Socket (1A)

- Socket

Young Won Lim
11/22/2012

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Server Side Steps

- Create a socket with the socket() system call
- Bind the socket to an address using the bind() system call. For a server socket on the Internet, an address consists of a port number on the host machine.
- Listen for connections with the listen() system call
- Accept a connection with the accept() system call. This call typically blocks until a client connects with the server.
- Send and receive data

sockaddr

```
int bind(int sockfd, const struct sockaddr *addr,socklen_t addrlen);
```

When a socket is created,
it exists in a name space (address family)
but has no address assigned to it.
bind() assigns the address specified by addr
to the socket referred to by sockfd.
addrlen specifies the size, in bytes,
of the address structure pointed to by addr.

It is normally necessary to assign a local address using bind()
before a SOCK_STREAM socket may receive connections

sockaddr

```
struct sockaddr {
    sa_family_t sa_family;
    char        sa_data[14];
}
```

AF_INET	: ip
AF_INET6	: ipv6
AF_UNIX	: unix
AF_APPLETALK	: ddp
AF_PACKET	: packet
AF_X25	: x25
AF_NETLINK	: netlink

```
Bind the socket to an address
For a server socket on the Internet
an address - a port number on the host machine.
```

sockaddr_in

```
struct sockaddr_in
{
  short            in_family; /* must be AF_INET */
  u_short          sin_port;
  struct  in_addr  sin_addr;
  char             sin_zero[8]; /* Not used, must be zero */
};
```

```
typedef uint32_t in_addr_t;
struct in_addr
{
    in_addr_t s_addr;
};
```

the IP address of the host. Server →
the IP address of the server machine
→ a symbolic constant INADDR_ANY

```
struct sockaddr_in serv_addr, cli_addr;
bzero((char *) &serv_addr, sizeof(serv_addr));
/* sets all values in a buffer to zero */

int portno;
portno = atoi(argv[1]);
```

```
serv_addr.sin_family          = AF_INET;
serv_addr.sin_port            = htons(portno);
serv_addr.sin_addr.s_addr     = INADDR_ANY;

bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr))
```

Endian

unsigned long **htonl**(unsigned long)
   host to network conversion for long ints (4 bytes)
unsigned short **htons**(unsigned short)
   host to network conversion for short ints (2 bytes)
unsigned long **ntohl**(unsigned long)
   network to host conversion for long ints
unsigned short **ntohs**(unsigned short)
   network to host conversion for short ints

Big endian:
the highest order byte is stored at A
the lowest order byte is stored at address A+3.

Little endian:
the least significant byte is stored at A
the most significant byte is at address A+3.

Computer networks are big endian

listen() and accept()

listen(sockfd,5);

5: the size of the **backlog queue**, i.e., the number of connections that can be **waiting** while the process is handling a particular connection.

struct sockaddr_in serv_addr, cli_addr;

clilen = sizeof(**cli_addr**);
newsockfd = accept(sockfd, (struct sockaddr *) &**cli_addr**, &**clilen**);

The accept() system call causes the process to block until a client connects to the server.
This wakes up the process when a connection from a client has been successfully established.
It returns a new file descriptor, and all communication on this connection should be done using the new file descriptor.

read() & write()

```
bzero(buffer,256);
n = read(newsockfd,buffer,255);
```

```
n = write(newsockfd,"I got your message",18);
```

Client Side Steps

- Create a socket with the socket() system call
- Connect the socket to the address of the server using the connect() system call.
- Send and receive data

hostent

```
int                    sockfd, portno, n;
struct sockaddr_in     serv_addr;
struct hostent         *server;
```

h_name	Official name of the host.
h_aliases	A zero terminated array of alternate names for the host.
h_addrtype	The type of address being returned; C urrently always **AF_INET**.
h_length	The length, in bytes, of the address.
h_addr_list	A pointer to a list of network addresses for the named host. Host addresses are returned in network byte order.

```
struct  hostent
{
  char        *h_name;         /* official name of host */
  char        **h_aliases;     /* alias list */
  int         h_addrtype;      /* host address type */
  int         h_length;        /* length of address */
  char        **h_addr_list;   /* list of addresses from name server */
  #define h_addr  h_addr_list[0]  /* address, for backward compatiblity */
};
```

```
struct sockaddr_in
{
  short             in_family; /* must be AF_INET */
  u_short           sin_port;
  struct  in_addr   sin_addr;
  char              sin_zero[8]; /* Not used, must be zero */
};
```

```
typedef uint32_t in_addr_t;
struct in_addr
{
   in_addr_t s_addr;
};
```

the IP address of the host. Server → the IP address of the server machine → a symbolic constant INADDR_ANY

gethostbyname()

```
int                     sockfd, portno, n;
struct sockaddr_in      serv_addr;
struct hostent          *server;
```

```
server = gethostbyname(argv[1]);

struct hostent *gethostbyname(char *name)
```

Takes such a **name** as an argument and returns a pointer to a **hostent** containing information about that host.

The field char *h_addr contains the **IP address**.

```
struct  hostent
{
  char        *h_name;          /* official name of host */
  char        **h_aliases;      /* alias list */
  int         h_addrtype;       /* host address type */
  int         h_length;         /* length of address */
  char        **h_addr_list;    /* list of addresses from name server */
  #define h_addr  h_addr_list[0]  /* address, for backward compatiblity */
};
```

gethostbyname()

```
int                    sockfd, portno, n;
struct sockaddr_in     serv_addr;
struct hostent         *server;
```

```
void bcopy(char *s1, char *s2, int length)

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);
```

```
struct sockaddr_in
{
  short             in_family;
  u_short           sin_port;
  struct  in_addr   sin_addr;
  char              sin_zero[8];
};
```

```
typedef uint32_t in_addr_t;
struct in_addr
{
    in_addr_t s_addr;
};
```

```
struct  hostent
{
  char      *h_name;        /* official name of host */
  char      **h_aliases;    /* alias list */
  int       h_addrtype;     /* host address type */
  int       h_length;       /* length of address */
  char      **h_addr_list;  /* list of addresses from name server */
  #define h_addr  h_addr_list[0]  /* address, for backward compatiblity */
};
```

gethostbyname()

```
int                     sockfd, portno, n;
struct sockaddr_in      serv_addr;
struct hostent          *server;


connect(sockfd, &serv_addr, sizeof(serv_addr))

bzero(buffer,256);
 fgets(buffer,255,stdin);

n = write(sockfd,buffer,strlen(buffer));

n = read(sockfd,buffer,255);
```

Reference

References

[1] http://en.wikipedia.org/
[2] http://www.linuxhowtos.org/manpages/2/bind.htm
[3] http://cs.baylor.edu/~donahoo/practical/CSockets/textcode.html
[4] http://www.cs.rpi.edu/~moorthy/Courses/os98/Pgms/socket.html