

**INSTITUTO TECNOLÓGICO DE COSTA RICA
ADMINISTRACIÓN DE TECNOLOGÍAS DE INFORMACIÓN
LENGUAJES DE PROGRAMACIÓN**

TAREA PROGRAMADA I

**ESTUDIANTES:
LUIS DIEGO COTO MATA
ERICK VARGAS VICTOR
MARIAM RAMIREZ CORDERO**

PROFESOR: ANDRÉI FUENTES LEIVA

I SEMESTRE 2013.

Tabla de contenidos

	Pág.
Descripción del problema.....	1
Diseño del programa.....	2
Librerías Usadas.....	4
Análisis de Resultados.....	6
Manual de Usuario.....	8
Conclusión Personal.....	10
Bibliografía.....	11

Descripción del problema

Con el fin de familiarizarnos con el lenguaje C, se debe crear un programa que tiene como función principal enviar y recibir archivos entre computadoras. Para desarrollo de dicho programa, se requiere la investigación de los sockets, que es básicamente un procedimiento o un método para la comunicación entre un programa del cliente y un programa del servidor en una red. El programa debe constar de un programa emisor (encargado de enviar archivos), y otro receptor (encargado de recibir archivos).

Una vez que estén desarrollados los programas necesarios, se deben agrupar en uno solo llamado Transmisor, que va a tener un socket para enviar archivos, y otro para recibir únicamente. Este programa posteriormente se debe bifurcar en dos procesos: Emisor (cliente) y Receptor (servidor). En cuanto al envío de archivos, el proceso cliente se encarga del socket enviar, que no se ve afectado si se está recibiendo un archivo. Para la recepción de archivos, el socket de recibir descargará, guardará en el sistema, mostrará el nombre y el tamaño del archivo que está recibiendo.

Diseño del programa

Para iniciar la tarea, empezamos diseñando dos programas por separado: uno que mandara archivos y otro que hace función de receptor. En la función cliente que desarrollamos se indica que se utilizan comunicaciones orientadas a conexión TCP y UDP. El SOCK_STREAM es para TCP y SOCK_DGRAM para UDP. Además se indica la dirección local, mediante el siguiente código:

```
struct sockaddr_in *direcc_loc = malloc(sizeof(struct sockaddr_in));

{ (*direcc_loc).sin_family = AF_INET;

  (*direcc_loc).sin_addr.s_addr = INADDR_ANY;

  (*direcc_loc).sin_port = htons(atoi(argv[3]));

  bind(desc, (struct sockaddr *)direcc_loc, sizeof(struct sockaddr_in));

}
```

Y de la misma manera, la dirección remota, que es a donde se desea enviar archivos:

```
struct sockaddr_in *direcc_rem = malloc(sizeof(struct sockaddr_in));

{

  (*direcc_rem).sin_family = AF_INET;

  (*direcc_rem).sin_addr.s_addr = inet_addr(argv[1]);

  (*direcc_rem).sin_port = htons(atoi(argv[2])); }
```

Cuando se va a hacer la conexión, para conectarse al servidor se indica la dirección remota y los detalles del archivo. Si ocurre un error, el programa

envía un mensaje. Si la conexión es exitosa, también se envía un mensaje pero en este caso indicando que el envío del archivo se logro. Finalmente se cierra el socket y la conexión:

```
fclose(archivoEnviar);

close(desc);

free(direcc_loc);

free(direcc_rem);
```

En cuanto al servidor, se indica la dirección remota y la conexión entrante. Se imprime un mensaje de 'conexión establecida' si se logra la recepción. De la misma manera que en el cliente, si aparece un error a la hora de abrir el archivo, el programa lo indica. Finalmente, se cierra el socket y se liberan los espacios en memoria utilizados por las estructuras del emisor y el receptor:

```
close(desc);

free(direcc_loc);

free(direcc_rem);}
```

Una vez que logramos que sirvieran por separado, probándolos en diferentes computadoras, empezamos el proceso de unión. Para lo que desarrollamos un main que realiza mediante una llamada a fork(), se bifurca en 2 procesos en donde uno llama al servidor, encargado de los archivos entrantes, y otro al cliente que envía archivos. Se utilizarán estructuras para almacenar datos como la dirección de la otra computadora a la que se desea enviar archivos, los puertos a utilizar por cada equipo, entre otras.

Uno de los problemas que se encontraron durante la implementación de la tarea programada, fue al momento de terminar los procesos, ya que aunque terminaran la ejecución del código, seguían "vivos". Para finalizarlos se decidió utilizar la función kill().

Librerías usadas

#include <stdio.h>

Permite uso de funciones para la interacción con dispositivos de entrada y salida del computador.

#include <stdlib.h>

Permite el uso de funciones aritméticas, números aleatorios, conversión de tipos, manejo de memoria dinámica, control de procesos, etc.

#include <string.h>

Incluye funciones para el manejo de cadenas.

#include <arpa/inet.h>

Funciones para el manejo de directorios.

Análisis de resultados

- Se logró el objetivo principal, que era establecer una conexión entre varios computadores mediante los protocolos TCP y UDP.
- Se consiguió amoldarse y adaptarse al lenguaje de programación C
- Se obtuvieron resultados positivos en cuanto al envío y recepción de archivos mediante el uso de los sockets. Los argumentos, como por ejemplo: ipRemota, puertoRemoto y puertoLocal, fueron utilizados como se esperaba y reconocidos por el sistema de forma correcta.
- Los programas Cliente y Servidor trabajan de manera adecuada a la hora de mandar archivos, cada uno con su socket correspondiente.
- Se logró bifurcar el programa en dos procesos mediante el uso de fork() en el main de la función.
- Cada mensaje emitido dependiendo del proceso que se logre, es de diferente color.
- Cuando se termina la ejecución del programa, se escribe un mensaje que indica su fin y el cierre de los sockets.

Manual de usuario

Para ejecutar el programa, esto se debe digitar en la terminal de su sistema Linux:

- gcc transmisor.c -o trasmisor.o

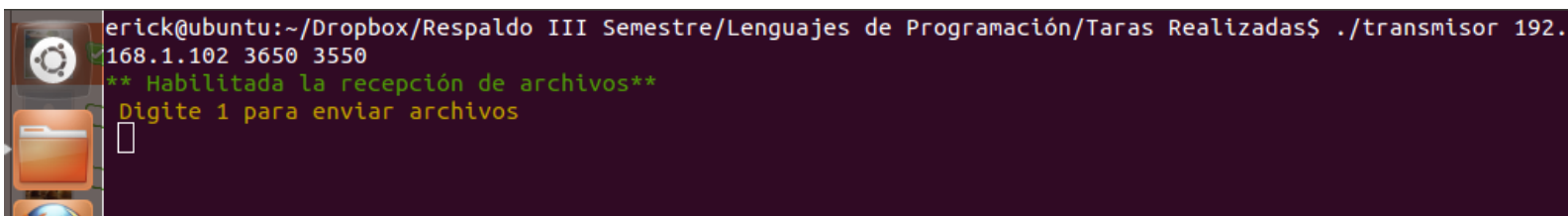
Para compilar el programa que se desarrolló.

- ./transmisor ipRemota puertoRemoto PuertoLocal

Para correrlo, ejemplo : ./transmisor 192.168.1.109 3650 3550

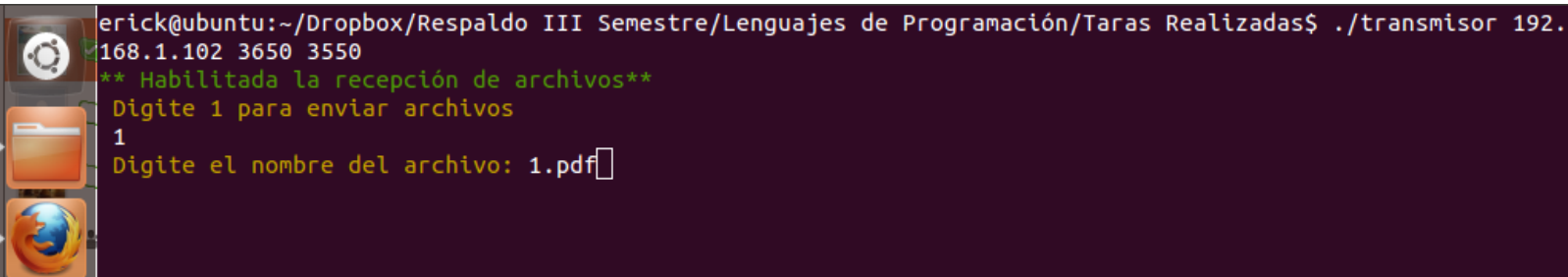
A continuación, una secuencia de imágenes con un ejemplo de una corrida del programa, que muestra los pasos básicos para el envío y recepción de archivos:

1. Recién Corrido: se escribe ./transmisor 192.168.1.109 3650 3550. Para enviar un archivo de debe ingresar '1'.

A screenshot of a Linux terminal window. The prompt is 'erick@ubuntu:~/Dropbox/Respaldo III Semestre/Lenguajes de Programación/Taras Realizadas\$'. The command entered is './transmisor 192.168.1.102 3650 3550'. The output shows a green checkmark, followed by '** Habilitada la recepción de archivos**' in green, and 'Digite 1 para enviar archivos' in yellow. A cursor is visible on the line below the prompt.

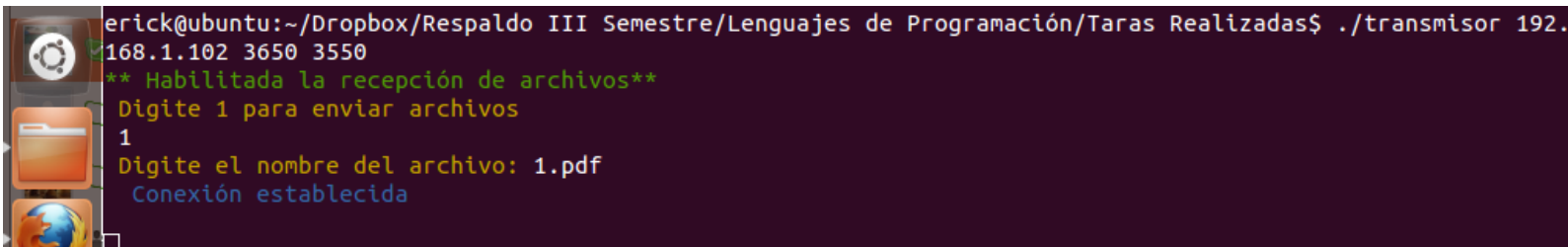
```
erick@ubuntu:~/Dropbox/Respaldo III Semestre/Lenguajes de Programación/Taras Realizadas$ ./transmisor 192.168.1.102 3650 3550
** Habilitada la recepción de archivos**
Digite 1 para enviar archivos
█
```


2. Digitar el nombre del archivo: después de que se ingresa el número, se imprime la indicación para escribir el nombre del archivo que se desea enviar. En este caso el nombre es '1.pdf'.

A terminal window with a dark purple background. The prompt is 'erick@ubuntu:~/Dropbox/Respaldo III Semestre/Lenguajes de Programación/Taras Realizadas\$'. The user has entered './transmisor 192.168.1.102 3650 3550'. The program outputs '** Habilitada la recepción de archivos**' in green. It then prompts 'Digite 1 para enviar archivos' in yellow. The user has entered '1'. It then prompts 'Digite el nombre del archivo: 1.pdf' in yellow, with a cursor at the end of the text.

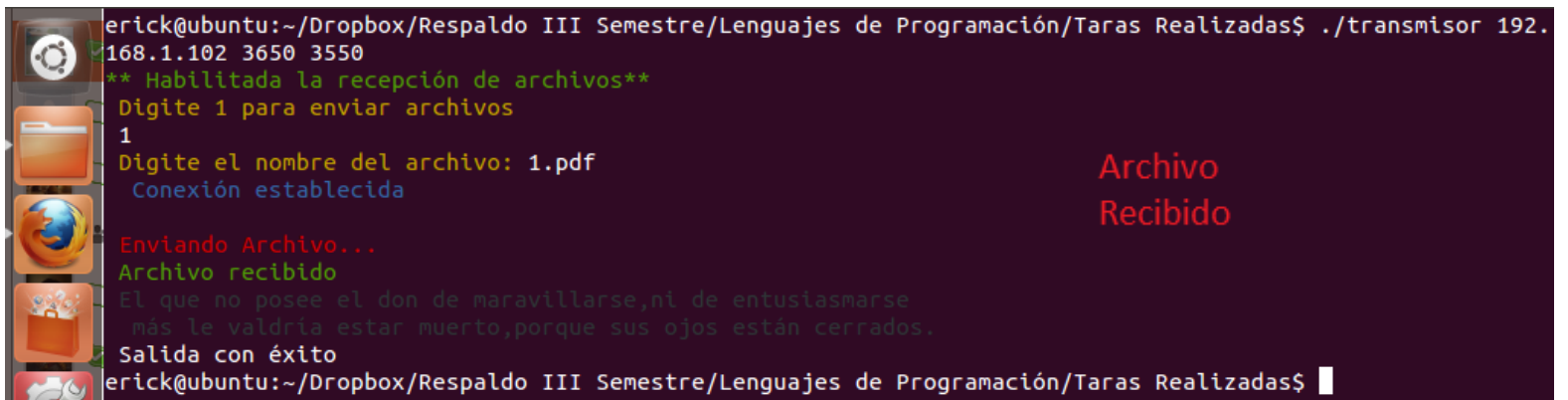
```
erick@ubuntu:~/Dropbox/Respaldo III Semestre/Lenguajes de Programación/Taras Realizadas$ ./transmisor 192.168.1.102 3650 3550
** Habilitada la recepción de archivos**
Digite 1 para enviar archivos
1
Digite el nombre del archivo: 1.pdf
```

3. Conexión Establecida: si la conexión es exitosa, el programa imprime mensaje señalándolo.

A terminal window showing the next step. The prompt is the same. The program outputs 'Conexión establecida' in blue after the user entered the filename '1.pdf'.

```
erick@ubuntu:~/Dropbox/Respaldo III Semestre/Lenguajes de Programación/Taras Realizadas$ ./transmisor 192.168.1.102 3650 3550
** Habilitada la recepción de archivos**
Digite 1 para enviar archivos
1
Digite el nombre del archivo: 1.pdf
Conexión establecida
```

4. Envío completado: una vez que se manda el archivo con éxito, se imprime 'Archivo recibido', y se finaliza el proceso.

A terminal window showing the final steps. The program outputs 'Enviando Archivo...' in red, 'Archivo recibido' in green, and a quote 'El que no posee el don de maravillarse,ni de entusiasmarse más le valdria estar muerto,porque sus ojos están cerrados.' in green. It then outputs 'Salida con éxito' in green. The prompt returns to the user. To the right of the terminal, the text 'Archivo Recibido' is written in red.

```
erick@ubuntu:~/Dropbox/Respaldo III Semestre/Lenguajes de Programación/Taras Realizadas$ ./transmisor 192.168.1.102 3650 3550
** Habilitada la recepción de archivos**
Digite 1 para enviar archivos
1
Digite el nombre del archivo: 1.pdf
Conexión establecida
Enviando Archivo...
Archivo recibido
El que no posee el don de maravillarse,ni de entusiasmarse
más le valdria estar muerto,porque sus ojos están cerrados.
Salida con éxito
erick@ubuntu:~/Dropbox/Respaldo III Semestre/Lenguajes de Programación/Taras Realizadas$
```

Archivo
Recibido

Conclusiones personales

Los sockets son un canal de comunicación, una manera de lograr que dos programas se transmitan datos utilizando un cliente y un servidor. Al realizar esta tarea se aprendió sobre los elementos indispensables para poder lograr la conexión y mandar uno o más archivos y que es necesario incluir en el código del programa, tales como dirección remota, dirección local, etc.

También es importante que aprendamos a buscar modos de acopiar datos, documentos, apuntes, reseñas que nos ayuden a cumplir con todos los requisitos de la tarea. Además de ampliar el conocimiento, se fomentó la investigación, exploración e indagación de formas para recolectar información.

Para nosotros, como administradores de tecnologías de información es muy importante tener un conocimiento amplio sobre los distintos lenguajes de programación. En el caso de esta tarea, a causa de la gran popularidad del lenguaje C y las oportunidades de trabajo que puedan surgir relacionadas con el mismo, es de gran beneficio poder familiarizarnos con el desarrollo de programas de este tipo.

Nos entretuvo bastante y consideramos de mucho provecho para nuestro aprendizaje la realización de esta tarea programada.

Referencias

Páginas Web:

<http://www.fismat.umich.mx/mn1/manual/node2.html>

http://www.chuidiang.com/clinix/sockets/sockets_simp.php

<http://www.masadelante.com/faqs/socket>

<http://www.masadelante.com/faqs/socket>

<http://es.tldp.org/Tutoriales/PROG-SOCKETS/prog-sockets.html>

<http://www.thegeekstuff.com/2011/12/c-socket-programming/>

Videos:

[http://www.youtube.com/watch?v=-](http://www.youtube.com/watch?v=-Gpmkk8Zghk)

[Gpmkk8Zghk&playnext=1&list=PL78386798B27755B9&feature=results_main](http://www.youtube.com/watch?v=-Gpmkk8Zghk&playnext=1&list=PL78386798B27755B9&feature=results_main)