Network Programming from Scratch in C

Lecture VDO 8

Socket Programming

2.1

- **Table of Contents**

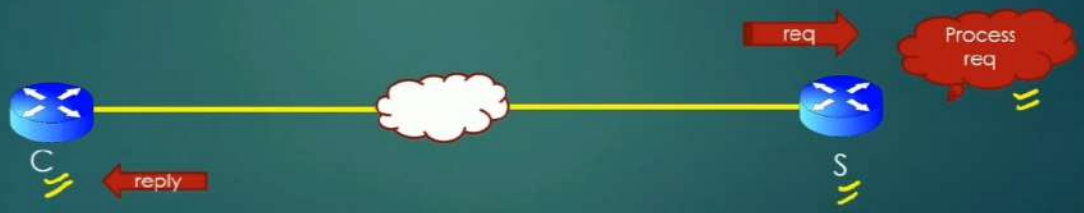
Socket programming

Lecture VDO 8

Socket Programming -> Client and Server based Model

2.2

- Server is any machine which receives the request, process it and optionally returns back the result
- Client is any machine which initiates the request
- We will quickly write a simple Client Server Hello world model



- Server never initiates the communication

Lecture VDO 8

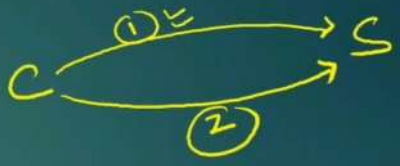
Socket Programming -> Select And Accept System Calls

- Linux OS provide two system calls to enable us write efficient Network Applications
  - Select()
  - Accept()
- In C, it is difficult to write a socket based application without the use of Select and Accept System calls
- From interview point of view – Understanding these two calls is very important
- Let us first grasp the idea of these two APIs. Having understood these two, remaining socket programming APIs would automatically fall in place

3.1

Lecture VDO 8

Socket Programming -> Select And Accept System Calls



- High level Linux Socket based Communication Design

- Messages (Or requests) send by the client to the server can be categorized into two types :
  - Connection initiation request messages
    - This msg is used by the client to request the server to establish a dedicated connection.
    - Only after the connection has been established, then only client can send Service request messages to server.

    And

  - Service Request Messages
    - Client can send these msg to server once the connection is fully established.
    - Through these messages, Client requests server to provide a service

- Servers identifies and process both the type of messages very differently

Lecture VDO 8

Socket Programming -> Select And Accept System Calls

- High level Socket Communication Design

Msg type	Client side API	Server side API
Connection initiation request msgs	connect()	accept()
Service request msgs	sendmsg(), sendto()	recvmsg(), recvfrom()

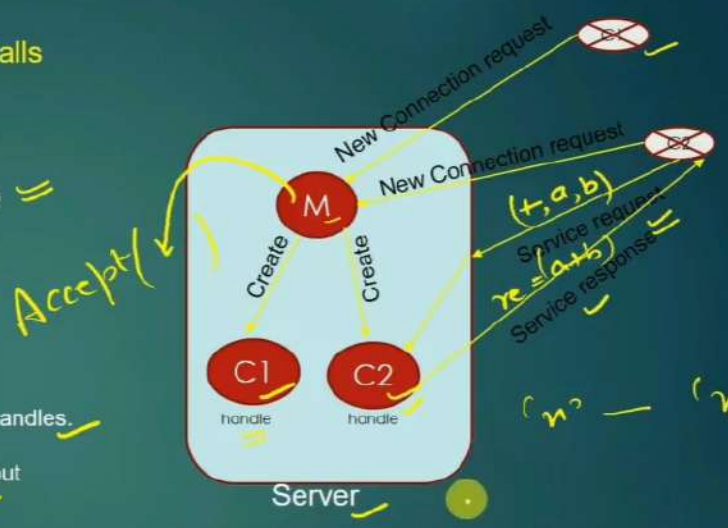
3.3

Lecture VDO 8

Socket Programming -> Select And Accept System Calls

- Life Cycle of a Client Server Communication

When Server boots up, it creates a master socket using socket()



M is the mother of all Client Handles. M gives birth to all Client handles.

Once Client handles are created for each client, Server carries out Communication with the client using Client handle (and not M)

Server Has to maintain the database of connected client handles

M is only used to create new client handles. M is not used for data exchange with already connected clients.

Accept() is the system call used on server side to create client handles.

In Linux Terminology, handles are called as "file descriptors" which are integer numbers. Client handles are called ""communication file descriptos" and M is called Master socket file descriptor

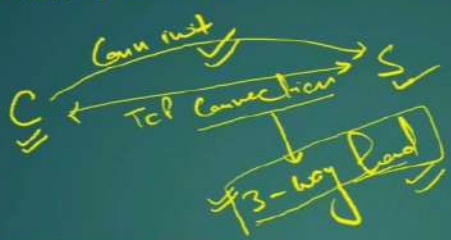
4.1

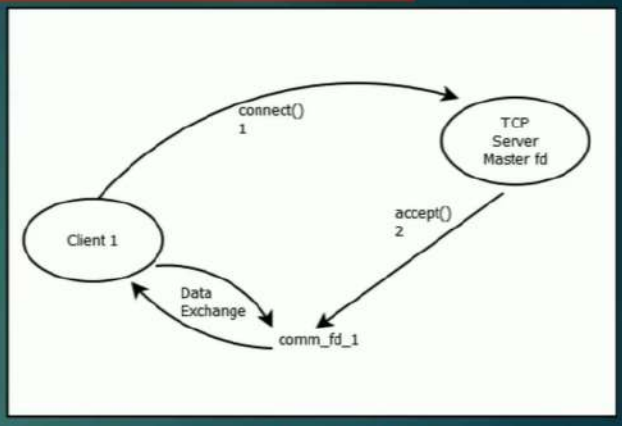
ACCEPT SYSTEM CALL

Lecture VDO 8

4.2

Socket Programming -> Accept() System Call





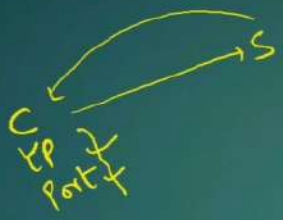
- Accept()
  - Accept() system call is used by the TCP server (not UDP server) to accept the new client connection request
  - Remember, TCP is a connection oriented protocol, meaning – client and server needs to establish dedicated connection first before they can participate in any data exchange activity
  - For example, when you login into facebook, your machine sends connection request to Facebook servers. Facebook servers accept your connection request using accept() system call. Now, you can send data to facebook servers, and facebook servers can update your page with notifications/feeds etc.
  - The sole purpose of Accept() just to establish the connection between two machines (client and server) and carrying out TCP 3-way handshake (refer to your standard books)
  - Accept() is called only on server side, and it returns the handle to the connection with client. Using this handle Server carries out all future communication with the client. This handle is called *Communication file descriptor.*

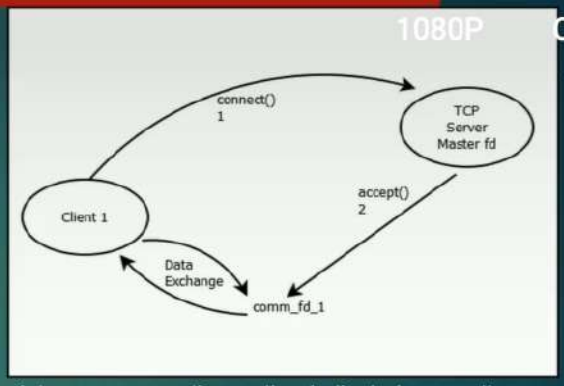
4.3

ACCEPT CALL SYNOPSIS

Lecture VDO 8

Socket Programming -> Accept() System Call



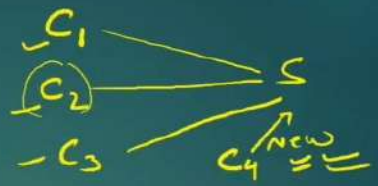


- Accept()
  - Accept() returns a **file descriptor** (comm_socket_fd), which is just an integer, representing as the dedicated connection handle between connected client and server. Server carries out all communication with client using this returned integer value.
  - comm_socket_fd = accept(master_sock_tcp_fd, (struct sockaddr *)&client_addr, &addr_len);
    - master_sock_tcp_fd is the fd of the socket which server has opened to listen to new client connection requests using *socket()*
    - client_addr is the structure which contains client info – ip address and tcp port no of client which has just connected
    - addr_len = size of predefined structure - *struct sockaddr* which is just a constant.
    - There are some more APIs – sendto and recvfrom, but there explicit discussion is not worth.
    - You will understand through code walk-through.

5.1

Lecture VDO 8

Socket Programming -> Select() System Call



- Select() system calls allows the server machine to MONITOR multiple client connected connections and check which client has send the data to process
  >> *Just like the class monitor keep an eye on all the students of the class at the same time*
- Select() system call is ***blocking*** System call – meaning, when it is executed, the code execution halts (similar to scanf/getch)
- Select() unblocks when either of two things happen on server side :
  1. New connection request from new client arrives
  2. Data request from existing connected client arrives
- When select() unblocks, server needs to check whether it's a new connection request Or new data request on existing connection. In the later case, server need to find which client has send the data.
- When server starts, the first thing it does is to create a master socket – A socket to detect arrival of new connection request from new client
- This master socket, Once created, gives birth to rest of the communication file descriptors for clients. Let us visualize this scenario.
- C provides a Data structure called *fd_set* which is a collection (or a set) of file descriptors. fd_set readfds;

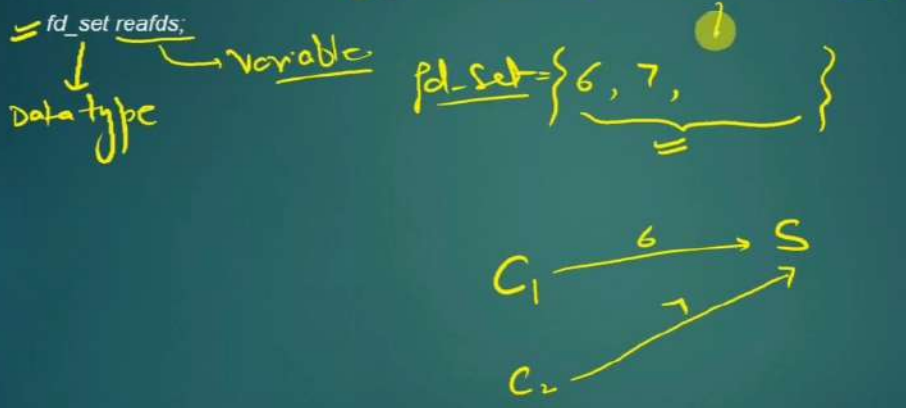
fd_set → {6, 8, 9}

5.2

Lecture VDO 8

Socket Programming -> fd_set

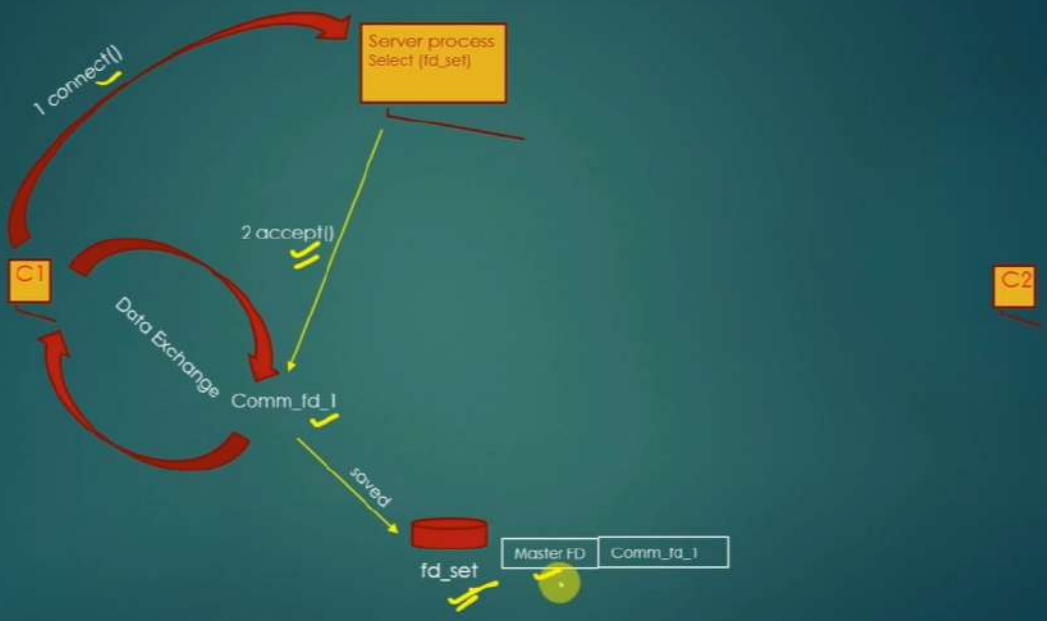
- C provides a Data structure called *fd_set* which is a collection (or a set) of file descriptors




6.1
ACEEPT & SELECT
WORKING TOGETHER

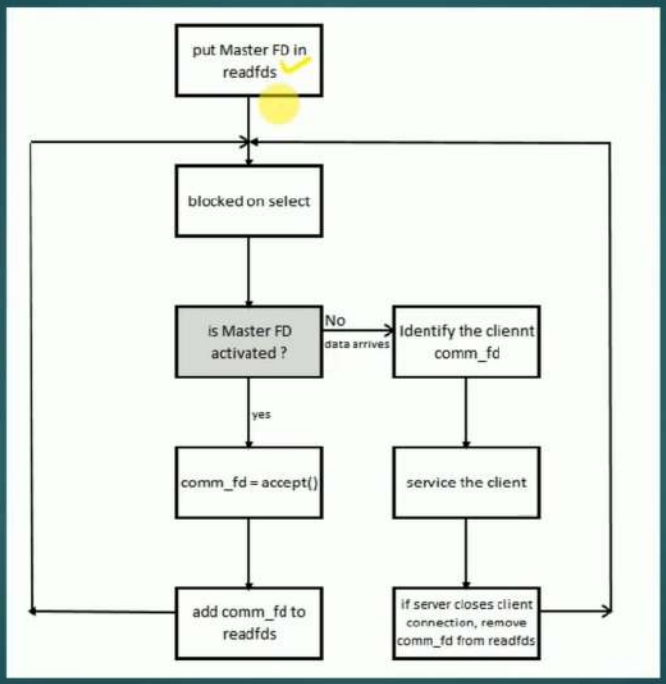
Select and accept together in action

6.2



Multiplexed Server process state machine diagram

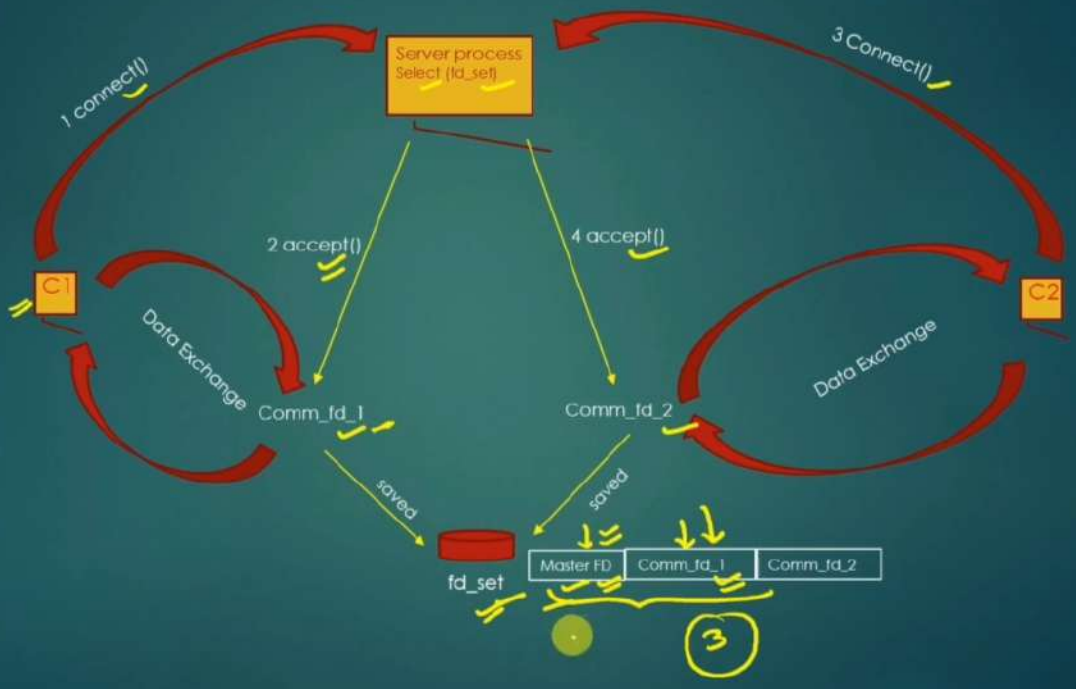
6.3



Time for a Code walk …

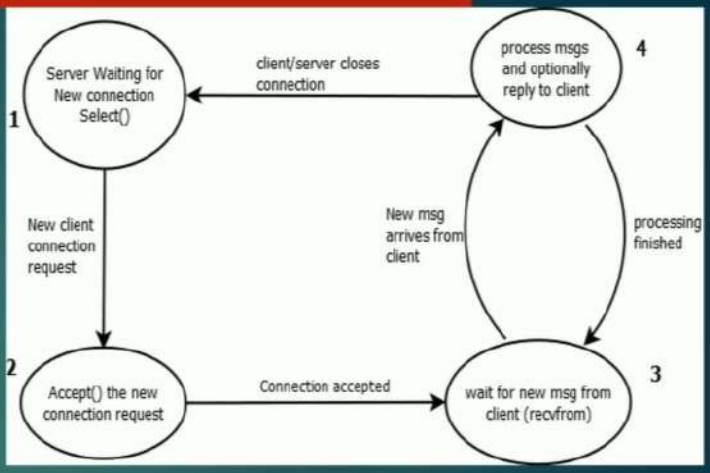
Select and accept together in action

6.4



Lecture VDO 8

Socket Programming -> Server Implementation



- Observations
  - Server cannot accept other client's request while it is processing the current client (oscillating between stages 3 and 4).
  - Only one client can be served at a time.
  - This kind of design is used where there is only and only one dedicated client of server. Such server processes are used in system soft-wares where there is only one client and one server.
  - This server design cannot handle multiple clients connections
  - Remember, Servers are just another process which accepts some data from another process , nothing special.
  - Server doesn't generate any data, it just replies to the client after processing client's data. Server by themselves never sends msg to client without Client's initiation first.

7.1

Network Programming from Scratch in C

Lecture VDO 8

Socket Programming -> TCP Client Implementation

7.2

Code Walk For TCP client

1. Initialize variables
2. Specify Server Credentials
3. Create a Communication socket
4. Connect with Server (Send connection initiation request to server)
5. Send data to Server
6. Receive response
7. (optional) close the connection
8. Goto 5 if wants to communicate more with the server over same connection

- No port number assignment by the programmer
- No bind() , no fd_set, no accept()

Network Programming from Scratch in C

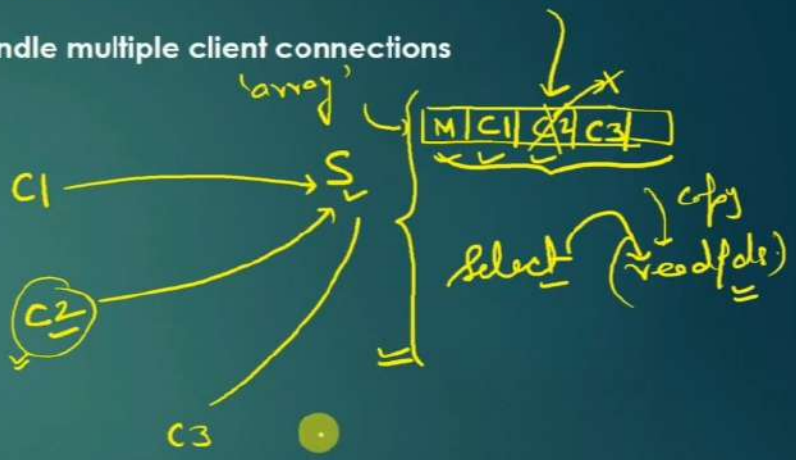
Lecture VDO 8

8.1

Socket Programming -> Multiplexed TCP Server Implementation

Code Walk For TCP Server which can handle multiple client connections

1. Initialize variables
2. Create Master socket
3. Bind
4. Listen
5. Initialize and fill readfds
6. Select
7. Accept the connection
8. service the client requests
9. close the connection
10. Goto 5



9.1

SOCKET PROGRAMMING

CONCLUSION

Lecture VDO 8

Socket Programming -> Server Implementation

Socket programming APIs

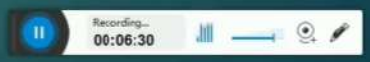
1. socket() – Used to create a TCP/UDP master socket. Used on server and Client side
2. select() – Used for monitoring multiple file descriptors. Can be used on both client and server sides. Blocking system call. Blocks until new connection request Or data arrived on FDs present in readfds set.
3. accept() – Used on TCP server side. Used to accept the client connection request and complete 3-way handshake with client
4. bind() – Used on TCP/UDP server side. Used by the Server application process to inform operating system the criteria of packets of interests
5. listen() – Used on TCP/UDP server side. Used by the Server application process to inform operating system the length of Queue of incoming connection request/data request from clients
6. recvfrom() – Used on the TCP/UDP server and client side. Used by the Server/client process to read the data arrived on communication file descriptors. This call is blocking call by default. Process block if data hasn't arrived yet.
7. sendto() – Used to send data to client/server. Used on Server and client sides.
8. close() – Used to close the connection. Used by both – clients and Server

9.3

Lecture VDO 8

Socket Programming -> Select() Accept() code walk through

- Observations
  - Server Can accept and maintain upto 31 client's connection request (1 being Master skt fd)
  - Server can service one client at a time, however can maintain 31 connections
  - If multiple client's data request comes together (upto n), all client will be serviced but one by one, where n is the value specified in listen()
  - With the increase in no of clients, clients can begin experience latency as they are serviced one by one
  - Thus, this server design is way better than previous one, but has scalability limitations
  - Scalability limitation will always be there, no matter how you design the server. To handle more no of clients, we need to deploy more independent server machines. That is why Facebook, Google have data centers with thousands of servers to handle millions of client request.
  - So don't expect a single server machine to handle significant no of clients. Such a server is not limited by software design, but by hardware limitations.



Network Programming from Scratch in C

Lecture VDO 8

Socket Programming -> UDP Programming

9.4

- How to create UDP server machine and UDP client machine !
- If you have learnt creating TCP server machine, UDP programming is way much simpler than TCP programming …
- Homework : Write your UDP Server and UDP client. Verrryyy easy ….. As compared to TCP counterparts. Infinite material available all over internet. Google…