Points earned
0

Back to all evaluation sheets

Webserv

You should evaluate 3 student in this team

Introduction

Please follow the rules below:

- Remain polite, courteous, respectful, and constructive throughout the evaluation process. The community's well-being depends on it.
- Work with the student or group being evaluated to identify potential issues in their project. Take time to discuss and debate the problems identified.
- Understand that there may be differences in how peers interpret the project instructions and scope. Always keep an open mind and grade as honestly as possible. Pedagogy is effective only when peer evaluations are taken seriously.

Guidelines

Please follow the guidelines below:

- Only grade the work submitted to the Git repository of the evaluated student or group.
- Double-check that the Git repository belongs to the student(s) and that the project is the one expected. Ensure that git clone is used in an empty folder.
- Carefully verify that no malicious aliases are used to deceive the evaluator into grading non-official content.
- If applicable, review any scripts used for testing or automation together with the student.
- If you haven't completed the assignment you're evaluating, read the entire subject before starting the evaluation.
- Use the available flags to report an empty repository, a non-functioning program, a Norm error, or cheating. The evaluation process ends with a final grade of 0 (or -42 for cheating). However, except in cases of cheating, students are encouraged to review the work together to identify mistakes to avoid in the future.
- Remember that no segfaults or other unexpected program terminations will be tolerated during the evaluation. If this occurs, the final grade is 0. Use the appropriate flag.
- You should not need to edit any files except the configuration file, if it exists. If editing a file is necessary, explain the reasons to the evaluated student and ensure mutual agreement.
- Verify the absence of memory leaks. All memory allocated on the heap must be properly freed before the program ends.
- You may use tools like leaks, valgrind, or e_fence to check for memory leaks. If memory leaks are found, tick the appropriate flag.

Attachments

Please download the attachments below:

- subject.pdf
- tester
- ubuntu_cgi_tester
- cgi_tester ubuntu_tester

Mandatory Part

Check the code and ask questions

Check the code and ask questions

- Launch the installation of siege with homebrew.
- Ask explanations about the basics of an HTTP server.
- Ask what function the group used for I/O Multiplexing.
- Ask for an explanation of how does select() (or equivalent) work.
- Ask if they use only one select() (or equivalent) and how they've managed the server to accept and the client to read/write.
- The select() (or equivalent) should be in the main loop and should check file descriptors for read and write AT THE SAME TIME. If not, the grade is 0 and the evaluation process ends now.
- There should be only one read or one write per client per select() (or equivalent). Ask the group to show you the code from the select() (or

equivalent) to the read and write of a client.

- Search for all read/recv/write/send on a socket and check that, if an error is returned, the client is removed.
- Search for all read/recv/write/send and check if the returned value is correctly checked (checking only -1 or 0 values is not enough, both should be checked).
- If errno is checked after read/recv/write/send, the grade is 0 and the evaluation process ends now.
- Writing or reading ANY file descriptor without going through the select() (or equivalent) is strictly FORBIDDEN.
- The project must compile without any re-link issue. If not, use the 'Invalid compilation' flag.
- If any point is unclear or is not correct, the evaluation stops.

Yes | No

Points earned
0

Configuration

In the configuration file, check whether you can do the following and test the result:

- Search for the HTTP response status codes list on the internet. During this evaluation, if any status codes is wrong, don't give any related points.
- Setup multiple servers with different ports.
- Setup multiple servers with different hostnames (use something like: curl --resolve example.com:80:127.0.0.1 http://example.com/).
- Setup default error page (try to change the error 404).
- Limit the client body (use: curl -X POST -H "Content-Type: plain/text" --data "BODY IS HERE write something shorter or longer than body limit").
- Setup routes in a server to different directories.
- Setup a default file to search for if you ask for a directory.
- Setup a list of methods accepted for a certain route (e.g., try to delete something with and without permission).

Yes | No

Basic checks

Using telnet, curl, prepared files, demonstrate that the following features work properly:

- GET, POST and DELETE requests should work.
- UNKNOWN requests should not result in a crash.
- For every test you should receive the appropriate status code.
- Upload some file to the server and get it back.

Yes | No

Check CGI

Pay attention to the following:

- The server is working fine using a CGI.
- The CGI should be run in the correct directory for relative path file access.
- With the help of the students you should check that everything is working properly. You have to test the CGI with the "GET" and "POST" methods.
- You need to test with files containing errors to see if the error handling works properly. You can use a script containing an infinite loop or an error; you are free to do whatever tests you want within the limits of acceptability that remain at your discretion. The group being evaluated should help you with this.

The server should never crash and an error should be visible in case of a problem.

Yes | No

Check with a browser

Check with a browser

- Use the reference browser of the team. Open the network part of it, and try to connect to the server using it.
- Look at the request header and response header.
- It should be compatible to serve a fully static website.
- Try a wrong URL on the server.
- Try to list a directory.

- Try a redirected URL.
- Try anything you want to.

Yes | No

Port issues

Port issues

- In the configuration file setup multiple ports and use different websites. Use the browser to ensure that the configuration works as expected and shows the right website.
- In the configuration, try to setup the same port multiple times. It should not work.
- Launch multiple servers at the same time with different configurations but with common ports. Does it work? If it does, ask why the server should work if one of the configurations isn't functional. Keep going.

Yes | No

Siege & stress test

Siege & stress test

- Use Siege to run some stress tests.
- Availability should be above 99.5% for a simple GET on an empty page with a siege -b on that page.
- Verify there is no memory leak (Monitor the process memory usage. It should not go up indefinitely).
- Check if there is no hanging connection.
- You should be able to use siege indefinitely without having to restart the server (take a look at siege -b).

Yes | No

Points earned
0

Bonus Part

Cookies and session

Evaluate the bonus part if, and only if, the mandatory part has been entirely and perfectly done, and the error management handles unexpected or bad usage. In case all the mandatory points were not passed during the defense, bonus points must be totally ignored.

There is a working session and cookies system on the webserver.

Yes | No

CGI

CGI

There is more than one CGI system.

Yes | No

Ratings

OK | Outstanding | Empty Work | Incomplete Work | Invalid Compilation

Cheat | Crash | Concerning Situations | Leaks | Forbidden Functions

Points earned
0