2p* is 0, then the stderr (unit 2 of the remote command) will be made the same as the stdout 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 rshd(8).

The **rresvport**() function is used to obtain a socket to which an address with a Privileged Internet port is bound. This socket is suitable for use by **rcmd**() and several other functions. Privileged Internet ports are those in the range 0 to 1023. Only the super-user is allowed to bind an address of this sort to a socket.

The **iruserok**() and **ruserok**() functions take a remote host's IP address or name, as returned by the gethostbyname(3) routines, two user names and a flag indicating whether the local user's name is that of the super-user. Then, if the user is *NOT* the super-user, it checks the */etc/hosts.equiv* file. If that lookup is not done, or is unsuccessful, the *.rhosts* in the local user's home directory is checked to see if the request for service is allowed.

If this file does not exist, is not a regular file, is owned by anyone other than the user or the super-user, or is writable by anyone other than the owner, the check automatically fails. Zero is returned if the machine name is listed in the "*hosts.equiv*" file, or the host and remote user name are found in the "*.rhosts*" file; otherwise **iruserok**() and **ruserok**() return -1. If the local domain (as obtained from gethostname(3)) is the same as the remote domain, only the machine name need be specified.

The **iruserok**() function is strongly preferred for security reasons. It requires trusting the local DNS at most, while the **ruserok**() function requires trusting the entire DNS, which can be spoofed.

The functions with an "_af" or "_sa" suffix, i.e., **rcmd_af**(), **rresvport_af**() and **iruserok_sa**(), work the same as the corresponding functions without a suffix, except that they are capable of handling both IPv6 and IPv4 ports.

The "_af" suffix means that the function has an additional *af* argument which is used to specify the address family, (see below). The *af* argument extension is implemented for functions that have no

binary address argument. Instead, the *af* argument specifies which address family is desired.

The "_sa" suffix means that the function has general socket address and length arguments. As the socket address is a protocol independent data structure, IPv4 and IPv6 socket address can be passed as desired. The *sa* argument extension is implemented for functions that pass a protocol dependent binary address argument. The argument needs to be replaced with a more general address structure to support multiple address families in a general way.

The functions with neither an "_af" suffix nor an "_sa" suffix work for IPv4 only, except for **ruserok**() which can handle both IPv6 and IPv4. To switch the address family, the *af* argument must be filled with AF_INET, or AF_INET6. For **rcmd_af**(), PF_UNSPEC is also allowed.

ENVIRONMENT

RSH When using the **rcmd**() function, this variable is used as the program to run instead of rsh(1).

DIAGNOSTICS

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

The **rresvport**() function 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. The error code EAGAIN is overloaded to mean ‘‘All network ports in use.’’

SEE ALSO

rlogin(1), rsh(1), intro(2), rlogind(8), rshd(8)

W. Stevens and M. Thomas, *Advanced Socket API for IPv6*, RFC2292.

W. Stevens, M. Thomas, and E. Nordmark, *Advanced Socket API for IPv6*, RFC3542.

HISTORY

Most of these functions appeared in 4.2BSD. The **rresvport_af**() function appeared in RFC2292, and was implemented by the WIDE project for the Hydrangea IPv6 protocol stack kit. The **rcmd_af**() function appeared in draft-ietf-ipngwg-rfc2292bis-01.txt, and was implemented in the WIDE/KAME IPv6 protocol stack kit. The **iruserok_sa**() function appeared in discussion on the IETF ipngwg mailing list, and was implemented in FreeBSD 4.0.