Sockets

Dong-kun Shin
Embedded Software Laboratory
Sungkyunkwan University
http://nyx.skku.ac.kr

Echo Client (1)


```
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>

#define MAXLINE 80

int main (int argc, char *argv[]) {
   int n, cfd;
   struct hostent *h;
   struct sockaddr_in saddr;
   char buf[MAXLINE];
   char *host = argv[1];
   int port = atoi(argv[2]);

   if ((cfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
      printf("socket() failed.\n");
      exit(1);
   }
```

Echo Client (2)


```
    if ((h = gethostbyname(host)) == NULL) {
        printf("invalid hostname %s\n", host);
        exit(2);
    }
    bzero((char *)&saddr, sizeof(saddr));
    saddr.sin_family = AF_INET;
    bcopy((char *)h->h_addr, (char *)&saddr.sin_addr.s_addr, h->h_length);
    saddr.sin_port = htons(port);

    if (connect(cfd,(struct sockaddr *)&saddr,sizeof(saddr)) < 0) {
        printf("connect() failed.\n");
        exit(3);
    }
    while ((n = read(0, buf, MAXLINE)) > 0) {
        write(cfd, buf, n);
        n = read(cfd, buf, MAXLINE);
        write(1, buf, n);
    }
    close(cfd);
}
```

bind()

- int bind (int sockfd, struct sockaddr *myaddr,
  socklen_t addrlen)
  - **bind()** gives the socket **sockfd** the local address **myaddr**.
  - **myaddr** is **addrlen** bytes long.
  - Servers bind their well-known port when they start.
  - If a TCP server binds a specific IP address to its socket, this restricts the socket to receive incoming client connections destined only to that IP address.
  - Normally, a TCP client let the kernel choose an ephemeral port and a client IP address.

listen()

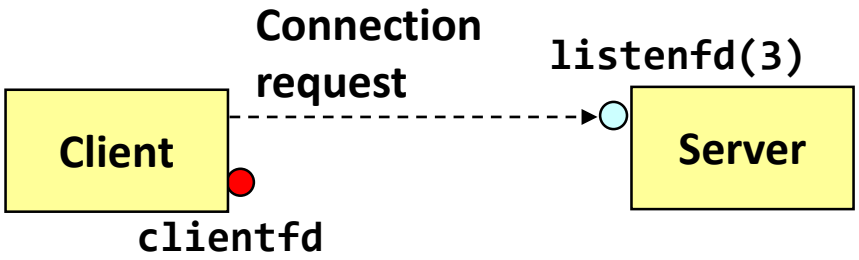
- int listen (int sockfd, int backlog)
  - **listen()** converts an unconnected socket into a passive socket, indicating that the kernel should accept incoming connection requests.
    - When a socket is created, it is assumed to be an active socket, that is, a client socket that will issue a **connect().**
  - **backlog** specifies the maximum number of connections that the kernel should queue for this socket.
  - Historically, a backlog of 5 was used, as that was the maximum value supported by 4.2BSD.
    - Busy HTTP servers must specify a much larger backlog, and newer kernels must support larger values.

accept() (1)

- int accept (int sockfd, struct sockaddr *cliaddr, socklen_t *addrlen)
  - **accept()** blocks waiting for a connection request.
  - **accept()** returns a connected descriptor with the same properties as the listening descriptor.
    - The kernel creates one connected socket for each client connection that is accepted.
    - Returns when the connection between client and server is created and ready for I/O transfers.
    - All I/O with the client will be done via the connected socket.
  - The **cliaddr** and **addrlen** arguments are used to return the address of the connected peer process (the client)

accept() (2)

1. Server blocks in `accept`, waiting for connection request on listening descriptor `listenfd`.



2. Client makes connection request by calling and blocking in `connect`.

3. Server returns `connfd` from `accept`. Client returns from `connect`. Connection is now established between `clientfd` and `connfd`.

accept() (3)

- Listening descriptor
  - End point for client connection requests
  - Created once and exists for lifetime of the server
- Connected descriptor
  - End point of the connection between client and server
  - A new descriptor is created each time the server accepts a connection request from a client.
  - Exists only as long as it takes to service client.
- Why the distinction?
  - Allows for concurrent servers that can communicate over many client connections simultaneously.

Echo Server (1)


```
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include <arpa/inet.h>

#define MAXLINE 80

int main (int argc, char *argv[]) {
   int n, listenfd, connfd, caddrlen;
   struct hostent *h;
   struct sockaddr_in saddr, caddr;
   char buf[MAXLINE];
   int port = atoi(argv[1]);

   if ((listenfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
      printf("socket() failed.\n");
      exit(1);
   }
```

Echo Server (2)


```
bzero((char *)&saddr, sizeof(saddr));
saddr.sin_family = AF_INET;
saddr.sin_addr.s_addr = htonl(INADDR_ANY);
saddr.sin_port = htons(port);
if (bind(listenfd, (struct sockaddr *)&saddr,
         sizeof(saddr)) < 0) {
   printf("bind() failed.\n");
   exit(2);
}
if (listen(listenfd, 5) < 0) {
   printf("listen() failed.\n");
   exit(3);
}
while (1) {
   caddrlen = sizeof(caddr);
   if ((connfd = accept(listenfd, (struct sockaddr *)&caddr,
                        &caddrlen)) < 0) {
      printf ("accept() failed.\n");
      continue;
   }
```

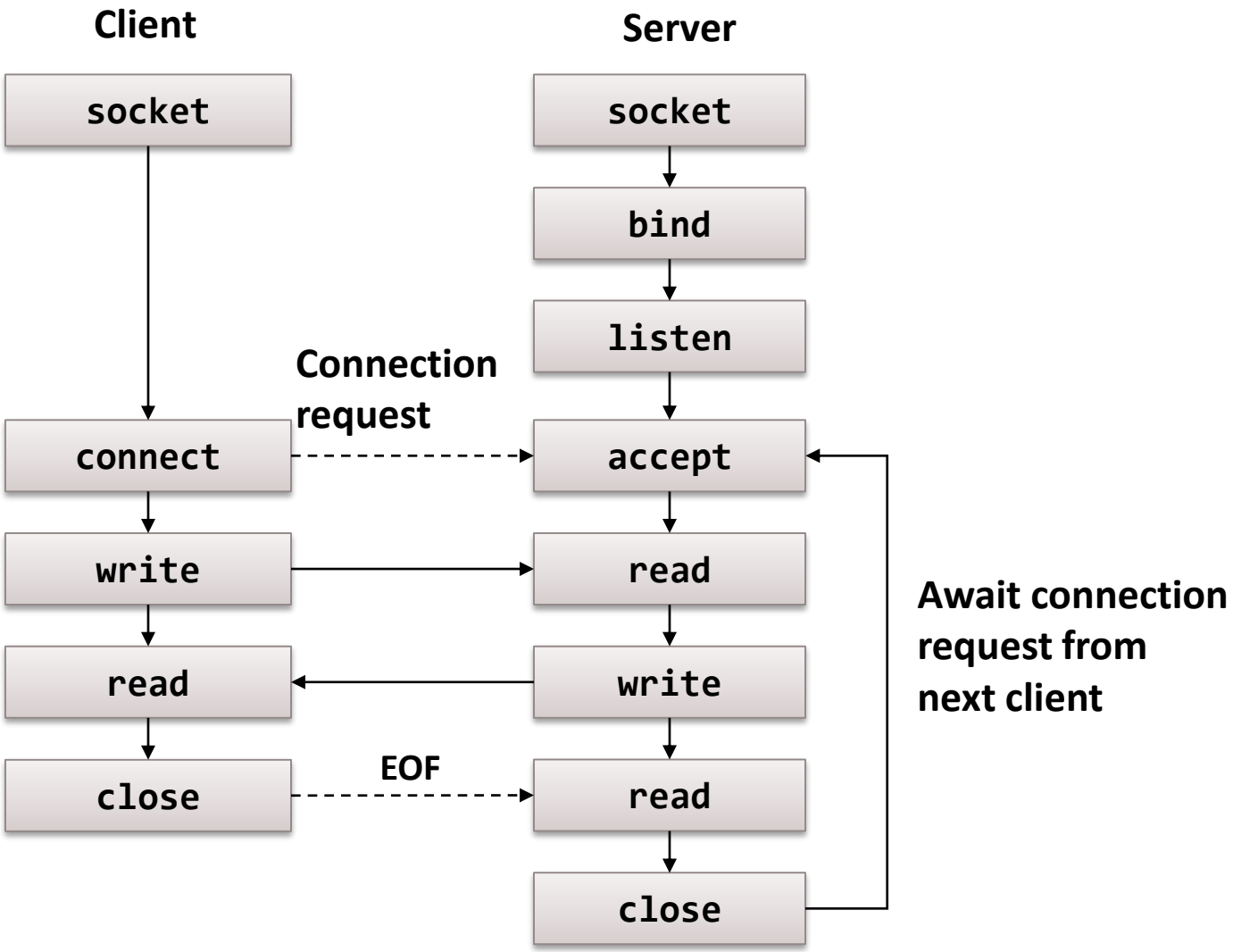
Echo Server (3)


```
        h = gethostbyaddr((const char *)&caddr.sin_addr.s_addr,
            sizeof(caddr.sin_addr.s_addr), AF_INET);
        printf("server connected to %s (%s)\n",
            h->h_name,
            inet_ntoa(*(struct in_addr *)&caddr.sin_addr));

        // echo
        while ((n = read(connfd, buf, MAXLINE)) > 0) {
           printf ("got %d bytes from client.\n", n);
           write(connfd, buf, n);
        }

        printf("connection terminated.\n");
        close(connfd);
    }
}
```

Echo Server (4)



Exercise

- **Client program**
  - Server (127.0.0.1)로 수식 전달
    - Ex. 1 + 231 * 43

- **Server program**
  - Client에서 전달받은 수식 계산하여 client로 전송