**EXPERIMENT 4:**

**AIM: Implementation of Select(),of getpeername () system call.**

Advanced Socket System Calls : Programs to demonstrate the usage of Advanced socket system

calls like getsockopt( ),setsockopt( ),getpeername ( ),getsockname( ),readv( ) and writev( ).

getpeername - get the name of the peer socket

#include <sys/socket.h>

int getpeername(int socket, struct sockaddr \*address, socklen\_t \*address\_len);

The getpeername() function retrieves the peer address of the specified socket, stores this address

in the sockaddr structure pointed to by the address argument, and stores the length of this address

in the object pointed to by the address\_len argument.

If the actual length of the address is greater than the length of the supplied sockaddr structure,

the stored address will be truncated.

If the protocol permits connections by unbound clients, and the peer is not bound, then the value

stored in the object pointed to by address is unspecified.

getsockname - get the socket name

#include <sys/socket.h>

int getsockname(int socket, struct sockaddr \*address, socklen\_t \*address\_len);

The getsockname() function retrieves the locally-bound name of the specified socket, stores this

address in the sockaddr structure pointed to by the address argument, and stores the length of

this address in the object pointed to by the address\_len argument.

If the actual length of the address is greater than the length of the supplied sockaddr structure,

the stored address will be truncated.

If the socket has not been bound to a local name, the value stored in the object pointed to by

address is unspecified.

STEPS:

Include the header files

Create a TCP Socket.

Fill in the socket address structure (with server information)

Bind the Address and port using bind() system call.

If Socket name is to be retrieved, then include getsockname() system call with appropriate

options set. If Peer address of the specified socket is to be retrieved, then include getpeername()

system call with appropriate options set.

**EXPERIMENT 5:**

**AIM: Implementation of gesockopt (), setsockopt () system calls.**

There are various options which can be set for a socket and there are multiple ways to set options

that affect a socket. Of these, setsockopt() system call is the one specifically designed for this

purpose. Also, we can retrieve the option which are currently set for a socket by means

of getsockopt() system call.

int setsockopt(int socket, int level, int option\_name, const void \*option\_value, socklen\_t

option\_len);

The socket argument must refer to an open socket descriptor. The level specifies who in the

system is to interpret the option: the general socket code, the TCP/IP code, or the XNS code.

This function sets the option specified by the option\_name, at the protocol level specified by the

level, to the value pointed to by the option\_value for the socket associated with the file descriptor

specified by the socket. The level argument specifies the protocol level at which the option

resides. To set options at the socket level, we need to specify the level argument

as SOL\_SOCKET. To set options at other levels, we need to supply the appropriate protocol

number for the protocol controlling the option. The option\_name specifies a single option to set.

The option\_name and any specified options are passed uninterpreted to the appropriate protocol

module for interpretations. The list of options available at the socket level (SOL\_SOCKET) are:

SO\_DEBUG

Turns on recording of debugging information. This option enables or disables debugging in the

underlying protocol modules. This option takes an int value. This is a boolean option.

SO\_BROADCAST

Permits sending of broadcast messages, if this is supported by the protocol. This option takes an

int value. This is a boolean option.

SO\_REUSEADDR

Specifies that the rules used in validating addresses supplied to bind() should allow reuse of local

addresses, if this is supported by the protocol. This option takes an int value. This is a boolean

option.

SO\_KEEPALIVE

Keeps connections active by enabling the periodic transmission of messages, if this is supported

by the protocol. This option takes an int value. If the connected socket fails to respond to these

messages, the connection is broken and processes writing to that socket are notified with a

SIGPIPE signal. This is a boolean option.

SO\_LINGER

Lingers on a close() if data is present. This option controls the action taken when unsent

messages queue on a socket and close() is performed. If SO\_LINGER is set, the system blocks

the process during close() until it can transmit the data or until the time expires. If SO\_LINGER

is not specified, and close() is issued, the system handles the call in a way that allows the process

to continue as quickly as possible. This option takes a linger structure, as defined in the

&lt;sys/socket.h&gt; header, to specify the state of the option and linger interval.

SO\_OOBINLINE

Leaves received out-of-band data (data marked urgent) in line. This option takes an int value.

This is a boolean option.

SO\_SNDBUF

Sets send buffer size. This option takes an int value.

SO\_RCVBUF

Computer Networks

Department of Computer Science and Engineering

Sets receive buffer size. This option takes an int value.

SO\_DONTROUTE

Requests that outgoing messages bypass the standard routing facilities. The destination must be

on a directly-connected network, and messages are directed to the appropriate network interface

according to the destination address. The effect, if any, of this option depends on what protocol is

in use. This option takes an int value. This is a boolean option.

SO\_RCVLOWAT

Sets the minimum number of bytes to process for socket input operations. The default value for

SO\_RCVLOWAT is 1. If SO\_RCVLOWAT is set to a larger value, blocking receive calls

normally wait until they have received the smaller of the low water mark value or the requested

amount. (They may return less

than the low water mark if an error occurs, a signal is caught, or the type of data next in the

receive queue is different than that returned, e.g. out of band data). This option takes an int value.

Note that not all implementations allow this option to be set.

SO\_RCVTIMEO

Sets the timeout value that specifies the maximum amount of time an input function waits until it

completes. It accepts a timeval structure with the number of seconds and microseconds

specifying the limit on how long to wait for an input operation to complete. If a receive operation

has blocked for this much time without receiving additional data, it returns with a partial count or

errno set to [EAGAIN] or [EWOULDBLOCK] if no data were received. The default for this

option is zero, which indicates that a receive operation will not time out. This option takes a

timeval structure. Note that not all implementations allow this option to be set.

SO\_SNDLOWAT

Sets the minimum number of bytes to process for socket output operations. Non-blocking output

operations will process no data if flow control does not allow the smaller of the send low water

mark value or the entire request to be processed. This option takes an int value. Note that not all

implementations allow this option to be set.

SO\_SNDTIMEO

Sets the timeout value specifying the amount of time that an output function blocks because flow

control prevents data from being sent. If a send operation has blocked for this time, it returns

with a partial count or with errno set to [EAGAIN] ore [EWOULDBLOCK] if no data were sent.

The default for this option is zero, which indicates that a send operation will not time out. This

option stores a timeval structure. Note that not all implementations allow this option to be set.

For boolean options, 0 indicates that the option is disabled and 1 indicates that the option is

enabled.Options at other protocol levels vary in format and name.