Report By Team - 9

Team Members : 1.Ayush Gupta
2.Ayush Singh
3.Dipesh Jaswani
4.Paras Pipre

Assignment 1 : A simple client server program with the help of socket programming.

Socket: An interface between an application process and transport layer.

*The application process can send/receive messages to/from another application process (local or remote)via a socket

* In Unix jargon, a socket is a file descriptor – an integer associated with an open file.

* Types of Sockets: Internet Sockets, unix sockets, X.25 sockets etc Internet sockets characterized by IP Address (4 bytes) and port number (2 bytes).

Types of Internet Sockets

* Stream Sockets (SOCK_STREAM)
-Connection oriented
- Rely on TCP to provide reliable two-way connected communication

* Datagram Sockets (SOCK_DGRAM)
-Rely on UDP
-Connection is unreliable

Connection Oriented Protocol

Server **Client**

```
socket()
   |
bind()                  socket()
    |                        |
listen()      ←---  connect()
    |                      |
 accept()                  |
    |                      |
  send()  --------→ recv()

  recv()  ←-------- send()
     |                  |
  close()           close()
```

Socket() --- Get the file descriptor

*int socket(int domain, int type, int protocol);
- domain should be set to AF _ INET
-type can be SOCK_STREAM or SOCK_DGRAM
-set protocol to 0 to have socket choose the correct protocol based on type
-socket() returns a socket descriptor for use in later system calls or -1 on error

Dealing with IP address

*int inet _ aton(const char *cp, struct in_addr *inp);
* Example usage:

```
struct sockaddr _ in my_addr;
 my_addr.sin_family = AF_INET;
 my_addr.sin_port = htons(MYPORT);
inet _ aton("10.0.0.5",&(my_addr.sin_addr));
memset(&(my_addr.sin_zero),'\0',8);
```

*inet _ aton() gives non-zero on success and zero on failure.
* To convert binary IP to string: inet_noa()
   printf("%s",inet_ntoa(my_addr.sin_addr));

connect() -- Hello!

*Connects to a remote host
* int connect(int sockfd, struct sockaddr *serv_addr, int addrlen)
- sockfd is the socket descriptor returned by socket()
- serv _ addr is pointer to struct sockaddr that contains information on destination IP address and port

- addrlen is set to sizeof(struct sockaddr) returns -1 on error
* At times, you don't have to bind() when you are using connect().

accept() - Thank you for calling !

*accept() gets the pending connection on the port you are listen()ing on
* int accept(int sockfd, void *addr, int *addrlen);
- sockfd is the listening socket descriptor
- information about incoming connection is stored in addr which is a pointer to a local struct sockaddr_in
- addrlen is set to sizeof(struct sockaddr_in)
- accept returns a new socket file descriptor to use for this accepted connection and -1 on error

send() and recv() -- Let’s talk !

*int recv(int sockfd, void *buf, int len, int flags);
-sockfd is the socket descriptor to read from
- buf is the buffer to read the information into
- len is the maximum length of the buffer
-set flags to 0 for now
-recv() returns the number of bytes actually read into the buffer or -1 on error
-If recv() returns 0, the remote side has closed connection on you

close() - Bye !

*int close(int sockfd);
-Closes connection corresponding to the socket descriptor and frees the socket descriptor
-Will prevent any more sends and recvs