Unix Socket

unix network programming

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

Sockets are communication points on the same or different computers to exchange data. Sockets are supported by Unix, Windows, Mac, and many other operating systems.

The tutorial provides a strong foundation by covering basic topics such as network addresses, host names, architecture, ports and services before moving into network address functions and explaining how to write client/server codes using sockets.

Audience

This tutorial has been designed for everyone interested in learning the data exchange features of Unix Sockets.

Prerequisites

Learning Sockets is not at all a difficult task. We assume that you are well versed with the basic concepts of C programming.

Copyright & Disclaimer

Table of Contents

1. UNIX SOCKETS – WHAT IS A SOCKET?

Sockets allow communication between two different processes on the same or different machines. To be more precise, it's a way to talk to other computers using standard Unix file descriptors. In Unix, every I/O action is done by writing or reading a file descriptor. A file descriptor is just an integer associated with an open file and it can be a network connection, a text file, a terminal, or something else.

To a programmer, a socket looks and behaves much like a low-level file descriptor. This is because commands such as read() and write() work with sockets in the same way they do with files and pipes.

Sockets were first introduced in 2.1BSD and subsequently refined into their current form with 4.2BSD. The sockets feature is now available with most current UNIX system releases.

Where is Socket Used?

A Unix Socket is used in a client-server application framework. A server is a process that performs some functions on request from a client. Most of the application-level protocols like FTP, SMTP, and POP3 make use of sockets to establish connection between client and server and then for exchanging data.

Socket Types

There are four types of sockets available to the users. The first two are most commonly used and the last two are rarely used.

Processes are presumed to communicate only between sockets of the same type but there is no restriction that prevents communication between sockets of different types.

- **Stream Sockets**: Delivery in a networked environment is guaranteed. If you send through the stream socket three items "A, B, C", they will arrive in the same order - "A, B, C". These sockets use TCP (Transmission Control Protocol) for data transmission. If delivery is impossible, the sender receives an error indicator. Data records do not have any boundaries.

- **Datagram Sockets**: Delivery in a networked environment is not guaranteed. They're connectionless because you don't need to have an open connection as in Stream Sockets - you build a packet with the destination information and send it out. They use UDP (User Datagram Protocol).

- **Raw Sockets**: These provide users access to the underlying communication protocols, which support socket abstractions. These sockets are normally datagram oriented, though their exact characteristics are dependent on the interface provided by the protocol. Raw sockets are not intended for the general user; they have been provided mainly for those interested in developing new communication protocols, or for gaining access to some of the more cryptic facilities of an existing protocol.

- **Sequenced Packet Sockets**: They are similar to a stream socket, with the exception that record boundaries are preserved. This interface is provided only as a part of the Network Systems (NS) socket abstraction, and is very important in most serious NS applications. Sequenced-packet sockets allow the user to manipulate the Sequence Packet Protocol (SPP) or Internet Datagram Protocol (IDP) headers on a packet or a group of packets, either by writing a prototype header along with whatever data is to be sent, or by specifying a default header to be used with all outgoing data, and allows the user to receive the headers on incoming packets.

What is Next?

The next few chapters are meant to strengthen your basics and prepare a foundation before you can write Server and Client programs using *socket*. If you directly want to jump to see how to write a client and server program, then you can do so but it is not recommended. It is strongly recommended that you go step by step and complete these initial few chapters to make your base before moving on to do programming.

2. UNIX SOCKETS – NETWORK ADDRESSES

Before we proceed with the actual stuff, let us discuss a bit about the Network Addresses — the IP Address.

The IP host address, or more commonly just IP address, is used to identify hosts connected to the Internet. IP stands for Internet Protocol and refers to the Internet Layer of the overall network architecture of the Internet.

An IP address is a 32-bit quantity interpreted as 48-bit numbers or octets. Each IP address uniquely identifies the participating user network, the host on the network, and the class of the user network.

An IP address is usually written in a dotted-decimal notation of the form N1.N2.N3.N4, where each Ni is a decimal number between 0 and 255 decimal (00 through FF hexadecimal).

Address Classes

IP addresses are managed and created by the *Internet Assigned Numbers Authority* (IANA). There are five different address classes. You can determine which class an IP address is in by examining the first four bits of the IP address.

- **Class A** addresses begin with **0xxx**, or **1 to 126** decimal.
- **Class B** addresses begin with **10xx**, or **128 to 191** decimal.
- **Class C** addresses begin with **110x**, or **192 to 223** decimal.
- **Class D** addresses begin with **1110**, or **224 to 239** decimal.
- **Class E** addresses begin with **1111**, or **240 to 254** decimal.

Addresses beginning with **01111111**, or **127** decimal, are reserved for loopback and for internal testing on a local machine [You can test this: you should always be able to ping **127.0.0.1**, which points to yourself]; Class D addresses are reserved for multicasting; Class E addresses are reserved for future use. They should not be used for host addresses.

Example

Class	Leftmost bits	Start address	Finish address
A	0xxx	0.0.0.0	127.255.255.255
B	10xx	128.0.0.0	191.255.255.255

C	110x	192.0.0.0	223.255.255.255
D	1110	224.0.0.0	239.255.255.255
E	1111	240.0.0.0	255.255.255.255

Subnetting

Subnetting or Subnetworking basically means to branch off a network. It can be done for a variety of reasons like network in an organization, use of different physical media (such as Ethernet, FDDI, WAN, etc.), preservation of address space, and security. The most common reason is to control network traffic.

The basic idea in subnetting is to partition the host identifier portion of the IP address into two parts:

- A subnet address within the network address itself; and
- A host address on the subnet.

For example, a common Class B address format is N1.N2.S.H, where N1.N2 identifies the Class B network, the 8-bit S field identifies the subnet, and the 8-bit H field identifies the host on the subnet.

3. UNIX SOCKETS – NETWORK HOST NAMES

Host names in terms of numbers are difficult to remember and hence they are termed by ordinary names such as Takshila or Nalanda. We write software applications to find out the dotted IP address corresponding to a given name.

The process of finding out dotted IP address based on the given alphanumeric host name is known as **hostname resolution**.

A hostname resolution is done by special software residing on high-capacity systems. These systems are called Domain Name Systems (DNS), which keep the mapping of IP addresses and the corresponding ordinary names.

The /etc/hosts File

The correspondence between host names and IP addresses is maintained in a file called *hosts*. On most of the systems, this file is found in ***/etc*** directory.

Entries in this file look like the following:

```
# This represents a comments in /etc/hosts file.
127.0.0.1       localhost
192.217.44.207    nalanda metro
153.110.31.18     netserve
153.110.31.19     mainserver centeral
153.110.31.20     samsonite
64.202.167.10   ns3.secureserver.net
64.202.167.97   ns4.secureserver.net
66.249.89.104   www.google.com
68.178.157.132  services.amrood.com
```

Note that more than one name may be associated with a given IP address. This file is used while converting from IP address to host name and vice versa.

You would not have access to edit this file, so if you want to put any host name along with IP address, then you would need to have root permission.