# I OBJETIVO

Conocer y utilizar los sockets como uno de los mecanismos utilizados para la comunicación entre procesos en un ambiente distribuido

# II BIBLIOGRAFÍA


- Sistemas Distribuidos, Coulouris, Dellimore, Kindberg, Prentice Hall
- Beginniing Linux Programming, Neil Matthew & Richard Stones, Wrox Press
- UNIX Programación Práctica, Key A. Robbins & Steven Robbins, Prentice Hall


# III RECURSOS

- Una estación de trabajo con UNIX/Linux o una conexión remota.
- Un editor de texto
- El compilador de C de UNIX/Linux
- Puede utilizarse Windows para parte de la práctica

# IV PREREQUISITOS

- Uso del sistema operativo Linux
- Lenguaje de programación C/C++
- Hilos Posix y/o otra API
- Puede utilizarse Windows y otro lenguaje que no sea C para parte de la práctica

# V DESCRIPCIÓN DE LA PRÁCTICA

## *1 Un proceso cliente y un servidor locales*

En los ejemplos que se muestran en la Figura 1 y Figura 2, tenemos los ejemplos de un cliente y servidor. El cliente envía un carácter al servidor, este lo incrementa y lo devuelve al cliente.

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
```

```
int main()

{

    int sockfd;
    int len;
    struct sockaddr_in address;
    int result;
    char ch = 'A';



/* Crear un socket para el cliente */

    sockfd = socket(AF_INET, SOCK_STREAM, 0);



/* Nombrar el socket, de acuerdo con el server */

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = inet_addr("127.0.0.1");
    address.sin_port = htons(9734);
    len = sizeof(address);
    result = connect(sockfd, (struct sockaddr *)&address, len);

    if(result == -1) {
        perror("oops: client1");
        exit(1);
    }

    write(sockfd, &ch, 1);
    read(sockfd, &ch, 1);
    printf("char from server = %c\n", ch);
    close(sockfd);
    exit(0);
}
```

**Figura 1. Un programa cliente que se conecta a un servidor local**

```
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>

int main()
{
    int server_sockfd, client_sockfd;
    int server_len, client_len;
```

```
    struct sockaddr_in server_address;
    struct sockaddr_in client_address;


/* Crear un socket sin nombre para el server */

    server_sockfd = socket(AF_INET, SOCK_STREAM, 0);



/* Nombrar el socket */

    server_address.sin_family = AF_INET;
    server_address.sin_addr.s_addr = htonl(INADDR_ANY);
    server_address.sin_port = htons(9734);
    server_len = sizeof(server_address);
    bind(server_sockfd, (struct sockaddr *)&server_address, server_len);
    listen(server_sockfd, 5);
    while(1) {
        char ch;
        printf("server waiting\n");
        client_sockfd = accept(server_sockfd,
        (struct sockaddr *)&client_address, &client_len);
        read(client_sockfd, &ch, 1);
        ch++;
        write(client_sockfd, &ch, 1);
        close(client_sockfd);
    }
}
```

**Figura 2.- Un programa servidor que espera conexiones de un cliente local**

## 1.1 Preguntas

¿Qué significa que el comportamiento en el envío o recepción de un mensaje sea bloqueante o no bloqueante?

¿El comportamiento de las primitivas `send` y `receive` que en este caso corresponden a `write` y `read`, tienen un comportamiento bloqueante o no bloqueante?

¿Es posible que usando sockets se pueda cambiar el comportamiento en el envío o recepción de mensajes a bloqueante o a no bloqueante?, ¿Cuál es el comportamiento establecido por default?

¿Qué sucedería si el cliente o el servidor no tuvieran la función `htons()` al especificar el número de puerto?

Si se modifica el cliente y el servidor de manera que en vez de ser un protocolo de flujo (`SOCK_STREAM` en la llamada `socket`) sea un protocolo de datagramas `SOCK_DGRAM` (en la llamada `socket`), ¿funciona igual?, ¿qué sucede?, explique el por que.

# 2 Un proceso cliente y el servidor remoto

Modifique los programas de la Figura 1 y Figura 2, para que estos puedan establecer conexión en dos máquinas diferentes dentro de la misma red.

## 2.1 Preguntas

¿Puede el proceso servidor que se muestra en la Figura 2 aceptar conexiones de cualquier cliente en otra IP?, ¿por qué?, ¿es necesario hacer alguna modificación para que esto sea posible?, si así es describe cual,

# 3 Un servidor con múltiples clientes

El ejemplo de la Figura 3 muestra un servidor que espera la conexión de un cliente, cuando un cliente se conecta crea un hilo que mantendrá la conexión con el cliente, mientras el hilo principal esperará la conexión de un cliente nuevo.

Pruebe este proceso servidor en la red de manera que múltiples clientes puedan conectarse simultáneamente.

```
#include <sys/types.h>
#include <sys/socket.h>
#include <pthread.h>
#include <stdio.h>
#include <netinet/in.h>
#include <signal.h>
#include <unistd.h>

#define MAX_CLIENT 10

int totcltes=0;
int client_sockfd[MAX_CLIENT];
void *coneccion_clte(void *arg);

int main()
{
    int server_sockfd;
    int server_len, client_len;
    struct sockaddr_in server_address;
    struct sockaddr_in client_address;

    int parametro[MAX_CLIENT];
```

```
    pthread_t tid[MAX_CLIENT];
    int i;

    server_sockfd = socket(AF_INET, SOCK_STREAM, 0);
    server_address.sin_family = AF_INET;
    server_address.sin_addr.s_addr = htonl(INADDR_ANY);
    server_address.sin_port = htons(9734);
    server_len = sizeof(server_address);
    bind(server_sockfd,(struct sockaddr *)&server_address,server_len);

      /* Crear una cola de conexiones */
    listen(server_sockfd, 5);

    while(totcltes<MAX_CLIENT)      //
    {
        printf("server esperando, clientes atendidos %d\n",totcltes);

        /* Aceptar conexión */
        client_sockfd[totcltes] = accept(server_sockfd,(struct sockaddr *)
&client_address, &client_len);

        /* Crea un hilo que atiende al nuevo cliente */
        parametro[totcltes]=totcltes;
        pthread_create(&tid[totcltes],NULL,coneccion_clte,(void *)
&parametro[totcltes]);
        totcltes++;
    }

    for(i=0;i<MAX_CLIENT;i++)
        pthread_join(tid[i],NULL);
}

void *coneccion_clte(void *arg)
{
    char ch;
    int *n=(int *) arg;
    int mynum=*n;
    read(client_sockfd[mynum], &ch, 1);
    sleep(5);
    ch++;
    write(client_sockfd[mynum], &ch, 1);
    close(client_sockfd[mynum]);
    return;
}
```

**Figura 3. Un servidor que permite la conexión de múltiples clientes**

## 3.1 Preguntas

¿Por qué ahora `client_sockfd` es un arreglo?

¿Cómo sabe cada uno de los hilos que número le corresponde?

¿Cómo se te ocurre que el servidor al recibir el carácter de un cliente enviarlo a todos los demás clientes que estén conectados?

### *4 Minichat*

Basándose en los ejemplos anteriores, diseñe un chat que constará de dos programas: un programa cliente y un programa servidor. Múltiples procesos clientes pueden ejecutarse y establecer conexión con un único servidor. De esta manera múltiples clientes envían mensajes a un servidor que distribuye los mensajes a todos los clientes permitiendo que todos puedan escribir y ver lo que todos los demás escriben.

Cuando un cliente inicia su ejecución pedirá un nombre de usuario o nick para que el sistema lo identifique con un nombre. El cliente establece conexión con el servidor y permite que el usuario escriba texto que será enviado al servidor. El cliente también recibirá los mensajes del servidor y los mostrará en pantalla indicando el nombre de usuario que lo origina. Un usuario podrá abandonar el chat tecleando `exit` en el texto a enviar.

El servidor aceptará las conexiones de los clientes y por cada cliente creará un hilo que será quien mantiene la conexión cliente-servidor (ver ejemplo de la Figura 3). Cada que el servidor recibe un mensaje de un cliente deberá distribuirlo a todos los clientes que se encuentran conectados.

## VI Requisitos para el desarrollo

Opcionalmente puede desarrollar parte del chat, es decir, algunos programas en plataformas que no sean Linux y/ lenguajes que no sean C, pero al menos uno de los programas (cliente o servidor) deberá desarrollarse en C bajo la plataforma Linux.

# VII ENTREGA Y EVALUACIÓN

## 1 Evaluación

**No incluya líneas de código en sus programas de las cuales desconozca su funcionamiento. El código no conocido será anulado en el funcionamiento de la práctica.**

## 2 Equipos

Esta práctica se hará en equipos (máximo 2 integrantes), es necesario que en la revisión esté el equipo completo ya que el integrante que no se presente no tendrá calificación en la práctica.

Importante: Al indicarse que el trabajo debe ser desarrollado por equipos, se entiende que no se permite colaboración entre equipos, cualquier evidencia de esto será considerada como plagio.

## 3 Fecha

La entrega es el Jueves 1° de Noviembre y se hará subiendo en el apartado correspondiente en Moodle los archivos fuentes del chat con su respectivo archivo `Makefile` en un archivo con extensión .ZIP, el tiempo límite de entrega es las 23:55 Hrs del 1° de Noviembre.

# 4 Evaluación

| | | | | | |
|---|---|---|---|---|---|
| Puntualidad en las revisiones | El equipo estuvo completo y puntual en todas las sesiones de revisión. | | Si hubo dos o más sesiones con el equipo, el equipo estuvo completo y puntual en casi todas las sesiones de revisión | | Si solo hubo una sesión de revisión, el equipo no estuvo completo o no fue puntual. Si fueron dos o más sesiones de revisión, en más de una sesión el equipo no estuvo completo o fue puntual |
| | +5 | | +2.5 | | 0 |
| Especificaciones de entrega | La entrega del producto cumple con todas las especificaciones indicadas en el documento de la práctica, por ejemplo, los archivos se entregan de acuerdo a las formas indicadas en el documento de la práctica. | | | | La entrega del producto no cumple con al menos una de las especificaciones indicadas en el documento de la práctica |
| | +5 | | | | 0 |
| Funcionamiento | El producto cumple con todas las especificaciones indicadas en el documento y no tiene fallas | El producto muestra una falla no esperada. | El producto está incompleto (falta máximo aprox 50%), pero lo demás puede funcionar bien | El producto está incompleto (falta máximo aprox 50%) y además muestra fallas o el producto está incompleto (falta máximo aprox 66%) | El producto no funciona o no está incompleto (más del 66%). |
| | +80 | +60 | +40 | +20 | 0 |
| Interfaz con el usuario | El producto funciona y pudo ser utilizado sin necesidad de recibir indicaciones por el desarrollador, tiene instrucciones claras para ser utilizado. | | El producto funciona, pero hubo necesidad de recibir alguna indicación para su uso por parte del desarrollador del producto | | El producto carece de instrucciones claras para ser utilizado y requiere que alguno de los desarrolladores esté presente para su utilización o no puede utilizarse debido a que no está completo |
| | +5 | | +3.5 | | 0 |
| Claridad en el código | El código es claro, usa nombres de variables adecuadas, está debidamente comentado e indentado. Puede ser entendido por cualquier otra persona que no intervino en su desarrollo. | | El código carece de claridad, puede ser entendido por cualquier persona ajena a su desarrollo pero con cierta dificultad. | | El código carece de comentarios, está mal indentado, usa nombres de variables no adecuadas. |
| | +5 | | +3.5 | | 0 |
| Defensa del producto | Todos los que presentan la práctica son capaces de explicar cualquier parte del producto presentado | | Uno de los que presenta la práctica muestra dudas sobre alguna parte del desarrollo del producto presentado | | Más de un integrante no muestra evidencia de que conoce el producto, o si el trabajo fue individual, el desarrollador duda sobre el desarrollo del producto que presenta. |
| | x 1(puntos se multiplican por 1) | | x 0.5 (puntos se multiplican por 0.5) | | x 0 (puntos se multiplican por 0) |
| Sobresaliente 20 % | Tiene 1 en todos los puntos anteriores.<br>El producto entregado es sobresaliente, muestra tener la calidad para ser expuesto como un producto representativo de la carrera<br>Hay evidencia de que los desarrolladores se documentaron y muestran aprendizajes más allá de lo esperado | | | | No tiene 1 en todos los puntos anteriores, o<br>el producto entregado no es sobresaliente y no muestra tener la calidad para ser expuesto como un producto representativo de la carrera o no<br>hay evidencia de que los desarrolladores se documentaron y muestran aprendizajes más allá de lo esperado |
| | +20 | | | | 0 |