NAME

rmtopen, rmtclose, rmtread, rmtwrite, rmtseek, rmtioctl - operate on a connection to a remote tape server (**-lrmt**)

SYNOPSIS

#include <schily/librmt.h>
#include <schily/rmtio.h>

int rmtopen(int remfd, char *pathname, int fmode);

int rmtclose(int remfd);

int rmtread(int remfd, char *buf, int count);

int rmtwrite(int remfd, char *buf, int count);

int rmtseek(int remfd, off_t offset, int whence);

int rmtioctl(int remfd, int cmd, int count);

DESCRIPTION

rmtopen()

sends an open request to the remote server, *remfd* is a file descriptor previously obtained from a call to **rmtgetconn**(3), *pathname* is a path to be opened on the server side and *fmode* is the same as the third argument for **open**(2). If the remote server does not support the GNU open flag enhancements, only the lowest two bits in the open flags can be send to the remote server.

If the open requests succeeds, **rmtopen**(3) takes care of telling the remote side to switch to **RMT protocol VERSION 1**

rmtclose()

sends a close request to the remote server, *remfd* is a file descriptor previously obtained from a call to **rmtgetconn**(3).

rmtread()

sends a read request to the remote server, *remfd* is a file descriptor previously obtained from a call to **rmtgetconn**(3). The other parameters are the same as for a local **read**(2) call. **rmtread**(3) will fail if there was no previous successful **rmtopen**(3) before.

rmtwrite()

sends a write request to the remote server, *remfd* is a file descriptor previously obtained from a call to **rmtgetconn**(3). The other parameters are the same as for a local **write(2)** call. **rmtwrite**(3) will fail if there was no previous successful **rmtopen**(3) before.

rmtseek()

sends a seek request to the remote server, *remfd* is a file descriptor previously obtained from a call to **rmtgetconn**(3). The other parameters are the same as for a local **lseek**(2) call. **rmtseek**(3) will fail if there was no previous successful **rmtopen**(3) before.

rmtioctl()

sends a **MTIOCTOP** request to the remote server, **cmd** and **count** are the values that should be filled into **struct mtop**. See **mtio(4)** for more information.

RETURN VALUES

rmtopen()

returns a value >= 0 if the remote open succeeds.

rmtclose()

returns a value >= 0 if the remote close succeeds.

rmtread()

returns the return value obtained the remote **read**(2) request.

rmtwrite()

returns the return value obtained by the remote **write**(2) request.

rmtseek()

returns the return value obtained by the remote **lseek**(2) request.

rmtioctl()

returns the return value obtained by the remote **ioctl(f, MTIOCTOP, struct mtop *)** call.

ERRORS

All functions return -1 on error and set **errno** to the **errno** value retrieved from the remote server.

EXAMPLES

```
int     remfd;
char    *remfn;
char    host[256];
```

```
int     iosize = 10240;  /* socket send/receive size to set up */


if ((remfn = rmtfilename(filename)) != NULL) {
        rmthostname(host, sizeof (host), filename);

        if ((remfd = rmtgetconn(host, iosize, 0)) < 0)
                comerrno(EX_BAD, "Cannot get connection to '%s'.\n",
                        /* errno not valid !! */            host);
}

if (rmtopen(remfd, remfn, mode) < 0)
        comerr("Cannot open '%s'.\n", remfn);

if (rmtread(remfd, buf, sizeof(buf)) < 0)
        comerr("Read error on '%s'.\n", remfn);

rmtclose(remfd);
```

SEE ALSO

rmt(1), **rsh**(1), **ssh**(1), **rcmd**(3), **rmtinit**(3), **rmtdebug**(3), **rmtrmt**(3), **rmtrsh**(3), **rmthostname**(3), **rmtfilename**(3), **rmtgetconn**(3), **rmtopen**(3), **rmtioctl**(3), **rmtclose**(3), **rmtread**(3), **rmtwrite**(3), **rmtseek**(3), **rmtxstatus**(3), **rmtstatus**(3), **_mtg2rmtg**(3), **_rmtg2mtg**(3), **errmsgno**(3), **mtio**(4)

NOTES

While all other known **rmt** implementations limit the size of a single **rmt** command to 64 bytes, this implementation limits all standard command lines to 80 bytes and the file name to 4096 bytes.

Note that this may cause problems if the remote **rmt** server implementation is directly derived from the historic BSD server code. This is true for all BSD systems, for SunOS up to 5.9 and even for the GNU **rmt** server.

As some of the other **rmt** servers do not even implement bound checking, be prepared that other server implementations may dump core or at least stop honoring the **rmt** protocol.

If you like to use long file names, make sure that you also use the **schily rmt** server.

BUGS

If local and remote **errno** values do not match, programs may get confused.

Mail other bugs and suggestions to **schilytools@mlists.in-berlin.de** or open a ticket at **https://codeberg.org/schilytools/schilytools/issues**.

The mailing list archive may be found at:

https://mlists.in-berlin.de/mailman/listinfo/schilytools-mlists.in-berlin.de.

AUTHOR

librmt has been written in 1990 by Joerg Schilling. In 1995, support for **RMT VERSION 1** has been added. **librmt** is now maintained by the schilytools project authors.

SOURCE DOWNLOAD

The source code for **librmt** is included in the **schilytools** project and may be retrieved from the **schilytools** project at Codeberg at

https://codeberg.org/schilytools/schilytools.

The download directory is

https://codeberg.org/schilytools/schilytools/releases.