FUJITSU Software BS2000 POSIX-SOCKETS

Version V10.0A
October 2016

Readme

1 Introduction

This readme file contains the known changes to the following manual:

POSIX
SOCKETS/XTI for POSIX
Edition March 2005

2 Product Components and Installation

2.1 BS2000 Installation

Following BS2000 files may be installed by IMON:

	MODE	SHARE	ACCESS	MIGRATE
$<TSOS>.SYSLIB.POSIX-SOCKETS.100	COM	Y	R	I
$<TSOS>.SKMLIB.POSIX-SOCKETS.100	X86	Y	R	I
$<TSOS>.SYSRME.POSIX-SOCKETS.100.D	COM	Y	R	I
$<TSOS>.SYSRME.POSIX-SOCKETS.100.E	COM	Y	R	I
$<TSOS>.SYSSII.POSIX-SOCKETS.100	COM	Y	R	A
$<TSOS>.SYSLNK.POSIX-SOCKETS.100.PTH	COM	Y	R	I
$<TSOS>.SKULNK.POSIX-SOCKETS.100.PTH	X86	Y	R	I

- SYSLIB.POSIX-SOCKETS.100 contains header files and modules in the /390 format for compilation and linking software by means of BS2000 and for the installation of the Sockets library /usr/lib/libsocket.a to the POSIX file system.

- SKMLIB.POSIX-SOCKETS.100 contains header files and modules in the X86 format for compilation and linking software by means of BS2000 and for the installation of the Sockets library /usr/lib/libsocket.a resp. /usr/lib/X86/libsocket.a to the POSIX file system.

- SxxLNK.POSIX-SOCKETS.100.PTH contains the PThread variants of the modules (as a LLM SOCKGM).
  They will be loaded dynamically at run-time by certain POSIX-SOCKETS applications. These variants are not installed in the POSIX filesystem.

2.2 Installation to the POSIX File System

The installation to the POSIX file system has to be done using the POSIX installation program. (/START-POSIX-INSTALLATION, function Install Packages on POSIX). See manual "POSIX Basics" for more details.

The installation to the POSIX file system requires about 7 MB of temporary disk space in /tmp and about 7 MB of permanent disk space per each installed variant in /usr/lib or /usr/lib/X86.

The installation will take some time.

If the installation should not have been executed correctly, the text file /tmp/install.libsocket.err would contain notes on the error reasons.

Installed Libraries:

If only the BS2000 file SYSLIB.POSIX-SOCKETS.100 is installed, only the library `/usr/lib/libsocket.a` with the /390 code variant will be created in the POSIX file system.

If several variants of the BS2000 libraries are installed, several variants of the Sockets libraries will be created in the POSIX file system, too:

`/usr/lib/libsocket.a`	with /390 format
`/usr/lib/X86/libsocket.a`	with X86 format

In this case the linking with the X86 variant can be done by using the compiler option
`-L /usr/lib/X86`

Installed Files:

```
/usr/lib/libsocket.a
/usr/lib/libxnet.a
/usr/include/netdb.h
/usr/include/arpa/inet.h
/usr/include/arpa.inet.h
/usr/include/net/if.h
/usr/include/net.if.h
/usr/include/netinet/in.h
/usr/include/netinet.in.h
/usr/include/sys/byteorder.h
/usr/include/sys.byteorder.h
/usr/include/sys/netconfig.h
/usr/include/sys.netconfig.h
/usr/include/sys/socket.h
/usr/include/sys.socket.h
/usr/include/sys/sockio.h
/usr/include/sys.sockio.h
/usr/include/sys/un.h
/usr/include/sys.un.h
/usr/include/sys/xti_inet.h
/usr/include/sys.xti_inet.h
/usr/include/xti.h
```

Activation of multicast functions in netinet/in.h by:
`#define IP_MULTICAST`

3 Functions

3.1 Additional Functions

In addition to the functions described in the manual the following functions are supported, too:

```
bindresvport( int sd , struct sockaddr_in *sin )

rcmd( char **ahost , u_short rport , char *locuser ,
          char *remuser , char *cmd , int *fd2p )

rcmd_af( char **ahost , u_short rport , char *locuser ,
          char *remuser , char *cmd , int *fd2p , int af )

rexec( char **ahost , int rport , char *name , char *pass ,
          char *cmd , int *fd2p )

s_fcntl( int des , int cmd , int arg ) /* same as fcntl() */
s_ioctl( int des , int cmd , int arg ) /* same as ioctl() */

void s_set_uppernameatgethostbyname( int mode ) /* bs2 special */

int t_rcvv(int, struct t_iovec *, unsigned int, int *)
int t_rcvreldata(int, struct t_discon *)
int t_rcvvudata(int, struct t_unitdata *, struct t_iovec *,
         unsigned int, int *)

int t_sndv(int, const struct t_iovec *, unsigned int, int)
int t_sndreldata(int, struct t_discon *)
int t_sndvudata(int, struct t_unitdata *, struct t_iovec *,
         unsigned int)

int t_sysconf(int)
```

The function `getsockopt()` supports following option not described in the manual:

```
getsockopt(fd, SOL_SOCKET, SO_BS2ERROR, &optval, &optlen);
```

The variable '`optval`' has to be a '`struct so_bs2error`'. The function call provides the internal error code (BCAM return code) and the corresponding 'errno' of the last error that appeared for this socket. Different from `getsockopt(..., SO_ERROR, ...)` that error code will not be reset after the query.
Both, '`SO_BS2ERROR`' and '`struct so_bs2error`' are defined in `sys/socket.h`.

The functions `getsockopt()` and `setsockopt()` support following options not described in the manual:

```
getsockopt(fd, IPPROTO_TCP, TCP_KEEPIDLE,  &optval, &optlen);
getsockopt(fd, IPPROTO_TCP, TCP_KEEPINTVL, &optval, &optlen);
getsockopt(fd, IPPROTO_TCP, TCP_KEEPCNT,   &optval, &optlen);

setsockopt(fd, IPPROTO_TCP, TCP_KEEPIDLE,  &optval, optlen);
setsockopt(fd, IPPROTO_TCP, TCP_KEEPINTVL, &optval, optlen);
setsockopt(fd, IPPROTO_TCP, TCP_KEEPCNT,   &optval, optlen);
```

The type of the parameter '`optval`' must be '`int *`' in all cases. The options modify the behavior if sending of control messages is enabled by the option SOL_SOCKET/SO_KEEPALIVE.

TCP_KEEPIDLE
This option defines the time (in seconds) the connection has to be idle before the first control message is sent. Valid range: 120 to 32767 seconds.

TCP_KEEPINTVL
This option defines the interval (in seconds) after which the sending of control messages is repeated. Valid range: 120 to 32767 seconds.
The transport system BCAM does not support different values for the options TCP_KEEPIDLE and TCP_KEEPINTVL. It uses the lower one of the two values if both are non-zero.

TCP_KEEPCNT
This option shall define the number of control message after which the connection will be dropped. Valid range: 1 to 127.
Even though this option is supported by POSIX-SOCKETS, it has no influence on the behavior of the transport systems BCAM.

The function ioctl() supports following option not described in the manual (as of BCAM V23):

Request	*arg	Function
SIOCGIFHWADDR	struct ifreq	Get hardware address (MAC) of the interface

SIOCGIFHWADDR
The hardware address (MAC) is returned in the *ifr_hwaddr* member for the interface specified
with the *ifr_name* member.
Restriction: The hardware address is returned, which was valid at startup of the POSIX subsystem.
Dynamic changes are not taken into account.

The restriction for IPv6 described for the function *ioctl()* with *SIOCGLIFNETMASK* does no longer apply (as of BCAM V23):

SIOCGLIFNETMASK
The IPv4 subnetwork mask or the IPv6 prefix is returned in the *lifr_addr* member for the interface specified with the *lifr_name* member.

Modifications of the network interface configuration

If dynamic modifications of the network interface configuration of the transport system BCAM take place, i.e. adding or removing an interface or changing parameters of an interface, there is no longer a restart of the POSIX subsystem required to make these modifications effective for POSIX sockets.

Improvement of the close processing of connected sockets

The previous restrictions on the 'close()' for connected sockets have been removed. A "true graceful disconnect" is now supported and the option SO_LINGER with timeout > 0 no longer leads to delays in the 'close ()'.

3.2 Multihoming

At hosts with multiple IP interfaces (IP addresses) POSIX-SOCKETS applications are able to bind separate server sockets for each interface to the same port at a time. Thus every server socket processes only those requests arriving via the interface the socket was bound to.

Instead of that (but not at the same time) a server socket can process request to a port arriving via any interface (`INADDR_ANY`).

Restriction:

At a time it is not possible to bind two sockets to one port processing requests independent on the interface one for IPv4 (`INADDR_ANY`) only and the other for IPv6 (`IN6ADDR_ANY`) only.

3.3 File Descriptor Transfer

The functions `sendmsg()`/`recvmsg()` can be used to transmit/receive file descriptors using either the `msg_accrights` array or a `SOL_SOCKET/SCM_RIGHTS` control message.

Only socket file descriptors cat be transmitted/received.

For further details read the description in the "`sys/socket.h`" header file.

3.4 Functional Dependencies to another Products

The functions `gethostbyname()` and `gethostbyaddr()` do support DNS (Domain Name Service) if the DNS resolver of the product InternetServices is running or the subsystem SOC6 of the product OpenNetServer is loaded.

The functions `getipnodebyname()` and `getipnodebyaddr()` do support DNS for IPv4 and IPv6 addresses if the subsystem SOC6 is loaded.

3.5 Omitted Functions

The function `gethostname()` is no longer contained in POSIX-SOCKETS but only in CRTE and CRTE-BASYS.

4 Restrictions

4.1 Using Sockets of Domain AF_UNIX

The system wide limit is 500.

The adjustments and limits of the transport system BCAM are also valid for these sockets.

4.2 Coexistence of Sockets of Domains AF_INET and AF_INET6

In general at one port sockets of the domains AF_INET and AF_INET6 can coexist.

This does not apply if one socket is bound to the address `INADDR_ANY (0.0.0.0)` and another socket is to be bound to the address `IN6ADDR_ANY (::0)`. The second bind() call would be rejected in this case with `EADDRINUSE`.

Workaround:

The application can bind a socket to the `IN6ADDR_ANY (::0)` address which processes request from both domains. Request from the `AF_INET` domain will be delivered to that socket with IPv4-mapped addresses `(::ffff:a.b.c.d)`.