Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Principios de sistemas operativos

Rust the future machine

Desarrollado por

- #### Jose Daniel Peñaranda Umaña (2013039845)
- #### Carlos Mora Murillo (2017238926)

Alajuela, I-S 2021

Introdución

Los Web Servers comúnmente trabajan en un mínimo de sus capacidades sin embargo existen picos de uso y es donde se debe recurrir a técnicas para poder satisfacer la demanda, ellas son el pre forked y el pre thread técnicas que nos ayudarán a manejar de mejor forma los recursos disponibles en el servidor.

Ambiente de Desarrollo

La tarea se implementó en los siguientes ambientes de trabajo:

- Ubuntu en su última versión como sistema operativo
- VScode como IDE de desarrollo
- Clang como compilador
- Make para la crear los archivos ejecutables
- Rust

Estructuras de datos usadas y funciones

Pre Forked Server

IniciarServidor()

Esta función se encarga de tomar los parámetros del input y establecer todas las configuraciones para el servidor.

ServidorManager()

Esta funcion se encarga de manejar el ciclo de ejecucion del servidor y mantener la recepcion de clientes, ademas verifica si se supero el numero maximo de clientes permitidos.

CrearSocket()

Esta funcion se encarga de crear el socket del servidor.

CrearProcesos()

Esta funcion se encarga de crear el pool de procesos y ponerlos en espera.

BindSocket()

Se encarga de enlazar con el servidor su socket de funcionamiento.

PreThread Server

IniciarServidor()

Esta función se encarga de tomar los parámetros del input y establecer todas las configuraciones para el servidor.

ServidorManager()

Esta funcion se encarga de manejar el ciclo de ejecucion del servidor y mantener la recepcion de clientes, ademas verifica si se supero el numero maximo de clientes permitidos.

CrearSocket()

Esta funcion se encarga de crear el socket del servidor.

CrearHilos()

Esta funcion se encarga de crear el pool de hilos y ponerlos en espera.

BindSocket()

Se encarga de enlazar con el servidor su socket de funcionamiento.

FTPClient en C

CrearSocket()

Esta funcion se encarga de crear el socket por el que el cliente va a comunicarse con el servidor.

Conectar()

Esta funcion se encarga de establecer una conexion con el servidor.

main()

En esta funcion se controla el flujo del cliente es donde se procesan las solicitudes y se le dan los coomandos al servidor para se ejecutados.

Instrucciones para ejecutar el programa

Las instrucciones para ejecutar y compilar nuestra tarea son:

Make en la carpeta principal del proyecto para compilar el servidor en pre thread

Para el servidor preforked es necesario utilizar el comando clang

clang -g preforked-FTPserver.c -o preforked-FTPserver

Para el caso del cliente es necesario entrar a la carpeta del cliente y correr el comando clang

clang -g FTPClient.c -o FTPClient

una vez hecho esto seguir las siguientes instrucciones

- Prethread Server: prethread-webserver -n -w -p
- Preforked Server: preforked-webserver -n -w -p
- Cliente http: ftpclient [lista de comandos]

Actividades realizadas por estudiante

Carlos Mora M

Actividad	Fecha	Tiempo
Se inicio la investigación sobre el trabajo a realizar	09/04/2021	1.5 horas
Se realizo lectura sobre manejo de procesos en C	11/04/2021	2 horas
Se realizo lectura e investigación sobre la implementación de procesos en C	15/04/2021	3 horas
Se realizaron pruebas de manejo de puertos en C	15/04/2021	3.5 horas
Se realizo la primer implementación de manejo de procesos en C	18/04/2021	2.5 horas
Se hizo la primer implementación de manejo de puertos en C	18/04/2021	2.5 horas
Se realizó más investigación sobre la implementación de servidores en C	18/04/2021	3 horas
Se creo el primer prototipo de servidor	19/04/2021	5 horas

Actividad	Fecha	Tiempo
Se creo un versión más pulida del servidor	19/04/2021	2 horas
Se investigo sobre el estrés test y el cliente	20/04/2021	3 horas
Se implemento el cliente	26/04/2021	3 horas
Se realizaron las pruebas de funcionamiento	26/04/2021	30 minutos
Se realizo la documentación escrita de la tarea	26/04/2021	3 horas

Horas Totales: 34 horas.

Daniel Peñaranda

Actividad	Fecha	Tiempo
Se inicio la investigación sobre la tarea	09/04/2021	1.5 horas
Se realizo investigación sobre manejo de hilos en C	09/04/2021	2 horas
Se realizo investigación sobre la implementación de hilos en C	09/04/2021	2 horas
Se realizaron pruebas de manejo de puertos en C	10/04/2021	1.5 horas
Se realizo la primer prueba de manejo de hilos en C	10/04/2021	2.5 horas
Se hizo la primer implementación de manejo de hilos en C	10/04/2021	2.5 horas
Se realizó más investigación sobre la implementación de servidores en C	12/04/2021	3 horas
Se creo el primer prototipo de servidor	16/04/2021	5 horas
Se creo un versión más pulida del servidor	17/04/2021	2 horas
Se investigo sobre Rust y los clientes ftp	18/04/2021	3 horas
Se implemento el primer prototipo de cliente rust	18/04/2021	4 horas
Se realizaron las pruebas de funcionamiento	20/04/2021	30 minutos
Se hicieron pruebas y mejoras de funcionamiento de cliente rust	25/04/2021	3 horas
Se trabajo en la documentación de la tarea	26/04/2021	2 horas

Horas Totales: 34 y media horas.

Autoevaluación

- FTPServer: 40 % / 30% • Implementación del Pre-thread • Implementación del Pre-forked • Servir archivos grandes correctamente • Descargar un archivo por FTP desde un navegador u otro cliente FTP
- Implementación de Protocolos: 10 % / 0 %
- ftpclient en Rust: 10 % / 5 %
- ftpclient en C: 10 % / 10 %
- Stress-Client: 10 % / 0 %
- Documentación: 20 % / 20 %
- Kick-off: 5 % extra o -20 % si no se entrega / 5%

Carlos Mora M

5-Aprendizaje de pthreds. 4-Aprendizaje de forks. 4-Aprendizaje de comunicacion entre procesos. 5-Aprendizaje de sockets

Daniel Peñaranda

5-Aprendizaje de pthreds. 4-Aprendizaje de forks. 4-Aprendizaje de comunicacion entre procesos. 5-Aprendizaje de sockets.

Estado Final del Proyecto

Se logro realizar todo lo referente a servidor en hilos y procesos ademas se logro implementar satisfactoriamente el cliente en lenguaje c sin embargo la parte que involucra al lenguaje Rust no se pudo llevar a acabo en su totalidad, esto por la falta de documentacion y por el poco conocimiento sobre el funcionamiento dle mismo.

Git Comits

https://github.com/Daniel-penau/Rust-The-Future-Machine/commits/main

Lecciones Aprendidas

Carlos Mora M

Durante esta tarea se aprendió mucho sobre el manejo de los procesos independientemente, además de entender a fondo su funcionamiento y cómo estos requieren una memoria compartida para su funcionamiento, aprendí a crear segmentos de memoria y utilizarlos como una tabla de índices para cada proceso, consiguiendo mantenerlos abiertos y utilizarlos a conveniencia.

También fue una tarea que me enseñó mucho sobre los distintos protocolos de manejo de archivos y cómo estos son utilizados, también aprendí a correr programas de C desde python mediante una terminal y mantener estos en hilos individuales de ejecución.

Me pareció una tarea muy enriquecedora académicamente, pero pesada por el tiempo que requiere y el nivel de investigación que hay que realizar para poder entender y trabajar con hilos y procesos, por momentos fue frustrante ya que C incrementa en gran cantidad la dificultad del trabajo.

Daniel Peñaranda

Se aprendio mucho sobre el manejo de hilos, utilizando de manera intenligente la funcionalidad que dan los phtreads, especialmente con sus modos *cond...* y *mutex...* Además, se repaso C y hasta se llegaron a aprender muchas funcionalidades que este lenguaje brinda, notando aún más la importancia y flexibilidad que da el poder manejar memoria.

Se aprendio mucho del manejo y funcionamiento de Rust, más que en mi caso no he tenido la oportunidad de trabajar a fondo con él antes. Aunque no se logro implementar completamente como para ver su funcionamiento en nuestra tarea, con la teoria aprendida y las pruebas (aunque no del todo satisfactorias) dieron una muy valiosa experiencia para futuros trabajos que se realicen.

Bibliografía

<pthread.h>. Retrieved 5 November 2020, from https://pubs.opengroup.org/onlinepubs/007908799/xsh/pthread.h.html

Campos, O. (2011). Multiprocesamiento en Python: Threads a fondo, introducción. Retrieved 5 November 2020, from https://www.genbeta.com/desarrollo/multiprocesamiento-en-python-threads-a-fondo-introduccion

Doherty, J. (2014). What exactly is a pre-fork web server model?. Retrieved 5 November 2020, from https://stackoverflow.com/questions/25834333/what-exactly-is-a-pre-fork-web-server-model

NA, G. (2020). Gisi NA. Retrieved 5 November 2020, from https://www.youtube.com/channel/UC4QlswVP3DPsOI_Ub2q4m3A/videos

Ortiz, S. (2018). Como funciona la función fork(). Retrieved 5 November 2020, from https://es.stackoverflow.com/questions/179414/como-funciona-la-funci%C3%B3n-fork

What is a pre-fork worker model for a server? : learnprogramming. (2014). Retrieved 5 November 2020, from https://www.reddit.com/r/learnprogramming/comments/25vdm8/what_is_a_prefork_worker_model_for_a_server/