Socket functions provide an application programmer interface (API) to send and receive data over a network.

**Socket Functions**

- **gethostbyname**: get the IP address for an IP name
- **gethostname**: get the name of the local machine
- **socket**: create a new socket
- **connect**: make connection to remote host
- **send**: transmit data through active connection
- **recv**: receive data through active connection
- **close**: terminate use of a socket
- **bind**: attach a network address to a socket
- **listen**: wait for incoming messages
- **accept**: begin using incoming connection

**Sequence of Socket Calls**
**gethostbyname**

hostent = gethostbyname(“IP name”)

- returns a pointer to a hostent structure.
- provides the IP address for an IP name.

**socket**

descriptor = socket(protofamily, type, protocol)

- Returns socket descriptor used in subsequent calls
  - protofamily selects protocol family;
    - AF_INET - Internet protocols
  - type selects type of communication
    - SOCK_DGRAM - connectionless
    - SOCK_STREAM - connection-oriented
  - protocol specifies protocol within protocol family:
    - IPPROTO_TCP - selects TCP
    - IPPROTO_UDP - selects UDP

**connect**

connect(socket, saddress, saddrlen)

- Client uses connect to establish connection to server
- socket holds descriptor of socket to use
- saddress is a struct sockaddr that identifies server
- saddrlen gives length of saddress

**connect (cont.)**

- Blocks until connection completed (accepted)
- Usually used with connection-oriented transport protocol
- Can be used with connectionless protocol
- Marks local socket with server address
- Implicitly identifies server for subsequent messages
### send

**send**

```c
send(socket, data, length, flags)
```

- Used to send data through a connected socket
- `socket` identifies socket
- `data` points to data to be sent
- `length` gives length of data (in bytes)
- `flags` indicate special options
  - zero is a nice flag

### sendto

**sendto**

```c
sendto(socket, data, length, flags, destaddr, addrlen)
```

- Used for **unconnected** sockets by explicitly specifying destination
- `sendto` adds additional parameters:
  - `destaddr` - struct sockaddr destination address
  - `addrlen` - length of destaddr

### recv

**recv**

```c
recv(socket, buffer, length, flags)
```

- Used to receive incoming data through connected socket
- `socket` identifies the socket
- Data copied into `buffer`
- At most `length` bytes will be received
- `flags` give special options
- Returns number of bytes actually received
  - 0 implies connection closed
  - -1 implies error

### recvfrom

**recvfrom**

```c
recvfrom(socket, buffer, length, flags, sndraddr, addrlen)
```

- Like `recvfrom` (in reverse!)
- Address of source copied into `sndraddr`
- Length of address in `addrlen`
close
close(descriptor)

- Terminates use of socket descriptor
- descriptor contains descriptor of socket to be closed

gethostname
gethostname(hostname, buffersize)

- puts the IP name of the local computer in the hostname string.

bind
bind(socket, localaddr, address)

- Initially, socket has no addresses attached
- bind selects either local, remote or both addresses
- server binds local port number for incoming messages
- client binds remote address and port number to contact server

listen
listen(socket, queuesize)

- Server uses listen to wait for incoming connections
- socket identifies socket through which connections will arrive (address)
- New connection requests may arrive while server processes previous request
- Operating system can hold requests on queue
- queuesize sets upper limit on outstanding requests. Usually 2 or 3 will work.
accept

accept(socket, caddress, caddrlen)

- Server uses accept to accept the next connect request
- accept call blocks until connection request arrives
- Returns a new socket with server's end of new connection
- Old socket remains unchanged and continues to field incoming requests
- caddress returns struct sockaddr client address;

Socket Address Format

struct sockaddr_in {
    u_char sin_len;  /* total length of address */
    u_char sin_family; /* family of the address */
    u_short sin_port;  /* protocol port number */
    struct in_addr sin_addr; /* IP address */
    char sin_zero[8]    /* unused */
};

Host entry structure Format

Host entry structure returned by gethostbyname

struct hostent {
    char *h_name;    /* official 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 addr from DNS */
};

#define h_addr h_addr_list[0]    /* address, for backward compatibility */

htons Format Conversion

int = htons( short )

- Host TO Network Short
- Converts a short int to network standard format.
- Swaps bytes if necessary.
ntohs Format Conversion

int = ntohs( short)

- Network TO Host Short
- Converts a 16 bit integer in network standard format to a short in the local host format.
- Swaps bytes if necessary.

Microsoft Extensions

WSAStartup( version, &wsaData);

- Must be called before any other socket function.
- WSADATA is an structure that will receive information about the socket implementation on this system, such as the implementation version.

WSACleanup();

- Closes all sockets.
- WSAStartup must be called to use any socket function again.