CIT595: Computer Systems II

Final assignment: MiniRSA

Author: Yuhui Xie

Last revised: 04/16/2012

Including files:

MiniRSA java package

Approach and structures:

1. in the main function of srv.c, make a server listener, and let the srvfd be a global variable. All clients will connect to this srvfd.

2. when a new client request comes in, let the server listens for the client: connfd = listenFor(srvfd).

3. create thread and pass the connfd to the thread function.

4. whenever a new client request comes in, a new thread will be created. The threads work parallel and no data shared except the srvfd.

5. read the client buffer, n = read(connfd, buffer, sizeof(buffer));

6. in the thread function, parse the client request string to get the host and uri.

7. create a file using the parsed url as name. e.g., if the uri is "www.google.com/12345", the filename will be "www.google.com-12345-".

8a. if the file doesn't exist, which means client hasn't visit this website before. My server request is:

GET http://www.google.com HTTP/1.1

Host: www.google.com

Connection: close

8b. if the file exist and the client didn't send the "If-Modified-Since: " header, add the If-Modified-Since header to my server request by using the current system time:

GET http://www.google.com HTTP/1.1

Host: www.google.com

If-Modified-Since: Mon, 09 Apr 2012 06:24:59 GMT (current system time)

Connection: close

8c. the file exist and client sent the "If-Modified-Since: " header, if the client request time is earlier than the corresponding cache file modified time, my server will send back a "HTTP/1.1 304 Not Modified" response and exit the thread.

If client request time is later than the corresponding cache file modified time, use the "If-Modified-Since: " time as my If-Modified-Since header to my server request and use the current system time:

GET http://www.google.com HTTP/1.1

Host: www.google.com

If-Modified-Since: Mon, 09 Apr 2012 06:24:59 GMT (client If-Modified-Since time)

Connection: close

All other client requests header will be ignore.

9a. make the connection to the host, webfd = connectTo(host, 80));

9b. if webfd == -1, which means the connection fails. The server will try to read data from local cache file (caching function).

10. send the request to host.

11. read a block of data from remote server response.

12a. read the remote server response to the file I create before, in several blocks(block number depends on the block size, this can be easily changed by revising the constant BUF\_SIZE).

12b. if remote server responses "HTTP/1.1 304 Not Modified", go to read local cache file.

13. write the local cache file to client connfd.

14. record the log information (client request, remote server response, cache file read) appropriately.

15. free heap stuff and exit thread.

Program usage:

This program implement server function, caching function and HTTP proxy function.

1. inpute "make" to create cli and srv execute files.

2. For terminal connect to my proxy server:

2-a, open one terminal, input "./srv port#" for a HTTP sever or proxy function.

2-b, open another terminal,

input "./cli port# GET http://www.google.com/ HTTP/1.1" or

"./cli port# GET http://www.upenn.edu/ HTTP/1.1 If-Modified-Since: Thu, 12 Apr 2012 21:50:05 EDT"

at client terminal to send the request to my server. You must input the exactly as the format shown above. Note: don't forget the "/" after the url.

4. For browser connect to server/proxy:

4-a, open one terminal, input "./srv port#" for a HTTP sever or proxy function. This will set the connection port be 80 which can connect to a remote server.

4-b, open a browser (chrome and firefox were tested), input url at browser address bar and press enter.

5. For browser opening a local file, input

"http://localhost:1234/test.txt" at address bar and press enter.

6. the server will keep running till user input "ctrl c" to interrupt the program. This will call signal handler to close the srvfd.

7. all saved cache files will saved in the cli-srv folder. Make clean can clean up those .o files but can't clean these cache files.

Test cases:

1. web server function:

client terminal input:

./cli 3333 GET http://www.upenn.edu/ HTTP/1.1 If-Modified-Since: Thu, 12 Apr 2012 21:50:05 EDT

client terminal output:

HTTP/1.1 200 OK

Server: Apache/2.2.22 (Unix) mod\_ssl/2.2.22 OpenSSL/0.9.8e-fips-rhel5

Content-Type: text/html

X-Pad: avoid browser bug

Date: Fri, 13 Apr 2012 02:20:52 GMT

Content-Length: 17096

Connection: close

<!DOCTYPE html>.......

logfile:

Web server response:

HTTP/1.1 200 OK

Update local file:

www.upenn.edu-

2. web caching function(after the previous visiting):

client terminal input:

./cli 3333 GET http://www.upenn.edu/ HTTP/1.1 If-Modified-Since: Thu, 12 Apr 2012 21:50:05 EDT

client terminal output:

HTTP/1.1 304 Not Modified

logfile:

Fri, 13 Apr 2012 02:23:56 GMT:

Received request:

GET http://www.upenn.edu/ HTTP/1.1 If-Modified-Since: Thu, 12 Apr 2012 21:50:05 GMT

My server response: HTTP/1.1 304 Not Modified

3. www.google.com was tested at terminal and browser. Can properly connect to google server and cache from local disc.

4. some other website were tested including www.6park.com, www. mitbbs.com, www.upenn.edu, etc. These websites can be displayed appropriately.

5. firefox and google chrome were used as browser to test my server.

Known issues:

www.yahoo.com can be connected. However, the browser can't display it appropriately. This may because some special format of the web page which can't save and then display appropriately.