Sede Central de Cartago

Escuela de Ingeniería en Computación

Curso de Principios de Sistemas Operativos

Profesora:

Erika Marín Shumann

Estudiantes:

Brandon Dinarte 2015088894

Julian Salinas 2015114132

Proyecto: Servidor Web

Fecha de Entrega: Jueves 21 de septiembre de 2017

Tabla de Contenidos

Introducción

El presente documento detalla información relevante acerca del primer proyecto del curso de Principios de Sistemas Operativos del segundo semestre del 2017. Este proyecto consiste en el desarrollo de un servidor en lenguaje C, dicho servidor puede iniciar en diferentes modalidades de operación para la recepción de solicitudes de archivos por parte de un web browser o un cliente a desarrollarse. Las modalidades de operación son mediante una cola de proceso único, mediante múltiples procesos, mediante hilos y finalmente mediante un número máximo de hilos pre-creados.

Entre los archivos que se le pueden solicitar al servidor, existen imágenes y archivos de extensión “html” y “txt”. Por otra parte se debe desarrollar un cliente que funciona con interfaz de consola, para solicitar uno o más archivos al servidor que se defina.

El sistema fue implementado usando el lenguaje C, y su ejecución fue probada sobre la distribución Ubuntu de Linux.

Análisis Resumen de Resultados

En la versión final del proyecto, todas las funcionalidades se encuentran implementadas de forma correcta en relación a la especificación proporcionada por la profesora del curso. El cliente es capaz de conectarse al servidor que se encuentre activo en el momento, y solicitar uno o más archivos que se encuentren disponibles al servidor.

Las cuatro modalidades de inicio del servidor fueron implementadas y se encuentran con funcionalidad total, el servidor en modo cola, modo de multiprocesos, que utiliza la instrucción de fork(), el servidor que crea un nuevo hilo por cada solicitud de archivo que entre y el servidor que mantiene una cantidad de hilos creados según la definición del usuario, los cuales revisan constantemente si existen solicitudes pendientes de atender.

Casos de Prueba

La definición de las pruebas se basa en decisiones tomadas sobre la implementación de la solución para el proyecto de servidores. El primer aspecto que se toma en cuenta es el hecho de que no importa el tipo de servidor que se esté ejecutando, la atención a la solicitud del cliente siempre se hace de la misma manera, con lo cual se definen las siguientes pruebas con el fin de verificar la funcionalidad de los servidores sin preocuparse del tipo de archivo que se solicita.

Prueba # 1 – Un único archivo

Descripción: Se hace la solicitud de un único archivo cualquiera, a cada uno de los servidores desde el navegador. El archivo de prueba solicitado fue “test2.html”.

Resultados esperados: El servidor responde a la solicitud, retorna el archivo esperado y este es desplegado por el navegador.

Resultados obtenidos: Cada uno de los servidores retorna correctamente el archivo de prueba en un formato correcto por lo que el navegador lo despliega.

Puerto definido: 5000

Solicitud: http://localhost:5000/test2.html

Prueba # 2 – Múltiples solicitudes en poco tiempo (servidor FIFO)

Descripción: La idea de esta prueba es hacer múltiples solicitudes en el menor período posible de tiempo con las herramientas disponibles, usando el navegador, para analizar cómo se comporta el servidor al tener una cola de solicitudes y no poder atender todas al instante.

Resultados esperados: El servidor atiende una solicitud a la vez, según el orden en que fueron recibidas.

Resultados obtenidos: El servidor atiende una solicitud a la vez, según el orden en que fueron recibidas. Por lo tanto el navegador despliega una imagen una después de otra.

Puerto definido: 5000

Solicitud: http://localhost:5000/test3.png (La dirección se define como marcador y se abren 6 pestañas en segundo plano en sucesión lo más rápido posible)

Prueba # 3 – Múltiples solicitudes en poco tiempo (servidor de procesos)

Descripción: Misma descripción de la prueba #2.

Resultados esperados: El servidor genera un nuevo proceso por cada solicitud que recibe, por lo que no se acumulan solicitudes y el proceso se termina así mismo al final.

Resultados obtenidos: El servidor genera un nuevo proceso que atiende la solicitud recibida, por lo que el navegador despliega los resultados ya no de forma secuencial, si no de forma paralela, el proceso se mata a sí mismo al terminar.

Puerto definido: 5000

Solicitud: http://localhost:5000/test3.png (La dirección se define como marcador y se abren 6 pestañas en segundo plano en sucesión lo más rápido posible)

Prueba # 4 – Múltiples solicitudes en poco tiempo (servidor de threads)

Descripción: Misma descripción de la prueba #2.

Resultados esperados: El servidor genera un nuevo hilo por cada solicitud que recibe y el hilo se mata así mismo al terminar.

Resultados obtenidos: El servidor genera un nuevo hilo por cada solicitud, que muere al terminar, los resultados se despliegan paralelo similar a la prueba #3.

Puerto definido: 5000

Solicitud: http://localhost:5000/test3.png (La dirección se define como marcador y se abren 6 pestañas en segundo plano en sucesión lo más rápido posible)

Prueba # 5 – Múltiples solicitudes en poco tiempo (servidor de pre-threads)

Descripción: Misma descripción de la prueba #2.

Resultados esperados: Se inician los hilos al iniciar el servidor, al aceptar solicitudes, algún hilo consulta la cola de solicitudes (bloqueándola al trabajar), y atiende la solicitud eliminándola de la cola y prosigue a buscar nuevas solicitudes aceptadas.

Resultados obtenidos: Se comporta como se espera, y al sobrepasar la cantidad de hilos definidos, algunas solicitudes deben esperar un poco para ser atendidas.

Puerto definido: 5000

Solicitud: http://localhost:5000/test3.png (La dirección se define como marcador y se abren 6 pestañas en segundo plano en sucesión lo más rápido posible)

Prueba # 6 – Atención de solicitud (navegador, texto)

Descripción: Se hace la solicitud de un único archivo de extensión “txt” para comprobar su correcto envío y despliegue en el navegador. El tipo de servidor que atiende la solicitud no es relevante pues todos hacen el envío de archivos de la misma manera.

Resultados esperados: El navegador despliega el contenido del archivo solicitado.

Resultados obtenidos: El navegador despliega correctamente el archivo de texto.

Puerto definido: 5000

Solicitud: http://localhost:5000/test1.txt

Prueba # 7 – Atención de solicitud (navegador, html)

Descripción: Se hace la solicitud de un único archivo de extensión “html” para comprobar su correcto envío y despliegue en el navegador. Similar a la prueba #6.

Resultados esperados: El navegador despliega el contenido del archivo solicitado.

Resultados obtenidos: El navegador despliega correctamente en contenido web del archivo.

Puerto definido: 5000

Solicitud: http://localhost:5000/test2.html

Prueba # 8 – Atención de solicitud (navegador, imagen)

Descripción: Se hace la solicitud de un único archivo de extensión “png” para comprobar su correcto envío y despliegue en el navegador. Similar a la prueba #6.

Resultados esperados: El navegador despliega el contenido del archivo solicitado.

Resultados obtenidos: El navegador despliega correctamente la imagen solicitada.

Puerto definido: 5000

Solicitud: http://localhost:5000/test3.png

Prueba # 9 – Atención de solicitud (cliente)

Descripción: Se hace la solicitud de los tres archivos solicitados en las pruebas anteriores al mismo tiempo, desde el cliente implementado en consola. El servidor específico no es relevante pues todos reciben las solicitudes de forma igual.

Resultados esperados: Los tres archivos son recibidos y almacenados en la dirección especificada por el usuario.

Resultados obtenidos: Se genera un hilo por cada solicitud, que recibe la respuesta del servidor y se almacena cada uno de los archivos solicitados en un principio.

Puerto definido: 5000

Solicitud: ./C_Servers -c localhost 5000 /Desktop/ “test1.txt”,”test2.html”,”test3.png”

Evidencias del funcionamiento

Como parte de las pruebas, se adjuntan algunas capturas de pantalla relevantes para evidenciar el correcto funcionamiento del sistema. En la evidencia #1 se muestra una solicitud de un archivo existente desde el navegador web (Mozilla Firefox), y la información que genera en el lado del servidor. Para la evidencia #2 se utiliza el cliente para solicitar dos archivos, uno que existe y uno que no existe ("archivoInexistente.txt"), de manera que en el cliente se obtenga retroalimentación sobre el error para encontrar el archivo solicitado. Y finalmente en la evidencia #3 se muestra el resultado que despliega el navegador ante la ausencia del archivo solicitado ("archivoInexistente.txt").

Evidencia #1

Tipo de servidor: Threads

Fuente: Browser

Solicitud: http://localhost:5000/test3.png

Información desplegada en el servidor:

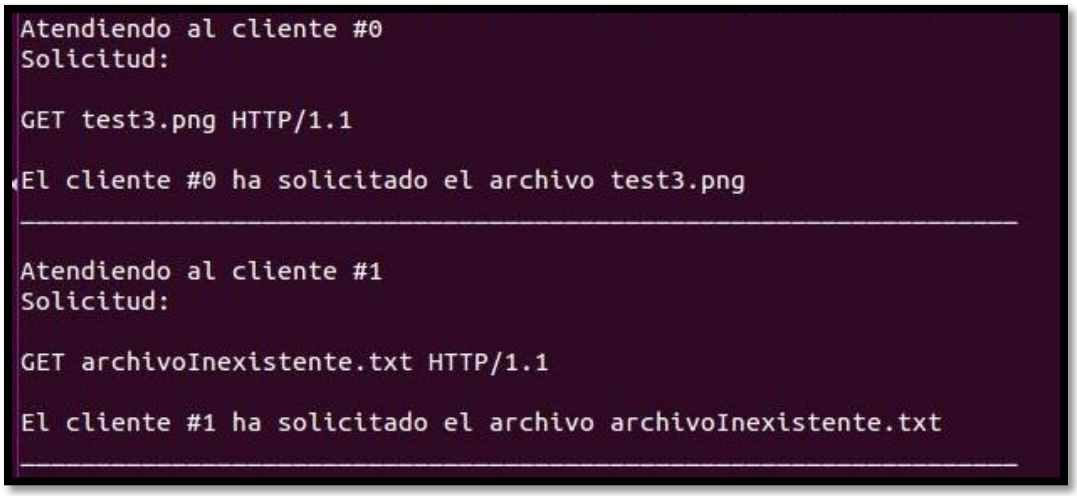
Evidencia #2

Tipo de servidor: FIFO

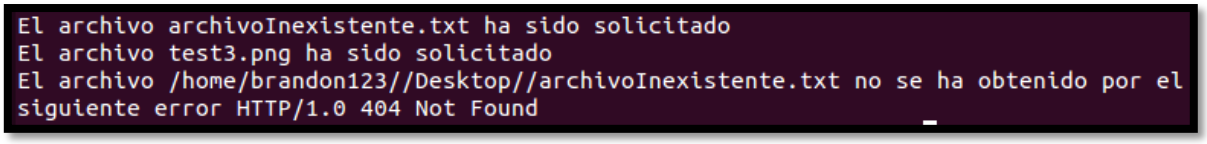
Fuente: Cliente

Solicitud: ./C_Servers -c localhost 5000 /Desktop/ “test3.png”,”archivoInexistente.txt”

Información desplegada en el servidor:



Información desplegada en el cliente:

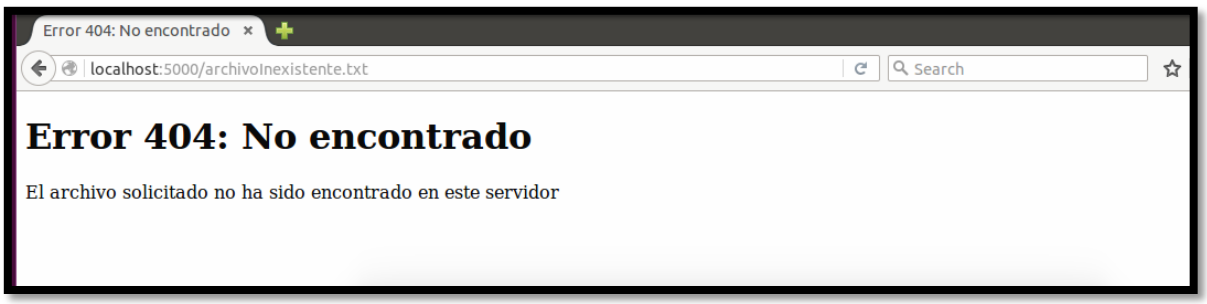


Evidencia #3

Tipo de servidor: Procesos

Solicitud: localhost:5000/archivoInexistente.txt

Información desplegada en el navegador:



Experiencias Obtenidas

En general la implementación de los métodos de operación de los servidores es muy sencilla una vez que se han definido los protocolos de comunicación, dichos protocolos son los que presentan más complejidad, desde la formulación de la solicitud de parte del cliente, hasta el procesamiento de la respuesta obtenida desde el servidor para generar el archivo a guardarse o desplegarse.

En términos de lo servidores el que presenta una ligera mayor complejidad es el que opera con hilos pre-cargados, esto puesto que junto con el manejo de hilos, requiere de la implementación de un mutex sencillo, que sirve para bloquear la zona crítica de código en la que los hilos revisan la cola de solicitudes recibidas por el servidor para atenderlas en caso de existir. En el caso de los demás servidores, la implementación es muy sencilla y sin contratiempos, una vez que se define una forma general a los cuatro para iniciar el socket, la función que atiende solicitudes sin importar el tipo, la función de envío de la respuesta, entre otros.

Mejor opción de servidor

Según la totalidad de pruebas realizadas al finalizar el proyecto, se pueden observar varios aspectos en relación a cada servidor. En el caso del servidor FIFO, es claro que al realizar gran cantidad de peticiones en sucesión, requiere de más tiempo para brindar todas las respuestas necesarias pues no divide la carga de trabajo y la ejecuta de forma secuencial.

Con respecto al servidor que crea un proceso por solicitud para su atención y al que crea un hilo por solicitud, es difícil notar las diferencias de rendimiento entre ambos, aunque está claro que ambos tienen el aspecto negativo del tiempo perdido por los procedimientos constantes de creación de nuevos procesos o hilos. Aún así con base a la simplicidad de crear un nuevo proceso en comparación a la creación de un nuevo hilo, podría decirse que los procesos resultan más sencillos de implementar.

Finalmente, con el servidor que atiende solicitudes mediante hilos previamente cargados, parece obtenerse el mejor rendimiento pues se evita la constante creación de hilos, claro está, aparece un nuevo costo de procesamiento para que los hilos estén revisando la cola de solicitudes que han sido aceptadas por el servidor, pero aparenta ser menor al costo de la creación de nuevos hilos. También hay que tener en cuenta que por el ambiente en el que se realizan las pruebas, pocos hilos pre-creados son suficientes para atender todas las solicitudes al instante prácticamente, en un ambiente real, se necesitarían muchos más hilos pre-creados para obtener un tiempo de respuesta aceptable por parte del servidor, pero aún en ese caso la opción seguiría siendo el servidor de hilos pre-creados, solo que con una mayor cantidad de hilos disponibles.

Compilación

El proceso para la compilación del proyecto requiere de los siguientes aspectos.

1. Para compilar el proyecto es necesario contar con make y cmake 3.7 como mínimo, de lo contrario será necesario abrir una terminal con Ctrl + Alt + T y ejecutar los siguientes comandos para realizar su instalación:
    a. sudo apt-get install make && sudo apt-get install cmake
    b. Nota: Estos programas están disponibles por defecto en el Ubuntu 17.10.

2. Una vez obtenidas estas dependencias, es necesario abrir una terminal ubicada dentro del proyecto. En esta colocamos el siguiente comando para crear el ejecutable en un carpeta llamada Bin:
    a. mkdir Bin && cd Bin && cmake .. && make
    b. Nota: El proyecto ya contiene la carpeta Bin, por tanto, para ejecutar el programa solamente es necesario accederla.

Ejecución

Consideraciones a tomar en cuenta.

- Se definió el archivo Readme.md que contiene información detallada de los comandos para la ejecución del proyecto.
- Los archivos considerados disponibles se encuentran en la carpeta Files.
- Los pasos para una ejecución normal son los siguientes.

1. Mediante la consola de comandos, iniciar el ejecutable “C_Servers” en modo servidor junto con el puerto deseado. El primer parámetro define el modo de servidor.
    a. –f: FIFO, -k: Procesos, -t: Threads, -p: Pre-Threads
    b. Ejemplo -> ./C_Servers -f 5000

2. Para hacer la solicitud de archivos se debe iniciar el navegador web o el cliente.
    a. Para iniciar el cliente, en consola ejecutar:
        i. ./C_Server -c localhost 5000 "test1.txt","test2.html"
    b. Si se ejecuta el navegador, ingresar en la barra de URL la dirección:
        i. localhost:<puerto>/archivo.extensión
        ii. Ejemplo -> localhost:5000/test1.txt

Librerías de hilos

La biblioteca de hilos más común es la de <pthread>, misma que fue utilizada para la realización del proyecto. Pthread parece ser la utilizada de forma más común, puesto que cumple con el estándar de C POSIX.

También existe la implementación de C del Proyecto GNU, “GNU C Library”. Dicha librería cuenta con las funciones necesarias para la implementación de hilos en un nivel muy similar a las proporcionadas por pthread.

Efecto de la plataforma

Según la experiencia obtenida durante el desarrollo del proyecto se puede establecer que las implementaciones de las librerías para el manejo de hilos varían entre diferentes plataformas, aún cuando “pthread” se encuentra implementado para Windows y para Linux, existen algunas diferencias que no permiten la compatibilidad total entre código escrito para una plataforma con respecto a la otra. También hay que destacar que según la plataforma aparecen diferentes funciones en las librerías estándar de C, lo cual es de esperarse pues es común de ver.

Conclusiones

- Según la experiencia con los diferentes modos de ejecución del servidor, el modo de pre-thread es el que brinda mejores resultados, al menos en el ambiente en el que se probó para el proyecto, en el que las solicitudes son pocas. Puesto que elimina la constante creación de nuevos hilos o procesos, al mismo tiempo que permite la división de trabajo.
- Se resaltó la importancia del manejo de zonas críticas cuando existen recursos compartidos, pues resultó imperativo el controlar y bloquear la lista de solicitudes aceptadas por el servidor, para que los diferentes hilos no respondieran al mismo cliente.