Sistemas Operativos
2017/2018

Práctica 2

Comunicaciones en red
Y
Concurrencia

“COMUNICACIONES EN RED”

Implementar una aplicación cliente-servidor mediante la utilización de sockets cuya funcionalidad será la transmisión de un archivo desde el servidor al cliente.

El ejercicio constará de dos partes (procesos):

- El servidor. Proceso que estará esperando a que le lleguen conexiones desde el cliente y, una vez establecida la conexión, transferirá al cliente el contenido de un archivo denominado “Google.html”. El contenido del archivo será el código fuente de la página web de Google.
- El cliente. Proceso que establecerá la conexión con el servidor, recibirá el contenido del archivo y lo mostrará por pantalla.

El cliente se ejecutará en la máquina local y se invocará con la orden:

Cliente IP_Servidor

El servidor se lanzará con la orden **Servidor** y debe estar en todo momento escuchando por el puerto 9999, preparado para aceptar conexiones. Tras una petición de conexión por parte de un cliente, debe crear un hijo que será el encargado de realizar todas las operaciones necesarias para la transferencia del archivo.

La puntuación del ejercicio será la siguiente:

- Implementación correcta del cliente y su estructura. **(1,5 puntos)**
- Implementación correcta del servidor y su estructura. **(1,5 puntos)**
- Transferencia del archivo. **(3 puntos)**

LLAMADAS AL SISTEMA RELACIONADAS CON LAS COMUNICACIONES EN RED

Estructuras necesarias

<sys/socket.h>, **struct sockaddr**.
<netinet/in.h>, **struct in_addr; struct sockaddr_in** (incluye un campo in_addr).
<sys/types.h>

Llamadas al sistema principales

socket
Permite establecer el canal de comunicación, está definida en <sys/socket.h> y necesita <sys/types.h>.

Ejemplo:
conector = socket(AF_INET,SOCK_STREAM,0) //(Familia protocolos, tipo transporte, protocol).

bind
Se emplea para comunicar a la red la dirección del canal. Utiliza como parámetro la estructura sockaddr_in definida en el archivo de cabecera <netinet/in.h>

Bind(conector, addr, addrlen) //(descriptor del conector, dirección servidor, longitud dir)

listen
Disponibilidad para recibir peticiones de servicio.

listen(conector, tamaño_cola) // (descriptor del conector, cola peticiones pendientes)

accept
Bloquea al servidor en espera de peticiones de conexión por parte de los clientes.

accept(conector, addr, addrlen) ///(descriptor del conector, dir. servidor, longitud dir)

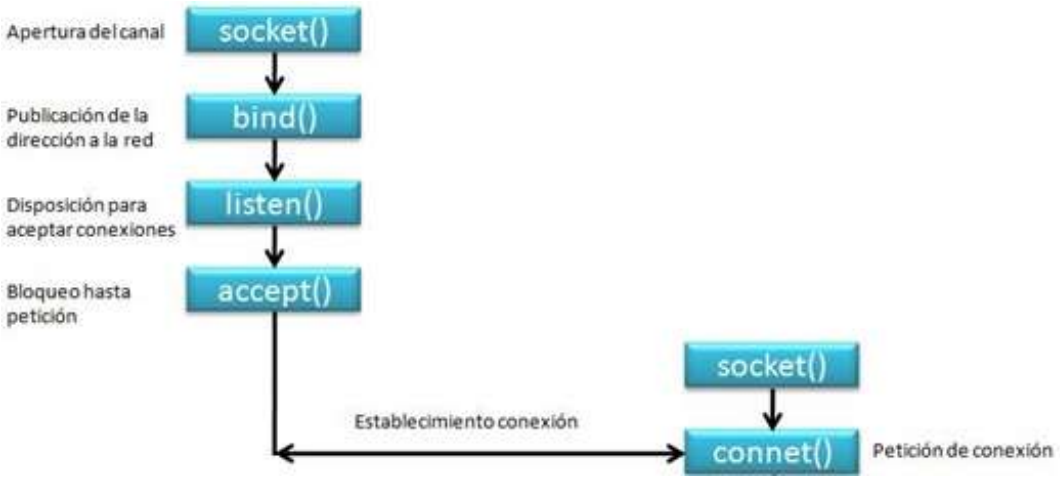
connect
Es invocada por el proceso cliente para establecer una conexión, el servidor debe estar esperando.

connect(conector_cliente, addr, addrlen) // (descriptor conector, dirección servidor, longitud)

Funciones de ayuda
inet_addr(cadena) //traduce una cadena de caracteres en dirección IP.
inet_ntoa(in_addr) //conversión inversa.

htonl, htons, ntohl, ntohs. Convierten datos unsigned long y unsigned short del formato de host al formato de la red y viceversa.

Estructura de toda aplicación cliente-servidor



“CONCURRENCIA”

Sincronización de procesos. Semáforos. Diseñar un programa de concurrencia mediante el entorno JBACI.

http://code.google.com/p/jbaci/

Enunciado

Implementar mediante semáforos el problema del productor consumidor con buffer limitado estudiado en clase de teoría.

Notas

Para la corrección y calificación del ejercicio además de la corrección del programa se valorarán los siguientes aspectos:

- Uso correcto de los semáforos atendiendo a los dos principales aspectos de exclusión mutua y sincronización
- Uso del interfaz gráfico para la claridad del problema

La puntuación del ejercicio será de **4 puntos.**

FECHA DE ENTREGA: **Semana del 13 al 17 de noviembre**