Universidad Nacional ARTURO JAURETCHE

Sistemas Distribuidos

Laboratorio 2
Sockets
Threads – OpenMP V2.1

Objetivos

Los objetivos de este laboratorio son:

- Familiarizarse con el uso de las librerias Pthread y OpenMP de entornos UNIX para procesamiento concurrente y paralelo para multiprocesamiento distribuido
- Familiarizarse con el uso de Sockets y diseño de clentes servidor con multi hilo.
- Desarrollo de aplicaciones peer to peer
- Tecnicas de comunicación por TCP/IP como UDP. A nivel middleware

Breve descripción

Se utilizarán técnicas de cliente servidor con hilos para el desarrollo de interacción de un servidor y multiples clientes..

Herramientas

Para la realización del laboratorio se utilizarán varios ejemplos de código, que se describen en las secciones siguientes y una imagen de linux con la que se realizó la primer práctica:

A continuación se describen los datos necesarios para su operación:
La imagen de linux es en formato OVA (para ser utilizada en VirtualBox 3.0 o superior)

Imagen de linux:

Versión de linux Debian 10 sin entorno grafico.
Usuario: linuxtest
password: 123456
Usuario: root
password: 123456

Carpetas compartidas

En virtualbox, definir como carpeta compartida el sitio en donde se guardarán los fuentes en la máquina host. El nombre de la carpeta es indistinto, pero cuando se realice la configuración de VB, para carpetas compartidas, el recurso se debe llamar “share”

script de montaje: mount.sh
carpeta de montaje /home/linuxtest/mnt
recurso compartido host: share

Librerías instaladas

Librerias necesarias: gcc, gcc-multilib, pthread, open-mp, sockets

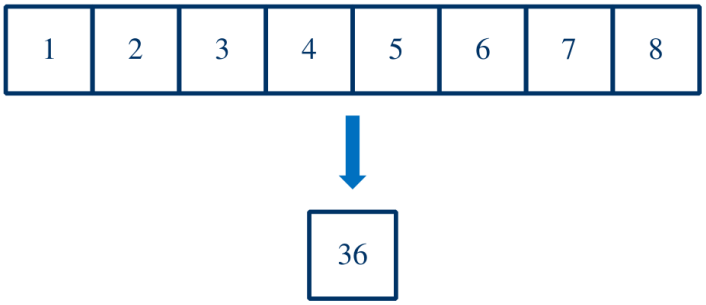
Ejercicios de ejemplo

Para realizar esta práctica se proveen cuatro ejercicios de ejemplo:

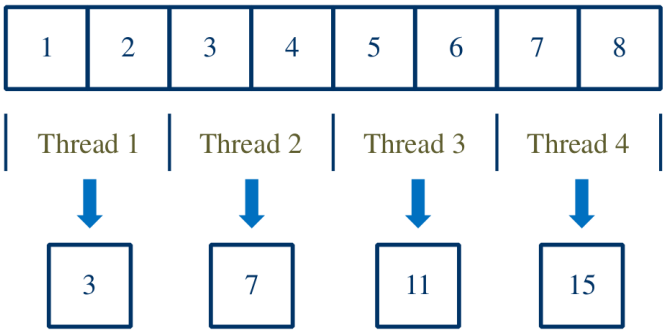
pthread.c

Describe un ejemplo de paralelización de la suma de un vector mediante la librería Pthread.

Un número de threads (numWorkers) debe sumar los elementos de un vector global con size elementos.



Cada thread se ocupa de sumar una porción del vector. Por ejemplo, con 4 threads:



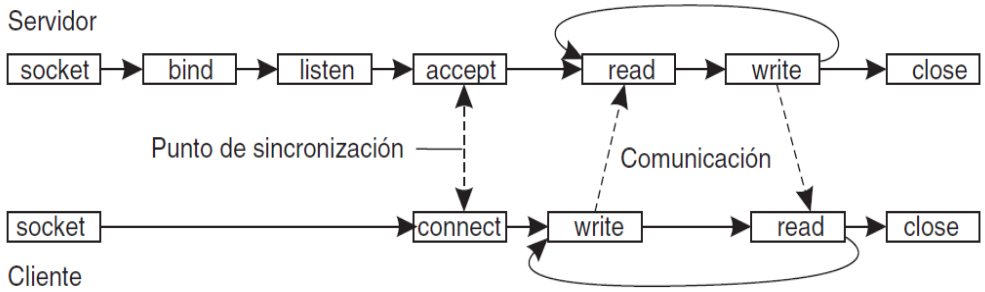
Linea de compilación: gcc pthread.c -o pthread -lpthread

Ejecución: ./pthread [size][workers]

Ejemplo de cliente servidor multi-hilo

Como se vió en clase los sockets pueden tener los siguientes estados o primitivas:

Primitiva	Significado
Socket	Crea un nuevo punto final de comunicación
Bind	Asocia una dirección local a un socket
Listen	Anuncia la conveniencia de aceptar conexiones
Accept	Bloquea a quien llama hasta que llega una petición de conexión
Connect	Activa el intento de establecer una conexión
Send	Envía algunos datos a través de la conexión
Receive	Recibe algunos datos a través de la conexión
Close	Libera la conexión



Ejemplo básico

Compilación:

gcc socket_server.c -o server -lnsl -pthread

gcc socket_client.c -o client -lnsl -pthread

Se debe ejecutar primero el servidor y luego clientes.

En consolas separadas de la misma máquina virtual (con alt+-F1, alt+-F2, alt+-F3, alt+-F4 para moverse entre consolas) y ejecutar en cada una:

Ejecución:

$./server

$./client

Ejemplo de chat cliente servidor.

El siguiente ejemplo de código es un cliente- servidor multi hilo en donde el servidor es un servidor de chat y los clientes son usuarios. El servidor recibe los mensajes de cada usuario y lo reenvía a cada cliente. La característica particular de esta aplicación es que el servidor soporta multiples clientes.

El código fuente de los aplicativos se pueden descargar del siguiente link:

https://github.com/lovenery/c-chatroom

Para compilarlo se debe descomprimir el archivo en una carpeta, y luego ejecutar make (nota: debe estar instalado el paquete “make”).

Compilación:

$make

Con esto se ejecuta el Makefile que tiene las reglas de compilación adecuadas.

Luego se ejecuta en una consola

$./server.out

Y abrir dos consolas mas (con alt+-F1, alt+-F2, alt+-F3, alt+-F4 para moverse entre consolas) y ejecutar en cada una:

$./client.out

Le pedirá logue y luego accederá al chat. Podrá ver en el servidor los logs de los buffer e intercambio de información.

Para este ejercicio lo que se observa es que cada cliente genera dos hilos, uno para transmitir y otro para recibir. Mientras que el servidor genera un hilo por cada cliente que se conecta.

Notese que tanto en el ejemplo anterior como en este ultimo la dirección IP de los clientes servidores es localhost o lo que es lo mismo 127.0.0.1 (definido en /etc/hosts)

Ejemplo OpenMP

Para OpenMP se tienen dos ejemplos:

ej1openmp.c

Generacion de varios threads y modificacion del numero de threads mediante:

export OMP_NUM_THREADS=4

Compilación: gcc ej1openmp.c -o ej1openmp -fopenmp

Ejecución: $./ej1openmp

ej2openmp.c

Es el mismo del ejemplo “pthread” pero con OpenMP.

Compilación: gcc ej2openmp.c -o ej2openmp -fopenmp

Ejecución: $./ej2openmp [size][workers]

Tareas a realizar por el alumno

1. Se deben ejecutar/ compilar y verificar funcionamiento de todos los ejemplos entregados para este laboratorio.

- Para los casos de pthread y open-mp se ejecutarán en una sola máquina.
- Para los casos de los clientes servidores, se debe utilizar como servidor la máquina que se habia definido como master y como clientes las máquinas nodos. (Tener en cuenta que se deben adaptar los códigos dado que están preparados para correr como localhost)

2. Se debe reescribir el código del c-chatroom con open-mp.
3. Desarrollar un sistema de chat del tipo peer 2 peer estructurado, de tal forma que:
    - El servidor se utilice solo como vinculo inicial entre los clientes.
    - La comunicación sea punto a punto entre clientes.
    - Los clientes pasen a ser clientes / servidores al mismo tiempo.
    - La comunicación entre clientes sea UDP y con chekeo por checksum
    - Cuando un cliente se conecta al servidor, este espera hasta visualizar que cliente está conectado, luego cuando uno de los clientes solicite la comunicación con otro, el servidor los vincula y solo mantiene estado de conexión de los clientes por si aparece algún tercero que quiera contactarlos

Calificación

La calificación va de 0 a 8 para los primeros 2 ejercicios

Para obtener una calificación superior a 8 se debe realizar el ejercicio 3.

Formato de entrega

- Se debe entregar informe que contenga.
    - Carátula con nombre y apellido del alumno, fecha y título del trabajo práctico.
    - Descripción de realización de cada punto de ser necesario con capturas de pantalla de los resultados.
    - Conclusiones
- La entrega es individual, en formato PDF y se debe adjuntar un paquete comprimido con los códigos realizados (comentados).