```
/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++

$Id: tcp.c,v 1.10 2001/08/10 01:59:33 davidc Exp $

Copyright (c) 2001 BeComm Corporation

Filename:

    tcp.c

Abstract:

    The posix implementation of the TCP portion of the SocketLib API.

Owner:

    David Costanzo (davidc)

----------------------------------------------------------------------------
----*/
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
#include <assert.h>
#include <errno.h>

#include <sys/time.h>
#include <unistd.h>

#include <netdb.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>

#define SOS_DEBUG_ZONE "/classes/socket/tcp"
#include <sosstrings.h>
#include <sosisocket.h>
#include "socket.h"
#include "tcp.h"

SOS_SOURCE_VERSION("$Id: tcp.c,v 1.10 2001/08/10 01:59:33 davidc Exp $");

/*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++
```

```
43 Constants
44 ------------------------------------------------------------------------
…  ----*/
45
46 #define ERROR_BIND_FAILED           SOS_Error
47 #define ERROR_SOCKET_FAILED         SOS_Error
48 #define ERROR_BIND_FAILED           SOS_Error
49 #define ERROR_CONNECT_FAILED        SOS_Error
50 #define ERROR_GETSOCKNAME_FAILED    SOS_Error
51
52
53 /* size of buffer to allocate for each read */
54 static const size_t g_TcpRecvSize = 1500;
55
56 /* largest amount to call send() with */
57 static const size_t g_MaxSendSize = 2048;
58
59 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
…  ++++++
60 Data Types
61 ------------------------------------------------------------------------
…  ----*/
62
63 /* TCP socket context to be passed back and
64    forth with caller of this library */
65 struct _SOS_ISOCKET_TCP {
66     int                         NativeSocket;
67
68     pthread_mutex_t             Mutex;          /* non-recursive */
69     int                         RefCount;
70
71     SOS_ISOCKET_TCP_CONNECT     ConnectedCallback;
72     SOS_ISOCKET_TCP_RECEIVE     RecvCallback;
73     SOS_ISOCKET_CLOSED          ClosedCallback;
74     SOS_ISOCKET_ADDRESS         SocketAddr;
75     void*                       UserContext;
76
77     SOS_BOOLEAN                 PeerHasClosed;
78     SOS_BOOLEAN                 WeHaveClosed;
79
80     pthread_t                   NativeConnectionThread;
81 };
82
83
```

```
85      SOS_ISOCKET_TCP     Socket;
86
87      pthread_t           NativeListenThread;
88  };
89
90
91  /* TcpConnectionWorkerProc types */
92
93  typedef struct _CONNECTION_PARAM            CONNECTION_PARAM;
94
95  typedef enum _TCP_CONNECT_FUNCTION_ID {
96      TCP_CONNECT_FUNCTION_ID_TcpOnConnect,
97      TCP_CONNECT_FUNCTION_ID_TcpOnReceive,
98      TCP_CONNECT_FUNCTION_ID_TcpOnClose,
99  } TCP_CONNECT_FUNCTION_ID;
100
101 struct _CONNECTION_PARAM {
102
103     TCP_CONNECT_FUNCTION_ID  FunctionId;
104
105     union _CONNECTION_PARAMS {
106
107         struct _CONNECTION_PARAM_ON_CONNECT {
108             SOS_ISOCKET_TCP *       Socket;
109             void *                  OutUserContext;
110         } OnConnect;
111
112
113         struct _CONNECTION_PARAM_ON_RECEIVE {
114             SOS_ISOCKET_TCP *       Socket;
115             void*                   Buffer;
116             size_t                  BufferLength;
117             SOS_FREEPROC            BufferFree;
118         } OnReceive;
119
120
121         struct _CONNECTION_PARAM_ON_CLOSE {
122             SOS_ISOCKET_TCP *       Socket;
123         } OnClose;
124
125     } Params;
126 };
127
128
/* TcpSendWorkerProc types */
```

```
130
131 typedef struct _TCP_SEND_WORKER_PROC TCP_SEND_WORKER_PROC;
132
133 struct _TCP_SEND_WORKER_PROC {
134     int         NativeSocket;
135     void *      Buffer;
136     size_t      BufferLength;
137     SOS_STATUS  OutStatus;
138 };
139
140
141 /*++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
… ++++++
142 Internal Routines
143 ----------------------------------------------------------------------
… ----*/
144
145
146 /*++
147 Routine Name:
148
149     TcpSocketCreate
150
151 Routine Description:
152
153     This routine allocates and initializes an SOS_ISOCKET_TCP struct.
154     It does not call any sockets routines.
155     The structure is returned with a reference count of 1.
156
157 Parameters:
158
159     SOCKET NativeSocket - [in]
160         The native socket handle.
161
162     SOS_ISOCKET_TCP_CONNECT ConnectedCallback - [in]
163         The user-defined function to call when a socket connection
164         has been established.
165
166     SOS_ISOCKET_TCP_RECEIVE RecvCallback - [in]
167         The user-defined RecvCallback.
168
169     SOS_ISOCKET_CLOSED ClosedCallback - [in]
170         The user-defined ClosedCallback.
171
172     const struct sockaddr_in* RemoteSockAddr - [in]
```

```
173         The remote socket address.
174         This parameter is optional.
175
176     const SOS_ISOCKET_ADDRESS* SocketAddr - [in]
177         The socket address.
178
179     void* UserContext - [in]
180         The user context to pass into RecvCallback and ClosedCallback.
181
182 Return Value:
183
184     SOS_ISOCKET_TCP* -
185         A pointer to a newly allocated SOS_ISOCKET_TCP structure with
186         a reference count of 1.
187         NULL, if the structure could not be created.
188
189 --*/
190 static
191 SOS_ISOCKET_TCP*
192 TcpSocketCreate(
193     int                         NativeSocket,
194     SOS_ISOCKET_TCP_CONNECT     ConnectedCallback,
195     SOS_ISOCKET_TCP_RECEIVE     RecvCallback,
196     SOS_ISOCKET_CLOSED          ClosedCallback,
197     const SOS_ISOCKET_ADDRESS * SocketAddr,
198     void*                       UserContext
199 )
200 {
201     SOS_ISOCKET_TCP*  tcpSocket;
202
203     /* allocate enough size for the larger SOS_ISOCKET_TCP_LISTEN */
204     tcpSocket = malloc(sizeof(SOS_ISOCKET_TCP_LISTEN));
205     if (tcpSocket != NULL) {
206
207         int initStatus;
208
209         initStatus = pthread_mutex_init(&tcpSocket->Mutex, NULL);
210         if (initStatus == 0) {
211
212             tcpSocket->NativeSocket      = NativeSocket;
213             tcpSocket->ConnectedCallback = ConnectedCallback;
214             tcpSocket->ClosedCallback    = ClosedCallback;
215             tcpSocket->RecvCallback      = RecvCallback;
216             tcpSocket->SocketAddr        = *SocketAddr;
```

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.