

I OBJETIVO

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

Desarrollar un protocolo simple para una comunicación fiable en un sistema no orientado a conexión.

Usar el lenguaje XML o JSON para la representación de los datos en un sistema distribuido.

II BIBLIOGRAFÍA

- Sistemas Distribuidos, Coulouris, Dellimore, Kindberg, Prentice Hall
- Beginning 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 o una conexión remota.
- Un editor de texto
- El compilador de C de UNIX

IV PREREQUISITOS

- Sistema operativo Linux
- Lenguaje de programación C/C++
- Hilos Posix y/o otra API

V DESCRIPCIÓN DE LA PRÁCTICA

1 *Minichat*

Basándose en el chat desarrollado en la práctica anterior, diseñe un chat que constará de dos programas: un programa cliente y un programa servidor. Múltiples procesos clientes pueden ejecutarse y mandar mensajes a un único servidor que se encargará de distribuir los mensajes entre a los demás clientes.

Cuando un cliente inicia su ejecución pedirá un nombre de usuario o nick para que el sistema lo identifique con un nombre.

El programa cliente debe interactuar con el usuario que puede realizar las siguientes acciones:

- 1) Enviar un mensaje público que será enviado a todos los usuarios.
- 2) Consultar los usuarios conectados.
- 3) Enviar un mensaje privado a un usuario.
- 4) Salir del chat.

2 *Requisitos para el desarrollo*

Tanto el cliente y el servidor deberán ser desarrollados en diferentes lenguajes, es decir, el cliente en un lenguaje y el servidor en otro lenguaje. Por ejemplo: cliente en Java o Python y el servidor en C/C++.

Para esta práctica será necesario definir un lenguaje para intercambiar mensajes entre cliente y servidor, este lenguaje será basado en XML o JSON.

Los mensajes se enviarán como texto, este texto contiene el mensaje en formato XML o JSON.

La comunicación con sockets será con datagramas, es decir, no orientado a conexión. Para esto es necesario diseñar un protocolo entre cliente-servidor que asegure que los mensajes se están recibiendo. En el Ejemplo 1 y Ejemplo 2 tenemos un programa cliente y servidor para la plataforma Linux.

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

#define SRV_IP "127.0.0.1"

void diep(char *s)
{
    perror(s);
    exit(1);
}

#define BUFLen 512
#define NPACK 10
#define PORT 9930

int main(void)
{
    struct sockaddr_in si_other;
    int sockfd, i, result;
```

```

int slen=sizeof(si_other);
char buf[BUFLen];

// Crear el socket
sockfd=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
if (sockfd==-1)
    diep("socket");

memset((char *) &si_other, 0, sizeof(si_other));

// Nombrar el socket
si_other.sin_family = AF_INET;
si_other.sin_port = htons(PORT);
result=inet_aton(SRV_IP, &si_other.sin_addr);
if (result==0)
    diep("inet_aton() failed\n");

for (i=0; i<NPACK; i++)
{
    printf("Sending packet %d\n", i);
    sprintf(buf, "This is packet %d\n", i);

    // Enviar paquete
    result=sendto(sockfd, buf, BUFLen, 0, (void *) &si_other, slen);
    if (result==-1)
        diep("sendto()");
}

close(sockfd);
return 0;
}

```

Ejemplo 1.- Cliente que envía paquetes en la red con datagramas

```

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

#define BUFLen 512
#define NPACK 10
#define PORT 9930

void diep(char *s)
{
    perror(s);
    exit(1);
}

int main(void)

```

```
{
    struct sockaddr_in si_me, si_other;
    int sockfd, i, result;
    int slen=sizeof(si_other);
    char buf[BUFLen];

    // Crear el socket
    sockfd=socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    if (sockfd==-1)
        diep("socket");

    memset((char *) &si_me, 0, sizeof(si_me));

    // Nombrar el socket
    si_me.sin_family = AF_INET;
    si_me.sin_port = htons(PORT);
    si_me.sin_addr.s_addr = htonl(INADDR_ANY);
    result=bind(sockfd, (struct sockaddr *) &si_me, sizeof(si_me));
    if (result==-1)
        diep("bind");

    for (i=0; i<NPACK; i++)
    {
        // Recibir paquete
        result=recvfrom(sockfd, buf, BUFLen, 0, (void *) &si_other,
&slen);
        if (result==-1)
            diep("recvfrom()");
        printf("Received packet from %s:%d\nData:
%s\n\n",inet_ntoa(si_other.sin_addr), ntohs(si_other.sin_port), buf);
    }

    close(sockfd);
    return 0;
}
```

Ejemplo 2.- Servidor que recibe los paquetes (datagramas) del cliente.

VI ENTREGA Y EVALUACIÓN

1 *Evaluación*

Se califica en la sesión de clases del día Jueves 15 de Noviembre. **Favor de traer sus laptops.**



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), se revisará en clase en la sesión del Jueves 15 de Noviembre y 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 Miércoles 14 de Noviembre y se hará subiendo en el apartado correspondiente en Moodle los archivos (programas fuentes, programas ejecutables y un documento que explique su instalación y/o uso) en un archivo con extensión .ZIP, el tiempo límite de entrega es las 23:55 Hrs del 24 de Abril. Subir en un archivo ZIP los programas fuentes en la sección de Moodle correspondiente.

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. El producto entregado es sobresaliente, muestra tener la calidad para ser expuesto como un producto representativo de la carrera 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 el producto entregado no es sobresaliente y no muestra tener la calidad para ser expuesto como un producto representativo de la carrera o no hay evidencia de que los desarrolladores se documentaron y muestran aprendizajes más allá de lo esperado	
	+20				0	