

NAME

`rcmd`, `rresvport`, `iruserok`, `ruserok`, `rcmd_af`, `rresvport_af`, `iruserok_af`, `ruserok_af` – routines for returning a stream to a remote command

LIBRARY

Standard C library (*libc*, *-lc*)

SYNOPSIS

```
#include <netdb.h> /* Or <unistd.h> on some systems */

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

int rresvport(int *port);

int iruserok(uint32_t 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 **restrict ahost, unsigned short inport,
             const char *restrict locuser,
             const char *restrict remuser,
             const char *restrict cmd, int *restrict fd2p,
             sa_family_t af);

int rresvport_af(int *port, sa_family_t af);

int iruserok_af(const void *restrict raddr, int superuser,
                const char *restrict ruser, const char *restrict luser,
                sa_family_t af);
int ruserok_af(const char *rhost, int superuser,
               const char *ruser, const char *luser,
               sa_family_t af);
```

Feature Test Macro Requirements for glibc (see `feature_test_macros(7)`):

`rcmd()`, `rcmd_af()`, `rresvport()`, `rresvport_af()`, `iruserok()`, `iruserok_af()`, `ruserok()`, `ruserok_af()`:

Since glibc 2.19:

`_DEFAULT_SOURCE`

glibc 2.19 and earlier:

`_BSD_SOURCE`

DESCRIPTION

The `rcmd()` function is used by the superuser to execute a command on a remote machine using an authentication scheme based on privileged port numbers. The `rresvport()` function returns a file descriptor to a socket with an address in the privileged port space. The `iruserok()` and `ruserok()` functions are used by servers to authenticate clients requesting service with `rcmd()`. All four functions are used by `thersh(8)` server (among others).

rcmd()

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 nonzero, then an auxiliary channel to a control process will be set up, and a file 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)**.

rresvport()

The **rresvport()** function is used to obtain a socket with a privileged port bound to it. This socket is suitable for use by **rcmd()** and several other functions. Privileged ports are those in the range 0 to 1023. Only a privileged process (on Linux, a process that has the **CAP_NET_BIND_SERVICE** capability in the user namespace governing its network namespace) is allowed to bind to a privileged port. In the glibc implementation, this function restricts its search to the ports from 512 to 1023. The *port* argument is value-result: the value it supplies to the call is used as the starting point for a circular search of the port range; on (successful) return, it contains the port number that was bound to.

iruserok() and ruserok()

The **iruserok()** and **ruserok()** functions take a remote host's IP address or name, respectively, two usernames and a flag indicating whether the local user's name is that of the superuser. Then, if the user is *not* the superuser, 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 superuser, is writable by anyone other than the owner, or is hardlinked anywhere, the check automatically fails. Zero is returned if the machine name is listed in the *hosts.equiv* file, or the host and remote username are found in the *.rhosts* file; otherwise **iruserok()** and **ruserok()** return -1 . If the local domain (as obtained from **gethostname(2)**) is the same as the remote domain, only the machine name need be specified.

If the IP address of the remote host is known, **iruserok()** should be used in preference to **ruserok()**, as it does not require trusting the DNS server for the remote host's domain.

***_af() variants**

All of the functions described above work with IPv4 (**AF_INET**) sockets. The "*_af*" variants take an extra argument that allows the socket address family to be specified. For these functions, the *af* argument can be specified as **AF_INET** or **AF_INET6**. In addition, **rcmd_af()** supports the use of **AF_UNSPEC**.

RETURN VALUE

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. On failure, it returns -1 and sets *errno* to indicate the error. The error code **EA_GAI** is overloaded to mean: "All network ports in use".

For information on the return from **ruserok()** and **iruserok()**, see above.

VERSIONS

The functions **iruserok_af()**, **rcmd_af()**, **rresvport_af()**, and **ruserok_af()** functions are provided since glibc 2.2.

ATTRIBUTES

For an explanation of the terms used in this section, see **attributes(7)**.

Interface	Attribute	Value
rcmd() , rcmd_af()	Thread safety	MT-Unsafe
rresvport() , rresvport_af()	Thread safety	MT-Safe
iruserok() , ruserok() , iruserok_af() , ruserok_af()	Thread safety	MT-Safe locale

STANDARDS

Not in POSIX.1. Present on the BSDs, Solaris, and many other systems. These functions appeared in 4.2BSD. The "*_af*" variants are more recent additions, and are not present on as wide a range of systems.

BUGS

iruserok() and **iruserok_af()** are declared in glibc headers only since glibc 2.12.

SEE ALSO

rlogin(1), **rsh(1)**, **rexec(3)**, **rexecd(8)**, **rlogind(8)**, **rshd(8)**