```
        for (i=0; i<ntohl(resp.rf_len); i++) {
                *buff++ = resp.rf_data[i];
        }
        rfptr->rfpos += ntohl(resp.rf_len);

        signal(Rf_data.rf_mutex);
        return ntohl(resp.rf_len);
}
```

Rflread begins by checking argument *count* to verify that the request is in range. It then verifies that the pseudo-device has been opened and the mode allows reading. Once the checking is complete, *rflread* performs the *read* operation: it forms a message, uses *rfscomm* to transmit a copy to the server and receive a response, and interprets the response.

If *rfscomm* returns a valid response, the message will include the data that has been read. *Rflread* copies the data from the response message into the caller's buffer, updates the file position, and returns the number of bytes to the caller.

20.13 Writing To A Remote File (rflwrite)

Writing to a remote file follows the same general paradigm as reading from a remote file. Driver function *rflwrite* performs the *write* operation; the code can be found in file *rflwrite.c*:

```
/* rflwrite.c - rflwrite */

#include <xinu.h>

/*------------------------------------------------------------------------
 * rflwrite  -  Write data to a remote file
 *------------------------------------------------------------------------
 */
devcall rflwrite (
          struct dentry *devptr,        /* Entry in device switch table */
          char  *buff,                  /* Buffer of bytes              */
          int32 count                   /* Count of bytes to write      */
        )
{
        struct  rflcblk *rfptr;         /* Pointer to control block     */
        int32   retval;                 /* Return value                 */
        struct  rf_msg_wreq  msg;       /* Request message to send      */
        struct  rf_msg_wres resp;       /* Buffer for response          */
        char    *from, *to;             /* Used to copy name            */
```