Ex. No. 7	**HTTP Web Client**	**Karthik D**
13 September 2021	**UCS1511 - Networks Lab**	**195001047**

Aim

To implement a HTTP web client program using C socket program with the help of a TCP-based client

Question

Implement a HTTP web client program to download the webpage using C socket programming as follows:

- Read the name of the serveras command line argument
- Get the address of the server using gethostbyname() that returns the pointer to network data structure for given host
- Create a TCP socket using socket()
- Connect to remote server 5.Send request using a GET /path/filename HTTP/1.1\r\n request using either send() or write().
- Receive the response using either recv() or read()
- Parse the response to find out if the request succeeded and what format the file data is being sent as
- Receive the file data, if present, using either recv() or read()and write the downloaded page intofileundera different name in a local folder
- Close the socket and the file

Algorithm

(a) Client Script

Step 1: Create a network socket with parameters suitable for an end-point of TCP based communication

Step 2: Parse the command line arguments to obtain the resource URL

Step 3: Separate the domain name and the path to the resource from the URL

Step 4: Using the network system call - getaddrinfo() and the domain name of the server, retrieve its address. Use port number 80 for HTTP connection and filter the results to contain only TCP connections

Step 5: Connect the client socket to the server using the connect() call, specifying the server address

Step 6: Prepare a GET request header structure as follows:

```
GET <resource_path> HTTP/1.1
Host: <domain_address>
Connection: close
```

Step 7: Forward the header as a request to the server using the TCP client socket with a write() call

Step 8: Block and wait using the read() network call to receive a response from the server

Step 9: Parse the response to ensure that the HTTP status was OK (response code: 200). Also ensure that the resource sent in the response body is a PDF (for this test case)

Step 10: Open a file on the local system, write the response body into the file and close the file using the open(), write() and close() system calls for file handling

Step 11: Close the created sockets using the *close()* system call and terminate the process

C Program Code

1. tcp_socket.h - TCP connection helper functions

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

#ifndef tcp_socket_h
   #include "tcp_socket.h"
#endif

#ifndef http_utils_h
   #include "http_utils.h"
#endif
```

```c
int get_default_fileperm(){
   mode_t curr_umask = umask(022);
   umask(curr_umask);
   return 0666-curr_umask;
}


int main(int argc, char **argv) {
   if(argc<2){
       printf("\nSupply web address\n");
       exit(1);
   }
   // Partition web
   char *web_address = *(argv+1);
   char *hostname = get_hostname(web_address);
   char *resource_path = get_resource_path(web_address);

   // Collect server socket details
   AddrInfo *socket_details;
   // socket-filter
   AddrInfo socket_filter;
   memset(&socket_filter, 0, sizeof(socket_details));
   socket_filter.ai_family = AF_UNSPEC;
   socket_filter.ai_socktype = SOCK_STREAM;
   getaddrinfo("www.africau.edu", HTTP_PORT_STRING, &socket_filter,
&socket_details);

   char *data_buffer = (char*)malloc(sizeof(char)*BUFFER_SIZE);
   int data_size;
   int response;

   // Working Case:
www.w3.org/WAI/ER/tests/xhtml/testfiles/restores/pdf/dummy.pdf
   // Working Case: www.africau.edu/images/default/sample.pdf

   int client_socket = make_socket(socket_details);
   printf("\nConnecting to server...\n");
   response = connect_server(client_socket, socket_details);
   if(response<0){
       printf("\nCould not connect to server. Retry!");
```

```
    }

    // Prepare HTTP header
    char *header = prepare_get_header(hostname, resource_path);
    // Send Request
    send(client_socket, header, strlen(header), 0);
    printf("\nHTTP Request Sent\n");
    printf("\nRequest Header\n%s", header);

    // Get response
    // Store the response into a temporary file to parser for status
    char *temp_filename = "temp";
    int store_fd = open(temp_filename, O_WRONLY | O_CREAT,
get_default_fileperm());
    if(store_fd == -1){
        return -4;          // File not Readable
    }
    do{
        data_size = recv(client_socket, data_buffer, sizeof(data_buffer),
0);
        data_size = write(store_fd, data_buffer, data_size);
    }while(data_size!=0);
    close(store_fd);

    check_response_status(temp_filename);

    /*
    char *store_as = (char*)malloc(sizeof(char)*BUFFER_SIZE);
    printf("\nName your download: ");
    scanf(" %s", store_as);
    sprintf(store_as, "%s.pdf", store_as);

    // Close file and sockets
    if(!close(store_fd)){
        printf("\nPDF downloaded as '%s'\n", store_as);
    }
    else{
        printf("\nError when closing\n");     // Unexpected Error Occurred
    }
    */
```

```
   destroy_socket(client_socket);
   return 0;
}
```

2. http_utils.h - Utility functions for HTTP requests management

```

#ifndef http_utils_h
#define http_utils_h

#include<stdlib.h>
#include<string.h>

#include "tcp_socket.h"

#define HTTP_HEADERLINE_SIZE 500
#define HTTP_PORT_STRING "80"
#define CONTENT_TYPE_ATTR_NAME "Content-Type"
#define CONTENT_PDF_VALUE "application/pdf"
#define PDF_START_SYMBOL "%PDF"


// Read a file line-by-line
int read_line(int fd, char** buffer){
   char reader[2];
   int line_size = 0;
   char *line_buf = (char*)malloc(sizeof(char)*HTTP_HEADERLINE_SIZE);

   int bytes_read = read(fd, reader, 1);
   while(bytes_read!=0 && *(reader)!=10){
       // Until EOF or newline
       *(line_buf+line_size) = *(reader);
       bytes_read = read(fd, reader, 1);
       line_size++;
   }
   *(buffer) = line_buf;
   return line_size;
}

short startswith(char *prefix, char *string){
```

```c
   return strncmp(prefix, string, strlen(prefix)) == 0;
}

char* get_hostname(char *web_address){
   char *hostname = (char*)malloc(sizeof(char)*BUFFER_SIZE);
   char *remain = (char*)malloc(sizeof(char)*BUFFER_SIZE);
   sscanf(web_address, "%[^/]/%s", hostname, remain);
   return hostname;
}

char* get_resource_path(char *web_address){
   char *remain = (char*)malloc(sizeof(char)*BUFFER_SIZE);
   char *path = (char*)malloc(sizeof(char)*BUFFER_SIZE);
   sscanf(web_address, "%[^/]%s", remain, path);
   return path;
}

char* prepare_get_header(char* hostname, char* resource_path){
   char *header = (char*)malloc(sizeof(char)*BUFFER_SIZE*2);
   sprintf(header, "GET %s HTTP/1.1\r\nHost: %s\r\nConnection:
close\r\n\r\n", resource_path, hostname);
   return header;
}

short check_response_status(char *response_file, short disp_header){
   // Currently ensures successful PDF response
   // File readers
   int read_fd = open(response_file, O_RDONLY);
   char *line = (char*)malloc(sizeof(char)*HTTP_HEADERLINE_SIZE);
   // Parameter readers
   int status_code = -1;
   float version = -1;
   short type_read = 0;
   short body_start = 0;
   char *type = (char*)malloc(sizeof(char)*HTTP_HEADERLINE_SIZE);
   //type = NULL;
   int line_size = read_line(read_fd, &line);
   while(line_size!=0){
       fflush(stdout);
       if(version==-1){
           sscanf(line, "HTTP/%f %d", &version, &status_code);
       }
       else if(!type_read && startswith(CONTENT_TYPE_ATTR_NAME, line)){
           sscanf(line, "Content-Type: %s", type);
           type_read = 1;
       }
       else if(disp_header && !body_start){
           // Store the response head
           if(startswith(PDF_START_SYMBOL, line)){
               body_start = 1;
           }
           else{
               printf("\n%s", line);
           }
       }
       line_size = read_line(read_fd, &line);
   }
   close(read_fd);
   // Return validation result
   if(status_code==200 && strcmp(CONTENT_PDF_VALUE, type)==0){
       // Verified. PDF
       return 1;
   }
   return 0;
}

/*
HTTP/1.1 200 OK
Date: Sat, 11 Sep 2021 02:15:54 GMT
Content-Type: application/pdf
Content-Length: 3028
Connection: close
Last-Modified: Fri, 24 Feb 2017 17:42:38 GMT
ETag: "58b0708e-bd4"
Expires: Sun, 24 Apr 2022 02:07:03 GMT
Cache-Control: max-age=31536000
Host-Header: 8441280b0c35cbc1147f8ba998a563a7
X-Proxy-Cache-Info: DT:1
CF-Cache-Status: HIT
Age: 12096531
```

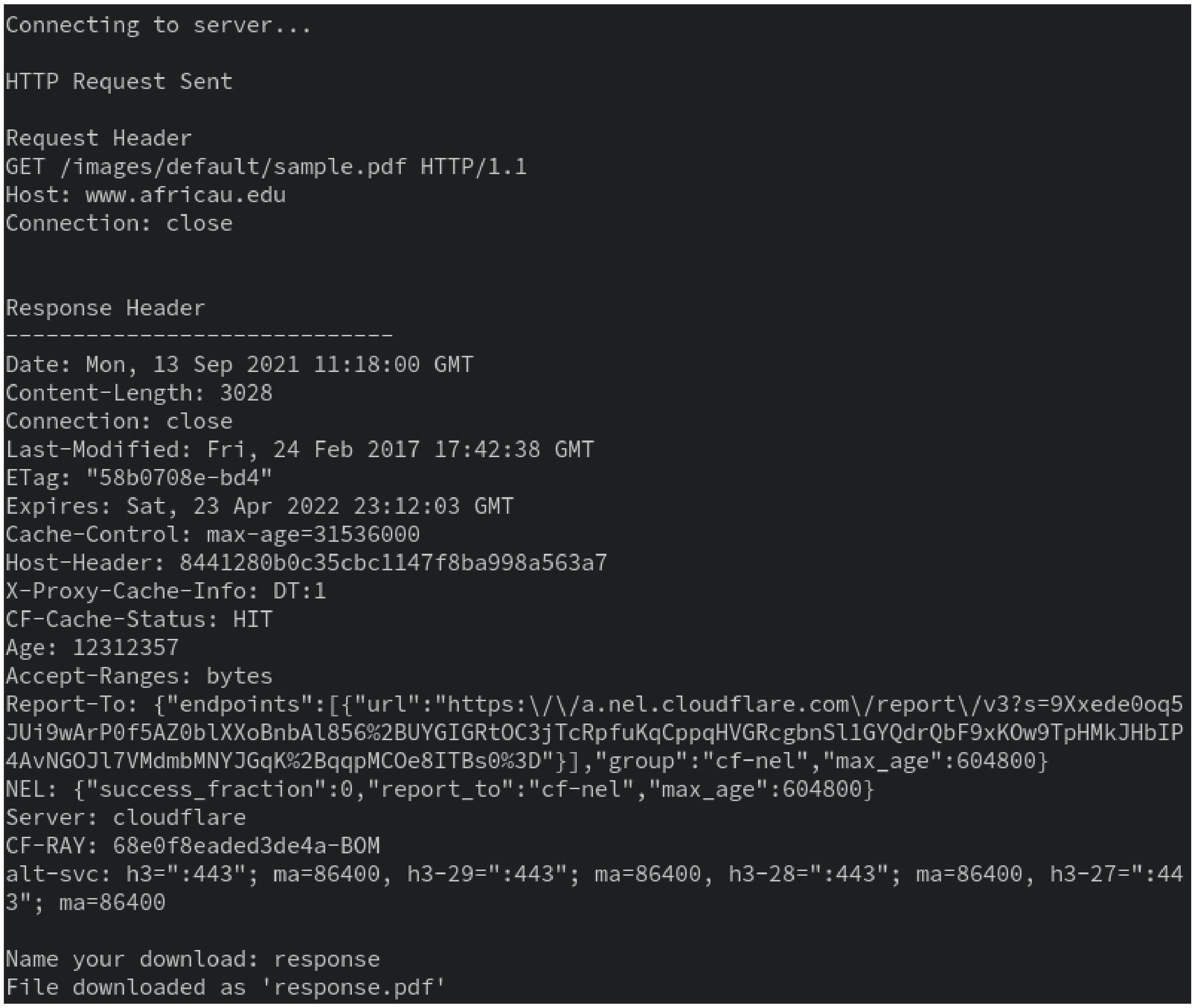
```
Accept-Ranges: bytes
Report-To:
{"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v3?s=QPStsF2
65LuROHuiaXlOcuf1V4hCQzjbSswac1pvFrW%2Bjks9%2BV7cfeocSAxy%2BUz2mLgeFzQCcHg
mt%2F49PIbg20yhJ2%2BvHhFfM%2FIHGZya3wUa7PiLAfboQDqWcL048FXwaOY%3D"}],"grou
p":"cf-nel","max_age":604800}
NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 68cd64131e434b1c-HYD
alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400, h3-28=":443";
ma=86400, h3-27=":443"; ma=86400
*/

#endif
```

3. main.c - Drive client script

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

#ifndef tcp_socket_h
   #include "tcp_socket.h"
#endif

#ifndef http_utils_h
   #include "http_utils.h"
#endif


int get_default_fileperm(){
   mode_t curr_umask = umask(022);
   umask(curr_umask);
```

```
    return 0666-curr_umask;
}


int main(int argc, char **argv) {
    if(argc<2){
        printf("\nSupply web address\n");
        exit(1);
    }
    // Partition web
    char *web_address = *(argv+1);
    char *hostname = get_hostname(web_address);
    char *resource_path = get_resource_path(web_address);

    // Collect server socket details
    AddrInfo *socket_details;
    // socket-filter
    AddrInfo socket_filter;
    memset(&socket_filter, 0, sizeof(socket_details));
    socket_filter.ai_family = AF_UNSPEC;
    socket_filter.ai_socktype = SOCK_STREAM;
    getaddrinfo("www.africau.edu", HTTP_PORT_STRING, &socket_filter,
&socket_details);

    char *data_buffer = (char*)malloc(sizeof(char)*BUFFER_SIZE);
    int data_size;
    int response;

    // Working Case:
www.w3.org/WAI/ER/tests/xhtml/testfiles/restores/pdf/dummy.pdf
    // Working Case: www.africau.edu/images/default/sample.pdf

    int client_socket = make_socket(socket_details);
    printf("\nConnecting to server...\n");
    response = connect_server(client_socket, socket_details);
    if(response<0){
        printf("\nCould not connect to server. Retry!");
    }

    // Prepare HTTP header
    char *header = prepare_get_header(hostname, resource_path);
    // Send Request
    send(client_socket, header, strlen(header), 0);
    printf("\nHTTP Request Sent\n");
    printf("\nRequest Header\n%s", header);

    // Get response
    // Store the response into a temporary file to parser for status
    char *temp_filename = "temp";
    int store_fd = open(temp_filename, O_WRONLY | O_CREAT,
get_default_fileperm());
    if(store_fd == -1){
        return -4;          // File not Readable
    }
    do{
        data_size = recv(client_socket, data_buffer, sizeof(data_buffer),
0);
        data_size = write(store_fd, data_buffer, data_size);
    }while(data_size!=0);
    close(store_fd);

    printf("\nResponse Header");
    printf("\n-----------------------------");
    check_response_status(temp_filename, 1);

    char *store_as = (char*)malloc(sizeof(char)*BUFFER_SIZE);
    printf("\nName your download: ");
    scanf(" %s", store_as);
    sprintf(store_as, "%s.pdf", store_as);

    // Rename the temporary save as the required file
    rename(temp_filename, store_as);
    printf("File downloaded as '%s'\n", store_as);
    // Close file and sockets
    destroy_socket(client_socket);
    return 0;
}
```

<u>Sample Output</u>



```
Connecting to server...

HTTP Request Sent

Request Header
GET /images/default/sample.pdf HTTP/1.1
Host: www.africau.edu
Connection: close


Response Header
-----------------------------
Date: Mon, 13 Sep 2021 11:18:00 GMT
Content-Length: 3028
Connection: close
Last-Modified: Fri, 24 Feb 2017 17:42:38 GMT
ETag: "58b0708e-bd4"
Expires: Sat, 23 Apr 2022 23:12:03 GMT
Cache-Control: max-age=31536000
Host-Header: 8441280b0c35cbc1147f8ba998a563a7
X-Proxy-Cache-Info: DT:1
CF-Cache-Status: HIT
Age: 12312357
Accept-Ranges: bytes
Report-To: {"endpoints":[{"url":"https:\/\/a.nel.cloudflare.com\/report\/v3?s=9Xxede0oq5
JUi9wArP0f5AZ0blXXoBnbAl856%2BUYGIGRtOC3jTcRpfuKqCppqHVGRcgbnSl1GYQdrQbF9xKOw9TpHMkJHbIP
4AvNGOJl7VMdmbMNYJGqK%2BqqpMCOe8ITBs0%3D"}],"group":"cf-nel","max_age":604800}
NEL: {"success_fraction":0,"report_to":"cf-nel","max_age":604800}
Server: cloudflare
CF-RAY: 68e0f8eaded3de4a-BOM
alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400, h3-28=":443"; ma=86400, h3-27=":44
3"; ma=86400

Name your download: response
File downloaded as 'response.pdf'
```

<u>Result</u>

Implemented a socket program in C language using TCP client to download web content from a HTTP server. Through this implementation, the following aspects were understood:

1. Working procedure of HTTP request-response process
2. Client based TCP connection procedures to a web server
3. Preparing and parsing HTTP headers