**SOCKET PROGRAMMING**

BY:SHRIKRISHNA BHAT(2SD15CS099)

THEORY

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.

Socket creation:

int sockfd = socket(domain, type, protocol)

sockfd: socket descriptor, an integer (like a file-handle)  
domain: integer, communication domain e.g., AF\_INET (IPv4 protocol) , AF\_INET6 (IPv6 protocol)  
type: communication type  
SOCK\_STREAM: TCP(reliable, connection oriented)  
SOCK\_DGRAM: UDP(unreliable, connectionless)  
protocol: Protocol value for Internet Protocol(IP), which is 0. This is the same number which appears on protocol field in the IP header of a packet.(man protocols for more details)

* Setsockopt:

int setsockopt(int sockfd, int level, int optname,

const void \*optval, socklen\_t optlen);

This helps in manipulating options for the socket referred by the file descriptor sockfd. This is completely optional, but it helps in reuse of address and port. Prevents error such as: “address already in use”.

* Bind:

int bind(int sockfd, const struct sockaddr \*addr,

socklen\_t addrlen);

After creation of the socket, bind function binds the socket to the address and port number specified in addr(custom data structure). In the example code, we bind the server to the localhost, hence we use INADDR\_ANY to specify the IP address.

* Listen:

int listen(int sockfd, int backlog);

It puts the server socket in a passive mode, where it waits for the client to approach the server to make a connection. The backlog, defines the maximum length to which the queue of pending connections for sockfd may grow. If a connection request arrives when the queue is full, the client may receive an error with an indication of ECONNREFUSED.

* Accept:

int new\_socket= accept(int sockfd, struct sockaddr \*addr, socklen\_t \*addrlen);

It extracts the first connection request on the queue of pending connections for the listening socket, sockfd, creates a new connected socket, and returns a new file descriptor referring to that socket. At this point, connection is established between client and server, and they are ready to transfer data.

Stages for Client

* Socket connection: Exactly same as that of server’s socket creation
* Connect:

int connect(int sockfd, const struct sockaddr \*addr,

socklen\_t addrlen);

The connect() system call connects the socket referred to by the file descriptor sockfd to the address specified by addr. Server’s address and port is specified in addr.

**select**() and **pselect**() allow a program to monitor multiple file descriptors, waiting until one or more of the file descriptors become "ready" for some class of I/O operation (e.g., input possible). A file descriptor is considered ready if it is possible to perform the corresponding I/O operation (e.g., **read**(2)) without blocking.

Four macros are provided to manipulate the sets.

* **FD\_ZERO**() clears a set.
* **FD\_SET**() and
* **FD\_CLR**() respectively add and remove a given file descriptor from a set.
* **FD\_ISSET**() tests to see if a file descriptor is part of the set;

this is useful after **select**() returns.

*nfds* is the highest-numbered file descriptor in any of the three sets, plus 1.

*timeout* is an upper bound on the amount of time elapsed before **select**() returns. It may be zero, causing **select**() to return immediately. (This is useful for polling.) If *timeout* is NULL (no timeout), **select**() can block indefinitely.

# RETURN VALUE

On success, **select**() and **pselect**() return the number of file descriptors contained in the three returned descriptor sets (that is, the total number of bits that are set in *readfds*, *writefds*, *exceptfds*) which may be zero if the timeout expires before anything interesting happens. On error, -1 is returned, and *errno* is set appropriately; the sets and *timeout* become undefined, so do not rely on their contents after an error.