Software Lab

Sockets

Roberto Farina
roberto.farina@cefriel.it

Summary

- General concepts
- Details about parameters
- System calls
- Basic operations
- Local sockets
- Internet domain sockets

General concepts

- BIDIRECTIONAL communication technique between processes residing on the same host or on different hosts
- Transmitted data are divided in packets
- Sockets are represented by file descriptors
  - It is possible to use canonical I/O functions
- Three main parameters
  - Communication style
  - Namespace
  - Protocol

Communication style

- How transmitted data are handled
- How they are addressed from the sender to the receiver
- Connection
  - It is granted that EVERY sent packet is received in the correct order
- Datagram
  - Packets can be lost or reordered because of the net
  - Every packet must be labeled with the address of the receiver

Namespace and protocol

- Namespace
  - Format of the addresses
    - Every address identifies one end of the socket
  - Local names = file names
  - Internet names = IP address + port
    - The port identifies the specific socket
- Protocol
  - The way data are transmitted
    - TCP/IP: most famous Internet protocol
    - AppleTalk
    - UNIX local communication protocol

Include and system calls

- `#include <sys/types.h>`
- `#include <sys/socket.h>`
- `socket` : to create a socket
- `close` : to destroy a socket
- `connect` : to connect two sockets
- `bind` : to assign an address to a server socket
- `listen` : to configure a socket in order to accept connections
- `accept` : to accept a connection; it creates a new socket for the communication
- `send` and `recv` to send and receive data

Creation and deletion

- Use `socket` for creation
  - Namespace
    - PF_LOCAL, PF_UNIX, PF_INET (PF = protocol families)
  - Communication style
    - SOCK_STREAM, SOCK_DGRAM
  - Protocol
    - Low level mechanism to transmit and to receive
    - Dependent from namespace-style couple
      - 0 is the best choice
  - Returns a file descriptor
- `close` to close a socket

Details on functions

- `bind`
  - Socket file descriptor
  - Pointer to a structure for the socket address
  - Structure length in bytes
- When an address is bound to a connection socket using `bind` it is necessary to invoke `listen` to declare it is a server
  - File descriptor and queue dimension
- `accept` to accept a connection
  - File descriptor and a pointer to a `sockaddr` structure filled with client data
  - Creates a new socket and returns the file descriptor

Connection

- The server
  - Creates a socket
  - Invokes `bind` to assign it an address
  - Invokes `listen` to allow connecting to the socket
  - Invokes `accept` to accept an incoming connection
- The client
  - Invokes `connect` with the address of the socket to which it wants to connect to

Local sockets

- Local namespace: PF_LOCAL or PF_UNIX
- Address format (`struct sockaddr_un`)
  - `sun_family` field set to AF_LOCAL
  - `sun_path` field to specify the file path to be used
    - Maximum length of 108 bytes
    - The process must have write permissions on the directory to be able to create new files
    - To connect to a socket a process must have read permissions on the file

Local sockets

- Use the SUN_LEN macro to calculate the length in bytes of the structure **`sockaddr`**
- Only for local processes
  - It is impossible to use on different hosts even if they share the same filesystem
- Invoke **`unlink`** on the file descriptor when socket are not used anymore

Internet domain sockets

- Internet namespace: PF_INET
- Address format (`struct sockaddr_in`)
  - `sin_family` field must be set to AF_INET
  - `sin_addr` to store the Interenet address as a 32 bit integer IP number
    - `gethostbyname` to convert IP addresses in dotted notation or names in 32 bit int
      - Returns a pointer to a `hostent` structure; the `h_addr` field contains the host IP number
  - `sin_port` to store the port number
    - To discriminate different sockets on the same host
    - `htons` function to convert the port number in network byte order

Socket pairs

- Pipes limited by the fact communication is unidirectional and only between related processes
- `socketpair` creates a connected socket couple
  - Bidirectional communication between related processes
  - Three initial parameters are the same as those used in `socket` (domain, style, protocol)
    - PF_LOCAL as domain
  - Additional parameter: integer array of dimension 2
    - Socket file descriptors
    - Similar to pipes