Instituto Tecnológico de Costa Rica

Lenguajes de Programación

Primera Tarea Programada

Profesor: Andrei Fuentes Leiva

Estudiantes:

Andrés Fernández

Luigui Madrigal

Primer Semestre2014

**Tabla de contenidos**

• Descripción del problema……………………………………………….3

• Diseño del programa…………………………………………………….4

• Librerías usadas………………………………………………………….5

• Análisis de resultados…………………………………………………....6

• Manual de usuario………………………………………………………..7

• Conclusión personal…………………………………………………….......................13

• Referencias…………………………………………………………….....14

**Descripción del problema**

La idea de esta tarea programada es crear un programa similar a un chat que utilice sockets para conectarse entre diferentes computadoras, se solicita que el programa cumpla las funcionalidades de agregar contactos, enviar y recibir mensajes. Para realizar esto se debe utilizar un fork para bifurcar los procesos en cliente y servidor y así poder manejar el envío y recibido de mensajes de parte de los dos. También se necesita un manejo de direcciones IP y puertos para realizar las conexiones necesarias para la comunicación. Se deben usar archivos (en nuestro caso usamos archivos de texto) para guardar contactos y evitar que la información sea volátil.

http://www.chuidiang.com/clinux/sockets/sockets\_simp.php

**Diseño del programa**

El programa al iniciar muestra un menú donde están las opciones: agregar contacto, enviar mensaje, recibir mensaje y salir, cada una identificadas con diferentes números. Para agregar contactos se solicitan los parámetros que son almacenados en un struct llamado listaContactos y en el archivo contactos.txt.

Los datos necesarios para la creación del contacto son el nombre, el puerto para la conexión y la dirección IP actual del ordenador.

Para iniciar una conversación el programa muestra la lista de contactos, cada uno con un número de puerto el cual es solicitado para asignárselo al struct y realizar la conexión. Terminado esto se realiza la creación de los sockets y la asociación de estos a puertos para así empezar el envío y recepción de mensajes, por medio de estos mismos que contienen los datos necesarios para la interacción. En esta parte se inicia el servidor que corresponde al primero que recibirá el mensaje.

Por otra parte se solicita el puerto de escucha y la dirección IP del servidor, con esto se inicia la sesión de cliente y se muestra la opción para enviar el primer mensaje. Tanto el servidor como el cliente tendrán la opción de recibir y enviar mensajes.

Por último escribiendo la palabra “salir” se cierra la conversación.

**Librerías usadas**

**#include <stdio.h> (standard input-output header)**

Contiene funciones de cabeceras de entrada/salida.

**#include <sys/socket.h>**

Librería para abrir un socket

**#include <stdlib.h>**

Librería para ejecutar procesos

**#include <unistd.h>**

Sirve para el control de procesos, contiene funciones para el manejo de directorios y archivos.

**#include <sys/types.h>**

Librería para el manejo de señales.

**#include <netinet/in.h>**

**Análisis de resultados**

**Objetivos alcanzados**

* Manejo de archivos para guardar información.
* Agregar contactos.
* Utilización de fork.
* Creación y conexión de sockets.
* Enviar y recibir mensajes.

**Objetivos no alcanzados**

* Cierre seguro de las conversaciones.
* Optimización a la hora de petición de datos.

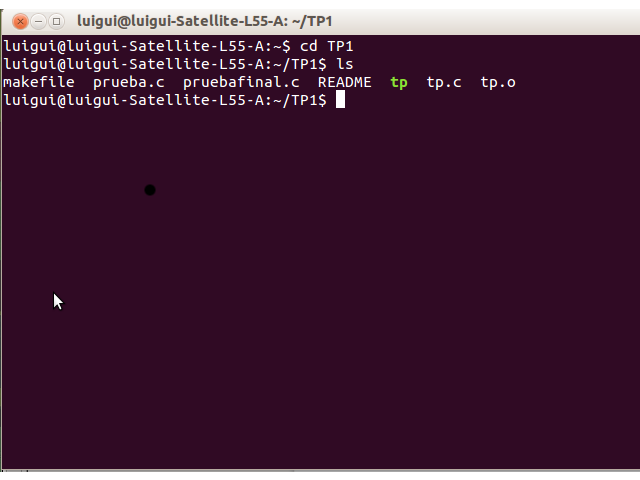
**Manual de usuario**

**Instrucciones**

**1. Abrir la terminal de linux**

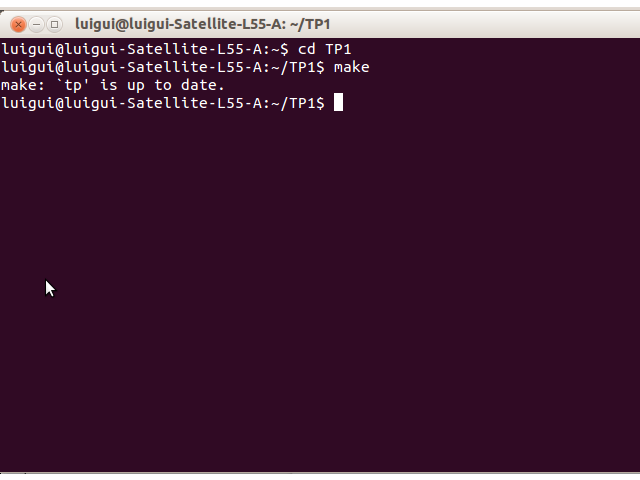
-Abrir consola o terminal

-Ingresar en la carpeta donde se encuentra ubicado el archivo “tp.c”



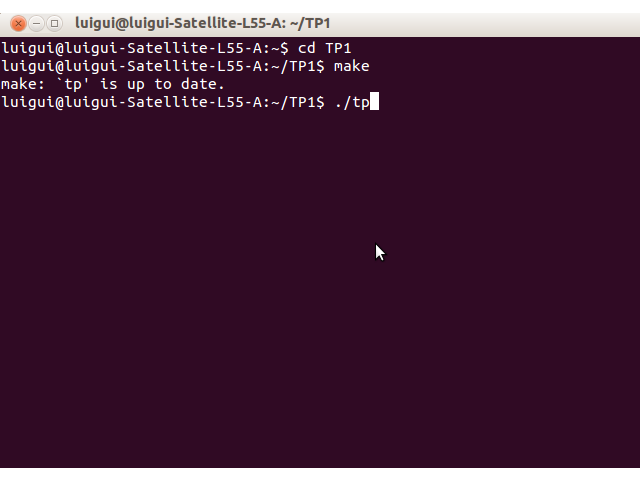
**2. Compilar el archivo**

Sin salirse del terminal, para compilar el archivo, escriba make.



**3. Iniciar archivo ejecutable**

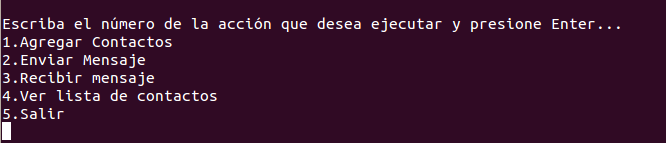
Para iniciar el programa escriba ./tp



**4. Menú**

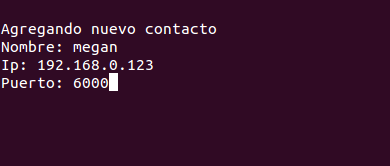
A continuación se muestra el menú con las opciones, escriba el número de la opción escogida y presione enter.

Ejemplo:



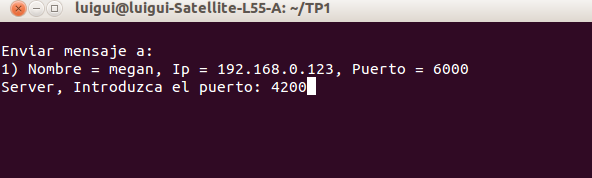
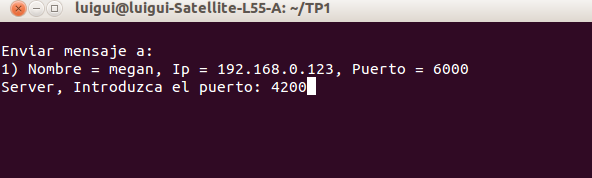
**5.Agregar contactos**

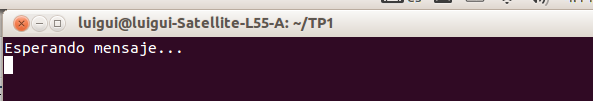
Presionamos 1 y enter, nos aparece el menú de agregar contacto, pregunta nombre, ip y puerto.



**6. Enviar mensaje**

Se nos muestra la lista de contactos, introducimos el puerto y presionamos enter.

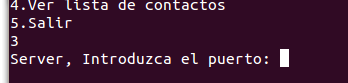




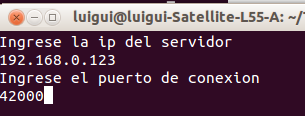
**6. Recibir mensajes**

Para realizar la conexión y enviar mensajes es necesario abrir otra terminal y ejecutar el programa de la misma manera que lo hicimos anteriormente.

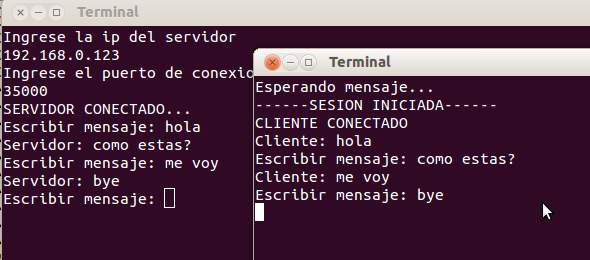
Nos aparece el menú, presionamos la opcion recibir mensajes.



Introducimos el puerto y presionamos enter. A continuación nos pide la ip del servidor y el puerto, presionamos enter cuando este correcto.



Se crea la conexión y se pueden transmitir mensajes desde cada terminal.



**7.Ver lista de contactos**

Para ver la lista de contactos simplemente seleccionamos la opción 4 en el menú.

**8. Salir del programa**

Para salir del programa durante el envío de mensajes, escriba la palabra “salir”.

Si está en el menú presione la opción 5.

**Conclusión personal**

C es un lenguaje de programación muy poderoso, aprendimos mucho de esta tarea pues debimos investigar sobre utilización de sockets, manejo de archivos y la utilización de la función fork, además de algunos conceptos básicos pero importantes de la programación en este lenguaje de bajo nivel, como el manejo de punteros, structs, arreglos ,etc.

Muy interesante esta tarea programada, la emulación de un programa como un chat o un whatsapp motivan y llaman nuestra atención, pues es algo que utilizamos todos los días, además, conocer a grandes rasgos cómo funcionan tanto externa como internamente estos programas estamos seguros nos será de utilidad en el futuro.

**Referencias**

<http://www.chuidiang.com/clinux/sockets/sockets_simp.php>