

# Networking, Part 3: Building a simple TCP Client

jakebailey edited this page on Dec 16, 2014 · 3 revisions

## Complete Simple TCP Client Example

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netdb.h>
#include <unistd.h>

int main(int argc, char **argv)
{
    int s;
    int sock_fd = socket(AF_INET, SOCK_STREAM, 0);

    struct addrinfo hints, *result;
    memset(&hints, 0, sizeof(struct addrinfo));
    hints.ai_family = AF_INET; /* IPv4 only */
    hints.ai_socktype = SOCK_STREAM; /* TCP */

    s = getaddrinfo("www.illinois.edu", "80", &hints, &result);
    if (s != 0) {
        fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(s));
        exit(1);
    }

    connect(sock_fd, result->ai_addr, result->ai_addrlen);

    char *buffer = "GET / HTTP/1.0\r\n\r\n";
    printf("SENDING: %s", buffer);
    printf("===\n");
    write(sock_fd, buffer, strlen(buffer));

    char resp[1000];
    int len = read(sock_fd, resp, 999);
    resp[len] = '\0';
    printf("%s\n", resp);

    return 0;
}
```

Example output:

```
SENDING: GET / HTTP/1.0

===
HTTP/1.1 200 OK
```

Edit

New Page

▼ Pages 51

Home

#Example Markdown

#Informal Glossary

#Piazza: When And How to Ask For Help

C Programming, Part 1: Introduction

C Programming, Part 2: Text Input And Output

C Programming, Part 3: Common Gotchas

C Programming, Part 4: Debugging

Deadlock, Part 1: Resource Allocation Graph

Deadlock, Part 2: Deadlock Conditions

File System, Part 1: Introduction

File System, Part 2: Files are inodes (everything else is just data...)


File System, Part 3: Permissions

File System, Part 4: Working with directories

File System, Part 5: Virtual file systems

Show 36 more pages...

Clone this wiki locally

 Clone in Desktop

Date: Mon, 27 Oct 2014 19:19:05 GMT  
Server: Apache/2.2.15 (Red Hat) mod\_ssl/2.2.15 OpenSSL/1.0.1e-fips mod\_jk/1.2.32  
Last-Modified: Fri, 03 Feb 2012 16:51:10 GMT  
ETag: "401b0-49-4b8121ea69b80"  
Accept-Ranges: bytes  
Content-Length: 73  
Connection: close  
Content-Type: text/html

Provided by Web Services at Public Affairs at the University of Illinois

# Comment on HTTP request and response

The example above demonstrates a request to the server using Hypertext Transfer Protocol. A web page (or other resources) are requested using the following request:

```
GET / HTTP/1.0
```

There are four parts (the method e.g. GET,POST,...); the resource (e.g. /index.html /image.png); the proctocol "HTTP/1.0" and two new lines (\r\n\r\n)

The server's first response line describes the HTTP version used and whether the request is successful using a 3 digit response code:

```
HTTP/1.1 200 OK
```

If the client had requested a non existing file, e.g. `GET /nosuchfile.html HTTP/1.0` Then the first line includes the response code is the well-known `404` response code:

```
HTTP/1.1 404 Not Found
```

Legal and Licensing information: Unless otherwise specified, submitted content to the wiki must be original work (including text, java code, and media) and you provide this material under a [Creative Commons License](#). If you are not the copyright holder, please give proper attribution and credit to existing content and ensure that you have license to include the materials.

