Backlog: Es una cola de conexiones entrantes en espera.

Tarea: Investigar el valor máximo que puede tomar el backlog

Opciones de socket

SO\_TIMEOUT:

SO\_REUSEADDR: Socket reuse address

SO\_KEEPALIVE: Se da un aviso cuando se ha cerrado la conexión del otro lado

SO\_OOBINLINE:

SO\_LINGER: Fija un temporizador de espera.

Si queremos mandar diferentes tipos de datos podemos usar el objeto DataOutputStream el cual es un flujo de bytes

**Sockets de datagrama**

|  |  |
| --- | --- |
| 16 bits | 16 bits |
| Puerto origen | Puerto destino |
| Longitud | Checksum |

Un paquete de datagrama

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Datagram packet   |  |  | | --- | --- | | InetAddress | Dirección | | Int | Puerto | | byte[] datos | | |

Con los sockets de datagrama podemos tener conexiones de tipo

Cliente 🡨🡪Cliente

Cliente 🡨🡪Servidor

Servidor 🡨🡪Servidor

El método void connect(InetAddress addresss, int port); existe para filtrar nuestro datagrama

Para que nuestro servidor pueda nuevamente recibir datagramas de quien sea se usa el método void disconnect();

La parte no bloqueante de un socket de datagrama utilizamos el método DatagramChannel getChannel(); Con esto podemos intercambiar entre la parte bloqueante y la no bloqueante.

El método bloolean isBound() nos dice si ya está ligado a una aplicación el socket

Estructura de la dirección unicast

|  |  |
| --- | --- |
| Id red | Id host |

Hacemos una and con la máscara de red y podemos saber el segmento de red donde se encuentra una máquina

¿Por qué no podemos conectarnos por ejemplo a una dirección 230.1.1.1?

Al conectarnos con un socket a otra máquina el DHCP obtiene la IP, la máscara, puerto de enlace y DNS.

Con la IP hace una operación AND para conocer el segmento de red en el que se encuentra la máquina, por ejemplo tenemos una máquina con una IP 10.0.0.1 y otra con una IP 20.0.0.1

Hacemos una operación AND 10.0.0.1 & 255.0.0.0 = 10.0.0.0 es el segmento de red donde estoy, posteriormente se hace lo mismo con la otra máquina usando MI máscara de red ya que es la única que conozco 20.0.0.1 & 255.0.0.0 = 20.0.0.0 como están en segmentos diferentes se envía la dirección al puerto de enlace.

Esto que se ha descrito solo ocurre con las estructuras unicast la estructura multicast es diferente

|  |
| --- |
| Id grupo |

El protocolo que debemos a usar es IGMP o protocolo de gestión de grupos de Internet

Se usa para escuchar datos pesados por ejemplo vídeo.

La estructura de un mensaje IGMP es la siguiente

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo | Temporizador | Checksum | ID grupo |

En el tipo tenemos por ejemplo Consultas cuyo valor es 0x11 o 17 en base 10

Reporte 0x12 o 18 en base 10 esto para IGMPv1

Reporte 0x16 o 22 en base 10 esto para IGMPv2

Reporte 0x22 o 34 en base 10 esto para IGMPv3

Valor más alto del TTL = 255

sudo ifconfig lo inet6 add 2001::1234:1/64 <- con esto configuramos nuestra tarjeta de red en localhost y añadimos la dirección IPv6CM

0.0.0.0 spclient.wg.spotify.com

sudo nano /etc/hosts

sudo service networking restart

ping6 ejemplo

Configuramos una DNS para nuestra máquina

SOCK\_STREAM -> IPPROTO\_TCP;

COSK\_DGRAM -> IP\_PROTO\_UDP

SOCK\_RAW

Datos numéricos se deben poner primero en formato de red antes de ser enviados, también una vez recibidos y antes de ser utilizados se deben poner en formato host

htons(short x); ntohs(short x);

htonl(long x); ntohl(long x);

Para ejecutar usamos los comandos

Programa 127.0.0.1 1234 “Un mensaje”

**TAREAS:**

* **Investigar longitud Backlog**
* **Echo java multimensaje - datagrama**
* **Multiarchivos java**
* **Agenda Java**
* **Ahorcado**

**Variables de condición**

Las usamos pata bloquear un hilo

Se utilizan junto con mutex

Condition es una interfaz para instanciar usamos el método newCondition de las clases ReentrantLock/ReadWriteLocj

El método signal “desbloquea” un hilo (El último)

Volatile es un modificador de acceso = “TOMA EL VALOR MÁS FRESCO”

**Semáforos:**

Supongamos que tenemos varios hilos y tenemos una sección que es crítica, entonces hay que controlar los hilos para que no ocurra alguna inconsistencia, con los semáforos podemos limitar el número de hilos que vamos a usar en ese bloque. Con los semáforos podemos hacer albercas de hilos

**Pipes:**

Usamos pipes para evitar crear muchísimos hilos, ya que cada hilo tiene su propia pila, que es una región en memoria

Para usarlos en C usamos

Pthread.h para compilar usamos

Gcc -pthread miprograma.c -o salida

int pthread\_create( identificador, tabla\_atributos”PTHREAD\_SCOPE\_PROCESS solo llega a el proceso, si es PTHREAD\_SCOPE\_SYSTEM está a nivel kernel el hilo”, apuntador\_a\_una\_función, si requieren parámetros (Solo uno, si queremos más usamos un struct) );

pthread\_equal(); Compara hilos

pthread\_drtach(); Vuelve independiente a un hilo

pthread\_join() bloquea el cuerpo del main mientras se ejecuta el cuerpo de un hilo, esta se invoca desde otro hilo

void pthread\_exit(); termina un hilo. Desde el mismo hilo

int pthread\_cancel(); evita que se ejecute un hilo pero no retorna nada

**MUTEX en C**

Debemos crear una variable tipo pthread\_mutex\_t podemos inicializar de forma estática

Si queremos hacerlo de forma dinámica usamos pthread\_mutex\_init(); donde el segundo parámetro es la prioridad del mutex

Para entrar a un bloque sincronizado usamos pthread\_mutex\_lock();

Si usamos pthreadmutex\_trylock, trata de bloquear el candado de acceso, si lo hizo bien retorna 1

Para quitar el candado pthread\_mutex\_unlock()

PARA PRIORIDAD

PTHREAD\_PRIO\_NONE

PTHREAD\_PRIO\_INHERIT: Más alta prioridad de los hilos en espera

PTHREAD\_PRIO\_PROTECT: Prioridad más alta de los hilos o mutex

Con el método pthread\_mutexattr\_settype()

PTHREAD\_MUTEX\_NORMAL: no detenta dead locks

PTHREAD\_MUTEX\_ERRORCHECK: verifica errores, por ejemplo que se intente bloquear algo bloqueado

PTHREAD\_MUTEX\_RECURSIVE: por ejemplo se desbloquean varios mutex para que posteriormente varios hilos puedan bloquear varios mutex

**SEMÁFOROS:**

Los utilizamos cuando más de un hilo va a entrar a la sección crítica.

La biblioteca a usar semaphore.h

El tipo de dato es sem\_t s:

Sem\_wait toma un permiso del semáforo si lo tomó se decrementa en una unidad. Si no hay permisos retorna -1 y se bloquea el hilo

Sem\_trywait lo mismo que sem\_wait pero sin el bloqueo

Sem\_post libera un permiso

Sem\_getvalue retorna los permisos restantes (Esto es cuando no tienen nombre los semáforo)

Para cuando son con nombre usamos sem\_open

**PIPES:**

Así os hilos pueden acceder al descriptor de archivos, y compartir variables ya que cada hilo tiene su propia pila. Una tubería con nombre es persistente, existe mientras no se elimine la tubería, además multiples hilos pueden acceder.

Sin nombre: Usamos unistd.h

Con nombre: <sys/types.h> y <sys/Stat.h>

Para los permisos

4 2 1

R W X

(1 1 1)2

Default permission: 777 (yo/grupo/otros)

Para abrir usamos la función open(nombre, bandera “modo”);

**SELECTORES**

Tenemos una variable tipo fd\_set la cual simboliza un conjunto, que almacena? Descriptores de I/O son descriptores de lo que se.

Tiene dos campos tv\_sec y tv\_unsec

**Sockets bloqueantes**

Para Java, tenemos la librería java.nio la cual tiene una clase abstracta Buffer la cual tiene clases hijas como ByteBuffer, CharBuffer, IntBuffer, y así para cada tipo primitivo de datos|