

2012

Tecnológico de Costa
Rica

[PROGRA_C LENGUAJES]

Contenido

Diseño del programa.....3

Descripción del Programa.....5

Librerías.....6

Manual de usuario.....7

Análisis de resultados.....14

Bibliografía.....15

Diseño del programa

Para el desarrollo del diseño del programa se inició con la investigación de C, sentencias, estructura, funciones y su funcionamiento en Linux. También se investigó sobre los sockets y fork, qué son, cómo funcionan y se analizaron ejemplos relacionados.

Con la información y conocimiento adquirido se desarrolló en papel todos los diferentes pasos para la realización y finalización del programa. Para esto se contó con todos los integrantes del grupo, y sus múltiples ideas. Luego se trasladaron las ideas a código fuente.

❖ **Decisiones tomadas:**

Entre las decisiones tomadas estuvieron:

- Usar TCP en vez de UDP: Esto debido a que el protocolo UDP sólo garantiza que, si un mensaje llega, llegue bien, y no garantiza que llegue o que lleguen todos los mensajes en el mismo orden que se envían.

- Utilización de la función fork: Se utilizó porque ésta da como resultado una bifurcación, representa la ramificación de cualquier proceso. En el caso del proyecto Messenger, se utiliza para poder enviar o recibir varios mensajes sin necesidad de una respuesta.

- Conexión del cliente con el servidor: Se utiliza la estructura while en cliente, esto permite hacer peticiones para conectarse a un servidor, en el momento en que encuentra algún servidor disponible en la IP y puerto indicado, realiza la conexión.

- Nombre del usuario: Cuando se realiza la ejecución del Messenger se envía como parámetros el IP remoto, el puerto remoto, el puerto local, además el nombre (nickname) del usuario, que se desplegará en la pantalla del usuario.

Lógica del programa:

Se compila y ejecuta el programa M1 en el cual se envía IP remoto, puerto remoto, puerto local y el nombre del usuario que se va a conectar. Luego el primer programa M1 se intenta conectar a otro IP en un ciclo, este ciclo finaliza cuando logra la conexión.

Luego se realizan validaciones de errores tales como:

- ✦ ERROR en caso de que no se pueda completar el bind
- ✦ ERROR en caso que falle el proceso de listen.
- ✦ ERROR en caso que no se pueda aceptar la conexión entrante.

Luego de obtener la conexión, se imprime en pantalla el mensaje de bienvenida.

Para enviar mensajes primero se valida que el último mensaje recibido no se la palabra "Adios", en caso contrario se imprime un mensaje de cierre del programa y finaliza la conexión.

Se emplea en el código realizado el método fork(), que permitirá realizar dos procesos, los cuales serán de ayuda para la comunicación entre el cliente y el servidor, además con esto se podrán mandar mensajes de manera bifurcada y simultánea, lo cual permitirá simular un msn.

Descripción del Programa

El proyecto consiste en el desarrollo de un programa de mensajería instantánea (messenger), el cual debe ser desarrollado en paradigma imperativo, específicamente en C y debe ser ejecutado en la plataforma de Linux.

Se debe implementar sockets en el desarrollo del programa, los sockets son mecanismos de comunicación pertenecientes a la capa de transporte del modelo OSI, en la que se puede emitir y recibir información de manera fiable y ordenada entre dos programas. Los sockets son usados en una modalidad de clienteservidor, en el cual una consulta o realiza peticiones y el otro se encuentra en estado de espera, cuando ya recibe la petición, envía una respuesta.

Los mensajes se conocerán sólo en tiempo de ejecución, ya que serán digitados en ese momento. El servidor recibirá como argumento el número de puerto en que funcionará, por otra parte, el servidor recibirá como argumento tanto el número de puerto como la IP del servidor al cual se deberá conectar.

El cliente debe iniciar la comunicación con el servidor, y esperar a que el servidor le devuelva una respuesta, lo cual no permite que el servidor envíe un segundo mensaje sin haber recibido respuesta del primer mensaje.

Después, mediante el uso de fork el programa se bifurca en dos procesos, el cliente y el servidor, lo que permitirá que los mensajes tanto enviados como los recibidos se van a mezclar en la pantalla. Para finalizar la ejecución del programa, uno de los usuarios debe escribir el mensaje "Adios", este es un indicador para que se cierren los sockets.

Primero el programa debe ejecutar el cliente y servidor en la misma computadora, luego deberán ejecutarse en computadoras diferentes en la misma red. Posteriormente podrían ejecutar los programas de manera remota.

Librerías

Es una recopilación de cabeceras y bibliotecas, que implementan operaciones comunes. El nombre y las características de cada función, se encuentran en un fichero denominado archivo de cabecera con extensión ".h". Las que se utilizaron

en el proyecto son:

- ❖ **<unistd.h>**: Es el nombre del archivo de encabezado que proporciona acceso al sistema operativo POSIX API, define varios tipos y constantes simbólicas, y declara funciones auxiliares.
- ❖ **<sys/types.h>**: Incluye definiciones que se utiliza para el recuento de bloqueo de archivos, para tamaños de bloque, para los tiempos del sistema de ciclos de reloj.
- ❖ **<errno.h>**: En ella se definen las macros que presentan un informe de error a través de códigos de error. La macro errno se expande a un lvalue con tipo int, que contiene el último código de error generado en cualquiera de las funciones utilizando la instalación de errno.
- ❖ **<stdio.h> (standard input-output header)**: Contiene las definiciones de macros, las constantes, las declaraciones de funciones y la definición de tipos usados por varias operaciones estándar de entrada y salida.
- ❖ **<sys/wait.h>**: Espera la terminación del proceso, para el uso de waitpid(), se definen las siguientes constantes simbólicas:
 - ✦ WNOHANG: No esperar por un hijo que está en ejecución.
 - ✦ WUNTRACED: Informa del estado del proceso hijo detenido.
- ❖ **<stdlib.h>**: Es para utilidades generales, y contiene los prototipos de funciones de C para gestión de memoria dinámica, control de procesos y otras.
- ❖ **<sys/socket.h>**: Define los siguientes elementos:
 - ✦ socklen_t: que es un tipo entero de la anchura.
 - ✦ sa_family_t: sin signo de tipo entero.
 - ✦ Sockaddr: se utiliza para definir una dirección de conector que se utiliza bind(), connect(), getpeername(), etc.
- ❖ **<netinet/in.h>**: Mediante typedef se incluye un tipo entero sin signo de exactamente 16 bits y un tipo entero sin signo de exactamente 32 bits. Y para in_addr: Incluye al menos in_addr_t s_addr
- ❖ **<sys/un.h>**: Incluye al menos la dirección de la familia y la ruta del socket.
- ❖ **<netdb.h>**: Puede hacer disponible el tipo in_port_t y el tipo in_addr_t.

Define la estructura hostent que incluye al menos el nombre oficial del host, un puntero a un array de punteros a los nombres de host alternativos, tipo de dirección, la longitud en bytes de la dirección.

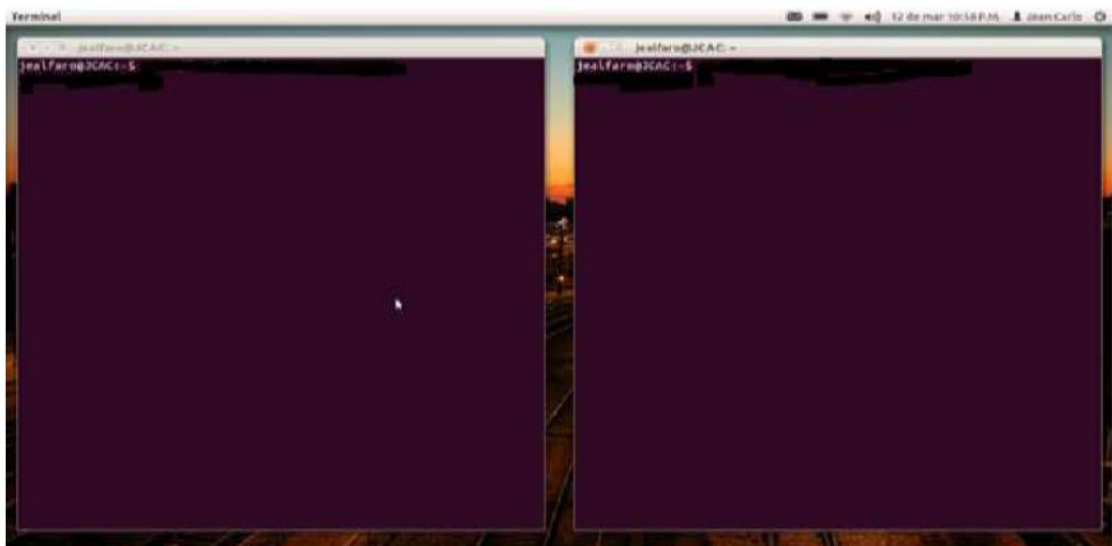
Manual de usuario

❖ Para realizar la compilación del programa en una misma máquina se deben realizar los siguientes pasos:

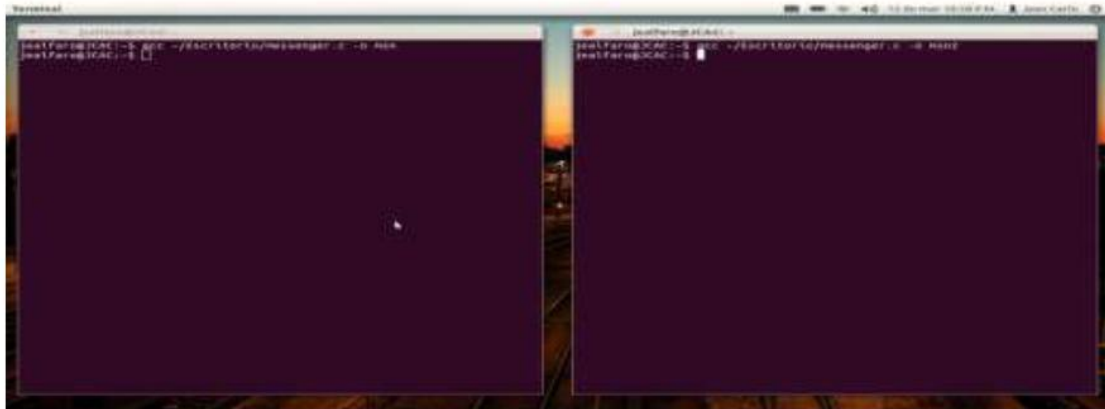
1. Abrir el Sistema Operativo Ubuntu
2. Iniciar



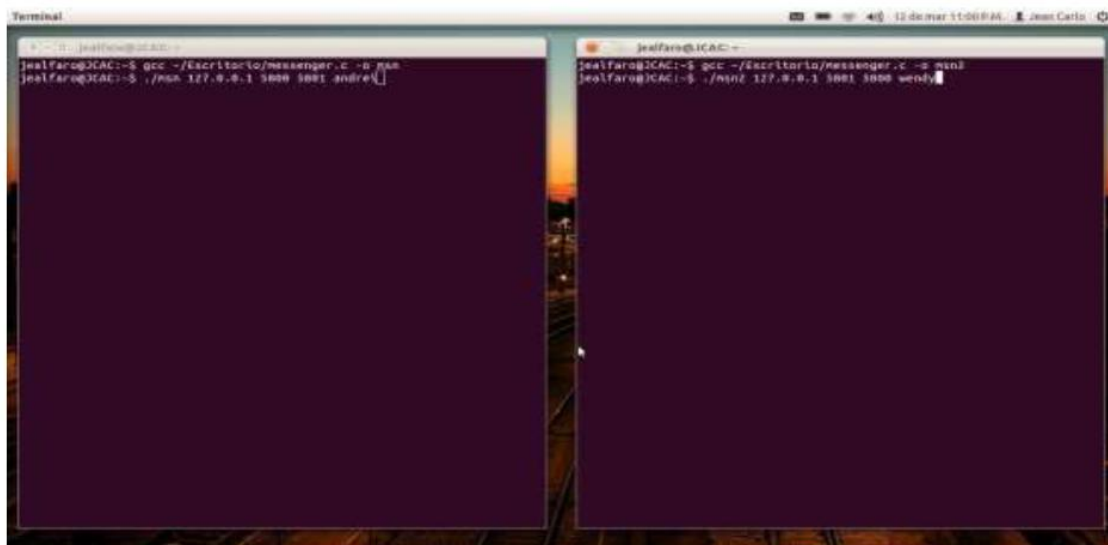
2. Iniciar las consolas



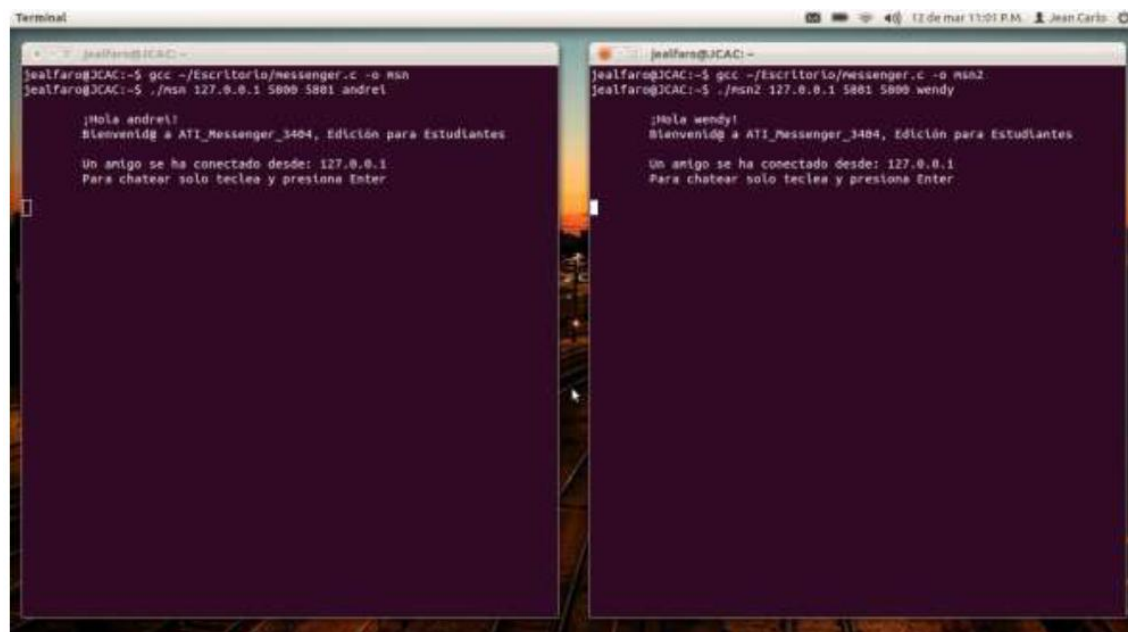
3. Luego se realiza la compilación, para ello se de colocar el comando `gcc nombreDelArchivo.c -o nombreArchivoEjecutable` y en la otra consola, se realiza igual. Si no se coloca el nombre del archivo ejecutables, se crea uno por defecto con el nombre `a.out`



4. Para ejecutar el programa se debe poner la siguiente sentencia:
`./nombreArchivoEjecutable IpRemoto puertoRemoto puertoLocal NicknameUsuario`. Se realiza igual en la otra consola intercambiando el orden de los puertos.



5. Aparece un mensaje de Bienvenida



```
Terminal
jealfaro@JCAC:~$ gcc -fEsritorio/messenger.c -o msn
jealfaro@JCAC:~$ ./msn 127.0.0.1 5800 5801 andrei

¡Hola andrei!
Bienvenid@ a ATI_Messenger_3404, Edición para Estudiantes.

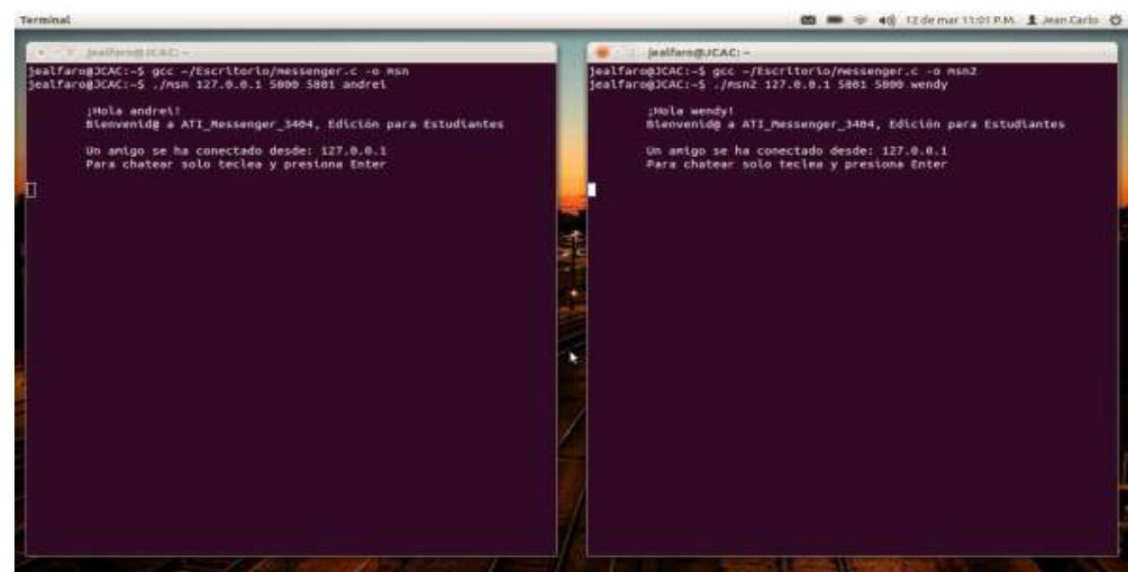
Un amigo se ha conectado desde: 127.0.0.1
Para chatear solo teclea y presiona Enter.

jealfaro@JCAC:~$ gcc -fEsritorio/messenger.c -o msn2
jealfaro@JCAC:~$ ./msn2 127.0.0.1 5801 5800 wendy

¡Hola wendy!
Bienvenid@ a ATI_Messenger_3404, Edición para Estudiantes.

Un amigo se ha conectado desde: 127.0.0.1
Para chatear solo teclea y presiona Enter.
```

Luego ya se puede realizar la comunicación y el envío de mensajes.



```
Terminal
jealfaro@JCAC:~$ gcc -fEsritorio/messenger.c -o msn
jealfaro@JCAC:~$ ./msn 127.0.0.1 5800 5801 andrei

¡Hola andrei!
Bienvenid@ a ATI_Messenger_3404, Edición para Estudiantes.

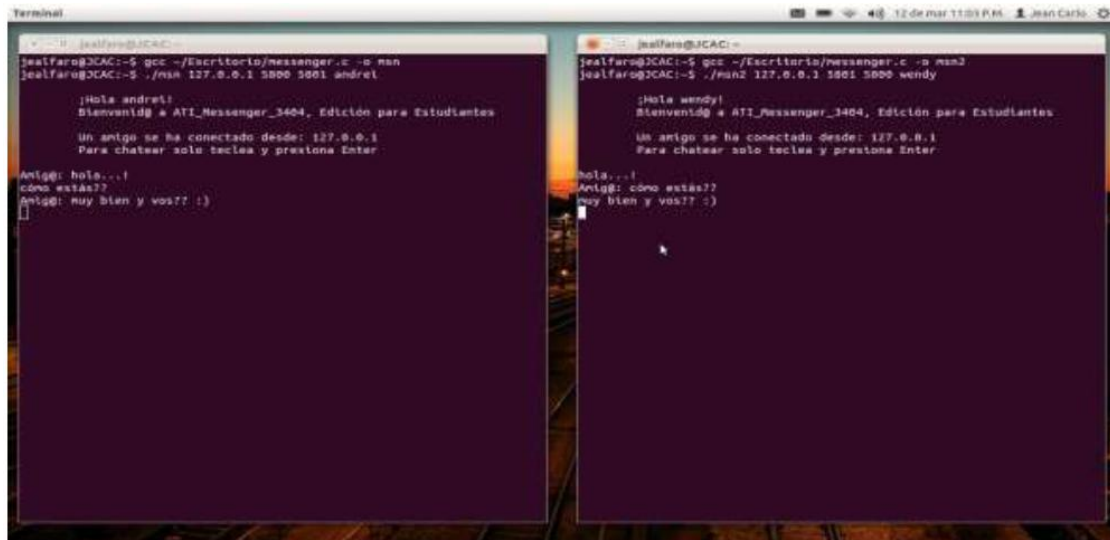
Un amigo se ha conectado desde: 127.0.0.1
Para chatear solo teclea y presiona Enter.

jealfaro@JCAC:~$ gcc -fEsritorio/messenger.c -o msn2
jealfaro@JCAC:~$ ./msn2 127.0.0.1 5801 5800 wendy

¡Hola wendy!
Bienvenid@ a ATI_Messenger_3404, Edición para Estudiantes.

Un amigo se ha conectado desde: 127.0.0.1
Para chatear solo teclea y presiona Enter.
```


7. Una vez enviado el mensaje, le aparecerá recibido a la otra persona y puede responder con otro mensaje para el otro usuario.



```
Terminal
jealfaro@JCAC:~$ gcc ~/Escritorio/messenger.c -o msn
jealfaro@JCAC:~$ ./msn 127.0.0.1 5000 5001 andrei

¡Hola andrei!
Bienvenido a ATI_Messenger_3404, Edición para Estudiantes

Un amigo se ha conectado desde: 127.0.0.1
Para chatear solo teclea y presiona Enter

Amigo: hola...!
¿cómo estás??
Amigo: muy bien y vos?? :)

jealfaro@JCAC:~$ gcc ~/Escritorio/messenger.c -o msn2
jealfaro@JCAC:~$ ./msn2 127.0.0.1 5001 5000 wendy

¡Hola wendy!
Bienvenido a ATI_Messenger_3404, Edición para Estudiantes

Un amigo se ha conectado desde: 127.0.0.1
Para chatear solo teclea y presiona Enter

hola...!
Amigo: ¿cómo estás??
muy bien y vos?? :)

jealfaro@JCAC:~$
```

8. Si desea terminar la comunicación debe digitar “Adiós”.



```
Terminal
jealfaro@JCAC:~$ gcc ~/Escritorio/messenger.c -o msn
jealfaro@JCAC:~$ ./msn 127.0.0.1 5000 5001 andrei

¡Hola andrei!
Bienvenido a ATI_Messenger_3404, Edición para Estudiantes

Un amigo se ha conectado desde: 127.0.0.1
Para chatear solo teclea y presiona Enter

Amigo: hola...!
¿cómo estás??
Amigo: muy bien y vos?? :)
Amigo: como va el curso de LP???
excelente por dicha, tengo unos guices para que los revisés...!
Amigo: bueno, nos vemos
Adiós
Terminaste la conversación,
Vuelve pronto...!

jealfaro@JCAC:~$ gcc ~/Escritorio/messenger.c -o msn2
jealfaro@JCAC:~$ ./msn2 127.0.0.1 5001 5000 wendy

¡Hola wendy!
Bienvenido a ATI_Messenger_3404, Edición para Estudiantes

Un amigo se ha conectado desde: 127.0.0.1
Para chatear solo teclea y presiona Enter

hola...!
Amigo: ¿cómo estás??
muy bien y vos?? :)
como va el curso de LP???
Amigo: excelente por dicha, tengo unos guices para que los revisés...!
bueno, nos vemos
Amigo: Adiós
Tu Amigo terminó la conversación,
Vuelve pronto...!

jealfaro@JCAC:~$
```

Algunos ejemplos de la conversación y realización de la bifurcación:



```
Terminal
jealfaro@JCAC:~$ gcc ~/Escritorio/messenger.c -o msn
jealfaro@JCAC:~$ ./msn 127.0.0.1 5000 5001 andrei

¡Hola andrei!
Bienvenido a ATI_Messenger_3404, Edición para Estudiantes

Un amigo se ha conectado desde: 127.0.0.1
Para chatear solo teclea y presiona Enter

Amigo: hola...!

jealfaro@JCAC:~$ gcc ~/Escritorio/messenger.c -o msn2
jealfaro@JCAC:~$ ./msn2 127.0.0.1 5001 5000 wendy

¡Hola wendy!
Bienvenido a ATI_Messenger_3404, Edición para Estudiantes

Un amigo se ha conectado desde: 127.0.0.1
Para chatear solo teclea y presiona Enter

hola...!
```

```
Terminal
jealfarq3CAC:~$ gcc -fEscritorio/messenger.c -o msn
jealfarq3CAC:~$ ./msn 127.0.0.1 5000 5001 andrei

;Hola andrei!
Bienvenido a ATI_Messenger_3484, Edición para Estudiantes
Un amigo se ha conectado desde: 127.0.0.1
Para chatear solo teclea y presiona Enter

Amigo: hola...!
¿cómo estás??
Amigo: muy bien y vos?? :}
Amigo: como se el curso de LP??
Amigo: excelente por dicho, tengo una quiza para que los revluda...!
Amigo: buena, así venia

Terminal
jealfarq3CAC:~$ gcc -fEscritorio/messenger.c -o msn2
jealfarq3CAC:~$ ./msn2 127.0.0.1 5001 5000 wendy

;Hola wendy!
Bienvenido a ATI_Messenger_3484, Edición para Estudiantes
Un amigo se ha conectado desde: 127.0.0.1
Para chatear solo teclea y presiona Enter

Hola...!
Amigo: ¿cómo estás??
Amigo: muy bien y vos?? :}
Amigo: como se el curso de LP??
Amigo: excelente por dicho, tengo una quiza para que los revluda...!
Amigo: buena, así venia
```

```
Terminal
jealfarq3CAC:~$ gcc -fEscritorio/messenger.c -o msn
jealfarq3CAC:~$ ./msn 127.0.0.1 5000 5001 andrei

;Hola andrei!
Bienvenido a ATI_Messenger_3484, Edición para Estudiantes
Un amigo se ha conectado desde: 127.0.0.1
Para chatear solo teclea y presiona Enter

Amigo: hola...!
¿cómo estás??
Amigo: muy bien y vos?? :}
Amigo: como se el curso de LP??
Amigo: excelente por dicho, tengo una quiza para que los revluda...!
Amigo: buena, así venia

Terminal
jealfarq3CAC:~$ gcc -fEscritorio/messenger.c -o msn2
jealfarq3CAC:~$ ./msn2 127.0.0.1 5001 5000 wendy

;Hola wendy!
Bienvenido a ATI_Messenger_3484, Edición para Estudiantes
Un amigo se ha conectado desde: 127.0.0.1
Para chatear solo teclea y presiona Enter

Hola...!
Amigo: ¿cómo estás??
Amigo: muy bien y vos?? :}
Amigo: como se el curso de LP??
Amigo: excelente por dicho, tengo una quiza para que los revluda...!
Amigo: buena, así venia
```

Análisis de resultados

Se logró concluir con todas las indicaciones especificadas en el documento de la primera tarea programada, conectando dos servidores, por medio de sockets y así poder comunicarse simulando un Messenger.

Se alcanzó la meta de obtener conocimiento relacionado con sockets, fork o bifurcación y aspectos relacionados con el paradigma imperativo, específicamente el lenguaje de programación C.

Aspectos no contemplados:

Relacionado a los puntos opcionales:

- ❖ No se desarrollo una aplicación grafica, y solo se puede ejecutar desde consola.
- ❖ No se incorporaron colores para la distinción de usuarios, solo indica el mensaje que envía el otro usuario en consola.

Conclusión

La primera tarea programada va muy acorde con los contenidos descritos en la carta al estudiante entregado al iniciar el curso.

Durante estas 5 semanas de curso, estudiamos el paradigma imperativo, al cual pertenece el lenguaje de programación C. Mismo que es utilizado para el desarrollo de esta primera tarea programada del curso.

Concluimos que se dificulta empezar una tarea programada sin previo estudio del proyecto como tal, por lo que se recomienda comprender en su totalidad el proyecto e ir detallando todas las partes del diseño para luego desarrollar el código fuente.

Se recomienda hacer un análisis profundo de los aspectos desconocidos en la especificación del proyecto es decir la función fork, implementación de sockets y un amplio estudio de las estructuras, sintaxis en general de C como lenguaje de programación.

Bibliografía

Recuperado el 03 de Marzo de 2012, de
http://mygnet.net/it/descargas/codigos/sockets_en_c_en_linux.1825.zip
Copyright © 2004, 2. 2. *YoLinux Tutorial: Fork, Exec and Process control*. Recuperado el
03 de
Marzo de 2012, de <http://www.yolinux.com/TUTORIALS/ForkExecProcesses.html>
Gcc. Recuperado el 03 de Marzo de 2012, de <http://doc.ubuntu-es.org/GCC>
Group, C. ©.-2. *The Open Group Base Specifications Issue 6*. Recuperado el 8 de Marzo
de 2012, de
<http://pubs.opengroup.org/onlinepubs/000095399/basedefs/sys/types.h.html>
Héctor Tejeda Villela. *Manual de C*. Recuperado el 02 de Marzo de 2012, de
<http://iate.oac.uncor.edu/~manuel/numerico/manual-de-C.pdf>
Ubutu.es. Recuperado el 06 de Marzo de 2012, de <http://www.ubuntu-es.org/node/14939>