Universidad Politécnica Internacional

Especificación Proyecto 2: Los servidores de C

Profesor: Bryan Mena Villalobos

Introducción

Los servidores Web son el motor que mueve al internet. A día de hoy hay muchas tecnologías que permiten crear un servidor fácilmente. Sin embargo, con este proyecto buscamos aprender las bases de los servidores y experimentar con threads y procesos.

Los estudiantes en **grupos de 2 personas** implementarán varios servidores Web en el **lenguaje de programación C**.

Condiciones del Proyecto

El proyecto **DEBE** estar almacenado en un repositorio **privado** de GitHub no se aceptarán proyectos que no estén almacenados en GitHub. Además **DEBEN** añadir al profesor como colaborador al repositorio. El usuario de GitHub del profesor es mena97villalobos, cualquier duda con respecto a cómo añadirlo como colaborador la pueden evacuar en clase o por los medios de contacto provistos en la primera clase. Esto con el fin de supervisar el desarrollo del proyecto y verificar que ambos miembros del grupo estén trabajando en el proyecto.

Si necesita más detalles con respecto a cómo trabajar un proyecto con GitHub pueden contactar al profesor y él los guiará por el proceso.

¿Por qué usar GitHub?

GitHub es uno de los VCS más famosos y utilizados en la industria del software. Es una habilidad clave para poder conseguir un trabajo en desarrollo de software. Aprender a utilizar GitHub no tiene mayor complejidad y le dará a los estudiantes una oportunidad controlada para experimentar el uso de herramientas de software más allá de los objetivos del curso. Además GitHub permite que el profesor supervise el proceso de desarrollo del proyecto. Dé feedback cuando sea necesario y permite a los estudiantes tener resguardado su código en la nube.

Se recomienda el uso del sistema operativo Linux en cualquiera de sus Distros (preferiblemente Ubuntu), esto dado que la programación de forks y de thread se simplifica en el sistema Linux. Sin embargo, la elección del sistema operativo queda a discreción de los estudiantes mientras el sistema operativo sea uno entre Windows, Linux y MacOS únicamente.

DEBEN utilizar el lenguaje de programación C en alguno de sus estándares modernos. **No se revisarán proyectos escritos en C++, C#, GO, Java, Python entre otros**. Solamente C puro.

Estructura del Proyecto

En este proyecto usted desarrollará 3 tipos de servidor Web utilizando el lenguaje de programación C. Cada tipo de servidor debe estar en su propio directorio, puede tener un directorio compartido con las siguientes funcionalidades:

- Validación de argumentos
    - Su programa recibe como argumento números de puertos, debe validar estos número de puertos
- Funcionalidades de HTTP:
    - Todo tipo de funciones que manejen acciones de HTTP pueden estar en el directorio compartidos (e.g Manejo de requests http, lectura de archivos, entre otros)
- MIME Types
    - El manejo de los MIME Types lo puede hacer en este directorio compartido
- Networking
    - Inicialización del socket
    - Cualquier funcionalidad relacionada a networking

Además de esto su servidor recibirá peticiones de un cliente, estas peticiones solicitan un archivo dentro de su servidor. Los archivos los puede almacenar en un directorio llamado serverroot. Más adelante veremos el tipo de archivos que puede enviar su servidor.

Posteriormente, sus 3 servidores deben estar almacenados en directorios diferentes.

Tipos de Servidores

Usted desarrollará 3 tipos de servidores:

- **Secuencial**: Solamente tendrá un proceso, este proceso debe manejar todos los requests que le lleguen de manera secuencial
- **Threaded**: Tendrá un proceso principal que configura todo el servidor, cada vez que llegue un request usted debe crear un **thread** para manejar este request. Ese servidor puede ser desarrollado basado en el servidor secuencial, lo único que debe hacer es crear un **thread** cada vez que llega un request
- **Forked**: Tendrá un proceso principal que configura todo el servidor, cada vez que llegue un request usted debe crear un **fork** para manejar este request. Ese servidor puede ser desarrollado basado en el servidor secuencial, lo único que debe hacer es crear un **fork** cada vez que llega un request

HTTP Requests

Cuando hablamos de internet por lo general hablamos de requests HTTP, hay diferentes tipos de requests (GET, POST, PUT, DELETE, entre otros), sin embargo, para este proyecto

solamente manejará requests GET, cualquier otro tipo de request debe ignorarlo, un ejemplo de cómo ignorar requests que no sean GET puede ser el siguiente:

```
if (strcmp(request_type, "GET") == 0) {
            get_file(fd, request_path);
            close(fd);
            return 0;
        } else {
            // Assume that POST and other request
types are ignored
            fprintf(stderr, "unknown request type
\"%s\"\n", request_type);
            close(fd);
            return 1;
        }
```

En este código comparamos el request_type con el string "GET", en caso que que sean iguales vamos a procesar el request (llamada a la función get_file), en caso contrario vamos a imprimir un error en la terminal notificando que hay un error desconocido.

Cada request será una URL donde el path final será el nombre del archivo que usted desea solicitar al servidor. Puede utilizar un navegador para testear su servidor o puede utilizar alguna herramienta como Insomnia o Postman

Tipos de Archivo

Su servidor debe ser capaz de manejar todo tipo de archivos, texto plano, audio, video, pdf. Estos archivos serán almacenados en la carpeta serverroot. Si un archivo está en esta carpeta el servidor debe ser capaz de enviarlo al cliente al ser solicitado. Para esto asegúrese de conocer y de manejar los MIME Types

Interfaz

Su servidor Web debe trabajar con una interfaz de línea de comandos.
Cada servidor debe poder iniciar con el número de puerto deseado como argumento, posterior a esto debe construir un sistema de Logs, cada vez que llegue un request al servidor debe imprimir información básica del request (Nombre del archivo solicitado como mínimo)

Para cada servidor debe construir un sistema para detectar input del usuario y matar todos los threads, forks, procesos. Puede utilizar lo siguiente:

- Al pasar por el método main cree un fork
- El proceso padre espera por input del usuario (específicamente la letra q del teclado)
- El proceso hijo maneja el servidor

Cuando el usuario presione la letra q el proceso padre puede enviar una interrupción al resto de procesos para que detengan su ejecución y mueran, por ejemplo

```
void key_listener() {
    int ch = 0;
     printf("Press q to kill all threads and terminate
server\n");
    // Wait for character "q" to be pressed
    while (ch != 113) {
        ch = getchar();
    }
    kill(0, SIGUSR1);
}


void signal_handler(int signal) {
    if (signal == SIGUSR1) {
        _exit(0);
    }
}
```

Medios para la Entrega

Los estudiantes deben **crear un Release en GitHub** antes de la fecha de entrega. Ese release debe contener el código que los estudiantes desean entregar para el primer proyecto.

En el release debe poder descargarse un archivo zip con el source code, el proyecto debe contener instrucciones para compilar, linkear y correr el proyecto. Verifiquen que el código del release se pueda descargar y correr con sus instrucciones.

No se aceptará ningún otro método de entrega. Ni correo, ni links a google drive. Solamente se aceptará un release en el repositorio de cada grupo de trabajo. Compruebe que puede hacer releases en GitHub y si no está familiarizado con el proceso de crear releases contáctese con el profesor con un tiempo prudente. No el mismo día de la entrega.

¿Por qué utilizar un release de Github?

Así el profesor puede asegurarse que no modifiquen el código después de la fecha de entrega. Cada release tiene un timestamp que indica cuando fue creado. Si el release fue creado posterior a la fecha de entrega no se tomará en cuenta para la revisión. Además, crear un release en GitHub toma solamente 3 clicks y es un proceso común en la industria, lo cual brindará a los estudiantes una oportunidad más de familiarizarse con herramientas utilizadas en la industria.
En caso de tener varios releases únicamente será tomado en cuenta para la evaluación el último release.

Documentación

Los estudiantes deben añadir un README.md al repositorio de GitHub
¿Qué debe cubrir este documento?

- Información de los estudiantes, nombre completo, carnet
- Información del curso y del profesor
- Cómo compilar, linkear (en caso de ser necesario) y como correr el proyecto
- Problemas que se encontraron programando este proyecto
- Cosas que aprendieron programando este proyecto

Cuadro de Evaluación

Rubro	**Valor**
Documentación	*5 PTS*
Interfaz de Línea de Comandos	*5 PTS*
Servidor Secuencial	*30 PTS*
Servidor Threaded	*30 PTS*
Servidor Forked	*30 PTS*
Total	**100 PTS**

Fecha de Entrega

La entrega del proyecto será en la semana 15 (27 de Agosto antes de las 11:59 pm). Sin excepción.

Recuerde que este proyecto representa un 20% de su nota final

PUNTOS EXTRA + 15 PTS

En caso que los estudiantes logren completar el proyecto y quieran ganar puntos extra se deja esta sección como opcional.

Su servidor debe proveer una página Web con interfaz gráfica realizada con HTML. En esta interfaz gráfica solamente debe mostrar todos los archivos disponibles en el servidor en una lista. El usuario puede clickear en cada elemento de la lista y se le debe redireccionar a la URL de descarga.

La lista de archivos disponibles debe ser construida dinámicamente basado en los contenidos de la carpeta **serverroot.**

No utilice ningún framework de desarrollo web. Debe utilizar C puro y archivos HTML. El archivo HTML puede ser autogenerado basado en una plantilla al iniciar el proceso del servidor.

Consideraciones Puntos Extra

1. Los puntos extra pueden cubrir otros puntos que hayan perdido en el curso, esto no significa que su nota final aumentará más del 100%
2. **Puntos extra por encima del 100% no tendrán ningún valor**
3. Los puntos extra son un incentivo para los estudiantes de ir más allá, **el profesor podrá vetar grupos de estudiantes de obtener puntos extra** en los casos que considere necesario (i.e. Entregar los puntos extra cuando las demás funcionalidades básicas del proyecto no están terminadas)
4. Para aplicar a puntos extra los estudiantes **deben obtener como mínimo 70 PTS** en el resto del proyecto
5. No se enfoquen en puntos extra, dejenlo para lo último en caso que tengan tiempo. Lo más importante es el proyecto principal.