

Tarea 3

- Estudiantes: Pablo Alberto Muñoz Hidalgo, Royner Miranda Segura
- Profesor: Kevin Moraga García
- Curso y Universidad: ITCR Sistemas Operativos
- Año: 2022

Introducción

Se debe de crear un HTTP server el cual implementa la técnica llamada pre thread en el lenguaje de programación Rust, este debe ser capaz de recibir usuarios utilizando POST, GET, HEAD, PUT, DELETE. También se debe crear un cliente HTTP el cual permita descargar un binario a través de una lista de comandos en los parámetros o bien interactuar con el servidor HTTP como cualquier otro cliente HTTP. Por último se debe crear un StressCMD que su objetivo de unir el HTTPClient y el StressCMD es saturar los WebServers hasta que estos se queden sin posibilidad de atender otro cliente más. En el lenguaje Python.

Ambiente de desarrollo

Se estará utilizando Ubuntu 20.04.4 LTS y como IDE se utilizará Visual Studio Code. Para probar la pagina web se usa Firefox 103.0.1 (64-bit). Además de un repositorio en github.

Estructuras de datos usadas y funciones:

Estructuras:

- Workers
- ThreadPool

Funciones:

Web Server:

- listener (Función para recibir todas las solicitudes del cliente al servidor, ademas maneja el uso y contador de hilos)
- handle_connection (Función para manejar la conexion con el cliente)
- comprobador (Función para manejar los argumentos de entrada)
- print_vector (Función para imprimir vectores)

HTTP Client:

- get (Función para enviar una solicitud de tipo GET al servidor)
- post (Función para enviar una solicitud de tipo POST al servidor)
- delete (Función para enviar una solicitud de tipo DELETE al servidor)
- put (Función para enviar una solicitud de tipo PUT al servidor)

Stress CMD:

- exitStress (Función que se encarga de "apagar el stress")
- stress (Función que realiza el stress)
- initializeStress (Función que se encarga de inicializa el stress con los parametros que recibe)

Instrucciones para ejecutar el programa:

1. Comenzar el web_server ejecutandolo desde consola
2. Despues es posible enviar solicitudes por medio del HTTPClient
3. En caso de que se quiera botar el server usar StressCMD
4. Disfrutar :)

Se deben usar las siguientes estructuras para cada ejecutable.

WebServer: `"./web_server prethread-WebServer -n -w -p "`

HTTPClient: `"./httpclient HTTPClient -h []"`

StressCMD: `"python3 stresscmd stress -n HTTPClient "`

Actividades realizadas por estudiante:

Fecha	Hora de Inicio	Hora de Finalización	Actividad realizada	Estudiante
09/09/2022	3:00 PM	10:00 PM	Investigación y creación del git	Pablo Muñoz
10/09/2022	8:00 AM	12:00 PM	Preparación del ambiente de desarrollo y más investigación	Pablo Muñoz
10/09/2022	3:00 PM	12:00 AM	Primeras implementaciones del WebServer	Pablo Muñoz
11/09/2022	7:00 AM	12:00 PM	Implementación de nuevos html y comenzar el StressCMD	Pablo Muñoz
11/09/2022	1:00 PM	6:00 PM	Creación del Client	Pablo Muñoz
11/09/2022	10:00 PM	12:00 AM	Contador de threads y mensaje de error	Pablo Muñoz
12/09/2022	2:00 PM	6:00 PM	StressCMD funcional, Client Funcional solo con GET y documentación	Pablo Muñoz
14/09/2022	6:00 PM	12:00 PM	Documentación, root como parámetro y programas funcionales desde consola	Pablo Muñoz
15/09/2022	9:30 PM	11:00 PM	Documentación y cambios finales	Pablo Muñoz
09/09/2022	6:00 PM	10:00 PM	Investigación sobre el web server	Royner Miranda

Fecha	Hora de Inicio	Hora de Finalización	Actividad realizada	Estudiante
10/09/2022	10:00 AM	4:00 PM	Implementación del web server con un hilo	Royner Miranda
11/09/2022	10:00 AM	5:00 PM	Investigación sobre el client y métodos HTTP	Royner Miranda
12/09/2022	7:00 PM	12:00 PM	Implementación método POST	Royner Miranda
13/09/2022	4:00 PM	9:00 PM	Implementación de los demás métodos HTTP	Royner Miranda
15/09/2022	5:00 PM	8:00 PM	Documentación	Royner Miranda

Horas totales: 45.5 (Pablo) + 30 (Royner) = 75.5 horas

Autoevaluación:

Estado del programa

El programa funciona perfectamente exceptuando la recepción de diferentes protocolos como Telnet o SNMP. Además no se pueden descargar documentos del servidor HTTP.

Problemas encontrados y limitaciones adicionales

Se encontraron problemas que no pudieron ser solucionados, en este caso no se logró que el servidor provea el servicio de descarga de archivos grandes, tampoco el parseo con otros protocolos diferentes a HTTP.

Evaluación

WebServer	Implementación de protocolos	HTTPclient en Rust	Stress-Client	Documentación	Kick-off
37/40	7/10	15/15	15/15	20/20	5/5

Autoevaluación:

Aprendizaje de pthreads	Aprendizaje de comunicación entre procesos	Aprendizaje de sockets	Estudiante
5	5	5	Pablo Muñoz
5	5	5	Royner Miranda

Reporte de commits:

commit 11b38a0821283abe4ff7658aca1b7827d69d9aef

Author: Pablo Munoz Hidalgo 53487847+Litecore50@users.noreply.github.com

Date: Thu Sep 15 21:59:45 2022 -0600

Version Final v1.0

commit 0d087f400717f788e7c7862f6a51ea82c9ae2e75

Author: Royner39 roynerarturo39@gmail.com

Date: Thu Sep 15 19:21:54 2022 -0600

Documentación completa

commit ac455a62ee072529859eae5d4fab1950346c01bc

Author: Pablo Munoz Hidalgo 53487847+Litecore50@users.noreply.github.com

Date: Wed Sep 14 23:12:25 2022 -0600

Documentacion actualizada

commit 7304641038e22d4090aba33468179515613ea08c

Author: Pablo Munoz Hidalgo 53487847+Litecore50@users.noreply.github.com

Date: Wed Sep 14 23:09:55 2022 -0600

Comentarios y documentacion actualizada

commit 92ae5c0c76d7fac0dcaaa60b2346288af56f1e02

Author: Pablo Munoz Hidalgo 53487847+Litecore50@users.noreply.github.com

Date: Wed Sep 14 22:48:03 2022 -0600

Código comentado

commit b3d8c79a8db64d380762a7f836d3b780b7f4ff98

Author: Pablo Munoz Hidalgo 53487847+Litecore50@users.noreply.github.com

Date: Wed Sep 14 22:33:29 2022 -0600

Rutas como parametros y comandos en consola al 100%

commit 7e82ef0fe1f4fb6cfc3caeb75b30002215bdc3a7

Author: Royner39 roynerarturo39@gmail.com

Date: Tue Sep 13 19:48:02 2022 -0600

Implementación de métodos POST, PUT, DELETE

commit 0accac8fb477ccd7e130e5323bcc38be5f2f50cd

Author: Royner39 roynerarturo39@gmail.com

Date: Tue Sep 13 08:55:31 2022 -0600

Caso Post

Falta probar

commit f9acb8af15216f105f3498d29a871f07c9aaa3d0

Author: Pablo Munoz Hidalgo 53487847+Litecore50@users.noreply.github.com

Date: Mon Sep 12 18:22:17 2022 -0600

Documentación

commit 68a4d74cc00ab0bec77f8c9b1d0a4fdb0b49707c

Author: Pablo Munoz Hidalgo 53487847+Litecore50@users.noreply.github.com

Date: Mon Sep 12 18:15:40 2022 -0600

Client y StressCMD funcional, solo método GET

commit d40b51d5c927b1131c9be074d6da9661607b0fca

Author: Pablo Munoz Hidalgo 53487847+Litecore50@users.noreply.github.com

Date: Sun Sep 11 23:08:03 2022 -0600

Actualizacion stressCMD, Webserver funcional, httpclient actualizado

commit 71f145e1979550381b79b79bb0631d093ebb9fa7

Author: Pablo Munoz Hidalgo 53487847+Litecore50@users.noreply.github.com

Date: Sun Sep 11 15:34:44 2022 -0600

StressCMD funcional, WebServer se cae y httpclient progreso

commit 653cc607e4c44cef0ffc95dc98899f0282322418

Author: Pablo Munoz Hidalgo 53487847+Litecore50@users.noreply.github.com

Date: Sat Sep 10 16:46:56 2022 -0600

WebServer funcional

commit 784ca2e2367a01223dd3685275a4fed42ef0e53a

Author: Pablo Munoz Hidalgo 53487847+Litecore50@users.noreply.github.com

Date: Sat Sep 10 15:02:06 2022 -0600

Create README.md

commit 492394198a0b347be09e286f43d89a06c493dcec

Author: Royner39 roynerarturo39@gmail.com

Date: Sat Sep 10 14:40:26 2022 -0600

Implementación de 404

commit 8f37ba844660919fb581cbad79ff4747565f9097

Author: Royner39 roynerarturo39@gmail.com

Date: Sat Sep 10 14:14:45 2022 -0600

Web Server básico

commit a641ff9042224ef229d0f4972cffb970c16e0c86

Author: Pablo Munoz Hidalgo 53487847+Litecore50@users.noreply.github.com

Date: Tue Sep 6 14:03:21 2022 -0600

REFERENCIAS!!!!!!!!!!!!!!

commit f87c05aa7059b11ec4c298730c710f38e33cd876

Author: Pablo Munoz Hidalgo 53487847+Litecore50@users.noreply.github.com

Date: Tue Sep 6 13:37:51 2022 -0600

Más referencias

commit fafbd3459997d8bdf091d2052d6c313018607e25

Author: Pablo Munoz Hidalgo 53487847+Litecore50@users.noreply.github.com

Date: Tue Sep 6 13:34:20 2022 -0600

Referencias útiles para la tarea

commit 760bbf2e83406c8645afc66d344f37c3f18f0b53

Author: Pablo Munoz Hidalgo 53487847+Litecore50@users.noreply.github.com

Date: Tue Sep 6 13:24:36 2022 -0600

Create test

Lecciones Aprendidas:



En esta tarea se aprendió muchísimo sobre threads y más que todo sobre el funcionamiento del protocolo HTTP y sus propiedades, el hacer que varios programas funcionen en conjunto también fue un reto interesante. Creemos que es una tarea bastante pesada sin embargo que deja muchas lecciones interesantes como el manejo de módulos en diferentes lenguajes de programación. Es una tarea que deja bastante que decir en un curriculum por lo que si se desea destacar recomendamos hacerla.

Además, aprendimos como crear un client-web server con la implementación de métodos como GET, POST, DELETE, entre otros; lo cual es bastante importante conocer.

Bibliografía:

[1]"Final Project: Building a Multithreaded Web Server - The Rust Programming Language", Doc.rust-lang.org, 2022. [Online]. Available: <https://doc.rust-lang.org/book/ch20-00-final-project-a-web-server.html>. [Accessed: 15- Sep- 2022].

[2]"Building a Single-Threaded Web Server - The Rust Programming Language", Doc.rust-lang.org, 2022. [Online]. Available: <https://doc.rust-lang.org/book/ch20-01-single-threaded.html>. [Accessed: 15- Sep- 2022].

- [3]"Simple HTTP server example for Rust", Gist, 2022. [Online]. Available: <https://gist.github.com/mjohnsullivan/e5182707caf0a9dbdf2d>. [Accessed: 15- Sep- 2022].
- [4]"DoS & DDoS attack", Tutorialspoint.com, 2022. [Online]. Available: https://www.tutorialspoint.com/python_penetration_testing/python_penetration_testing_dos_and_ddos_attack.htm. [Accessed: 15- Sep- 2022].
- [5]"Build software better, together", GitHub, 2022. [Online]. Available: <https://github.com/topics/dos-attacks?l=python>. [Accessed: 15- Sep- 2022].
- [6]"Code A DDOS Script In Python - NeuralNine", NeuralNine, 2022. [Online]. Available: <https://www.neuralnine.com/code-a-ddos-script-in-python/>. [Accessed: 15- Sep- 2022].
- [7]"Denial of Service attack in Python", Medium, 2022. [Online]. Available: <https://medium.com/@joel.barmettler/denial-of-service-attacks-do-not-always-have-to-flood-the-server-with-requests-to-make-him-shut-c630e59b731e>. [Accessed: 15- Sep- 2022].
- [8]"Multithreading in C - GeeksforGeeks", GeeksforGeeks, 2022. [Online]. Available: <https://www.geeksforgeeks.org/multithreading-c-2/>. [Accessed: 15- Sep- 2022].
- [9]"<pthread.h>", Pubs.opengroup.org, 2022. [Online]. Available: <https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/pthread.h.html>. [Accessed: 15- Sep- 2022].
- [10]"pthreads(7) - Linux manual page", Man7.org, 2022. [Online]. Available: <https://man7.org/linux/man-pages/man7/pthreads.7.html>. [Accessed: 15- Sep- 2022].
- [11]"What are GET, POST, PUT, PATCH, DELETE? A walkthrough with JavaScript's Fetch API.", Medium, 2022. [Online]. Available: <https://medium.com/@9cv9official/what-are-get-post-put-patch-delete-a-walkthrough-with-javascripts-fetch-api-17be31755d28>. [Accessed: 15- Sep- 2022].
- [12]"IBM Documentation", Ibm.com, 2022. [Online]. Available: <https://www.ibm.com/docs/es/ma/m/7.6.1?topic=api-put-post-delete-methods>. [Accessed: 15- Sep- 2022].
- [13]"HTTP Methods "GET", "POST", "PUT", "PATCH", "DELETE"", DEV Community  , 2022. [Online]. Available: <https://dev.to/qbentil/http-methods-get-post-put-patch-delete-1fhi>. [Accessed: 15- Sep- 2022].
- [14]"▷ Telnet qué es y para qué sirve 【 La más completa 】 ", Profesional Review, 2022. [Online]. Available: <https://www.profesionalreview.com/2019/01/20/telnet-que-es/>. [Accessed: 15- Sep- 2022].
- [15]H. [closed] and s. bachert, "How to send an HTTP request using Telnet", Stack Overflow, 2022. [Online]. Available: <https://stackoverflow.com/questions/15772355/how-to-send-an-http-request-using-telnet>. [Accessed: 15- Sep- 2022].
- [16]"Network Time Protocol - Wikipedia", En.wikipedia.org, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Network_Time_Protocol. [Accessed: 15- Sep- 2022].