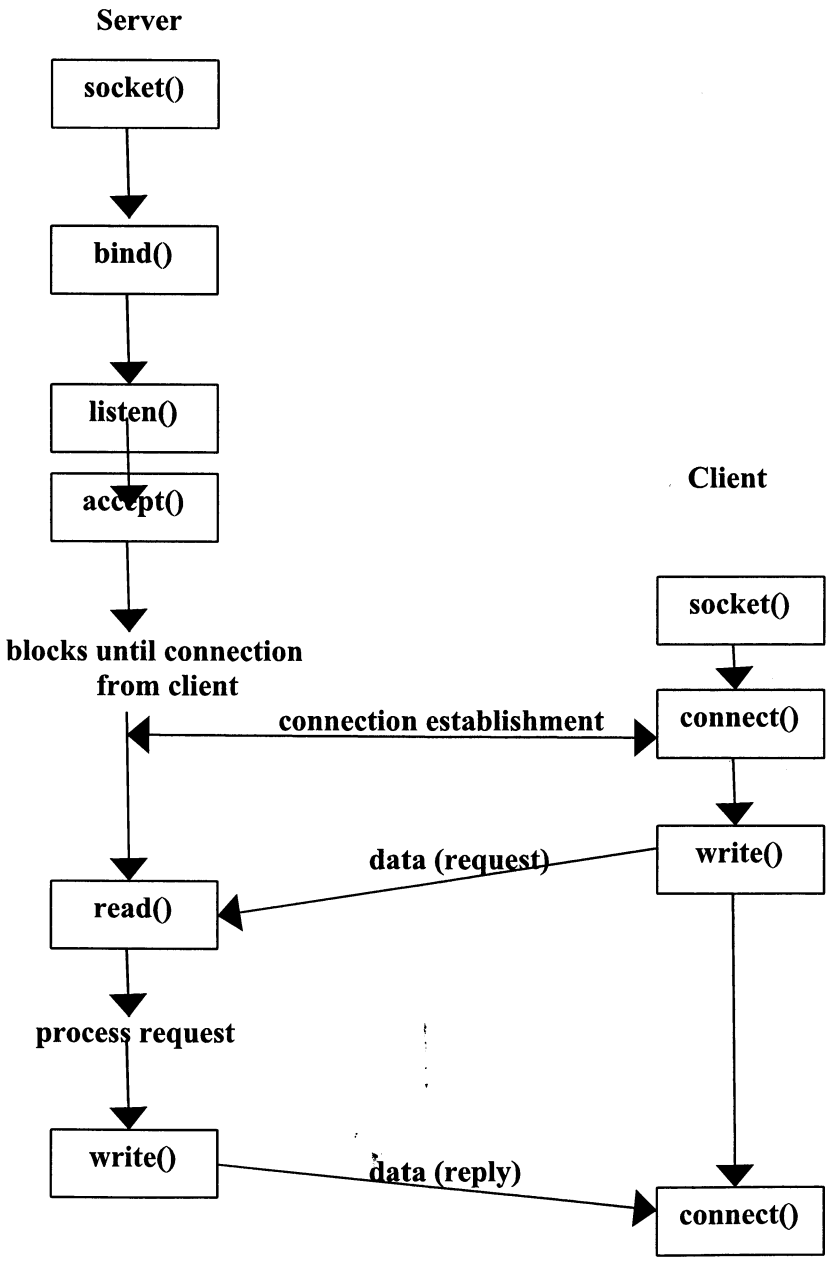
Lab 8: Client-Server Programming

Objective:

Students will write and compile interprocess programs that communicate each other through the network.

[1] Berkeley Socket system calls:



[2] Install Development Tools in both systems

[root@server /root]# **yum grouplist**

[root@server /root]# **yum –y groupinstall "Development Tools"**
[root@client /root]# **yum –y groupinstall "Development Tools"**

[3] Server Example:

[root@server /root]# **vi server.c**

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

void error(const char *msg)
{
   perror(msg);
   exit(1);
}

int main(int argc, char *argv[])
{
    int sockfd, newsockfd, portno;
    socklen_t clilen;
    char buffer[256];
    struct sockaddr_in serv_addr, cli_addr;
    int n;

    if (argc < 2) {
       fprintf(stderr,"ERROR, no port provided\n");
       exit(1);
    }
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0)
      error("ERROR opening socket");
    bzero((char *) &serv_addr, sizeof(serv_addr));
    portno = atoi(argv[1]);
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = INADDR_ANY;
    serv_addr.sin_port = htons(portno);
    if (bind(sockfd, (struct sockaddr *) &serv_addr,
          sizeof(serv_addr)) < 0)
          error("ERROR on binding");
    listen(sockfd,5);
    clilen = sizeof(cli_addr);
    newsockfd = accept(sockfd,
             (struct sockaddr *) &cli_addr,
             &clilen);
    if (newsockfd < 0)
        error("ERROR on accept");
    bzero(buffer,256);
    n = read(newsockfd,buffer,255);
    if (n < 0) error("ERROR reading from socket");
    printf("\n%s",buffer);
    n = write(newsockfd,"I got your message",18);
    if (n < 0) error("ERROR writing to socket");
    close(newsockfd);
    close(sockfd);
    return 0;
}
```

[4] Client Example:

[root@client /root]# **vi client.c**

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

void error(const char *msg)
{
   perror(msg);
   exit(0);
}

int main(int argc, char *argv[])
{
   int sockfd, portno, n;
   struct sockaddr_in serv_addr;
   struct hostent *server;

   char buffer[256];
   if (argc < 2) {
     fprintf(stderr,"usage %s hostname \n", argv[0]);
     exit(0);
   }
   //portno = atoi(argv[2]);
   portno = 51919;
   sockfd = socket(AF_INET, SOCK_STREAM, 0);
   if (sockfd < 0)
      error("ERROR opening socket");
   server = gethostbyname(argv[1]);
   if (server == NULL) {
      fprintf(stderr,"ERROR, no such host\n");
      exit(0);
   }
   bzero((char *) &serv_addr, sizeof(serv_addr));
   serv_addr.sin_family = AF_INET;
   bcopy((char *)server->h_addr,
       (char *)&serv_addr.sin_addr.s_addr,
       server->h_length);
   serv_addr.sin_port = htons(portno);
   if (connect(sockfd,(struct sockaddr *) &serv_addr,sizeof(serv_addr)) < 0)
      error("ERROR connecting");
   printf("Please enter the message: ");
   bzero(buffer,256);
   fgets(buffer,255,stdin);
   n = write(sockfd,buffer,strlen(buffer));
   if (n < 0)
       error("ERROR writing to socket");
   bzero(buffer,256);
   n = read(sockfd,buffer,255);
   if (n < 0)
       error("ERROR reading from socket");
   printf("%s\n",buffer);
   close(sockfd);
   return 0;
}
```

[5] Compile Server: ***(on the server machine)***
[root@server /root]# **gcc –o server server.c**

[6] Compile Client: ***(on the client machine)***
[root@client /root]# **gcc –o client client.c**

[7] Run Server:
[root@server /root]# **server 51919 &**

[8] Run Client: *(Make usre that your DNS know the server name)*
[root@client /root]# **client server.yourdomain.com 51919**

Test the client-server program.

Lab 8: Client-Server Programming Report

Name:

[1] What did you learn from this lab?

[2] Your troubleshooting procedures while you were doing this lab.

[3] What transport layer protocol this client-server model uses?

[4] Show your firewall configurations on both client and server.

[5] Modify the client-server program to "cookie server" or "chatting program".