TEC | Tecnológico de Costa Rica

Instituto Tecnológico de Costa Rica

My DOS

Tarea 2

Prof. Kevin Moraga Garcia

Emanuelle Jiménez Sancho - 2017136727

II Semestre 2020

Introducción

El proyecto se basa en crear un servidor tanto en pre-thread como pre-forked, este debe de demostrar su capacidad a la hora de ser atacado con múltiples peticiones. El servidor es capaz de poder mostrar videos e imágenes, también permite la funcionalidad del CGI, el cual es poder ejecutar un binario únicamente mediante peticiones.

Ambiente de desarrollo

El ambiente de trabajo fue muy simple, únicamente se utilizó sublime text, el sistema operativo fue una distribución de linux, **no** se utilizaron debuggers para la tarea.

Estructuras de datos usadas y funciones

main: Esta es la función principal, guarda los datos insertados de los parámetros.
route: Esta función se encarga de las rutas que tendrá en el servidor.
pthread_function: Esta función es la que llama el pthread, únicamente llama al response y crear el mutex de los semáforos.
respond: Esta función se encarga de la petición con el cliente, imprime y descarga los recursos necesarios para la página que se desee desplegar.
start_server: Esta función crea el servidor dado un puerto.
request_header: Devuelve el valor del componente del header a buscar.
print_protocol: Esta función responde a la petición que NO es de http y devuelve un número indicado si es http o no.
server_forever: Esta función es la principal para poder crear el server, en esta se encuentra el while infinito.

La única estructura de datos usada fue la del arreglo y los structs de los threads.

Instrucciones para ejecutar el programa

Antes de correr el programa este se debe compilar por medio del comando **make**

El programa se puede correr de la siguiente manera, este depende del ejecutable:

Server:

```
sudo ./prethread-webserver -n <max-connections> -w <path-www-root> -p <port>
sudo ./prefork-webserver -n <max-connections> -w <path-www-root> -p <port>
```

Clients:

```
python3 httpclient.py [http://url/resource.jpg]
./httpclient [http://url/resource.jpg]
```

Stress:

```
python3 stress.py -n <how-many-threads> httpclient <parameter of the
httpclient>
```

Actividades realizadas por estudiante

Fecha: 29 de octubre

Horas: 4 horas

Actividades: Únicamente se buscó información sobre los threads, tanto videos como información de linux.

Problemas: Ninguno.

Fecha: 30 de octubre

Horas: 2 horas

Actividades: Se empezó a buscar repositorios sobre webserver ya implementados, la tarea se basará en un repo en específico.

Problemas: Ninguno.

Fecha: 31 de octubre

Horas: 3 horas

Actividades: Se implementaron todos los request requeridos para la progra, de igual manera se comprendió el código en el que se basó.

Problemas: Ninguno por ahora.

Fecha: 1 de noviembre

Horas: 6 horas

Actividades: Se implementaron las imágenes, estas se basaron en un video, fue un merge del trabajo del video con la progra.

Problemas: No quería mostrar la imagen al principio, problemas con el filedescriptor.

Fecha: 2 de noviembre

Horas: 6 horas

Actividades: Se trató de hacer el CGI, esto causó problemas al principio por el parseo del url, se soluciona con facilidad, como solución se buscó guardar el output en un txt.

Problemas: Los parámetros no funcionaban.

Fecha: 3 de octubre

Horas: 10 horas

Actividades: Se termina el CGI y la implementación de los puertos, responde correctamente con el puerto necesitado, el stress y el cliente se terminaron.

Problemas: El pthread no sirve, problemas a la hora de hacer llamadas al response()

Fecha: 4 de octubre

Horas: 6 horas

Actividades: Las imágenes no querían descargarse por alguna razón, habían problemas con el header del request, se pensó en implementar un url separado

Problemas: Ninguno, la progra se termina

Fecha: 5 de octubre

Horas: 8 horas

Actividades: Se documenta el código, se insertan videos y las pruebas comienzan contra el server, se arreglan bugs.

Problemas: Ninguno

Autoevaluación

El programa se terminó por completo, todos los puntos se completaron exitosamente. Como se trabajó solo, entonces la evaluación sería 100, todo se completó a la perfección.

Commits

---

commit a55aa3f2e6baed5606a5b8f8678b4bf043ce6bb3
Author: Emanuelle <emanuellejs1999@gmail.com>
Date: Thu Nov 5 21:40:52 2020 -0600

    Documentación terminada

---

commit f4a7c6aded538759b33e783fc037ec64ec60f910
Author: Emanuelle <emanuellejs1999@gmail.com>
Date: Thu Nov 5 20:21:25 2020 -0600

    Trabajo terminado

---

commit 6a37bc5e65a3022116f400ff2f8e2b973ce4562e
Author: Emanuelle <emanuellejs1999@gmail.com>
Date: Wed Nov 4 00:13:55 2020 -0600

MD terminado

---

commit f086651ef4862bb5127a1ed3270135c8371aeb4e
Author: Emanuelle <emanuellejs1999@gmail.com>
Date: Mon Nov 2 22:52:42 2020 -0600

CGI terminado y ya puede abrir cualquier tipo de imagen

---

commit c8e4a9d82e7c6640cd0db0d66796d86dcd5818a8
Author: Emanuelle <emanuellejs1999@gmail.com>
Date: Sun Nov 1 22:30:27 2020 -0600

Se agregan imagenes

---

commit ecb5adbb74399019ca509926d430575ae489b38a
Author: Emanuelle <emanuellejs1999@gmail.com>
Date: Sat Oct 31 20:02:38 2020 -0600

Solo falta el get y el CGI

---

commit 10b02f359fc72cfa09eaf230fe35835603afa7f9
Author: Emanuelle <emanuellejs1999@gmail.com>
Date: Wed Oct 28 23:57:15 2020 -0600

Se arreglaron problemas del .git

---

commit 046eae446ecac7ad18a72a0f9a72310a2a8b39e9
Author: Emanuelle <emanuellejs1999@gmail.com>
Date: Wed Oct 28 23:56:10 2020 -0600

Documentación del código y argumentos terminados

---

commit 8e820398108a290191c50987c33b9dec3d42ba3e
Author: Emanuelle <emanuellejs1999@gmail.com>
Date: Wed Oct 28 00:42:10 2020 -0600

Stress terminado y clientes de python/C terminados

---

commit c4a5ac6d050430ba773ae165aed6b8f0087fd218
Author: Emanlui <emanuellejs1999@gmail.com>
Date: Fri Oct 23 16:08:43 2020 -0600

Initial commit

Lecciones Aprendidas del proyecto

Se aprendió bastantes sobre los threads y cómo funcionan, se complicó bastante la tarea cuando se trató de implementar los threads, se recomienda hacer la tarea con 1 semana de anticipación como mínimo.

Bibliografía

Video explicativo sobre cómo hacer un web server en c.

https://www.youtube.com/watch?v=Q1bHO4VbUck

Git que crea un WebServer en C cno fork

https://gist.github.com/laobubu/d6d0e9beb934b60b2e552c2d03e1409e

Se estudian las bibliotecas del video

https://github.com/afabbro/netinet/blob/master/in.c

Forro de sockets

https://www.gta.ufrj.br/ensino/eel878/sockets/index.html

Ejemplo de GET y POST

https://stackoverflow.com/questions/22077802/simple-c-example-of-doing-an-http-post-and-consuming-the-response

Uso de hilos en python

https://www.youtube.com/watch?v=J9wOU5uWrjw&ab_channel=codigofacilito

Obtener argumentos en python

https://www.tutorialspoint.com/python3/python_command_line_arguments.htm#:~:text=Python%20provides%20a%20getopt%20module,command%2Dline%20options%20and%20arguments.&text=serves%20two%20purposes%20%E2%88%92-,sys.,list%20of%20command%2Dline%20arguments.

Descargar archivos en python

https://stackoverflow.com/questions/51528727/how-to-download-a-pdf-with-python-or-curl

Descargar archivos en C con libcurl

https://curl.haxx.se/libcurl/c/url2file.html

Compilar el código del httpclient en c

https://stackoverflow.com/questions/28165518/use-libcurl-undefined-reference-to-curl-easy-init/28165569