4.9 close Function

The normal Unix close function is also used to close a socket and terminate a TCP connection.

```
#include <unistd.h>

int close (int sockfd);
```

Returns: 0 if OK, -1 on error

The default action of close with a TCP socket is to mark the socket as closed and return to the process immediately. The socket descriptor is no longer usable by the process: It cannot be used as an argument to read or write. But, TCP will try to send any data that is already queued to be sent to the other end, and after this occurs, the normal TCP connection termination sequence takes place (Section 2.6).

In Section 7.5, we will describe the SO_LINGER socket option, which lets us change this default action with a TCP socket. In that section, we will also describe what a TCP application must do to be guaranteed that the peer application has received any outstanding data.

Descriptor Reference Counts

At the end of Section 4.8, we mentioned that when the parent process in our concurrent server closes the connected socket, this just decrements the reference count for the descriptor. Since the reference count was still greater than 0, this call to close did not initiate TCP's four-packet connection termination sequence. This is the behavior we want with our concurrent server with the connected socket that is shared between the parent and child.

If we really want to send a FIN on a TCP connection, the shutdown function can be used (Section 6.6) instead of close. We will describe the motivation for this in Section 6.5.

We must also be aware of what happens in our concurrent server if the parent does not call close for each connected socket returned by accept. First, the parent will eventually run out of descriptors, as there is usually a limit to the number of descriptors that any process can have open at any time. But more importantly, none of the client connections will be terminated. When the child closes the connected socket, its reference count will go from 2 to 1 and it will remain at 1 since the parent never closes the connected socket. This will prevent TCP's connection termination sequence from occurring, and the connection will remain open.