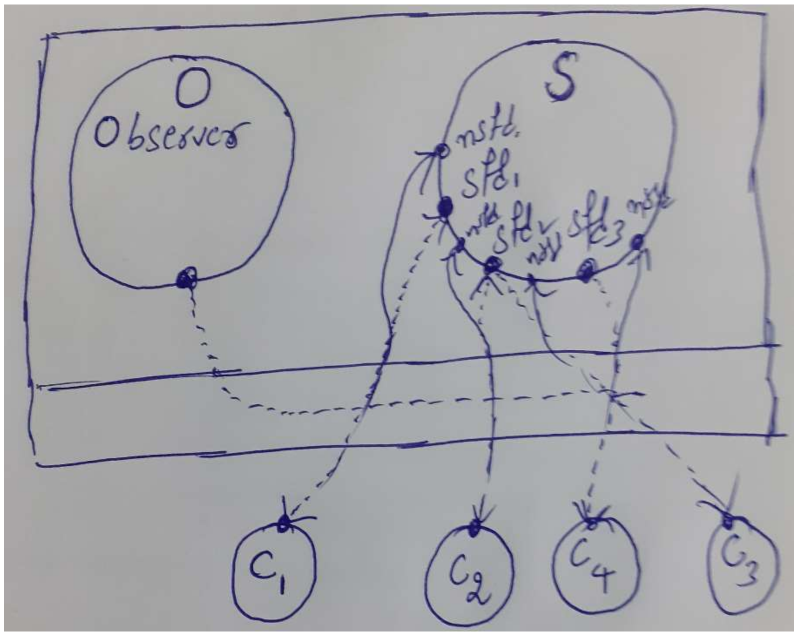
1. **Passport Issuer**: A process S opens 1 raw socket (rsfd) ,1 well known sfd(TCP) , 3 usfds (Unix Domain).And there exists 3 Separate Processes (A,B,C) which are in the same system of process S are connected to process S through 3 usfds (Unix Domain).Now In a Separate System Clients(c1,c2,c3) exists which are connected to raw socket of Process S. Now the process S will pass the rsfd(raw socket fd) to Process A, Process A will receive rsfd(raw socket fd) and it will broadcast all the documents names it requires to authorize and passes back rsfd to S, now S will pass rsfd to process B and Process B will broadcast all the document names it requires using rsfd and passes back rsfd to S. Similarly to Process C. After this if a Client wants to go through verification, it will connect to well known sfd in S. Process S will pass nsfd to A, and A only verifies that Client and sends its opinion to S. S only has to convey the result to the Client. Similar case with B and C.

   Implement all the processes in this scenario.

2. **Fake Server**

   Look at the following figure.



- Server Process S serves on sfd1 , sfd2, sfd3. <u>It so happened that with good intention, S has opened all the sockets with port reusability.</u>
- Clients Ci gets connected to any of the sfd.
- Observer Process observes the traffic (even S should not know that O is observing)
- As soon as process O, comes to know about a service S_i offered by S, then it also starts that service and can accept clients and even serves them.

Implement all the processes in this scenario.

3. **Nitnetd Super Server(Privileged):** Nitnetd super server is similar to inetd super server except on the limit of maximum number of clients for a particular service.
 Assume that Nitnetd server provides services S1, S2 on ports say 1, 2. The Nitnetd server can accept a maximum of 2 clients for S1 service on port 1, maximum of 3 clients for S2 service on port2. Once the maximum limit reaches on a port , Nitnetd will be rejecting connection requests of clients, until the running S1, S2 server processes exit after servicing some of the clients. Suppose at a moment 2 clients are being served by 2 server processes of S1. Then Nitnetd keeps rejecting of connection requests for port 1. Now, if any one S1 server process completes its service and exits, then Nitnetd can accept one client for S1 service on port1. At any time, if Nitnetd is not able to service clients because of maximum limit, it notifies to another side by server N2, which will be serving any number of the clients. <u>But a Client should be given first preference to get connected to Nitnetd</u>. (As Nitnetd is the main server , assume that it provides speedy service connection than that of N2.)

   Implement all the processes in this scenario.
   Hint : After all, the Service server processes are children of Nitnetd super server process.

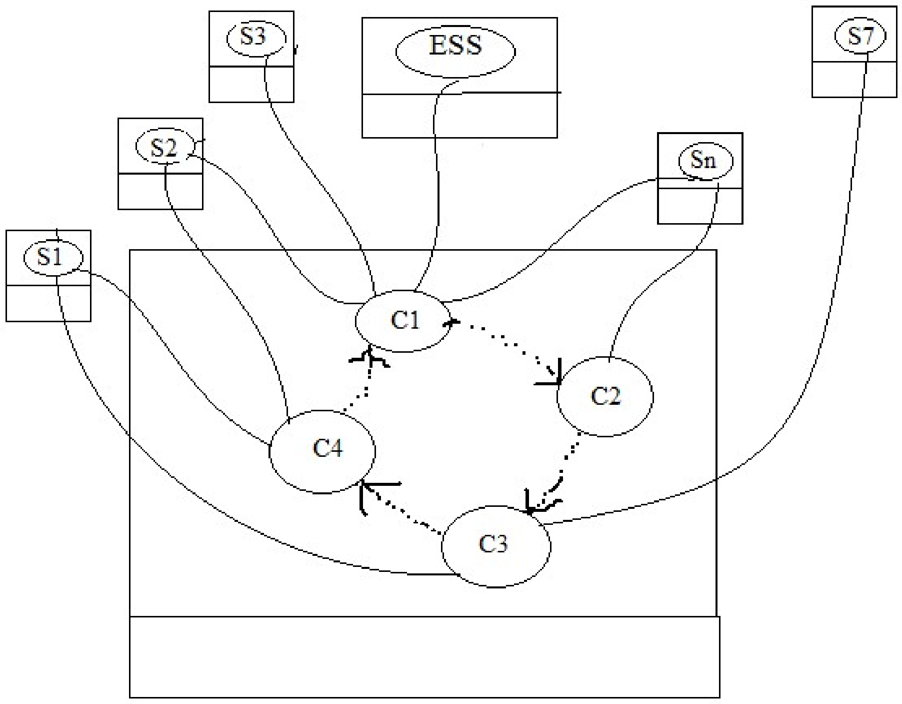
4. **ABC service provider:** S is a super server process which supposes to offer four services (S1, S2, S3, S4) by listening to four socket fds. There are three service provider server processes A, B, C in the same computer system. Process A is child of process S, whereas process B and process C are not children of S and A. (B, C are unrelated processes). All the four processes will be running (existing) before the arrival of first client request.
   As soon as the first client connect request arrives for a service (say S_i), {*in other words, as soon as the super server process S <u>notices</u> the first request for a service*}, the super server process S informs the process A to start accepting and serving clients for the service S_i. After process A accepts two clients, it stops accepting and also it sees that now, process B should start accepting and serving clients for the same service S_i. After process B accepts two clients, it also stops accepting and it sees that now, process C should start accepting and serving clients for the same service S_i. Soon after process C accepts two clients, it stops accepting and also it makes process S to know this. Now process S again starts the offering of services other than the service S_i. This flow is continued until all the services got served.

   Implement all the different processes in this scenario and client process.
   (Hint: Are you going to write code for S,A,B,C or S,A,B or S,A or S only ? The answer may not be in this list !).

5. **Standby Server:** A server process S provides a single service for multiple clients through separate individual threads (i.e. not by forking) for each connected client. There also exists a process called as Alternate Server (AS) <u>in another computer system.</u>(i.e. not in the computer system where S is running). Whenever process S needs to do some maintenance work of its own, it asks process AS to serve of all of its clients and informs about all the connected clients to AS. Process S also informs to its connected clients about its busy in maintenance, and tells them that they <u>can get service</u> from another server. But server S does not give the address or details of the server AS to clients. And server S also tells the clients that it will start servicing again at any time after it finishes its maintenance. It strictly instructs the clients to get served by it only soon after it says ready for the service again. Now, a scenario (arrangement) can be made out that all the clients will be getting service from AS. As and when a client notices the readiness of server S, it gets serviced only by S and stops getting from AS.

   Implement all the processes in this scenario.

6. **Exclusive Special Server(ESS)-circular sharing:** Look at the following figure.



ESS is a very special server such that only one of the client processes running at a particular computer system can get served by it. Suppose, unrelated client processes of C1, C2, C3, C4 running at the same computer system wish to get served by ESS. Assume that at the moment C1 is using the service of ESS though the connected sfd. After some time, C1 notifies C2 to use the service of ESS through the same sfd, and again after some time C2 notifies C3. After a while C3 notifies C4, and C4 in turn notifies C1, and so on as shown in figure. Any new client process can join in the circle of sharing at the rear.
A client process first checks whether there is already a connected sfd to ESS is existing or not. If there is no connection, then only it establishes the connection. That means, the first client process only makes a sfd connected to the ESS, and that sfd only will be used (shared circularly) among all the client processes which join in the circle for service of ESS.
A client process can also get connected to other servers, i.e. a client process will be getting served by other servers. But when a client process gets chance of service of ESS, it exclusively gets served (interacts) by ESS. After some time, it notifies the next client process in the circle and it continues its regular processing (get severed by other connected servers).

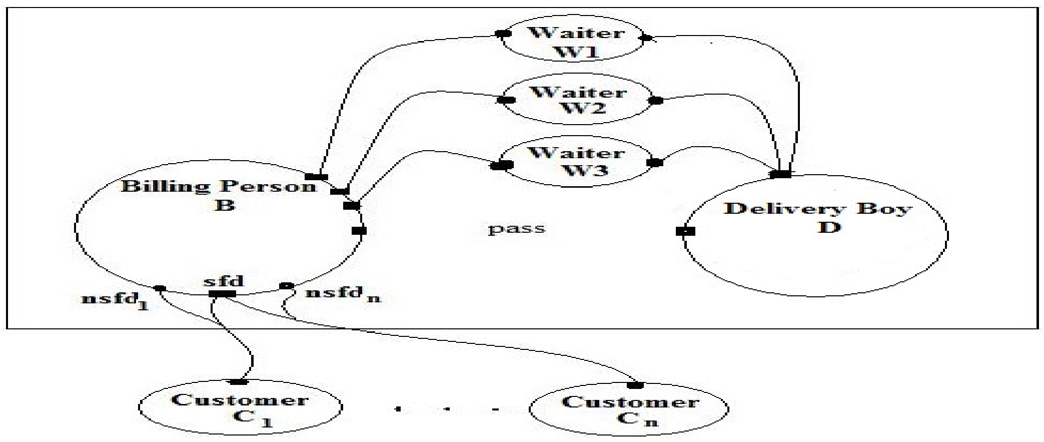
Implement a Client process C_i, server process S_i and ESS process. Usage of threads is also not allowed).

7. **Nitw-Pizza Takeaway**

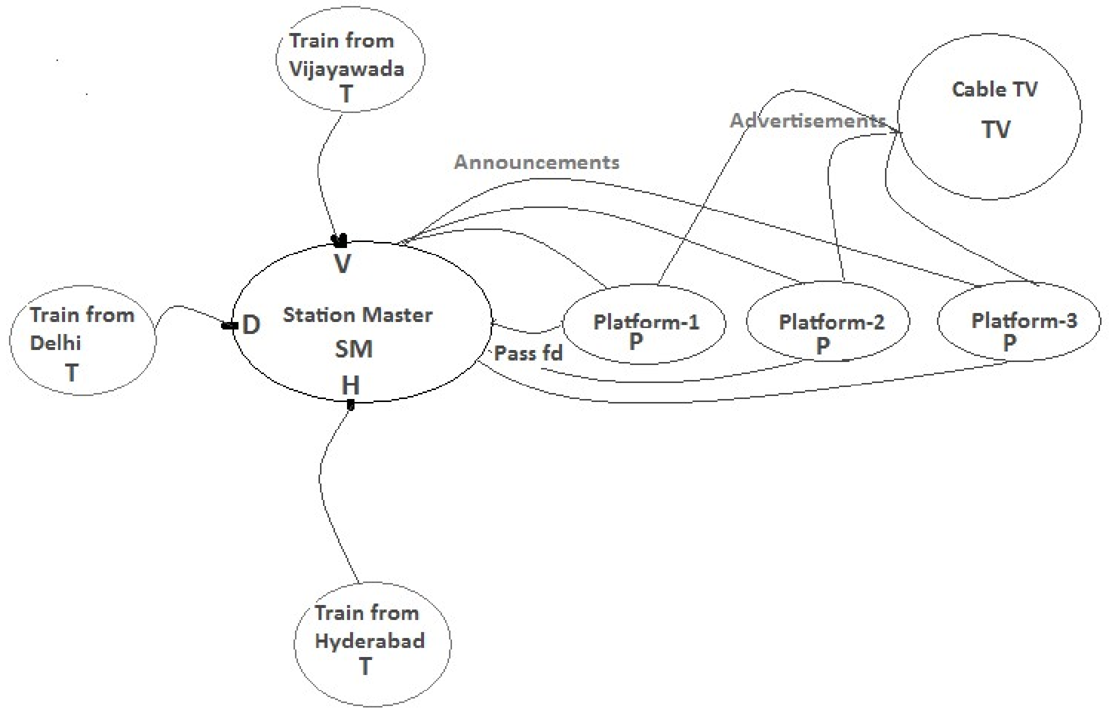
Consider the following different processes at a Pizza Takeaway stall as shown in figure below. Customers(C) contact Billing person(B). Billing person accepts Customer and takes his choice of Combo offer number and receives money for that bill. Then, Billing person handovers Customer to a Waiter(W). Waiter takes the order of menu items from Customer for that Combo offer and signals Billing person. After receiving signal from Waiter, the Billing person passes Customer to Delivery boy(D). Waiter packs those food items of the order and gives the parcel to Delivery boy. Now, Delivery boy handovers the parcel to the appropriate Customer as there may be many Customers at a time.

Implement all the different type of processes involved in this Pizza Takeaway stall Scenario.

(Note: Billing person contacts(knows) Waiters by name and Waiters also contact(know) Delivery boy by name only. Processes B, W, D belong to same computer system, whereas C is from a different computer system.)



8. **Nitw Rail Junction** : Consider the following different processes at a Railway Junction as shown in figure below. Trains (T) may arrive at directional points H, V, D of the Station Master (SM). SM can accept a T if any of the Platform (P) is free. After accepting a T, the SM process gives (sends) the announcement of arrival of T to all the platforms. Then SM handovers the accepted fd of T, to a free platform (P), so that the train (T) sends its arranged sequence of compartments, directly to the platform. Then, P displays the same. After the train leaves the station, it informs the platform (P). The process P notifies the same to SM. There is a Cable Television (TV) process in the railway station, which continuously sends advertisements to platforms for displaying.
Implement all the processes involved in this Railway Junction System Scenario.



14. **NIT-Cricket:** Assume that there is one batsman, one bowler, one umpire and four fielders in the NIT-Cricket ground. The bowler sends a fd (i.e. a file descriptor represents a ball) to the batsman and notifies the umpire. (i.e. has bowled the ball). If the batsman doesn't reply within a specified time, he is declared as out, else, the batsman reads data from the file (fd), which contains two values – (speed, spin). Then he uses a function which takes these two numbers and generates a random number 'r' between 0 and 40. The batsman announces this number 'r'. (i.e. has hit the ball). The fielders are bearing(wearing) numbers as 10, 20, 30, 40. If the number 'r' is a multiple of either 4 or 6, the batsman gets runs as 4(boundary) or 6(sixer). Otherwise, the fielder with the number closest to 'r', informs the umpire that he got 'catch', then, umpire declares that the batsman is out. If the umpire has not been reported by any fielder, he announces the runs, as mentioned above. The fd (i.e. ball) is to be returned to the umpire and then to the bowler for next bowling. Assume that batsman, bowler and umpire are in one same system and fielders are in different systems.
Write essential code for all different processes of the above cricket scenario.

(As the batsman is from CSE, NITW, if he hits, it would be either a boundary(4) or a sixer(6), because he is playing single handed on his OWN !!!)