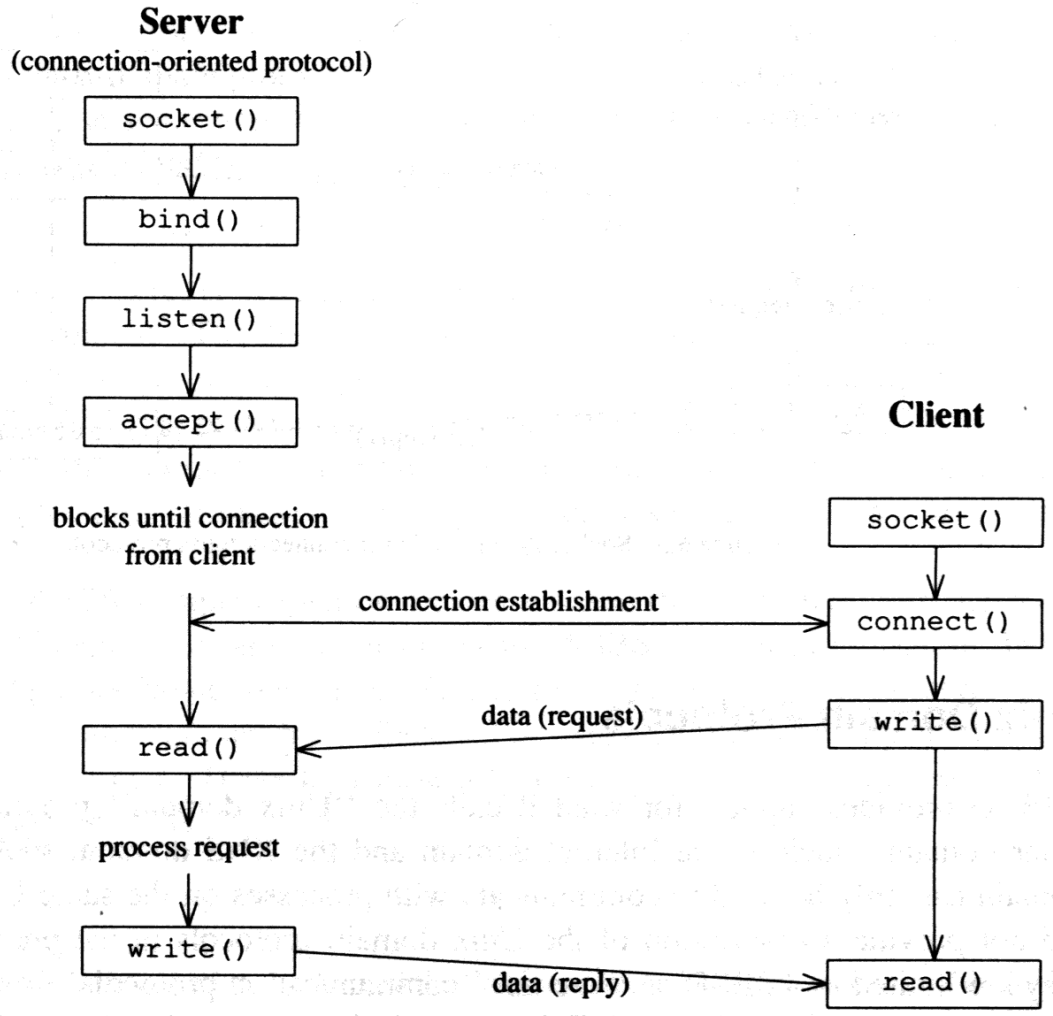
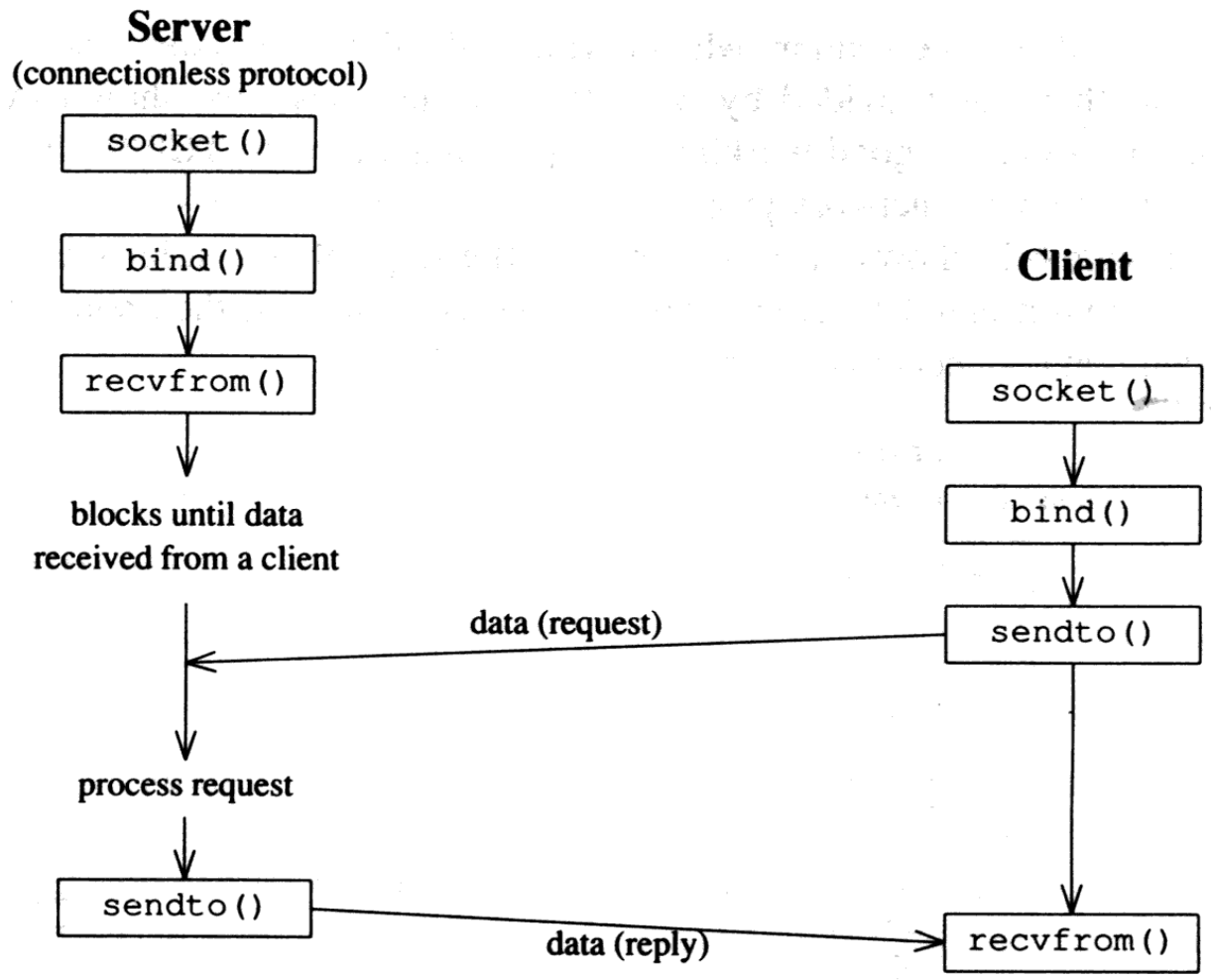
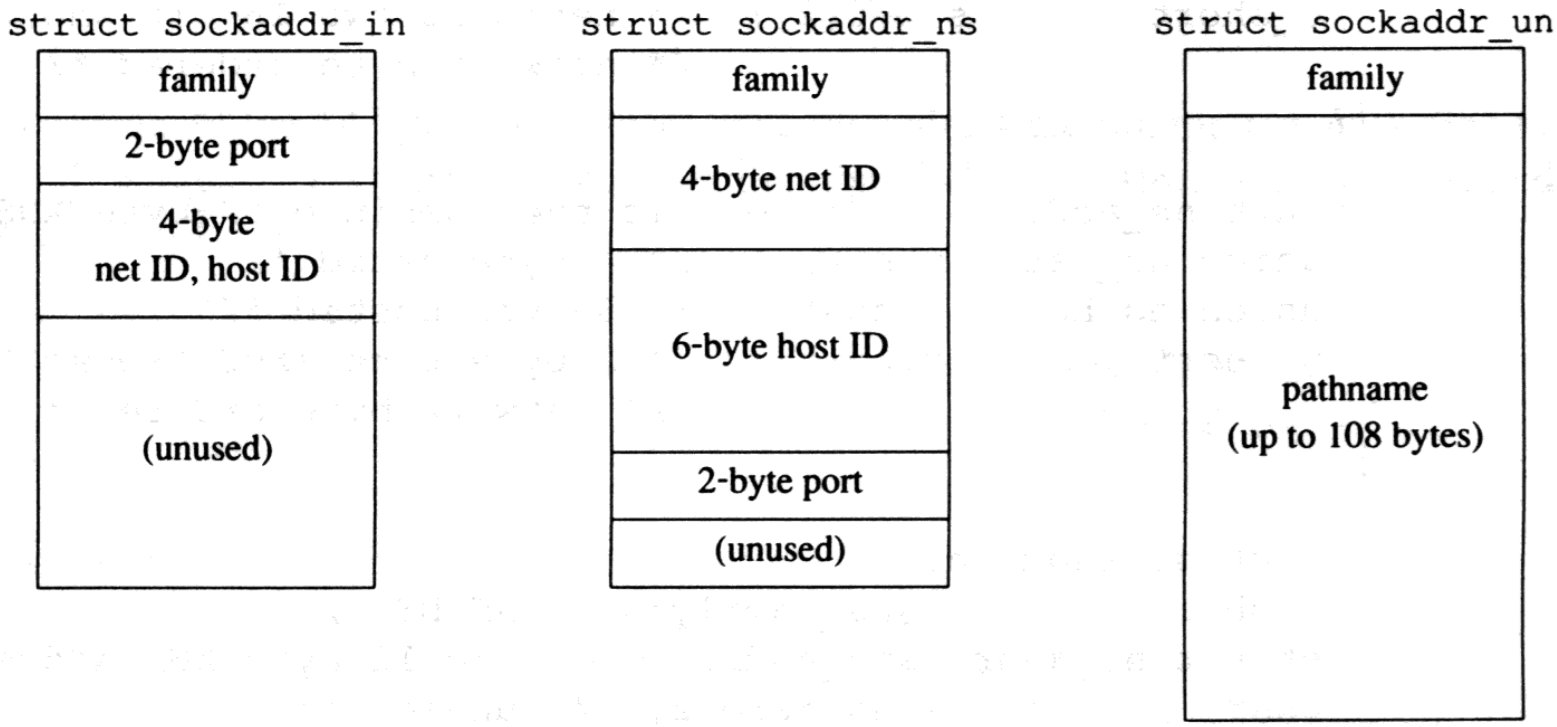
Connection-Oriented Protocol



Connectionless Protocol



Socket Addresses



Example (TCP): Client -- (1)

```
main(argc, argv)
int  argc;
char        *argv[];
{
     int                  sockfd;
     struct sockaddr_in           serv_addr;
     pname = argv[0];

     // Fill in the structure "serv_addr" with the address of the
     // server that we want to connect with.

     bzero((char *) &serv_addr, sizeof(serv_addr));
     serv_addr.sin_family      = AF_INET;
     serv_addr.sin_addr.s_addr = inet_addr(SERV_HOST_ADDR);
     serv_addr.sin_port   = htons(SERV_TCP_PORT);
```

Example(TCP): Client -- (2)

```
    // Open a TCP socket (an Internet stream socket).
    if ( (sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
            err_sys("client: can't open stream socket");

    // Connect to the server.
    if (connect(sockfd, (struct sockaddr *)
                &serv_addr, sizeof(serv_addr)) < 0)
            err_sys("client: can't connect to server");

    str_cli(stdin, sockfd);          /* do it all */

    close(sockfd);
    exit(0);
}
```

Example(TCP): Client -- (3)

```
#define   MAXLINE         512
str_cli(FILE *fp, int sockfd)
{
    int     n;
    char   sendline[MAXLINE], recvline[MAXLINE + 1];
    while (fgets(sendline, MAXLINE, fp) != NULL) {
            n = strlen(sendline);
            if (writen(sockfd, sendline, n) != n) err_sys("str_cli: writen error ");

            // Now read a line from the socket and write it to our standard output.
            n = readline(sockfd, recvline,  MAXLINE);
            if (n < 0) err_dump("str_cli: readline error");
            recvline[n] = 0;     /* null terminate */
            fputs(recvline, stdout);
    }
}
```

Example(TCP): Server -- (1)

```
#include   "inet.h"

main(argc, argv)
int  argc;
char         *argv[];
{
     int                         sockfd, newsockfd, clilen, childpid;
     struct sockaddr_in          cli_addr, serv_addr;

     pname = argv[0];

     /*
      * Open a TCP socket (an Internet stream socket).
      */
     if ( (sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0)
          err_dump("server: can't open stream socket");
```

Example(TCP): Server -- (2)

```
/*
 * Bind our local address so that the client can send to us.
 */

bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family      = AF_INET;
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_addr.sin_port        = htons(SERV_TCP_PORT);

if (bind(sockfd, (struct sockaddr *)
        &serv_addr, sizeof(serv_addr)) < 0)
    err_dump("server: can't bind local address");

listen(sockfd, 5);
```

Example(TCP): Server -- (3)

```
    for ( ; ; ) {
        clilen = sizeof(cli_addr);
        newsockfd = accept(sockfd, (struct sockaddr *) &cli_addr, &clilen);
        if (newsockfd < 0)  err_dump("server: accept error");
        if ( (childpid = fork()) < 0)       err_dump("server: fork error");
        else if (childpid == 0) {   /* child process */
            /* close original socket */
            close(sockfd);
            /* process the request */
            str_echo(newsockfd);
            exit(0);
        }
        close(newsockfd); /* parent process */
    }
}
```

Example(TCP): Server -- (4)

```
#define    MAXLINE          512
str_echo(sockfd)
int  sockfd;
{
     int      n;
     char     line[MAXLINE];

     for ( ; ; ) {
          n = readline(sockfd, line, MAXLINE);
          if (n == 0)     return;    /* connection terminated */
          else if (n < 0)     err_dump("str_echo: readline error");
          if (writen(sockfd, line, n) != n)
                    err_dump("str_echo: writen error");
     }
}
```

Example(TCP): Server -- (6)

```
int readline(int fd, char * ptr, int maxlen)
{
    int     n, rc;                  char        c;
    for (n = 1; n < maxlen; n++) {
        if ( (rc = read(fd, &c, 1)) == 1) {
                *ptr++ = c;
                if (c == '\n')      break;
        } else if (rc == 0) {
            if (n == 1)             return(0);      /* EOF, no data read */
            else                    break;          /* EOF, some data was read */
        } else
            return(-1);         /* error */
    }
    *ptr = 0;
    return(n);
}
```

Example(UDP): Client -- (1)

```
main(int argc, char *argv[])
{
        int                             sockfd;
        struct sockaddr_in  cli_addr, serv_addr;

        pname = argv[0];

        /*
         * Fill in the structure "serv_addr" with the address of the
         * server that we want to send to.
         */

        bzero((char *) &serv_addr, sizeof(serv_addr));
        serv_addr.sin_family      = AF_INET;
        serv_addr.sin_addr.s_addr =  inet_addr(SERV_HOST_ADDR);
        serv_addr.sin_port = htons(SERV_UDP_PORT);
```

Example(UDP): Client -- (2)

```
// Open a UDP socket (an Internet datagram socket).
if ( (sockfd = socket(AF_INET,  SOCK_DGRAM, 0)) < 0)
        err_dump("client: can't open datagram socket");
// Bind any local address for us.
bzero((char *) &cli_addr, sizeof(cli_addr));  /* zero out */
cli_addr.sin_family      = AF_INET;
cli_addr.sin_addr.s_addr = htonl(INADDR_ANY);
cli_addr.sin_port        = htons(0);
if (bind(sockfd, (struct sockaddr *) &cli_addr, sizeof(cli_addr)) < 0)
        err_dump("client: can't bind local address");
dg_cli(stdin, sockfd, (struct sockaddr *) &serv_addr,
sizeof(serv_addr));
close(sockfd);
exit(0);
}
```

Example(UDP): Client -- (3)

```
#define    MAXLINE          512
dg_cli(FILE *fp, int sockfd, struct sockaddr *pserv_addr, int servlen)
{
    int     n;
    char    sendline[MAXLINE], recvline[MAXLINE + 1];
    while (fgets(sendline, MAXLINE, fp) != NULL) {
        n = strlen(sendline);
        if (sendto(sockfd, sendline, n, 0, pserv_addr, servlen) != n)
            err_dump("dg_cli: sendto error on socket");
        n = recvfrom(sockfd, recvline, MAXLINE, 0, 0, 0);
        if (n < 0)
            err_dump("dg_cli: recvfrom error");
        recvline[n] = 0;        /* null terminate */
        fputs(recvline, stdout);
    }
}
```

Example(UDP): Server -- (1)

```
main(int argc, char **argv)
{
    int                             sockfd;
    struct sockaddr_in  serv_addr, cli_addr;
    pname = argv[0];
    if ( (sockfd = socket(AF_INET,  SOCK_DGRAM, 0)) < 0)
        err_dump("server: can't open datagram socket");
    // Bind our local address so that the client can send to us.
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family      = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port        = htons(SERV_UDP_PORT);
    if (bind(sockfd, (struct sockaddr *)  &serv_addr, sizeof(serv_addr)) < 0)
        err_dump("server: can't bind local address");

    dg_echo(sockfd, (struct sockaddr *)
                &cli_addr, sizeof(cli_addr));
}
```

Example(UDP): Server -- (2)

```
#define     MAXMESG             2048
dg_echo(int sockfd, struct sockaddr *pcli_addr, int maxclilen)
{
    int     n, clilen;
    char    mesg[MAXMESG];

    for ( ; ; ) {
        clilen = maxclilen;
        n = recvfrom(sockfd, mesg, MAXMESG, 0, pcli_addr, &clilen);
        if (n < 0)  err_dump("dg_echo: recvfrom error");

        if (sendto(sockfd, mesg, n, 0, pcli_addr, clilen) != n)
            err_dump("dg_echo: sendto error");
    }
}
```