Vidal Sánchez José Antonio **Código:213496757**

Programación **Actividad 4**

Descripción de la práctica

Título de la actividad: Servidor TCP no bloqueante
Objetivo: Utilizar lecturas no bloqueantes para evitar bloqueos en el servidor.
Fecha de entrega:Semana Y

Problemática

Varias llamadas a función realizadas por el servidor son del tipo “bloqueante”. Una llamada bloqueante es aquella que no regresa hasta que haya cumplido su labor. En el servidor TCP la llamada a accept no retornará hasta que llegue un cliente nuevo. Si ningún cliente se conecta, la aplicación se quedará detenida indefinidamente hasta que llegue un cliente.

En el caso de un servidor, esto no puede ser lo deseado, puesto que tiene que atender otras funciones y/o otros clientes.

Una de las técnicas para lograr esto es un descriptor de archivo no-bloqueante. Un descriptor de archivo no-bloqueante causa el efecto de que cualquier función bloqueante, en vez de bloquear, retorna inmediatamente con un código de error (generalmente bueno).

Detalles técnicos

Para esta entrega, investiga la función fcntl para aplicar banderas no-bloqueantes (O_NONBLOCK) al descriptor de archivo.

Aplica los atributos de no-bloqueante al socket después de asociar el puerto con bind y antes de ejecutar accept. Revisa el código de error regresado por accept, y cuando no haya un cliente nuevo has que el servidor imprima un punto “.” y descanse por 500 milisegundos, luego tiene que volver a revisar con accept y si aún no llega algún cliente repetir de forma indefinida.

Se sugiere utilizar la función gettimeofday para generar el retraso de 500 milisegundos.

Entregables

Deberás entregar el código fuente (archivo .c o .cpp) junto con un reporte con las dificultades técnicas para realizar la práctica.

Ambos archivos comprimidos en un ZIP. Estrictamente prohibido utilizar formato rar.

Notas/Dificultades en la realización de la práctica

La mayor dificultad con la que me encontré en la realización de esta práctica fue el de la cuenta de tiempo. Revisando en internet, encontré que además de la función sugerida, existen las funciones *time(), clock(),* y aquellas contenidas en la librería *chrono*. Al final, me decidí por usar la sugerida *gettimeofday()* , debido a la unidad de tiempo que maneja (segundos+microsegundos) , además que de momento no estoy familiarizado con las librerías de c++11 .

En cuanto al uso de *fcntl* , en un principio simplemente añadí la bandera NON_BLOCK, en lugar de usar el or a nivel de bits. Corregí este error una vez que vi la razón para usarlo, la cual es respetar las banderas ya puestas. Considero que sería útil una función para automatizar este proceso; es decir, que tome la bandera la añada a las existentes y las vuelva a fijar con *fcntl*.