HTTP Server in C

Jun 12, 2018

C, HTTP, Sockets

David Egan

This article describes a simple http server socket in Linux.

SERVER SOCKET

The basic procedure:

- Create socket with `socket()` call
- `bind()` this to an IP and port where it can
- `listen()` for connections, then
- `accept()` connection and `send()` or `receive()` data to/from connected sockets

Note that if `struct sockaddr_in serverAddress.sin_addr.s_addr` is set to INADDR_ANY the socket is bound to all local interfaces. INADDR_ANY is a constant set to zero, defined in `netinet/in.h` . This will correspond to an IP address of 0.0.0.0 in the standard IPv4 notation. Note that `htonl(INADDR_LOOPBACK)` and `inet_addr("127.0.0.1")` are functionally equivalent.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <netdb.h> // for getnameinfo()

// Usual socket headers
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#include <arpa/inet.h>

#define SIZE 1024
#define BACKLOG 10  // Passed to listen()

void report(struct sockaddr_in *serverAddress);

void setHttpHeader(char httpHeader[])
{
    // File object to return
    FILE *htmlData = fopen("index.html", "r");

    char line[100];
    char responseData[8000];
    while (fgets(line, 100, htmlData) != 0) {
        strcat(responseData, line);
    }
    // char httpHeader[8000] = "HTTP/1.1 200 OK\r\n\n";
    strcat(httpHeader, responseData);
}

int main(void)
{
    char httpHeader[8000] = "HTTP/1.1 200 OK\r\n\n";
```

```
// Socket setup: creates an endpoint for communication, re
// -------------------------------------------------------
int serverSocket = socket(
    AF_INET,      // Domain: specifies protocol family
    SOCK_STREAM,  // Type: specifies communication semanti
    0             // Protocol: 0 because there is a single
);

// Construct local address structure
// -------------------------------------------------------
struct sockaddr_in serverAddress;
serverAddress.sin_family = AF_INET;
serverAddress.sin_port = htons(8001);
serverAddress.sin_addr.s_addr = htonl(INADDR_LOOPBACK);//i

// Bind socket to local address
// -------------------------------------------------------
// bind() assigns the address specified by serverAddress t
// referred to by the file descriptor serverSocket.
bind(
    serverSocket,                         // file descript
    (struct sockaddr *) &serverAddress,   // Address to be
    sizeof(serverAddress)                 // Size (bytes)
);

// Mark socket to listen for incoming connections
// -------------------------------------------------------
int listening = listen(serverSocket, BACKLOG);
if (listening < 0) {
    printf("Error: The server is not listening.\n");
    return 1;
}
report(&serverAddress);     // Custom report function
setHttpHeader(httpHeader);  // Custom function to set head
int clientSocket;
```

```
    // Wait for a connection, create a connected socket if a c
    // ------------------------------------------------------
    while(1) {
        clientSocket = accept(serverSocket, NULL, NULL);
        send(clientSocket, httpHeader, sizeof(httpHeader), 0);
        close(clientSocket);
    }
    return 0;
}

void report(struct sockaddr_in *serverAddress)
{
    char hostBuffer[INET6_ADDRSTRLEN];
    char serviceBuffer[NI_MAXSERV]; // defined in `<netdb.h>`
    socklen_t addr_len = sizeof(*serverAddress);
    int err = getnameinfo(
        (struct sockaddr *) serverAddress,
        addr_len,
        hostBuffer,
        sizeof(hostBuffer),
        serviceBuffer,
        sizeof(serviceBuffer),
        NI_NUMERICHOST
    );
    if (err != 0) {
        printf("It's not working!!\n");
    }
    printf("\n\n\tServer listening on http://%s:%s\n", hostBuf
}
```

PROTOCOL FAMILY

When setting up a socket using `socket()` the following protocols are available:

Name	Purpose	Man page
AF_UNIX , AF_LOCAL	Local communication	man unix
AF_INET	IPv4 Internet protocols	man ip
AF_INET6	IPv6 Internet protocols	man ipv6
AF_IPX	IPX - Novell protocols	
AF_NETLINK	Kernel user interface device	man netlink
AF_X25	ITU-T X.25 / ISO-8208 protocol	man x25
AF_AX25	Amateur radio AX.25 protocol	
AF_ATMPVC	Access to raw ATM PVCs	
AF_APPLETALK	AppleTalk	man ddp
AF_PACKET	Low level packet interface	man packet
AF_ALG	Interface to kernel crypto API	

You can acces this list by running `man socket` .

SOCKET TYPE

The socket has the indicated type, which specifies the communication semantics. Currently defined types are:

Type	Description
SOCK_STREAM	Provides sequenced, reliable, two-way, connection-based byte streams. An out-of-band data transmission mechanism may be supported. Used for TCP protocol.
SOCK_DGRAM	Supports datagrams - connectionless, unreliable messages of a fixed maximum length. Used for UDP protocol.
SOCK_SEQPACKET	Provides a sequenced, reliable, two-way connection-based data transmission path for datagrams of fixed maximum length; a consumer is required to read an entire packet with each input system call.
SOCK_RAW	Provides raw network protocol access.
SOCK_RDM	Provides a reliable datagram layer that does not guarantee ordering.
SOCK_PACKET	Obsolete and should not be used in new programs; see packet(7).

CONSTRUCTING A LOCAL ADDRESS STRUCTURE

If `serverAddress.sin_addr.s_addr` is set to `INADDR_ANY` the socket is bound to all local interfaces. `INADDR_ANY` is a constant set to zero, defined in `netinet/in.h` . This will correspond to an IP address of 0.0.0.0 in the standard IPv4 notation. Note that `htonl(INADDR_LOOPBACK)` and `inet_addr("127.0.0.1")` are functionally equivalent.

BIND SOCKET TO LOCAL ADDRESS

When a socket is created with socket(), it exists in a name space (address family) but has no address assigned to it. bind() assigns the address specified by serverAddress to the socket referred to by the file descriptor serverSocket. serverAddressLength specifies the size, in bytes, of the address structure pointed to by serverAddress.

This operation is known as “assigning a name to a socket”.

CONNECTING

The programme runs an infinite loop in which we wait and create a connected socket if a connection is pending.

The `accept()` function gets the first connection request on the queue of pending connections for the listening socket (in this case denoted by `serverSocket`). It then creates a new connected socket and returns a file descriptor referring to this socket. The newly created socket is NOT in a listening state. The original socket (`serverSocket`) is unaffected by this call.

USING GETNAMEINFO

Used in the `report()` function

```
int getnameinfo(
    const struct sockaddr *sa,  // pointer to a generic socket
    socklen_t salen,            // size of the above
    char *host,                 // Caller allocated buffer, wi
    socklen_t hostlen,          // Size of above
    char *serv,                 // Caller allocate buffer, wil
    socklen_t servlen,          // Size of above
    int flags);
```

REFERENCES

- getaddrinfo/getnameinfo Wikipedia