**Ex No: 2**

**Date:**

**STUDY OF BASIC FUNCTIONS OF SOCKET PROGRAMMING**

**AIM:**

To discuss some of the basic functions used for socket programming.

1.**man socket**

**NAME:**

Socket – create an endpoint for communication.

**SYNOPSIS**:

#include<sys/types.h>

#include<sys/socket.h>

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

**DESCRIPTION:**

* Socket creates an endpoint for communication and returns a descriptor.
* The domain parameter specifies a common domain this selects the protocol family which will be used for communication.
* These families are defined in <sys/socket.h>.

**FORMAT:**

|  |  |
| --- | --- |
| **NAME** | **PURPOSE** |
| PF\_UNIX,PF\_LOCAL | Local Communication. |
| PF\_INET | IPV4 Internet Protocols. |
| PF\_IPX | IPX-Novell Protocols. |
| PF\_APPLETALK | Apple Talk. |

* The socket has the indicated type, which specifies the communication semantics.

**TYPES:**

**1.SOCK\_STREAM:**

* Provides sequenced , reliable, two-way , connection based byte streams.
* An out-of-band data transmission mechanism, may be supported.

**2.SOCK\_DGRAM:**

* Supports datagram (connectionless, unreliable messages of a fixed maximum length).

**3.SOCK\_SEQPACKET:**

* Provides a sequenced , reliable, two-way connection based data transmission path for datagrams of fixed maximum length.

**4.SOCK\_RAW:**

* Provides raw network protocol access.

**5.SOCK\_RDM:**

* Provides a reliable datagram layer that doesn’t guarantee ordering.

**6.SOCK\_PACKET:**

* Obsolete and shouldn’t be used in new programs.

**2.man connect:**

**NAME:**

connect – initiate a connection on a socket.

**SYNOPSIS:**

#include<sys/types.h>

#include<sys/socket.h>

int connect(int sockfd,const (struct sockaddr\*)serv\_addr,socklen\_t addrlen);

**DESCRIPTION:**

* The file descriptor sockfd must refer to a socket.
* If the socket is of type SOCK\_DGRAM then the serv\_addr address is the address to which datagrams are sent by default and the only addr from which datagrams are received.
* If the socket is of type SOCK\_STREAM or SOCK\_SEQPACKET , this call attempts to make a connection to another socket.

**RETURN VALUE**:

* If the connection or binding succeeds, zero is returned.
* On error , -1 is returned , and error number is set appropriately.

**ERRORS:**

|  |  |
| --- | --- |
| EBADF | Not a valid Index. |
| EFAULT | The socket structure address is outside the user’s address space. |
| ENOTSOCK | Not associated with a socket. |
| EISCONN | Socket is already connected. |
| ECONNREFUSED | No one listening on the remote address. |

**3.man accept**

**NAME:**

accept/reject job is sent to a destination.

**SYNOPSIS:**

accept destination(s)

reject[-t] [-h server] [-r reason] destination(s)

**DESCRIPTION:**

* accept instructs the printing system to accept print jobs to the specified destination.
* The –r option sets the reason for rejecting print jobs.
* The –e option forces encryption when connecting to the server.

**4.man send**

**NAME:**

send, sendto, sendmsg - send a message from a socket.

**SYNOPSIS:**

#include<sys/types.h>

#include<sys/socket.h>

ssize\_t send(int s, const void \*buf, size\_t len, int flags);

ssize\_t sendto(int s, const void \*buf, size\_t len, int flags, const struct sock\_addr\*to, socklen\_t tolen);

ssize\_t sendmsg(int s, const struct msghdr \*msg, int flags);

**DESCRIPTION:**

* The system calls send, sendto and sendmsg are used to transmit a message to another socket.
* The send call may be used only when the socket is in a connected state.
* The only difference between send and write is the presence of flags.
* The parameter is the file descriptor of the sending socket.

**5.man recv**

**NAME:**

recv, recvfrom, recvmsg – receive a message from a socket.

**SYNOPSIS:**

#include<sys/types.h>

#include<sys/socket.h>

ssize\_t recv(int s, void \*buf, size\_t len, int flags);

ssize\_t recvfrom(int s, void \*buf, size\_t len, int flags, struct sockaddr \*from, socklen\_t\* from len);

ssize\_t recvmsg(int s, struct msghdr \*msg, int flags);

**DESCRIPTION:**

* The recvfrom and recvmsg calls are used to receive messages from a socket, and may be used to recv data on a socket whether or not it is connection oriented.
* If from is not NULL, and the underlying protocol provides the src addr , this src addr is filled in.
* The recv call is normally used only on a connection socket and is identical to recvfrom with a NULL from parameter.

**6.man read**

**NAME:**

read, readonly, return

**7.man write**

**NAME:**

write- send a message to another user.

**SYNOPSIS:**

write user[ttyname]

**DESCRIPTION:**

* write allows you to communicate with other users, by copying lines from terminal to ………
* When you run the write and the user you are writing to get a message of the form:

Message from yourname @yourhost on yourtty at hh:mm:…

* Any further lines you enter will be copied to the specified user’s terminal.
* If the other user wants to reply they must run write as well.

**8. ifconfig**

**NAME:**

ifconfig- configure a network interface.

**SYNOPSIS:**

ifconfig[interface]

ifconfig interface[aftype] options | address……

**DESCRIPTION:**

* ifconfig is used to configure the kernel resident network interfaces.
* It is used at boot time to setup interfaces as necessary.
* After that, it is usually only needed when debugging or when system tuning is needed.
* If no arguments are given, ifconfig displays the status of the currently active interfaces.

**9. man bind**

**SYNOPSIS:**

bind[-m keymap] [-lp sv psv]

**10. man htons/ man htonl**

**NAME:**

htonl, htons, ntohl, ntohs - convert values between host and network byte order.

**SYNOPSIS:**

#include<netinet/in.h>

uint32\_t htonl(uint32\_t hostlong);

uint16\_t htons(uint32\_t hostshort);

uint32\_t ntohl(uint32\_t netlong);

uint16\_t ntohs(uint16\_t netshort);

**DESCRIPTION:**

* The htonl() function converts the unsigned integer hostlong from host byte order to network byte order.
* The htons() converts the unsigned short integer hostshort from host byte order to network byte order.
* The ntohl() converts the unsigned integer netlong from network byte order to host byte order.

**11. man gethostname**

**NAME:**

gethostname, sethostname- get/set host name.

**SYNOPSIS:**

#include<unistd.h>

int gethostname(char \*name,size\_t len);

int sethostname(const char \*name,size\_t len);

**DESCRIPTION:**

* These functions are used to access or to change the host name of the current processor.
* The gethostname() returns a NULL terminated hostname(set earlier by sethostname()) in the array name that has a length of len bytes.
* In case the NULL terminated then hostname does not fit ,no error is returned, but the hostname is truncated.
* It is unspecified whether the truncated hostname will be NULL terminated.

**12. man gethostbyname**

**NAME:**

gethostbyname, gethostbyaddr, sethostent, endhostent, herror, hstr – error – get network host entry.

**SYNOPSIS:**

#include<netdb.h>

extern int h\_errno;

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

#include<sys/socket.h>

struct hostent \*gethostbyaddr(const char \*addr)int len, int type);

struct hostent \*gethostbyname2(const char \*name,int af);

**DESCRIPTION:**

* The gethostbyname() returns a structure of type hostent for the given hostname.
* Name->hostname or IPV4/IPV6 with dot notation.
* gethostbyaddr()- struct of type hostent / host address length
* Address types- AF\_INET, AF\_INET6.
* sethostent() – stay open is true(1).
* TCP socket connection should be open during queries.
* Server queries for UDP datagrams.
* endhostent()- ends the use of TCP connection.
* Members of hostent structure:

1. h\_name
2. h\_aliases
3. h\_addrtype
4. h\_length
5. h\_addr-list
6. h\_addr.

**RESULT**:

Thus the basic functions used for Socket Programming was studied successfully.