

Instituto Tecnológico de Costa Rica

Escuela de Computación

Lic. en Administración de Tecnologías de Información

Lenguajes de Programación

Andrei Fuentes

Chat en C

Grupo:

Adrián Díaz Meza

John Largaespada Pérez

Alonso Vargas Astúa

Tabla de contenidos

Tabla de contenidos	2
Descripción del programa	3
Diseño del programa	5
Librerías	6
Análisis de resultados	7
Manual de Usuario	8
Conclusión personal	9

Descripción del Problema

El objetivo de esta tarea es familiarizarse con el desarrollo de programas en C, mediante la creación de un programa de mensajería instantánea. Para realizar la comunicación entre las computadoras y poder permitir el intercambio de mensajes instantáneos entre una y otra se usarán sockets.

Los sockets son mecanismos de comunicación que pertenecen a la capa de transporte del modelo OSI, y son usados en una modalidad cliente-servidor, usando tanto comunicaciones orientadas a conexión (protocolo TCP confiable), como en comunicaciones no orientadas a conexión (protocolo UDP no confiable).

Para simplificar el desarrollo de la tarea, se nos recomienda empezar por una solución sencilla y luego irle añadiendo funcionalidad compleja. Inicialmente, se debe lograr que un proceso P1 envíe un mensaje (una hilera de caracteres) a otro proceso P2, y que el proceso P2 pueda responder a P1 con otro mensaje.

Usando sockets TCP, se debe escribir un programa cliente C1 y un programa servidor S1, ambos para Linux, de manera que C1 pueda enviar una hilera de caracteres al S1, el cual desplegará esos caracteres en pantalla y responderá con un mensaje, que podría ser algo similar a “Mensaje Recibido”. Posteriormente, el cliente terminará la conexión.

Los mensajes podrían tener una longitud máxima que podrá ser definida por los estudiantes. Los mensajes se conocerán sólo en tiempo de ejecución, ya que serán digitados en ese momento. Tanto C1 como S1 serán programas de consola. S1 recibirá como argumento el número de puerto en que funcionará, mientras que S1 recibirá como argumento tanto el número de puerto como la IP del servidor al cual se deberá conectar.

Al principio pueden ejecutar C1 y S1 en la misma computadora, sin usar una red. Luego deberán ejecutar los programas en computadoras diferentes en la misma red. Posteriormente podrían ejecutar los programas de manera remota, por ejemplo, C1 ejecutándose en las computadoras de los laboratorios y S1 ejecutándose en la computadora de la casa.

En los programas descritos anteriormente, se tiene que C1 debe iniciar la comunicación con S1, y esperar a que S1 le devuelva su respuesta, lo cual no permite que C1 envíe un segundo mensaje sin haber recibido respuesta del primer mensaje. Esto no permite la creación de un messenger en la que los dos participantes pueden escribir sus mensajes sin tener que esperar las respuestas del otro.

Para lograr el correcto funcionamiento del messenger, se deben de modificar los programas descritos anteriormente, para crear un programa messenger llamado M1, el cual creará 2 sockets: uno para enviar mensajes, y el otro sólo para recibir mensajes. Después, mediante el uso de fork() el programa se bifurca en dos procesos, el cliente y el servidor.

El proceso cliente atiende el socket de enviar, y cuando el usuario digite un mensaje, enviará el mensaje sin esperar una respuesta. El proceso servidor atenderá el socket de recibir, y cada vez que reciba un mensaje, lo desplegará en pantalla y continuará esperando por otro mensaje.

Tanto mensajes enviados como mensajes recibidos se van a mezclar en la pantalla, por lo que se deberá de poner un identificador al inicio de cada mensaje para poder identificar su origen, incluso se podrían usar colores para distinguir los mensajes enviados de los recibidos, como en el siguiente ejemplo:

Enviado: Hola

Recibido: Hola, como vas?

Enviado: Bien

Enviado: Y vos? Todo en orden?

Recibido: Si, todo en orden

Una copia de M1 se ejecutará en una computadora, y otra copia en otra. Para finalizar la ejecución, alguna de las copias escribirá el mensaje “Adios”, y se cerrarán los sockets. La otra copia detectará el mensaje “Adios”, y, después de desplegar el mensaje, cerrará sus sockets.

Diseño del Programa

Al inicio se contaba con dos códigos diferentes (un cliente y un servidor) que se ejecutaban al mismo tiempo para poder enviar mensajes entre ellos, en el programa “servidor” era el programa principal, el que definía las direcciones por las cuáles se iba a dar el enlace entre las entidades mencionadas anteriormente pero solamente en la misma computadora, en ese momento se debía correr primero el programa servidor para que montara la conexión, por medio de los sockets y otras implementaciones logramos hacer que se diera ya entre computadoras diferentes pero por medio de una conexión punto a punto, con el proceso fuimos involucrando aspectos como el comando `fork()` fuimos evolucionando el programa hasta llegar al punto de unir el cliente y el servidor en el mismo archivo y crear un programa “cliente-servidor” en el que el primero de los dos computadores en correrlo será el servidor y el segundo el cliente, no como en los prototipos anteriores en el que se debía correr estrictamente el programa servidor primero. Se realizó esta modificación con el fin de evitar molestias de usuarios al tener que correr siempre dos códigos diferentes.

Se le dio color a los textos ingresados por los usuarios con el fin de tener una mejor distinción de las escrituras.

Librerías

Para el correcto funcionamiento de este programa se requiere del uso de las siguientes librerías:

<i>Librería</i>	<i>Función</i>
#include <stdio.h>	Es la librería que contiene funciones de cabeceras de Entrada/Salida.
#include <stdlib.h>	Es la biblioteca necesaria en funciones aritméticas, números aleatorios y conversión de cadenas.
#include <string.h>	Es la biblioteca necesaria para el manejo de cadenas
#include <sys/utsname.h>	Obtiene el nombre e información del núcleo actual
#include <sys/types.h>	Librería que contiene funciones de búsqueda y ordenamiento de directorios y manipulación de archivos.
#include <netinet/in.h>	Librería que contiene encabezados de la familia del protocolo de internet.
#include <arpa/inet.h>	Librería que incluye las definiciones de las operaciones de internet.
#include <netdb.h>	Librería que contiene las definiciones de las operaciones de la base de datos de red.
#include <unistd.h>	Librería que contiene funciones de cabeceras para el manejo de directorios y archivos
#include <fcntl.h>	Librería que contiene las funciones de control de archivos.
#include "MES.h"	Se incluye con el fin de utilizar el kit de herramientas MES.

Análisis de Resultados

Aspectos Concluidas:

El programa cuenta con la mayoría de los requerimientos estipulados por el profesor, entre ellos contamos con:

- Programa escrito en el lenguaje C, además funciona en sistemas operativos de Linux.
- Uso de sockets para la conexión entre computadoras.
- EL programa se encuentra en un sólo código, permitiendo una conexión entre dos equipos en cualquier dirección.
- Bifurcación por medio del uso del comando fork().
- Se posee un identificador al inicio de cada mensaje para saber quién lo envía, además de un color diferente para cada usuario.

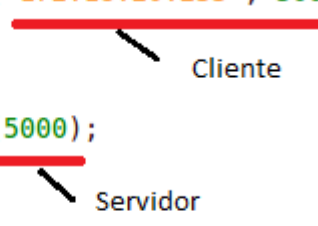
Aspectos Inconclusos:

- El programa no cuenta con la función del Adios requerida por el profesor.

Manual de Usuario

1. Se deben introducir en el el main los datos de la IP del servidor y el puerto en caso de que se vaya a ejecutar como servidor, o solamente el cliente en caso de que se vaya a ejecutar como cliente.

```
int main ()
{
    pid_t hijo;
    hijo = fork();
    if (hijo >= 0)
    {
        if (hijo == 0)
        {
            CL("172.18.20.233", 5000);
        }
        else
        {
            SL(5000);
        }
    }
    else
    {
        perror("fork");
        exit(0);
    }
    return 0;
}
```



2. Cuando los datos que se introducen en el main sean sean correctos se podra ejecutar el programa;

la ejecución del programa tiene el siguiente sistema:

1. el programa se inicializara ejecutando un cliente.
2. si hay un hay un servidor activo, el cliente se conectará a este.
3. si no hay un servidor el programa dejara de ser cliente y se convertira en servidor.

Despues de que los pasos anteriores sean ejecutados se podra dar inicio a la comunicación.

ejemplo de comunicación:

```
- cliente: hola
-hola cm estas?
- cliente: bien y ud?
-bien gracias
-
```

Conclusión Personal

Durante este trabajo fue de gran importancia el trabajo en equipo, con una buena la cooperación por parte de todos los integrantes del grupo, además de la investigación previa al arranque del proyecto ya que fueron factores clave para el éxito de este.

Aprendimos sobre la programación en un lenguaje de bajo nivel como lo es C, y al usarlo nos dimos cuenta de la gran similitud que tiene con Java, además al añadirle el uso de sockets nos ayudó a entender un poco más sobre el funcionamiento de la telemática y sus aplicaciones, teniendo en cuenta que los tres tenemos un gran interés en el dominio y administración de redes.

