Reading and writing bytes through a TCP connection in C language

Both reading and writing requests over TCP connected sockets may be only partially executed, this means not all bytes requested to be read or written are effectively read or written.

Functions and methods used for reading and writing a number of bytes return the number of bytes effectively read and written. If the return value is less than the requested, the remaining bytes must be read or written afterwards.

The following functions in C language ensure all requested bytes are effectively read and written. In Java language similar methods can be defined.

The readExactTimeOut() function also sets a timeout for the read operation.

```c
// returns zero on read error or timeout, on success returns 1
int readExactTimeOut(int sock, unsigned char *dataBuff, int dataLen, int timeoutSec) {
unsigned char *aux=dataBuff;
int r;
fd_set rs;
struct timeval to;

to.tv_usec=0; to.tv_sec=timeoutSec;
while(dataLen) {
        FD_ZERO(&rs); FD_SET(sock,&rs);
        r = select(sock+1,&rs,NULL,NULL,&to);
        if(r!=1) return(0); // timeout
        r = read(sock,aux,dataLen);
        if(r<1) return(0);  // read error or nothing read
        aux = aux+r; dataLen = dataLen-r;
        }
return(1);
}

// returns zero on write error, on success returns 1
int writeExact(int sock, unsigned char *dataBuff, int dataLen) {
unsigned char *aux=dataBuff;
int w;
while(dataLen) {
        w = write(sock,aux,dataLen);
        if(w<1) return(0);  // write error or nothing written
        aux = aux+w; dataLen = dataLen-w;
        }
return(1);
}
```