**What is the client/server Model?**

A client is someone or something which is requesting a service.

A server is someone or something that can fulfill a service request [provides a service].

Programs can act in a similar fashion since one program can request a service that another program can provide. The program requesting the service is a client and the service provider is the server.

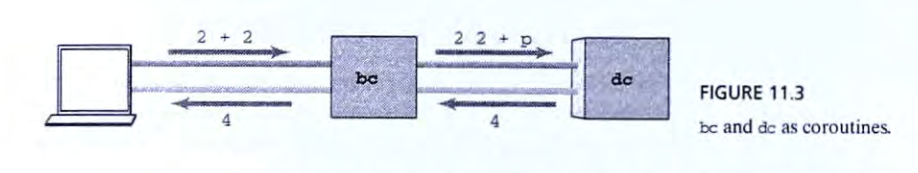
**What are co-routines?**

There are four mechanisms of data sources are:

1. Disk
2. Files
3. Pipes
4. Sockets [outside]

A co-routine is two programs in which are interacting with each other on a continuous basis. Both processes continue to run but control passes from one to the other as each completes its job. The service is also provided within the same system. Both programs are distinct processes that have separate stacks.

Example BC:

Takes input from user and then sends back a result. This is actually not a calculator because it acts as an interface to the actual program providing the calculator service. The reason is because BC is designed to take in information that the user is comfortable with. BC takes the information and turns it into information the calculator service can understand. These two programs communicate through pipes. These are examples of co-routines.

**What are sockets? How do you create one? What is it used for?**

Sockets are a means of interacting with processes that are running on separate systems. You can create one using the socket() system call.

sockid=socket(int domain, int type, int protocol)

You have to create a struct sockaddr\_in saddr; in this struct has information regarding the address, and we modify the information and then bind it in position 2.

1. Create Phone line

sock\_id = socket( PF\_INET, SOCK\_STREAM, 0 ); /\* get a socket \*/

1. Assign a number

if ( bind(sock\_id, (struct sockaddr \*)&saddr, sizeof(saddr)) != 0 oops( "bind" );

1. Allow for incoming calls

if ( listen(sock\_id, 1) != 0 ) oops( "listen" );

1. Wait for a call

sock\_fd = accept(sock\_id, NULL, NULL); /\* wait for call \*/

1. Transfer data

sock\_fp = fdopen(sock\_fd,"w"); /\* we'll write to the \*/

1. Hang up

fclose( sock\_fp ); /\* release connection \*/