NETLIB Programmer's Guide

January, 2002

This manual contains information about NETLIB, a library for TCP/IP network programming on VMS systems.

Revision/Update Information:	This is an updated manual.
Operating System and Version:	VAX/VMS V5.2 or later; OpenVMS Alpha V1.5 or later
Software Version:	NETLIB V2.3C

Matthew Madison
MadGoat Software

20 January 2002

Permission is granted to copy and redistribute this document for no commercial gain.

The information in this document is subject to change without notice and should not be construed as a commitment by the author. The author assumes no responsibility for any errors that may appear in this document.

DISCLAIMER: The author, the author's employer, and MadGoat Software make no representations or warranties with respect to the contents hereof and specifically disclaim any implied warranties of merchantability or fitness for any particular purpose.

AXP, DEC, VAX, VMS, and OpenVMS are trademarks of Digital Equipment Corporation.

UNIX is a trademark of Unix System Laboratories, Inc.

MultiNet and TCPware are registered trademarks of Process Software Corporation.

PathWay is a trademark of Attachmate, Inc.

Copyright ©2002 MadGoat Software. All Rights Reserved.

Contents

PREFACE v

CHAPTER 1 USING NETLIB 1–1

1.1 OVERVIEW 1–1
1.1.1 NETLIB Services 1–1

1.2 HEADER FILES 1–2

1.3 NETLIB DATA STRUCTURES 1–2
1.3.1 INADDRDEF Structure 1–2
1.3.2 SINDEF Structure 1–2
1.3.3 MXRRDEF Structure 1–2
1.3.4 NETLIB_DNS_HEADER Structure 1–3

1.4 BYTE-ORDER CONSIDERATIONS 1–3

1.5 SYNCHRONOUS VS. ASYNCHRONOUS OPERATION 1–3

1.6 I/O STATUS BLOCK 1–3

1.7 LINKING NETLIB PROGRAMS 1–3

1.8 CONTACTING THE AUTHOR 1–4

Part I ROUTINE DESCRIPTIONS

NETLIB_ACCEPT RTN–3
NETLIB_ADDRESS_TO_NAME RTN–5
NETLIB_ADDRTOSTR RTN–8
NETLIB_BIND RTN–9
NETLIB_CLOSE RTN–11
NETLIB_CONNECT RTN–12
NETLIB_CONNECT_BY_NAME RTN–14
NETLIB_DNS_EXPANDNAME RTN–16

NETLIB_DNS_MX_LOOKUP RTN–18
NETLIB_DNS_QUERY RTN–20
NETLIB_DNS_SKIPNAME RTN–23
NETLIB_GETPEERNAME RTN–24
NETLIB_GETSOCKNAME RTN–26
NETLIB_GETSOCKOPT RTN–28
NETLIB_GET_HOSTNAME RTN–30
NETLIB_HTON_LONG RTN–31
NETLIB_HTON_WORD RTN–32
NETLIB_LISTEN RTN–33
NETLIB_NAME_TO_ADDRESS RTN–35
NETLIB_NTOH_LONG RTN–38
NETLIB_NTOH_WORD RTN–39
NETLIB_READ RTN–40
NETLIB_READLINE RTN–42
NETLIB_SERVER_SETUP RTN–45
NETLIB_SETSOCKOPT RTN–47
NETLIB_SHUTDOWN RTN–50
NETLIB_SOCKET RTN–52
NETLIB_STRTOADDR RTN–54
NETLIB_VERSION RTN–55
NETLIB_WRITE RTN–56
NETLIB_WRITELINE RTN–58

APPENDIX A STATUS CODES A–1

TABLES

RTN–1 Domain Name Service query types RTN–21
RTN–2 Flag values for NETLIB_DNS_QUERY RTN–21
RTN–3 Flag values for NETLIB_READLINE RTN–43
RTN–4 Socket options RTN–48
RTN–5 Shutdown types RTN–50
RTN–6 Socket types RTN–52
A–1 NETLIB Status Codes A–1

Preface

There are several TCP/IP packages available for VMS systems today. Each provides a VMS-style programming interface, using the $QIO system service, and most also provide a ‘‘socket’’ programming library, based on the communications model developed for BSD UNIX. Unfortunately, there is no standard among all of the packages for the $QIO-based programming interface (most, but not all, emulate Digital’s, at least to some degree), and the $QIO-based interface is not very easy to use.

The socket libraries provided with these packages provide a somewhat easier-to-use programming interface, but don’t permit VMS-style asynchronous programming (using ASTs), and generally require at least a re-LINK, and sometimes source modifications, when moving from one vendor’s library to another.

NETLIB was originally developed to support MadGoat Software’s Message Exchange mail package, which needed to support many TCP/IP packages doing VMS-style asynchronous programming. NETLIB provides a consistent, VMS-style interface for TCP/IP-based network programs, operating with all of the currently available TCP/IP packages available today for VMS (with one minor exception). In addition, NETLIB allows for flexibility in the use of a TCP/IP package, by selecting the vendor-dependent library code at run-time, rather than link-time.

Intended Audience

This manual is intended for programmers writing applications that use NETLIB.

Document Structure

This document consists of an introductory chapter, a section containing routine descriptions, and an appendix describing the status codes returned by NETLIB routines.

Related Documents

The *NETLIB Installation Guide* describes how to install NETLIB.

1 Using NETLIB

This chapter discusses the NETLIB programming interface.

1.1 Overview

NETLIB provides a single programming interface that can be used with almost any TCP/IP package for VMS systems. While every package provides a $QIO interface and most also provide a socket library, the advantages to using NETLIB instead are:

- NETLIB allows VMS-style asynchronous programming (using ASTs) without having to use the $QIO interface directly. The $QIO programming interface varies from vendor to vendor.
- NETLIB selects its vendor-specific support code at run-time, allowing you to switch between TCP/IP packages without re-linking your program. You can even have the same application, installed just once, run over different TCP/IP packages at the same time on different nodes in a VMScluster system.
- NETLIB provides a VMS common language environment style of programming interface, making it easier to use from other languages than a UNIX-type socket library written for C programmers.

1.1.1 NETLIB Services

NETLIB provides services for writing TCP/IP-based network programs, both client and server. Servers can either be standalone, AST-driven programs, or can be single-threaded server processes ''forked'' by the ''master server'' provided with each TCP/IP package.

NETLIB services can be grouped into four basic areas:

- **Connection services**. These routines provide the services necessary for performing TCP/IP network I/O.
- **Line-mode services**. These routines provide a line-oriented network reads and writes, for line-oriented protocols that run over TCP.
- **DNS resolver services**. These routines provide host name and address lookup services. A direct DNS query routine is also provided.
- **Utility routines**. These are routines for doing byte-order conversions, IP address parsing and formatting, and other miscellaneous services.

1.2 Header files

NETLIB includes header files for both C and BLISS programming, called NETLIBDEF.H and NETLIBDEF.R32, respectively. The header files reside in the NETLIB_DIR: directory. These files define constants and structures that are used by the NETLIB programming interface.

1.3 NETLIB Data Structures

NETLIB uses several data structures in its programming interface. Some of these data structures (INADDRDEF and SINDEF) are common to all BSD socket-oriented programming libraries, one (MXRRDEF) is special to NETLIB. The header files provided with NETLIB defines these structures for BLISS and C programmers.

1.3.1 INADDRDEF Structure

The INADDRDEF structure is used for passing an IP address. The contents is (currently) a longword, in network byte order (see Section 1.4 for more information). Its definition is (in C) is:

```
struct INADDRDEF {
 unsigned long inaddr_l_addr;
}
```

1.3.2 SINDEF Structure

The SINDEF structure defines an IP ‘‘socket,’’ which is a combination of an IP address and a TCP or UDP port number. Its definition is:

```
struct SINDEF {
 unsigned short sin_w_family;
 unsigned short sin_w_port;
    struct INADDRDEF sin_x_addr;
    unsigned char sin_x_mbz[8];
}
```

The **sin_w_family** field should always be set to NETLIB_K_AF_INET (value 2). The **sin_w_mbz** field should always be zeroed.

This is a specific form of the more general SOCKADDRDEF structure for use with IP-based sockets. NETLIB currently only supports IP network programming, so other types of addresses should not be used.

1.3.3 MXRRDEF Structure

The MXRRDEF structure is used to return MX resource record information in a call to NETLIB_DNS_MX_LOOKUP:

```
struct MXRRDEF {
    unsigned int mxrr_l_preference;
    unsigned int mxrr_l_length;
    char         mxrr_t_name[128];
}
```

This structure is NETLIB-specific.

1.3.4 NETLIB_DNS_HEADER Structure

A domain name server accepts queries and returns replies prefixed with with a standard header. A definition of this header is provided in the NETLIB_DNS_HEADER structure. Refer to the NETLIBDEF header file and the appropriate DNS RFCs (such as RFC 1035) for more information about this header. Only those programs which use the NETLIB_DNS_ QUERY routine will need this header definition.

1.4 Byte-Order Considerations

In NETLIB routines, addresses and port numbers are expected to be in ''network order;'' that is, with the high-order byte first. This is opposite the natural byte-order for VAX and AXP systems running VMS, which places the low-order byte first. NETLIB provides routines that can be used to reverse byte order for use in the calls which expect it.

1.5 Synchronous vs. Asynchronous Operation

Most NETLIB routines can be executed synchronously (where the routine uses $QIOW) or asynchronously ($QIO is used instead of $QIOW). To have a routine called asynchronously, specify a non-zero **astadr** argument (and, optionally, the **astprm** argument, to have a parameter passed to the AST routine).

1.6 I/O Status Block

Most NETLIB routines include an optional parameter for a VMS-style I/O status block (IOSB), which is used to return status information to the calling program. The first word of the IOSB is the NETLIB status code for the call; the second word, for read and write operations, is the number of bytes transferred. The second longword of the IOSB is not used by NETLIB.

1.7 Linking NETLIB Programs

To link your program with NETLIB, include the following line in a LINK options file:

```
NETLIB_SHRXFR/SHARE
```

Referencing this logical name will link your program against NETLIB's transfer vector, which at run-time will activate the appropriate vendor-specific version of NETLIB (pointed to by the NETLIB_SHR logical name).

1.8 Contacting the Author

There is no formal support for NETLIB. If you have Internet connectivity, however, you may wish to subscribe to one or more of the following MadGoat Software mailing lists:

Info-MadGoat@MadGoat.Com

Discussion of MadGoat Software products by users and MadGoat developers. To subscribe, send a message to *Info-MadGoat-Request@MadGoat.Com* with the word SUBSCRIBE in the first line of the body of the message.

MadGoat-Announce@MadGoat.Com

Announcements of new releases and new products from MadGoat. To subscribe, send a message to *MadGoat-Announce-Request@MadGoat.Com* with the word SUBSCRIBE in the first line of the body of the message.

MadGoat-Bugs@MadGoat.Com

Address for reporting bugs in MadGoat Software products. Please include the name of the package and version in the subject header of the message, so the report can be more easily directed to the appropriate developer.

Part I Routine Descriptions

This part describes the each of the NETLIB routines.

NETLIB_ACCEPT—Accept an incoming connection

NETLIB_ACCEPT waits for an incoming connection and accepts it, creating a new socket for the new connection.

FORMAT

NETLIB_ACCEPT *socket ,newsocket [,remaddr] [,remaddr-size] [,remaddr-len] [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Socket that was established as a listener.

newsocket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

Socket for the accepted connection (created and returned by this routine).

remaddr

VMS Usage: **structure**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

Address of a socket_address structure into which the remote address/port information for the new connection is written.

remaddr-size

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Size of the **remaddr** structure.

remaddr-len

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

Longword into which the actual returned lenght of the socket address is written.

iosb

VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

I/O status block to receive the status for this call.

astadr

VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**

Address of an AST routine to be invoked on completion of this operation.

astprm

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Parameter to be passed to the AST routine.

DESCRIPTION

This routine is used as part of a ''passive'' open sequence, typically by servers that bind a socket to a specific port number, then establish a listener to handle the incoming connections. This call is used to accept an incoming connection, creating a new socket for the new connection. The listener socket is not affected by this call, so you can call NETLIB_ ACCEPT again to wait for another incoming connection on the same port after one accept completes.

If you want to know the address and port number of the remote end of the accepted connection, you can either use the **remaddr**, **remaddr-size**, and **remaddr-len** arguments on this call, or follow this call with a call to NETLIB_GETPEERNAME.

NETLIB_ADDRESS_TO_NAME—Get the host name for an IP address

NETLIB_ADDRESS_TO_NAME performs an inverse host lookup, returning a name associated with an IP address.

FORMAT

NETLIB_ADDRESS_TO_NAME *socket, [which], address, addrsize, hostname [,retlen] [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

The socket about which you wish to obtain the option information.

which

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

A code indicating the type of lookup to be performed. Possible values are NETLIB_K_LOOKUP_DNS (1), which requests that the name be looked up using the Domain Name System, and NETLIB_K_LOOKUP_HOST_TABLE (2), which requests that the name be looked up in the local host table. If omitted, the default is NETLIB_K_LOOKUP_DNS. See the Description section for information on package-specific restrictions with this parameter.

address

VMS Usage: **structure**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

An INADDRDEF structure containing the address to be looked up.

addrsize
VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**
The size of the address passed in the **address** parameter.

hostname
VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**
A buffer into which the returned host name will be written.

retlen
VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **write only**
mechanism: **by reference**
The returned length of the host name, in bytes.

iosb
VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**
I/O status block to receive the status for this call.

astadr
VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**
Address of an AST routine to be invoked on completion of this operation.

astprm
VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**
Parameter to be passed to the AST routine.

DESCRIPTION

This routine performs an address-to-name lookup on the specified IP address. Each TCP/IP package implements this function slightly differently, so there are some restrictions on the use of this routine.

For MultiNet and Pathway, the **which** argument is used to determine whether the lookup should occur using NETLIB's DNS resolver support or using the TCP/IP package's host-table access routines. For these packages, the host-table access method cannot be used at AST level. Also, if a DNS lookup is attempted and the system is not configured for DNS, an SS$_ UNSUPPORTED error will be returned.

For UCX, TCPware (NETLIB uses its UCX emulation mode), and CMU TCP/IP, the **which** argument is ignored, since these packages do not provide separate access to DNS and local host tables. For UCX, host table lookup will occur first, followed by a DNS lookup. For CMU, this order is reversed.

NETLIB_ADDRTOSTR—Format an IP address to a string

NETLIB_ADDRTOSTR is a utility routine for formatting an IP address as a character string.

FORMAT

NETLIB_ADDRTOSTR *address, string [,retlen]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

address

VMS Usage: **INADDRDEF structure**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**
An INADDRDEF structure containing the address to be formatted.

string

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**
The string to hold the formatted address.

retlen

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **write only**
mechanism: **by reference**
The number of characters written to **string**.

DESCRIPTION

This is a utility routine which formats an IP address to a character string, using the dotted-decimal format.

NETLIB_BIND—Bind a socket to an address/port

NETLIB_BIND sets the address and/or port number for a socket.

FORMAT

NETLIB_BIND *socket, socket-address, sockaddr-len [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Socket to be bound.

socket-address

VMS Usage: **structure**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Socket_address structure describing the addreses and port to which the socket is to be bound.

sockaddr-len

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Size of the socket address structure.

iosb

VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

I/O status block to receive the status for this call.

astadr

VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**

Address of an AST routine to be invoked on completion of this operation.

astprm

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Parameter to be passed to the AST routine.

DESCRIPTION

This routine is used primarily by server programs to bind a socket to a particular port, and is typically followed by a call to NETLIB_LISTEN to open a passive connection on that port.

If a program does not bind a socket to a particular local address/port combination, the underlying TCP/IP software will automatically assign an address and/or port.

NETLIB_CLOSE—Close down a socket

NETLIB_CLOSE closes a socket.

FORMAT

NETLIB_CLOSE *socket*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Socket to be closed.

DESCRIPTION

This routine deletes the specified socket, closing the associated network communications channel, if one is open. Once closed, the socket can no longer be used; you must open a new one with the NETLIB_SOCKET call.

NETLIB_CONNECT—Connect a socket to a remote address/port

NETLIB_CONNECT connects a socket to a remote address/port.

FORMAT

NETLIB_CONNECT *socket, socket-address, sockaddr-len [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Socket to be connected.

socket-address

VMS Usage: **structure**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Socket_address structure describing the addreses and port to which the socket is to be connected.

sockaddr-len

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Size of the socket address structure.

iosb

VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

I/O status block to receive the status for this call.

astadr

VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**
Address of an AST routine to be invoked on completion of this operation.

astprm

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**
Parameter to be passed to the AST routine.

DESCRIPTION

This routine is used primarily with TCP-based programs to establish a connection to a remote system. When used with UDP, this routine fixes the address to which subsequent UDP datagrams will be sent.

NETLIB_CONNECT_BY_NAME—Establish a TCP connection to a host by name

NETLIB_CONNECT_BY_NAME establishes a TCP connection to a host using the host's name, rather than its IP address.

FORMAT

NETLIB_CONNECT_BY_NAME *socket, hostname, port [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

A STREAM-type socket allocated with NETLIB_SOCKET.

hostname

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

A character string containing either a host name or an IP address in dotted-decimal format.

port

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **read only**
mechanism: **by reference**

The port number to connect to on the destination host. Unlike the SIN_W_PORT field of a SINDEF (socket address) structure, this number is specified in *host order*, not network order. NETLIB_CONNECT_BY_NAME will automatically convert the port number to network order for you.

iosb

VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**
I/O status block to receive the status for this call.

astadr

VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**
Address of an AST routine to be invoked on completion of this operation.

astprm

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**
Parameter to be passed to the AST routine.

DESCRIPTION

This routine converts the specified name or dotted-decimal address to an IP address and uses NETLIB_CONNECT to connect to the specified host and port. If multiple addresses are returned by the host name lookup, each address will be tried until a connection is established or all addresses have been tried.

NETLIB_DNS_EXPANDNAME—Expand a name in a DNS response

NETLIB_DNS_EXPANDNAME is a utility routine used for extracting a domain name from a DNS response.

FORMAT

NETLIB_DNS_EXPANDNAME *bufstart, buflen, bufptr, name [,retlen] [,skipcount]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

bufstart

VMS Usage: **pointer**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

A pointer to the start of the DNS response buffer.

buflen

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **read only**
mechanism: **by reference**

The size of the buffer, in bytes.

bufptr

VMS Usage: **pointer**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

A pointer to the area of the buffer containing the domain name to be expanded.

name

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**

A descriptor for a character string buffer into which the expanded name will be written.

retlen

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **write only**
mechanism: **by reference**

The returned length of the expanded name, in bytes.

skipcount

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **write only**
mechanism: **by reference**

The number of bytes in the buffer that were used for the domain name.

DESCRIPTION

This is a utility routine used when parsing a response returned by a call to NETLIB_DNS_QUERY. It expands a domain name stored in a DNS response out to the typical, human-readable, dotted-domain format.

This routine is needed because of the way domain names are represented in DNS responses. Refer to RFC 1035 for further information on the format of DNS queries and responses.

NETLIB_DNS_MX_LOOKUP—Look up MX records for a domain name

NETLIB_DNS_MX_LOOKUP looks up a host name, returning any MX records.

FORMAT

NETLIB_DNS_MX_LOOKUP *socket, hostname, mxrrlist, mxrrlist-size [,mxrrcount] [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Any socket allocated with NETLIB_SOCKET.

hostname

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

The host name to be looked up.

mxrrlist

VMS Usage: **array of structures**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

An array of MXRRDEF structures into which the MX records will be written.

mxrrlist-size

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

The number of elements in the **mxrrlist** array.

mxrrcount

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**
The actual number of MX records written to **mxrrlist**.

iosb

VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**
I/O status block to receive the status for this call.

astadr

VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**
Address of an AST routine to be invoked on completion of this operation.

astprm

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**
Parameter to be passed to the AST routine.

DESCRIPTION

This routine performs a DNS lookup on the specified name, returning any Mail Exchanger (MX) records for that name.

For MultiNet, TCPware, and Pathway, and UCX, this routine is a front-end that uses NETLIB_DNS_QUERY to perform the DNS queries. For CMU TCP/IP, the MX lookups are performed using NAMRES calls.

NETLIB_DNS_QUERY—Perform a DNS query

NETLIB_DNS_QUERY formats a domain name service query and returns the response from a DNS server.

FORMAT

NETLIB_DNS_QUERY *socket, name, [class], type, buffer, bufsize [,flags] [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

A socket allocated with NETLIB_SOCKET. Any socket can be used.

name

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

The domain name to be looked up.

class

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

The class of the query. For Internet domain names, this should be NETLIB_K_DNS_CLASS_IN, which is the default if this parameter is omitted.

type

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

The type of query. Valid type codes are specified by Internet RFCs on the DNS, the codes provided by NETLIB are listed in Table RTN–1.

Table RTN–1 Domain Name Service query types

Symbolic name	Value	Description
NETLIB_K_DNS_TYPE_A	1	Address (A) records
NETLIB_K_DNS_TYPE_NS	2	Name Server (NS) records
NETLIB_K_DNS_TYPE_CNAME	5	Canonical Name (CNAME) records
NETLIB_K_DNS_TYPE_SOA	6	Start-of-Authority (SOA) records
NETLIB_K_DNS_TYPE_WKS	11	Well-known-server (WKS) records
NETLIB_K_DNS_TYPE_PTR	12	Pointer (PTR) records
NETLIB_K_DNS_TYPE_HINFO	13	Host information (HINFO) records
NETLIB_K_DNS_TYPE_MX	15	Mail Exchanger (MX) records
NETLIB_K_DNS_TYPE_TXT	16	Text (TXT) records
NETLIB_K_DNS_QTYPE_ALL	255	Any available information

buffer

VMS Usage: **varying_arg**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

A buffer into which the DNS response will be copied.

bufsize

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **read only**
mechanism: **by reference**

The size of **buffer**, in bytes.

flags

VMS Usage: **longword_mask**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

A bitmask of flags specifying options for this query. Valid flags are listed in Table RTN–2. If omitted, the default value for this parameter is 1.

Table RTN–2 Flag values for NETLIB_DNS_QUERY

Symbolic name	Value	Description
NETLIB_M_DOMAIN_SEARCH	1	Perform a domain search on the default domain.
NETLIB_M_NO_RECURSION	2	Specify a non-recursive query.

iosb

VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

I/O status block to receive the status for this call.

astadr

VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**
Address of an AST routine to be invoked on completion of this operation.

astprm

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**
Parameter to be passed to the AST routine.

DESCRIPTION

This routine formats a DNS query and returns the response from the first name server to provide an answer that indicates either that the query succeeded, or that the name does not exist in the DNS. If the query fails due to an I/O error, that error status is returned and/or stored in the IOSB status word.

Even when this routine returns a success status, the caller should verify that the return code in the header of the DNS reply is NETLIB_K_DNS_RC_SUCCESS, indicating a successful lookup, before attempting to parse the rest of the reply.

For MultiNet, TCPware, and Pathway, and UCX, the name servers that are queried are those specified for those packages. With MultiNet, for example, the ones listed in the translation string for the logical name for MULTINET_NAMESERVERS are used. For the CMU TCP/IP package, the name servers used by NAMRES are not available to NETLIB, so you must define the logical name NETLIB_NAMESERVERS with the IP address(es) of the name server(s) you want NETLIB to use. If no name servers can be found, SS$_UNSUPPORTED is returned.

NETLIB_DNS_SKIPNAME—Skip a name in a DNS response

NETLIB_DNS_SKIPNAME is a utility routine used for parsing DNS responses which skips over a domain name.

FORMAT

NETLIB_DNS_SKIPNAME *bufptr, buflen*

RETURNS

VMS Usage: **longword_signed**
type: **longword (signed)**
access: **write only**
mechanism: **by value**

ARGUMENTS

bufptr

VMS Usage: **pointer**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

A pointer to the area of the DNS response buffer that begins the name to be skipped.

buflen

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **read only**
mechanism: **by reference**

A count of the number of bytes in the buffer, starting from **bufptr**.

DESCRIPTION

This is a utility routine used when parsing a response returned by a call to NETLIB_DNS_QUERY. It returns a count of the number of bytes to skip allocated to the domain name positioned in the buffer at location **bufptr**. If the operation fails, -1 is returned to indicate an error.

This routine is needed because of the way domain names are represented in DNS responses. Refer to RFC 1035 for further information on the format of DNS queries and responses.

NETLIB_GETPEERNAME—Get remote socket address

NETLIB_GETSOCKNAME returns the remote address and port information for a connected socket.

FORMAT

NETLIB_GETPEERNAME *socket, socket-address, sockaddr-size, sockaddr-retlen [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Socket for which the information is to be returned.

socket-address

VMS Usage: **structure**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

Socket_address structure which will hold the returned address/port information.

sockaddr-size

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Size of the socket address structure.

sockaddr-retlen

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

Returned actual length of the socket address.

iosb

VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**
I/O status block to receive the status for this call.

astadr

VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**
Address of an AST routine to be invoked on completion of this operation.

astprm

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**
Parameter to be passed to the AST routine.

DESCRIPTION

This routine is used to return the remote address and port information for a socket that is connected to a remote system.

NETLIB_GETSOCKNAME—Get local socket address

NETLIB_GETSOCKNAME returns the local address and port information for a socket.

FORMAT

NETLIB_GETSOCKNAME *socket, socket-address, sockaddr-size, sockaddr-retlen [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Socket for which the information is to be returned.

socket-address

VMS Usage: **structure**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

Socket_address structure which will hold the returned address/port information.

sockaddr-size

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Size of the socket address structure.

sockaddr-retlen

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

Returned actual length of the socket address.

iosb

VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**
I/O status block to receive the status for this call.

astadr

VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**
Address of an AST routine to be invoked on completion of this operation.

astprm

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**
Parameter to be passed to the AST routine.

DESCRIPTION

This routine is used to return the local address and port information for a socket that has been bound or connected.

NETLIB_GETSOCKOPT—Get socket options

NETLIB_GETSOCKOPT gets the current state of a socket option.

FORMAT

NETLIB_GETSOCKOPT *socket, level, option, value, valsize [,vallen] [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

The socket about which you wish to obtain the option information.

level

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

''Level'' of the option being set. The only level globally supported is NETLIB_K_LEVEL_SOCKET (value X'FFFF'). Some packages also support protocol-level option settings.

option

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword representing the option being set. Not all packages support all options, but most support the ones listed in Table RTN–4.

Consult your package-specific documentation on other available options and on option value ranges. The CMU TCP/IP package is not based on sockets, so this call will always return success status with 0 returned length on systems running that package.

value

VMS Usage: **varying_arg**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

Value to set for the option. Values vary from option to option.

valsize

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**
Size of the **value** argument, in bytes.

vallen

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**
Actual returned length of **value**, in bytes.

iosb

VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**
I/O status block to receive the status for this call.

astadr

VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**
Address of an AST routine to be invoked on completion of this operation.

astprm

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**
Parameter to be passed to the AST routine.

DESCRIPTION

This routine provides an interface to the **getsockopt** service provided by most TCP/IP packages (most are based on the BSD socket communications model). Note that not all options are available with all packages, and for the CMU TCP/IP this call is a no-operation—it will always return a success status, with 0 as the returned length of the option value.

NETLIB_GET_HOSTNAME—Get local host's name

NETLIB_GET_HOSTNAME returns the Internet host name of the local host.

FORMAT

NETLIB_GET_HOSTNAME *namdsc [,retlen]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

namdsc

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**
Character string buffer that gets the returned host name.

retlen

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **write only**
mechanism: **by reference**
The returned length of the host name.

DESCRIPTION

This routine returns the name of the local host by translating the appropriate logical name defined for the TCP/IP package running on the system.

NETLIB_HTON_LONG—Convert host-order longword to network order

NETLIB_HTON_LONG is a utility routine for converting a longword in the local host's byte order to a network-order longword.

FORMAT

NETLIB_HTON_LONG *value*

RETURNS

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

value

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

The longword to be converted.

DESCRIPTION

This is a utility routine which reverses the byte order of a longword to convert it from the native format for the local host (which is ''little-endian'' for VAX and AXP systems running VMS) to network order (which is ''big-endian''). The returned value is the network-order longword.

NETLIB_HTON_WORD—Convert host-order word to network order

NETLIB_HTON_WORD is a utility routine for converting a word in the local host's byte order to a network-order word.

FORMAT

NETLIB_HTON_WORD *value*

RETURNS

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

value

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **read only**
mechanism: **by reference**

The word to be converted.

DESCRIPTION

This is a utility routine which reverses the byte order of a word to convert it from the native format for the local host (which is ''little-endian'' for VAX and AXP systems running VMS) to network order (which is ''big-endian''). The return value is the network-order word.

NETLIB_LISTEN—Listen for incoming connections

NETLIB_LISTEN sets a socket up to listen for incoming connections.

FORMAT

NETLIB_LISTEN *socket [,backlog] [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Socket to be set up as a listener.

backlog

VMS Usage: **unsigned int**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

The number of backlogged connections to allow on this listener.

iosb

VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

I/O status block to receive the status for this call.

astadr

VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**

Address of an AST routine to be invoked on completion of this operation.

astprm

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Parameter to be passed to the AST routine.

NETLIB_LISTEN

DESCRIPTION

This routine is used as part of a ‘‘passive’’ open sequence, typically by servers that bind a socket to a specific port number, then establish a listener to handle the incoming connections. This call is usually followed by one or more invocations of NETLIB_ACCEPT to accept the incoming connections.

NETLIB_NAME_TO_ADDRESS—Get the IP address(es) for a host name

NETLIB_NAME_TO_ADDRESS looks up a host name, returning its IP address(es).

FORMAT

NETLIB_NAME_TO_ADDRESS *socket, [which], hostname, addrlist, addrlist-size [,addrcount] [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

The socket about which you wish to obtain the option information.

which

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

A code indicating the type of lookup to be performed. Possible values are NETLIB_K_LOOKUP_DNS (1), which requests that the name be looked up using the Domain Name System, and NETLIB_K_LOOKUP_HOST_TABLE (2), which requests that the name be looked up in the local host table. If omitted, the default is NETLIB_K_LOOKUP_DNS. See the Description section for information on package-specific restrictions with this parameter.

hostname

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

The host name to be looked up.

addrlist
VMS Usage: **array of structures**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**
An array of INADDRDEF structures into which the addresses will be written.

addrlist-size
VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**
The number of elements in the **addrlist** array.

addrcount
VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**
The actual number of addresses written to **addrlist**.

iosb
VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**
I/O status block to receive the status for this call.

astadr
VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**
Address of an AST routine to be invoked on completion of this operation.

astprm
VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**
Parameter to be passed to the AST routine.

DESCRIPTION

This routine performs a name-to-address lookup on the specified host name. Each TCP/IP package implements this function slightly differently, so there are some restrictions on the use of this routine.

For MultiNet and Pathway, the **which** argument is used to determine whether the lookup should occur using NETLIB's DNS resolver support or using the TCP/IP package's host-table access routines. For these packages, the host-table access method cannot be used at AST level. Also, if a DNS

lookup is attempted and the system is not configured for DNS, an SS$_ UNSUPPORTED error will be returned.

For UCX, TCPware (NETLIB uses its UCX emulation mode), and CMU TCP/IP, the **which** argument is ignored, since these packages do not provide separate access to DNS and local host tables. For UCX, host table lookup will occur first, followed by a DNS lookup. For CMU, this order is reversed.

NETLIB_NTOH_LONG—Convert network-order longword to host order

NETLIB_NTOH_LONG is a utility routine for converting a longword in the network byte order to a host-order longword.

FORMAT

NETLIB_NTOH_LONG *value*

RETURNS

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

value

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

The longword to be converted.

DESCRIPTION

This is a utility routine which reverses the byte order of a longword to convert it from network order to host order. The return value is the host-order longword.

NETLIB_NTOH_WORD—Convert network-order word to host order

NETLIB_NTOH_WORD is a utility routine for converting a word in the network byte order to a host-order word.

FORMAT

NETLIB_NTOH_WORD *value*

RETURNS

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

value

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **read only**
mechanism: **by reference**

The word to be converted.

DESCRIPTION

This is a utility routine which reverses the byte order of a word to convert it from network order to host order. The return value is the host-order word.

NETLIB_READ—Receive data from remote system

NETLIB_READ reads data sent by a remote system from a socket.

FORMAT

NETLIB_READ *socket, buffer [,socket-address] [,sockaddr-size] [,sockaddr-len] [,timeout] [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Socket to which data should be written.

buffer

VMS Usage: **unspecified**
type: **longword (unsigned)**
access: **write only**
mechanism: **by descriptor (fixed-length)**

The address of a descriptor for a buffer to receive the data.

socket-address

VMS Usage: **structure**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

Socket_address structure to receive the addreses and port from which the data is to be sent. Typically used only with unconnected UDP sockets).

sockaddr-size

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Size of the socket address structure. Typically used only with unconnected UDP sockets.

sockaddr-len

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Returned length of the socket address structure. Typically used only with unconnected UDP sockets.

timeout

VMS Usage: **delta_time**
type: **quadword (signed)**
access: **read only**
mechanism: **by reference**

Amount of time that NETLIB_READ should wait for the read to complete. If omitted, no timeout is used.

iosb

VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

I/O status block to receive the status for this call.

astadr

VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**

Address of an AST routine to be invoked on completion of this operation.

astprm

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Parameter to be passed to the AST routine.

DESCRIPTION

This routine reads data from the socket into the specified buffer. The read will complete when the buffer is full, or if the read times out (if **timeout** is specified). A fixed-length descriptor should be used for the buffer. To obtain the actual number of bytes read in this call, use an I/O status block.

NETLIB_READLINE—Read a “line” of data

NETLIB_READLINE reads a “line” of data from the socket.

FORMAT

NETLIB_READLINE *socket, buffer [,retlen] [,flags] [,timeout] [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Socket from which data should be read.

buffer

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**

The address of a descriptor for the buffer to hold the line.

retlen

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **write only**
mechanism: **by reference**

The address of a word which receives the actual length of the received line.

flags

VMS Usage: **bitmask**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

The address of a bitmask containing flags controlling this call. Valid flags are listed in Table RTN–3. If omitted, the default value for this parameter is 1.

Table RTN–3 Flag values for NETLIB_READLINE

Symbolic name	Value	Description
NETLIB_M_ALLOW_LF	1	Allows a bare linefeed as a line terminator, in addition to a CR/LF sequence.
NETLIB_M_FLUSH	2	Causes NETLIB_READLINE to return any data it may have in its internal buffers. See the routine description for more information.

timeout

VMS Usage: **delta_time**
type: **quadword (signed)**
access: **read only**
mechanism: **by reference**

The address of a VMS delta time value specifying the timeout for this read.

iosb

VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

I/O status block to receive the status for this call.

astadr

VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**

Address of an AST routine to be invoked on completion of this operation.

astprm

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**

Parameter to be passed to the AST routine.

DESCRIPTION

This routine is intended for TCP-based programs that use a protocol which delimits commands and replies with carriage-return/linefeed pairs. It completes only when the specified buffer is full; when a complete, terminated, line has been received; or if a timeout occurs. The line terminator is stripped from the returned data.

The NETLIB_M_FLUSH flag was added in NETLIB V2.2, for those programs that need to switch between line-mode reads using this routine and ‘‘raw’’ reads using NETLIB_READ on the same TCP stream. Use this flag after your last line-mode read, but before you switch to raw reads; it will return any data that NETLIB_READLINE has buffered but not yet

processed. If the internal buffer is empty when NETLIB_M_FLUSH is specified, the return status will be SS$_NORMAL and the returned length will be zero; otherwise, the buffered data and its actual length will be returned, without any line-mode processing performed.

NETLIB_SERVER_SETUP—Socket setup for "forked" servers

NETLIB_SERVER_SETUP creates and sets up the socket used by a server process "forked" from a TCP/IP package's "master server".

FORMAT

NETLIB_SERVER_SETUP *socket, socket-address, sockaddr-size*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

Returned socket handle for use in subsequent NETLIB calls.

socket-address

VMS Usage: **special_structure**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Socket address describing port being opened. See note in description section.

sockaddr-size

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Size of the structure passed in **socket-address** argument.

DESCRIPTION

This routine creates a NETLIB socket for the communications channel in a server program invoked by a TCP/IP package's ''master server.'' The **socket-address** and **sockaddr-size** arguments must be provided, although for all packages except CMU TCP/IP they are ignored (the actual address and port number are pre-determined).

On systems running TCPware, use a protocol type of BG_TCP for TCP-based services and BG_UDP for UDP-based services, when defining the service in TCPware's NETCU utility.

Note: For at least some TCP/IP packages, this routine should be called *before* any language-specific I/O routines (for example, the printf routine in the C run-time library) are called, to prevent the run-time system from interfering with the network channel.

NETLIB_SETSOCKOPT—Set socket options

NETLIB_SETSOCKOPT modifies attributes of a a socket.

FORMAT

NETLIB_SETSOCKOPT *socket, level, option, value, vallen [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

The socket you wish to modify.

level

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

''Level'' of the option being set. The only level globally supported is NETLIB_K_LEVEL_SOCKET (value X'FFFF'). Some packages also support protocol-level option settings.

option

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Longword representing the option being set. Not all packages support all options, but most support the ones listed in Table RTN–4.

NETLIB_SETSOCKOPT

Table RTN–4 Socket options

Symbolic name	Value	Description
NETLIB_K_OPTION_REUSEADDR	4	Allows a port number to be reused. Value is a longword, either 1 (on) or 0 (off). Default for most packages is off, but NETLIB turns this option on automatically unless you explicitly call NETLIB_SETSOCKOPT to set it.
NETLIB_K_OPTION_SNDBUF	4097	Sets socket buffer size for sends. Value is a longword, typically with a maximum value of 32768.
NETLIB_K_OPTION_RCVBUF	4098	Sets socket buffer size for receives. Value is a longword, typically with a maximum value of 32768.

Consult your package-specific documentation on other available options and on option value ranges. The CMU TCP/IP package is not based on sockets, so this call has no effect on systems running that package.

value

VMS Usage: **varying_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Value to set for the option. Values vary from option to option.

vallen

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Size of the **value** argument, in bytes.

iosb

VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

I/O status block to receive the status for this call.

astadr

VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**

Address of an AST routine to be invoked on completion of this operation.

astprm

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**
Parameter to be passed to the AST routine.

DESCRIPTION

This routine provides an interface to the **setsockopt** service provided by most TCP/IP packages (most are based on the BSD socket communications model). Note that not all options are available with all packages, and for the CMU TCP/IP this call is a no-operation.

NETLIB_SHUTDOWN—Shut down a connection

NETLIB_SHUTDOWN shuts down a connection without deleting the socket.

FORMAT

NETLIB_SHUTDOWN *socket [,shuttype] [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Socket to be set up as a listener.

shuttype

VMS Usage: **unsigned int**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Type of shutdown to be performed. Possible values are listed in Table RTN–5. If omitted, the default is NETLIB_K_SHUTDOWN_RECEIVER.

Table RTN–5 Shutdown types

Symbolic name	Value	Description
NETLIB_K_SHUTDOWN_RECEIVER	0	Prevent any further receives.
NETLIB_K_SHUTDOWN_SENDER	1	Prevent any further sends.
NETLIB_K_SHUTDOWN_BOTH	2	Prevent any further sends/receives.

This parameter has no meaning when using the CMU TCP/IP package.

iosb

VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

I/O status block to receive the status for this call.

astadr

VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**
Address of an AST routine to be invoked on completion of this operation.

astprm

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**
Parameter to be passed to the AST routine.

DESCRIPTION

This routine closes the network connection currently open on the specified socket, without deleting the socket. Any data still waiting to be received or sent can be discarded or retained by specifying an appropriate shutdown type in **shuttype**.

For the CMU TCP/IP package, the shutdown type is not used; the CMU close operation does not provide this option.

NETLIB_SOCKET—Create a NETLIB communication endpoint

NETLIB_SOCKET creates a "socket"—that is, a local endpoint for TCP/IP communication.

FORMAT

NETLIB_SOCKET *socket [,socket-type] [,socket-family]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

The returned socket context, for use in subsequent NETLIB calls.

socket-type

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by referencee**

Type of socket to be created. Allowed values are shown in Table RTN–6. If omitted, the default is NETLIB_K_TYPE_STREAM.

Table RTN–6 Socket types

Symbolic name	Value	Description
NETLIB_K_TYPE_STREAM	1	Connection-oriented socket (for TCP).
NETLIB_K_TYPE_DGRAM	2	Connectionless socket (for UDP).

socket-family

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Address/protocol family. Only NETLIB_K_AF_INET (value 2) is permitted, and is assumed by default if omitted.

DESCRIPTION

This routine creates a ‘‘socket’’ (an endpoint for network communication). One socket is required for each network connection. This will be the first call in most NETLIB-based programs. The only exception to this rule is a program that is intended to be run as a ‘‘forked’’ server from a TCP/IP package’s ‘‘master server’’, which will use the NETLIB_SERVER_SETUP routine instead.

NETLIB_STRTOADDR—Convert a dotted-address to binary form

NETLIB_STRTOADDR is a utility routine for converting a character string holding a dotted-decimal IP addreses into its binary equivalent.

FORMAT

NETLIB_STRTOADDR *string, address*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

string

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**
The string containing the dotted-decimal address.

address

VMS Usage: **INADDRDEF structure**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**
An INADDRDEF structure to hold the returned address.

DESCRIPTION

This is a utility routine which parses a character string containing an IP address in dotted-decimal format, returning the binary representation of that address in network order (suitable for a call to NETLIB_CONNECT, for example).

NETLIB_VERSION—Obtain version information for NETLIB

NETLIB_VERSION returns a character string containing the version of NETLIB.

FORMAT

NETLIB_VERSION *verstr [,retlen]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

verstr

VMS Usage: **char_string**
type: **character string**
access: **write only**
mechanism: **by descriptor**
A character string into which the version information is copied.

retlen

VMS Usage: **word_unsigned**
type: **word (unsigned)**
access: **write only**
mechanism: **by reference**
The length of the version information string.

DESCRIPTION

This routine, added in NETLIB V2.2, returns a copy of the version information for the NETLIB library installed on the running system. For V2.2, the string has the format ''NETLIB V2.2''.

Note: If you need to use this routine to make a run-time determination of available NETLIB features, and you wish to be compatible with NETLIB versions prior to V2.2, you should dynamically locate this routine's symbol name via LIB$FIND_IMAGE_SYMBOL. If the symbol is not found, then you are running a version of NETLIB that pre-dates V2.2.

NETLIB_WRITE—Send data to remote system

NETLIB_WRITE writes data to a socket for transmission to a remote system.

FORMAT

NETLIB_WRITE *socket, buffer [,socket-address] [,sockaddr-len] [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Socket to which data should be written.

buffer

VMS Usage: **unspecified**
type: **longword (unsigned)**
access: **read only**
mechanism: **by descriptor**

The address of a descriptor for the data to be sent.

socket-address

VMS Usage: **structure**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Socket_address structure describing the addreses and port to which the data is to be sent. Used only with unconnected UDP sockets).

sockaddr-len

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Size of the socket address structure. Used only with unconnected UDP sockets.

iosb

VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**
I/O status block to receive the status for this call.

astadr

VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**
Address of an AST routine to be invoked on completion of this operation.

astprm

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**
mechanism: **by value**
Parameter to be passed to the AST routine.

DESCRIPTION

This routine writes the specified data to the socket for transmission to the remote system. For UDP sockets that have not been connected to a specific remote address, the **socket-address** argument specifies the remote address/port to which the datagram is to be sent.

NETLIB_WRITELINE—Send data with CR/LF termination

NETLIB_WRITELINE writes data to a socket for transmission to a remote system, terminating the data with a CR/LF pair.

FORMAT

NETLIB_WRITELINE *socket, buffer [,iosb] [,astadr] [,astprm]*

RETURNS

VMS Usage: **cond_value**
type: **longword (unsigned)**
access: **write only**
mechanism: **by value**

ARGUMENTS

socket

VMS Usage: **longword_unsigned**
type: **longword (unsigned)**
access: **read only**
mechanism: **by reference**

Socket to which data should be written.

buffer

VMS Usage: **char_string**
type: **character string**
access: **read only**
mechanism: **by descriptor**

The address of a descriptor for the data to be sent.

iosb

VMS Usage: **io_status_block**
type: **longword (unsigned)**
access: **write only**
mechanism: **by reference**

I/O status block to receive the status for this call.

astadr

VMS Usage: **procedure**
type: **longword (unsigned)**
access: **call**
mechanism: **by reference**

Address of an AST routine to be invoked on completion of this operation.

astprm

VMS Usage: **user_arg**
type: **longword (unsigned)**
access: **read only**

mechanism: **by value**
Parameter to be passed to the AST routine.

DESCRIPTION

This routine functions identically to NETLIB_WRITE, but adds a carriage-return/ linefeed pair to the data sent to the remote system. It is intended for TCP-based programs that use a protocol which delimits commands and responses with CR/LFs.

Note: If you use NETLIB_WRITELINE asynchronously, you should only have one outstanding call at any given time, to ensure proper operation with all TCP/IP packages.

A Status Codes

NETLIB translates all TCP/IP status codes into the codes used by DEC TCP/IP Services. The status codes are described in Table A–1. For synchronous NETLIB calls, these codes are returned in R0 and also in the IOSB, if one is specified; for asynchronous calls, you should always specify an IOSB to receive the status of the network I/O operation.

Some NETLIB routines may also return status codes from other facilities, such as the LIB$ and STR$ routines when processing strings.

Table A–1 NETLIB Status Codes

Symbolic Name	Description
SS$_ABORT	Catch-all code for errors not covered by other codes.
SS$_INSFARG	Not enough arguments specified on NETLIB call.
SS$_BADPARAM	Invalid value specified for an argument to a NETLIB call. This code can also be returned by the underlying TCP/IP driver when an invalid value is specified for a network I/O operation.
SS$_NOSUCHNODE	Destination address required.
SS$_INSFMEM	Memory allocation failure.
SS$_NOPRIV	Insufficient privilege for requested operation.
SS$_ACCVIO	Invalid memory address specified.
SS$_FILALRACC	Socket already connected.
SS$_LINKDISCON	Network link has been disconnected.
SS$_TOOMUCHDATA	Result too large for user's buffer.
SS$_SUSPENDED	I/O operation would block.
SS$_NOTNETDEV	Network operation requested for non-network device.
SS$_PROTOCOL	Protocol type or option incorrect. Protocol or socket type not supported. Other general protocol failure.
SS$_ILLCNTRFUNC	Operation not supported on socket.
SS$_DUPLNAM	Address/port already in use.
SS$_IVADDR	Invalid address.
SS$_UNREACHABLE	Network or host unreachable.
SS$_RESET	Network connection has been reset.
SS$_LINKABORT	Connection aborted by network software.
SS$_CONNECFAIL	Connection failure, or reset by peer process.
SS$_NOLINKS	Socket is not connected.
SS$_SHUT	Socket, host, or network software has been shut down.
SS$_TIMEOUT	Operation timed out.

Table A–1 (Cont.) NETLIB Status Codes

Symbolic Name	Description
SS$_REJECT	Connection refused.