new sock → accept ⑤

אם Connect חזר ונוכל במספר accept יחזיר בהצלחה

אחרי שמוצאת לבקשה welcome socket

בדרך כלל אחרי accept צריך גם לבדוק אם יש בקשות

⇐ אחרי accept ⇐ thread פיצול ל clinet מסוים

וגם ימשיך ה main thread לקבל בקשות חדשות.

accept(sd, addr, addr_len)

Null, Null ← אם לא מעניין
לשמור

unique clinet או [illegible] מסוים ⇐ שולחת ריקים

IP, Port חוזרים במקום כי ריקים בפונקציה..

sent = write(sd, buffer, buffer_len) ⑥

שגיאה → -1

כמה → num_write ; אם sent קטן מ

כמות ביטים →

```
total = 0                       buff → char*
while(1)                 עבור TCP.
            sd , buf+total        - total
    Sent = write( ... , num_write)
    total+= Sent
    if(total == num-write)
        break;
```

UDP → sendto

nbytes = read (sd, buffer, len)

כמו write גם לא ניתנים.

if (nread = = 0) ← connection סגר

break;

בכל פעם

strstr / strtok . request ה נגמר → \r\n פעמיים

if (strcmp (buf, "\r\n\r\n"))

break;

אין צורך לקרוא הכל ה mega לשלוח

⇐ קוראים ושולחים (מוצגים למסך).

char* buff ✗ לא טוב

unsigned char* buff ✓ טוב

לא string ← בינרי

אם ה response : header

⇒ char*

אחרי \r\n\r\n הבא לא char*

240 בינרי אחרי כן ⇐ לא char*

[illegible] ← מוצגים למסך

כאשר client

↖

אין צורך

[illegible]

if(nread == 0)
break

סיכום :

Socket
↓
bind
↓
listen
↓
accept
⋮

client

Socket
↓
connect
↓
⋮

יצירת סוקט :

Socket I/O

① לקרוא ל connect ומלא struct &

אופציה ①

srv

IP & ווצגים אותו
הפוך

inet - addr(   )

↕

sin_addr

inet - ntoa (srv)

אופציה ②: יש גם לשגר

DNS ' קורא

get host by name

ממיקם
הפ(ו)לי:

perror

→ herror

srv_sin_port = hton(80)

אתחול של struct בניגוד של השרת.

conect
- אחרי מילוי הכתובת → DNS או נקוד הרשמינה.
- גודל ה Socket מסוג עניר sockaddr.

⇐ תקשורת מול השרת.

TCP: write ⑥ socket + connet socket יצירת שקע ①

sent ↗ read ⑦ bind ②

③

EX2 – HTTP client close(sock-fd) ⑧ listen connect

[illegible] Goals: [shutdown או] כמה. → listen(sd, qlen) ④

Seen

The purpose of this project is two-fold: size הגבלת מקסימלי → מיון לגלל גוקה ⑤

1. give students hands-on experience with socket programming
2. help students better understand application-level protocols by implementing a well-known protocol, HTTP.

accept

In this programming assignment, you will write an HTTP client. Students are not required to implement the full HTTP specification, but only a very limited subset of it.

פונקציה חוסמת

You will implement the following:

An HTTP client that constructs an HTTP request based on the user's command line input, sends the request to a Web server, receives the reply from the server, and displays the reply message on the screen. You should support only IPv4 connections.

[illegible] לפרק את הבקשה http client *
לקבל תשובה ← הדפסה

Background:

What is HTTP? HTTP stands for Hyper Text Transfer Protocol and is used for communication among web clients and servers. HTTP has a simple stateless client/server paradigm. The web client initiates a conversation by opening a connection to the server. Once a connection is set up, the client sends an HTTP request to the server. Upon receiving the HTTP request from the client, the server sends an HTTP response back to the client. An HTTP request consists of two parts: a header and a body. In this project, the basic HTTP request from a client doesn't contain a body. The first line of any request header should be:

Method Request-URI Version. An example HTTP1.1 request is:

לקבל את קשר
ונבנה את הבקשה

GET /index.html HTTP/1.1

Host: www.jce.ac.il

3 \r\n\r\n

The request header and body are separated by two sets of carriage return and line feed (\r\n). Since we do not need the body, the end of a header marks the end of a request. Using a C char string, the example request above should be: "GET /index.html HTTP/1.1\r\nHost: www.jce.ac.il\r\n\r\n". [illegible]

Get קובץ HTTP / גרסה

What is a URL?

Uniform Resource Locators (URLs) are formatted strings that identify resources in the web: documents, images, downloadable files, electronic mailboxes, etc. It generally has the format: Protocol://Host[:port]/Filepath.

Default = 80

אופציונלי

לפרק מחרוזת String

לצבי בכיתה:

הסבר:

In this project, when a port is not specified, the default HTTP port number of 80 is used. For example, a file called "foo.html" on HTTP server "www.yoyo.com" in directory "/pub/files" corresponds to this URL: http://www.yoyo.com/pub/files/foo.html. The default HTTP network port is 80; if an HTTP server resides on a different network port (say, port 1234), then the URL becomes http://www.yoyo.com:1234/pub/files/foo.html.

Program Description and What You Need to Do:

You will write the program client.c.

The Client

לקוח [illegible] [illegible]

post ⇐ body [illegible] -p

get ⇐ text argument [illegible] n [illegible] -r

The client takes two options "-p" and "-r" and a required argument <URL>.

Command line usage: client [–p n <text>] [–r n <pr1=value1 pr2=value2 …>] <URL>. The flags and the URL can come in any order, the only limitation is that the parameters should come right after the flag –r and the text should come right after the flag –p.

n=3 → Pr3=value 3

<URL> specifies the URL of the object that the client is requesting from the server. The URL

format is http://hostname[:port]/filepath.

The default request is GET request. Option "-p" along with its argument <text> specifies that this is a POST request. You should use POST method in your HTTP request instead of GET, you should also add a Content-length header and request body with the text when "-p" is specified in the command line. **The text after the -p flag can contain space and special characters but not new-line.**

Option "-r" along with its argument <n pr1=value1 pr2=value2 …> specify that the request has n parameters, and each of the parameters format is ‘name’=’value’ separated by space. The parameters should appear after the path, for example, if there are 2 parameters, the format will be: /path?pr1=value1&pr2=value2

You can assume that the URL has to start with http://

If the URL has no path, the path in the request is “/”.

In client.c, you need to:

Parsing in Command line

1. Parse the <URL> given in the command line. If there is a port, you should verify that it is a positive number under 2^16.
   The parsing is the easy part, don’t spend time on it.
   A suggested logic:

[illegible]

   a. If you see ‘-‘, then look for either p or r, if you don’t see any of them, print the Usage message and exit.

argv בגלל רווחים

      i. If you see -r, look for a number n (there migh~~t be more than one~~ space), if there is no number, print the Usage message and exit. After the number n, you should look for str=str, there can be spaces (in this case, spaces are not allowed within one parameter, only between parameters), after reading n arguments, go back to stage a. There can be 0 arguments when n=0.

[illegible]

הדגמה:

-r 3 P1=val1 P2=val2 ...

לחפש str=str 3 פעמים

אם יש פחות
שגיאה

אם יש יותר
לקחת
n הראשונים
ולהתעלם מהשאר.

מילה בגודל 12 טקסט

-p 12 abc . . - - ^

text

ii. If you see -p, look for a number n (there might be more than one space), if there is no number, print the Usage message and exit. After the number n, you should look for text, read n characters, and go back to stage a. if there are no n characters, print the Usage message and exit.

b. If there is no ‘-‘, then this is your URL. אם אין http:// שגיאה.

i. Check that it begins with http://, otherwise, print the Usage message and exit.

Port

ii. Read the domain name until you see either ‘:’, ‘/’ or end-of-string.

אם לא מתחיל

ב
-
חייב להיות מספר

1. If you see :, look for a positive number, which is less than 2^16, if not print the Usage message and exit. Port
2. If you see ‘/’, look for a path (can be also without a path).

Path

2. Construct an HTTP request based on the options specified in the command line
3. Connect to the server
4. Send the HTTP request to the server
5. Receive an HTTP response
6. Display the response on the screen.

After constructing the http request and before you send it to the server, print it to stdout in

the following format:

printf("HTTP request =\n%s\nLEN = %d\n", request, strlen(request)); ⟵

where request holds your constructed request.

After getting the response from the server and printing it to stdout, print the following

message:

printf("\n   Total received response bytes: %d\n",size);

where size is the number of characters in the response.

Your client should close connection after getting the file. You should use HTTP/1.1

Error handling:

1. In any case of a failure in one of the system calls, use perror(<sys_call>) and exit the program (for errors on gethostbyname call herror instead).
2. In any case of wrong command usage, print "Usage: client [-p n <text>] [-r n < pr1=value1 pr2=value2 …>] <URL>"

Enter a new line after each error message.

Examples:

1. ./client http://www.ptsv2.com/t/ex2
   Request:
   GET /t/ex2 HTTP/1.1
   Host: www.ptsv2.com

2. ./client -r 3 addr=jecrusalem tel=02-6655443 age=23 http://www.ptsv2.com/t/ex2
   Request:
   GET /t/ex2?addr=jecrusalem&tel=02-6655443&age=23 HTTP/1.1
   Host: www.ptsv2.com

3. ./client -p blabla http://www.ptsv2.com/t/ex2
   Request:
   POST /t/ex2 HTTP/1.1
   Host: www.ptsv2.com
   Content-length:6

   blala
4. ./client -p blabla -r 3 addr=jecrusalem tel=02-6655443 age=23 http://www.ptsv2.com/t/ex2

   Request:

   POST /t/ex2?addr=jecrusalem&tel=02-6655443&age=23 HTTP/1.1

   Host: www.ptsv2.com

   Content-length:6

   blala

5. ./client -r 3 addr=jecrusalem tel=02-6655443 age=23 -p blabla http://www.ptsv2.com/t/ex2
   The request is same as before.

6. Command line errors example:
   a. ./client -p -r 3 addr=jecrusalem tel=02-6655443 age=23 blabla http://www.ptsv2.com/t/ex2
      There is no number after -p.
   b. ./client -p 6 blabla -r addr=jecrusalem tel=02-6655443 age=23 http://www.ptsv2.com/t/ex2
      Has to be number after -r
   c. ./client -p 6 blabla -r 3 addr=jerusalem tel=02-6655443 age23 http://www.ptsv2.com/t/ex2
      The third parameter age23 is not of the right format: name=value
   d. ./client -p 6 blabla -r 3 addr=jerusalem tel=02-6655443 http://www.ptsv2.com/t/ex2

Either the url will be considered as the 3rd parameter and it is not of the right format or too few parameters, depends on the order you check the command line.

e. ./client -p 6 blabla -r 2 addr=jecrusalem tel=02-6655443 age=23 http://www.ptsv2.com/t/ex2
Too many parameters

f. ./client http://blala
This is not a usage error, you will fail when trying to get the IP address for that host.

Useful function:

Strchr, Strstr, strcat

Compile the client:

gcc -Wall –o client client.c

client is the executable file.

What to submit:

You should submit a tar file with client.c and README. Find README instructions in the course web-site.

Test the client:

You can use the client to connect to any HTTP server. You should try different URLs and options to make sure that the client works correctly.

In order to test parameters and post request, you can use the server at www.ptsv2.com, first go to that server, type your unique string in the 'lookup' box and press lookup. You'll redirected to a page www.ptsv2.com/t/your-string. This page is now your server and it shows the requests it gets. In order to test your requests, send them to www.ptsv2.com/t/your-string/post and you'll be able to see the details of the requests in the www.ptsv2.com/t/your-string page.