EMBEDDED OPERATING SYSTEMS

Embedded Linux on Beaglebone Black

Socket Types

- IPC / Unix Domain Sockets
  - Enable channel-based communication
    - Between processes on same physical device
  - Use the local system kernel
- Network Sockets
  - Enable bidirectional communication
    - Between processes on different devices on the network
  - Need an underlying network protocol
    - TCP / UDP

- APIs for usage are the same
  - Only the input argument enums change

Network Socket: What?

- **Network Socket**
  - **Software Structure**
  - Within a **Network Node**
  - Of a Computer Network
  - Serves as an **Endpoint**
  - For **Sending** and **Receiving** Data
  - Over the Network

- Bidirectional channel of communication
  - Between 2 machines on the Network

N/W Sockets: Communication

- Communication types:
  - Stream communication
    - Connection-oriented (TCP)
    - Reliable, error-free, sequenced, no message boundaries
      - Like “pipes” on the network
    - Underlying protocol retransmits on errors
      - Throws errors on broken connections
  - Datagram communication
    - Connectionless (UDP)
    - Each datagram is addressed individually
    - Order and reliability not guaranteed

Client-Server Communication

- The server
  - Creates a network socket
  - Binds to this socket
    - Host IP Address and Port Number (>1024)
  - Listens on this socket
  - Accepts connections from ‘clients’

- The client
  - Creates a network socket
  - Connects to the ‘server’
    - Using server’s IP Address and Port Number
  - Once server accepts connection
    - Sends / Receives data over the Network Socket

Socket API: socket()

- Socket creation

  #include <sys/types.h>
  #include <sys/socket.h>
  *int **socket**(int domain, int type, int protocol);*

  domain:
    - AF_UNIX / AF_LOCAL: IPC
    - AF_INET: IPv4, AF_INET6: IPv6
    - AF_CAN, AF_BLUETOOTH

  type:
    - SOCK_STREAM / SOCK_DGRAM

  protocol:
    - 0 / IPPROTO_TCP: for TCP
    - 1 / IPPROTO_UDP: for UDP

  Returns a socket fd / -1 on error and errno is set accordingly

Socket API: bind()

- Bind the socket to a name

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

  sockfd: FD returned by socket() call

  addr: struct sockaddr {   / sockaddr_in */*

  sa_family_t ***sin_family****; // AF_INET*

  in_port_t ***sin_port****; // Port number*

  struct in_addr ***sin_addr****; // IPv4 address*

  };

  addrlen: Size of addr

  Returns 0 / -1 and errno set accordingly

Socket APIs: listen(), accept()

- Listen to incoming requests

  *int **listen**(int sockfd, int backlog);*

  backlog: Max number of connections allowed

  Returns 0 / -1 (errno set)

- Accept an incoming request – blocking wait call

  *int **accept**(int sockfd, struct sockaddr *addr, socklen_t *addrlen);*

  addr: sockaddr struct using Client IP address

  Returns FD for communication (client_fd) / -1 (errno set)

Socket API: htonx(), connect()

- Convert host to network-byte order

  #include <arpa/inet.h>

  *uint32_t **htonl**(uint32_t hostlong); // convert long*

  *uint16_t **htons**(uint16_t hostshort); // convert short*

- Used by client to “connect” to server

  *int **connect**(int sockfd, struct sockaddr *addr, socklen_t *addrlen);*

Socket API: gethostbyname()

- Convert host name/IP to ***hostent*** structure

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

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 */
    };
```

Network Socket exercise

- ***sock-server.c***
  - Creates a network socket
    - Binds it to **127.0.0.1 (localhost)**
    - Listens on port **19000**
  - Accepts connections from client
  - Receives data from client
  - Sends it back as a echo

- ***sock-client.c***
  - Creates a network socket
  - Connects to the server’s port
  - Sends data to server
    - Over the network socket
  - Receives data from server
    - Over the network socket

THANK YOU!