```
=================================================================
IMPORTANT NOTE:
Use "man -s 3n ..." FOR THE MANUAL PAGES OF NETWORKING FUNCTIONS
=================================================================

1. socket - create an endpoint for communication

=============================================================
     cc  [ flag ... ] file ... -lsocket -lnsl [ library ... ]
     #include <sys/types.h>
     #include <sys/socket.h>

     int socket(int domain, int type, int protocol);
=============================================================

The domain parameter specifies a communications domain within which
communication will take place. Two possible domains are
     AF_UNIX - Unix domain
     AF_INET - Internet domain

The second argument is the type of socket. The socket has the indicated type,
which specifies the com-munication semantics. The common choices are:
     SOCK_STREAM - sequenced, reliable, two-way connection-based byte streams
     SOCK_DGRAM - datagrams which is connectionless, unreliable messages of a
                  fixed (typically small) maximum length

The third argument is the protocol. Protocol specifies a particular protocol
to be used with the socket. Use 0 for TCP/IP (stream sockets) and UDP/IP
(datagram sockets)

RETURN VALUES
     A -1 is returned if an error occurs.  Otherwise  the  return
     value is a descriptor referencing the socket.


=================================================================
2. bind - bind a name to a socket

=============================================================
     cc  [ flag ... ] file ... -lsocket -lnsl [ library ... ]
     #include <sys/types.h>
     #include <sys/socket.h>

     int bind(int s,  const struct sockaddr *name, socklen_t *namelen);
=============================================================

     bind() assigns a name to an unnamed socket. When a socket is
     created  with socket(), it exists in a name space (address
     family) but has no name assigned. bind() requests  that  the
     name pointed to by name be assigned to the socket.

RETURN VALUES
     If the bind is successful, 0 is returned. A return value  of
     -1  indicates  an  error.


=================================================================
3. sockaddr data structure.
=================================================================

/*
 * Structure used by kernel to store most
 * addresses. Defined in <sys/socket.h>
 */
struct sockaddr {
     sa_family_t     sa_family;      /* address family - AF_INET or AF_UNIX*/
     char            sa_data[14];    /* name of socket */
};
```

```
=====================================================================
4. sockaddr_in data structure.
=====================================================================

/*
 * Socket address, internet style. Defined in <netinet/in.h>
 */
struct sockaddr_in {
     sa_family_t     sin_family;  /* address family */
     in_port_t       sin_port;    /* port number */
     struct  in_addr sin_addr;    /* address of host */
     char            sin_zero[8];
};

=====================================================================
5. listen - listen for connections on a socket

=============================================================
     cc  [ flag ... ] file ... -lsocket -lnsl [ library ... ]
     #include <sys/types.h>
     #include <sys/socket.h>

     int listen(int s, int backlog);
=============================================================

     To accept  connections,  a  socket  is  first  created  with
     socket(),  a backlog for incoming connections is specified
     with listen() and then the  connections  are  accepted  with
     accept().   The  listen()  call applies only to sockets of
     type SOCK_STREAM.

     The backlog parameter defines the maximum length  the  queue
     of pending connections may grow to.

     If a connection request arrives with  the  queue  full,  the
     client  will receive an error with an indication of ECONNRE-
     FUSED for  AF_UNIX sockets. If the underlying protocol  sup-
     ports retransmission, the connection request may be ignored
     so that  retries  may succeed. For AF_INET sockets, the tcp
     will retry the connection. If the  backlog is not cleared by
     the  time  the  tcp times out, the connect will fail with
     ETIMEDOUT.

RETURN VALUES
     A 0 return value indicates success; -1 indicates an error.


=====================================================================
6. accept - accept a connection on a socket

=============================================================
     cc  [ flag ... ] file ... -lsocket -lnsl [ library ... ]
     #include <sys/types.h>
     #include <sys/socket.h>

     int accept(int s, struct sockaddr *addr, socklen_t *addrlen);
=============================================================

     The argument s is  a  socket  that  has  been  created  with
     socket()  and  bound to an address with bind(), and that
     is listening for connections after a call to listen(). The
     accept() function extracts the first connection on the queue
     of pending connections, creates a new socket with  the  pro-
     perties  of  s, and allocates a new file descriptor, ns, for
     the socket. If no pending connections  are  present  on  the
     queue and the socket is not marked as non-blocking, accept()
```

blocks the caller until a connection is present. If the socket is marked as non-blocking and no pending connections are present on the queue, accept() returns an error as described below. The accept() function uses the netconfig(4) file to determine the STREAMS device file name associated with s. This is the device on which the connect indication will be accepted. The accepted socket, ns, is used to read and write data to and from the socket that connected to ns; it is not used to accept more connections. The original socket (s) remains open for accepting further connections.

The argument addr is a result parameter that is filled in with the address of the connecting entity as it is known to the communications layer. The exact format of the addr parameter is determined by the domain in which the communication occurs.

The argument addrlen is a value-result parameter. Initially, it contains the amount of space pointed to by addr; on return it contains the length in bytes of the address returned.

The accept() function is used with connection-based socket types, currently with SOCK_STREAM.

RETURN VALUES

The accept() function returns -1 on error. If it succeeds, it returns a non-negative integer that is a descriptor for the accepted socket.

==========

7. connect - initiate a connection on a socket

==========

```
cc [ flag ... ] file ... -lsocket -lnsl [ library ... ]
#include <sys/types.h>
#include <sys/socket.h>

int connect(int s, const struct sockaddr *name, struct_t
namelen);
```

==========

The parameter s is a socket. If it is of type SOCK_DGRAM, connect() specifies the peer with which the socket is to be associated; this address is the address to which datagrams are to be sent if a receiver is not explicitly designated; it is the only address from which datagrams are to be received. If the socket s is of type SOCK_STREAM, connect() attempts to make a connection to another socket. The other socket is specified by name. name is an address in the communication space of the socket. Each communication space interprets the name parameter in its own way. If s is not bound, then it will be bound to an address selected by the underlying transport provider. Generally, stream sockets may successfully connect() only once; datagram sockets may use connect() multiple times to change their association. Datagram sockets may dissolve the association by connecting to a null address.

RETURN VALUES

If the connection or binding succeeds, 0 is returned. Otherwise, -1 is returned and sets errno to indicate the error.

==========

8. SOME OTHER USEFUL FUNCTIONS.

==========

1) void bcopy(const void *s1, void *s2, size_t n);

The bcopy() function copies n bytes from string s1 to the string s2. Overlapping strings are handled correctly.

2) struct hostent *gethostbyname(const char *name);

gethostbyname() searches for information for a host with the hostname specified by the character-string parameter name.

RETURN VALUES
Host entries are represented by the struct hostent structure defined in <netdb.h>:

```
struct hostent {
    char    *h_name;         /* canonical name of host */
    char    **h_aliases;     /* alias list */
    int     h_addrtype;      /* host address type */
    int     h_length;        /* length of address */
    char    **h_addr_list;   /* list of addresses */
};
```