Instituto Tecnologico de Costa Rica

Sistemas Operativos

III Tarea

Rust The Future Machine

Jean Marco Rojas

2015040717

Jonathan Guillén

201261579

Profesor
Sr. K. Moraga

Abril 27, 2021

1 Introduction

En la actualidad muchos de los FTPservers se encuentran subutilizados, pero en períodos de tiempo cortos, es posible que se sobre utilicen debido a la demanda del servicio. Esto sucede con muchos proveedores de entradas a eventos masivos. La configuración normalmente no contempla la limitación de los recursos. Existen dos técnicas que pueden ayudar a administrar mejor los recursos de un servidor, ellas son pre-thread y pre-forked. Los requerimientos funcionales con los que contará el proyecto serán:

- Pre-thread FTPServer: Crea un FTP server el cual implementa la técnica llamada prethread. Esta técnica consiste en crear previamente varios hilos de ejecución del método que atiende la solicitudes. Estos hilos se crean utilizando la biblioteca pthreads de Unix. Debe de recibir como parámetro el número de hilos N que se deben pre-crear, el FTPserver escuchará en el puerto estándar de FTP, y tendrá N hilos posibles para atender la solicitud, cada solicitud se atenderá por el primer hilo que esté disponible. En caso que no existan más disponibles mostrará un mensaje de error, indicando que se ha sobrepasado el número de clientes que pueden ser atendidos.

- Pre-forked FTPServer: Crea un FTP server el cual implemente la técnica llamada pre-forked. Esta técnica consiste en crear previamente varios procesos del método que atiende la solicitudes. Estos procesos se crean utilizando el system call estándar de Unix. Debe de recibir como parámetro el número de procesos N que se deben pre-crear, el FTPserver escuchará en el puerto estándar de FTP, y tendrá N procesos posibles para atender la solicitud, cada solicitud se atenderá por el primer proceso que esté disponible. En caso que no existan más disponibles mostrará un mensaje de error, indicando que se ha sobrepasado el número de clientes que pueden ser atendidos.

- FTPClient en Rust: Crea un cliente FTP el cual permita descargar un binario a través de una lista de comandos en los parámetros o bien interactuar con el servidor FTP como cualquier otro cliente FTP. Para ello utilice la biblioteca curl en el lenguaje de programación Rust.

- FTPClient en C: Crea un cliente FTP el cual permita descargar un binario a través de una lista de comandos en los parámetros o bien interactuar con el servidor FTP como cualquier otro cliente FTP. Para ello utilice el lenguaje de programación C.

- StressCMD: Crea una aplicación que reciba como parámetro un ejecutable (Además de los parámetros del ejecutable). Luego debe de crear la cantidad de hilos que el cliente especifique, con el objetivo de lanzar un ataque de Denegación de Servicio. El principal objetivo de unir el FTPClient y el StressCMD es de saturar los FTPservers hasta que estos se queden sin posibilidad de atender otro cliente más. En el lenguaje Rust.

2 Ambiente de desarrollo:

Cada miembro del equipo trabajara desde su propia máquina para la realización del código necesario para el proyecto, el problema presentado se dividirá en asignaciones y cada miembro seleccionará de cuales se hará cargo distribuyendo equitativamente las cargas de trabajo, esto se hará mediante sprints semanales, que nos permite dividir y observar el trabajo realizado por los integrantes del grupo, cada vez que se dé por terminada una asignación o parte de la misma se debe subir el trabajo realizado a Github de manera tal que permita el acceso a dicha información a los demás miembros del grupo, esto con el fin de tener una integración continua con versiones del trabajo realizado, poder realizar revisiones y comentar sobre el código realizado, además al final de cada sprint se hace una autoevaluación para detectar las debilidades del grupo de trabajo y así saber en qué mejorar. A continuación se detallan las herramientas utilizadas para la realización de esta tarea:

C: En el caso de este lenguaje se utilizará el compilador gcc y se programara utilizando Sublime text 3, Clion, VScode y Netbeans 8.2.

Rust: En su versión 1.51.0 y se programará utilizando, Clion, IntelliJ IDEA.

Latex: Este editor de texto será la herramienta principal para la realización de los documentos del kick off y la documentación final de la tarea.

GitHub: Como sistema de control de versiones.

Google Chrome: Como navegador web y donde se correrá el servidor, en su versión 89.0.4389.114.

Ubuntu: Como sistema operativo donde será desarrollada la tarea, en su versión 20.04.2.0.

3 Estructuras de datos usadas y funciones

sleep(): Pausa el hilo actual de ejecución del programa.
Main() del servidor: Levanta el servidor, espera por los clientes y envía y recibe los mensajes de los mismos.
clone(): Clona el contenido del archivo, para posteriormente ser enviado.
LOCAL: Contiene la variable de tipo String con la dirección ip y el puerto del servidor.
MSG_SIZE: Contiene la variable de tipo entero con el tamaño máximo de los mensajes, por defecto 32 bytes.
FILE_SIZE: Contiene la variable de tipo entero con el tamaño máximo del archivo por enviar, por defecto 1000 bytes.
Main() del cliente: Levanta la conexión con el servidor, en el puerto e ip especificados, y desde aquí hace las solicitudes al servidor.

4 Instrucciones para ejecutar el programa

Para ejecutar el programa se debe tener instalado los lenguajes en los que esta realizado este lenguaje, estos serian C y Rust. Para ejecutar el cliente y el

servidor en Rust se debe escribir el siguiente comando en la terminal en las carpetas del cliente y del server:

```
cargo run.
```

Para ejecutar el cliente y el servidor en C, se debe escribir el siguiente comando en la terminal para ejecutar el servidor:

```
./server 4444 8
```

Donde "4444" es el puerto, y "8" es el numero de procesos. Para ejecutar el cliente se debe escribir el siguiente comando en la terminal:

```
./client 4444
```

Donde "4444" es el puerto al cual se conectará el cliente

5 Actividades realizadas

- 8 de abril 2021 (Jean Marco): Buscar información sobre sockets en c y rust (3 horas).
- 11 de abril 2021 (Jean Marco): Buscar información sobre servidores FTP (2 horas).
- 12 de abril 2021 (Jonathan): Creación del repositório (2 horas).
- 15 de abril 2021 (Jonathan): Programar sockets en rust (4 horas).
- 17 de abril 2021 (Jean Marco): Programar sockets en C son forks (4 horas).
- 18 de abril 2021 (Jonathan): Continuar programando sockets FTP en rust (4 horas).
- 27 de abril 2021 (Jean Marco): Escribir la documentacion (3 horas).

6 Autoevaluación

En el transcurso del proyecto se logro la realizacion de los clientes y el servidor en los sockets, tanto en C como en Rust. Se logro que se puedan enviar archivos que esten en el server y que el cliente pueda recivir los archivos. No se logró realizar la implementacion de la prueba de estres.

6.1 Evaluación

- FTPServer: Completado
- Implementación de Protocolos: Completado
- ftpclient en Rust: Completado

- ftpclient en C: Completado
- Stress-Client: No Realizado
- Documentación: Completado
- Kick-off: Completado

6.2 Autoevaluación

[5] Aprendizaje de pthreds.
[4] Aprendizaje de forks.
[4] Aprendizaje de comunicacion entre procesos.
[5] Aprendizaje de sockets.

7 Lecciones Aprendidas y Recomendaciones

- Investigar y leer toda la documentación de rust posible, por que es bastante escasa y en youtube hay muy muy poca.
- Funcionamiento de los sockets a profundidad así como los servidores FTP.
- Investigar lectura y escritura de archivos en Rust.
- Investigar sobre sockets y como usarlos en Rust

8 Bibliografía

- Sockets en C de Unix/Linux. (2021). Retrieved 28 April 2021, from http://www.chuidiang.org/clinux/sockets/sockets_simp.php
- std::net - Rust. (2021). Retrieved 28 April 2021, from https://doc.rust-lang.org/std/net/index.html
- Unix sockets — Rust by Example. (2021). Retrieved 28 April 2021, from https://www.cs.brandeis.edu/ cs146a/rust/rustbyexample-02-21-2015/sockets.html
- Protocolo de transferencia de archivos - Wikipedia, la enciclopedia libre. (2021). Retrieved 28 April 2021, from https://es.wikipedia.org/wiki/Protocolo$_de_transferencia_de_archivo$