



The IPv6 Programming Environment

Rahul Banerjee

**Assistant Professor: CS&IS Group & Assistant Dean: DLP
Birla Institute of Technology & Science
Pilani (India)**

Home Page: <http://www.bits-pilani.ac.in/~rahul/>

E-mail: rahul@bits-pilani.ac.in

Copyright: Rahul Banerjee, BITS, Pilani (India)



A Quick tour to basics

- ◆ Every network protocol has its own definition of *Network Address*.
- ◆ In C, a protocol implementation provides a *struct sockaddr* as the elementary form of a *Network Address*.
- ◆ A sample definition of *struct sockaddr*

```
#include <sys/socket.h>

struct sockaddr {
    unsigned short sa_family;
    char sa_data [MAXSOCKADDRDATA]
}
```

A Quick tour to basics ...

- ◆ In Linux, sockets are created by the *socket()* system call.
- ◆ This call returns a *file descriptor* for the socket that is yet to be initialized.
- ◆ Then the socket is initialized by binding it to a particular protocol and address using the *bind()* system call.

```
#include <sys/socket.h>
```

```
int socket (int domain, int type, int protocol);
```

Here,

- ◆ the parameter **domain** specifies the PF;
- ◆ parameter **type** usually specifies either of SOCK_STREAM or SOCK_DGRAM;
- ◆ parameter **protocol** specifies the protocol to be used (0=> default protocol associated).

```
int bind (int sock, struct sockaddr * my_addr, int addrlen);
```

Here,

- ◆ the parameter **sock** is the socket-in-question;
- ◆ parameter **sockaddr** is the address of protocol;
- ◆ parameter **addrlen** is the length of the address for the local end-point.

A Quick tour to basics ...

- ◆ Next, *listen()* system call is executed for informing the system that the process is now ready to allow other processes establish a connection to this socket at the specified end-point.
- ◆ This step does not really establish a connection by itself, however!
- ◆ Now, the *accept ()* system call is made for accepting the connection requests.
- ◆ *accept ()* is a blocking call as it blocks until a process requests a connection. In case, the socket has been marked as 'non-blocking' by the *fcntl()* call, *accept()* would return an error if no process is requesting it for a connection.

```
#include <sys/socket.h>
```

```
int listen (int sock, int backlog);
```

Here,

- ◆ the parameter **sock** is the socket-in-question;
- ◆ parameter **backlog** is the number of connection requests that may be pending on the socket before any further connection requests are explicitly refused.

```
int accept (int sock, struct sockaddr *  
addr, int * addrlen);
```

- ◆ The *select ()* system call can also be made for determining if any connection request is currently pending to a socket.



A Quick tour to basics ...

- ◆ A Client attempts to connect to a Server by creating a socket, binding it to an address (optionally) and making the *connect()* call to the Server at the known address.



A Subset of Address Families Used in Linux Environment

- ◆ Unix / Linux Domain:
`AF_UNIX`
- ◆ TCP/IPv4 Domain
`AF_INET`
- ◆ TCP/IPv6 Domain
`AF_INET6`
- ◆ Novell NetWare Domain:
`AF_IPX`
- ◆ AppleTalk Domain:
`AF_APPLETALK`



A Subset of Protocol Families Used in Linux Environment

- ◆ Unix / Linux Domain:

PF_UNIX

- ◆ TCP/IPv4 Domain

PF_INET

- ◆ TCP/IPv6 Domain

PF_INET6

- ◆ Novell NetWare Domain:

PF_IPX

- ◆ AppleTalk Domain:

PF_APPLETALK



Socket Errors (ERRNO VALUES)

- ◆ ENOTSOCK
- ◆ EDESTADDRREQ
- ◆ EPROTOTYPE
- ◆ ENOPROTOOPT
- ◆ EPROTONOSUPPORT
- ◆ ESOCKTNOSUPPORT
- ◆ EPFNOSUPPORT
- ◆ EAFNOSUPPORT
- ◆ ENOTCONN
- ◆ ETIMEDOUT
- ◆ ECONNREFUSED
- ◆ EADDRINUSE
- ◆ EADDRNOTAVAIL
- ◆ ENETDOWN
- ◆ ENETUNREACH
- ◆ ENETRESET
- ◆ ECONNABORTED
- ◆ ECONNRESET
- ◆ ENOBUFS
- ◆ EISCONN
- ◆ EHOSTDOWN
- ◆ EHOSTUNREAD

This document was created with Win2PDF available at <http://www.daneprairie.com>.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.