Instituto Tecnológico de Costa Rica
Administración de Tecnologías de Información
Lenguajes de Programación
Prof: Andréi Fuentes L.
Marzo, 2013

TAREA PROGRAMADA I

Introducción

El objetivo de esta tarea es familiarizarse con el desarrollo de programas en C, mediante la creación de un programa de intercambio de archivos entre computadoras. Para realizar la comunicación entre las computadoras y poder permitir el envío de archivos entre una computadora 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).

Descripción del programa

Para simplificar el desarrollo de la tarea, se recomienda empezar por una solución sencilla y luego irle añadiendo funcionalidad compleja.

La idea es que se puedan enviar y recibir archivos, desde y hacia otras computadoras. Inicialmente, deben lograr que un programa Emisor envíe un archivo a otro programa Receptor, y que el programa Receptor pueda responder al programa Emisor en caso de que haya funcionado correctamente la transferencia. Una vez que se haya terminado la transferencia, se deberán terminar los programas.

Al inicio, pueden ejecutar el Emisor y el Receptor en la misma computadora. Luego deberán ejecutar los programas en computadoras diferentes en la misma red. Posteriormente podrían ejecutar los programas de manera remota, por ejemplo, Emisor ejecutándose en las computadoras de los laboratorios y Receptor ejecutándose en la computadora de la casa.

En los programas descritos anteriormente, el Emisor sólo puede enviar archivos, y el Receptor sólo puede recibir archivos. La idea es que ambos puedan enviar archivos de forma simultánea, para lo que se deberán hacer algunas modificaciones para incorporar esa funcionalidad.

La idea es que una vez que Emisor y Receptor estén desarrollados, se unifiquen en un mismo programa Transmisor. Este programa creará dos sockets: uno para enviar archivos, y el otro sólo para recibir archivos. Después, mediante el uso de `fork()` el programa se bifurca en dos procesos, el Emisor (cliente) y el Receptor (servidor).

El proceso cliente atiende el socket de enviar, y cuando el usuario ingrese el nombre de un archivo, enviará el archivo (independientemente de que el socket servidor esté recibiendo un archivo). El proceso servidor atenderá el socket de recibir, y cada vez que reciba un archivo, lo descargará, lo guardará en el sistema de archivos, mostrará el nombre y el tamaño del archivo en pantalla y continuará esperando por otro archivo.

Los archivos pueden tener un tamaño máximo, el cual podrá ser definido por los estudiantes. Los archivos pueden ser especificados en tiempo de ejecución.

La idea es que el programa se ejecute de la siguiente forma:

```
./transmisor ipRemota puertoRemoto puertoLocal
Digite el nombre del archivo:/Users/juan/Docs/file.txt
Enviando archivo…
Servidor remoto: file.txt recibido correctamente
Servidor local: recibiendo archivo file2.txt
Servidor local: file2.txt recibido correctamente
```

Los argumentos de línea de comandos del programa son los siguientes:

- `ipRemota`: es el ip del transmisor remoto
- `puertoRemoto`: es el puerto del transmisor remoto
- `puertoLocal`: es el puerto en el cual se deberá abrir el socket a nivel local para recibir archivos

Los nombres de archivos no deben recibirse como argumentos de línea de comandos, sino que serán digitados por el usuario en tiempo de ejecución (en el caso anterior, sería el texto en azul).

Pueden usar colores para diferenciar los mensajes emitidos por el transmisor local, y los mensajes recibidos del transmisor remoto.

Una copia del transmisor se ejecutará en una computadora, y otra copia en otra. Para finalizar la ejecución, alguna de las copias escribirá el mensaje "`Finalizar`", y se cerrarán los sockets. La otra copia detectará el mensaje "`Finalizar`", y, después de desplegar el mensaje, cerrará sus sockets. Deben usar ese mensaje para finalizar la ejecución del programa, ya que la idea es poder comunicar programas de diferentes grupos.

Funciones a investigar

Para desarrollar algunas de las funcionalidades del programa, deberán investigar el uso de los sockets de red en Linux. Además, deberán investigar cómo manejar diferentes procesos mediante el uso de fork.

Aspectos técnicos

El proyecto deberá estar escrito en el lenguaje de programación C (no C++), y deberá de funcionar en el sistema operativo Linux. En caso de requerir librerías adicionales para compilar y ejecutar el programa, deberán especificarlo en la documentación, ya que de lo contrario se descontarán puntos en la evaluación.

Documentación

La documentación es un aspecto de gran importancia en el desarrollo de programas, especialmente en tareas relacionadas con el mantenimiento de los mismos.

Para la documentación interna, deberán incluir comentarios descriptivos para cada función, con sus entradas, salidas, y restricciones.

La documentación externa deberá incluir:

- Tabla de contenidos
- Descripción del problema
- Diseño del programa: decisiones de diseño, algoritmos usados, diagramas lógicos
- Librerías usadas: sockets de Unix, etc
- Análisis de resultados: objetivos alcanzados, objetivos no alcanzados, y razones por las cuales no se alcanzaron los objetivos (en caso de haberlos)
- Manual de usuario: instrucciones uso, de ejecución y de compilación
- Conclusión personal

Evaluación

Documentación interna	2%
Documentación externa	13%
Funcionalidad de mensajería	45%
Revisión de tarea	40%

Aspectos administrativos

- La tarea vale un 10% de la nota del curso
- La tarea se hará en grupos de 2-3 personas.
- Fecha de entrega: Martes 2 de abril, 9 a.m. No se aceptan tareas entregadas después de esa fecha y hora.
- Los grupos deberán subir el código y la documentación de sus respectivas tareas a un repositorio en Github, de manera que el profesor pueda ver las contribuciones que las diferentes personas hacen al proyecto. La idea es que apenas empiecen a desarrollar la tarea, suban las contribuciones al repositorio, y no esperar a tener todo el código listo para subirlo.
- Deberán enviar un correo a andreifu@gmail.com, con copia a evelyn.madriz@gmail.com, en donde indiquen el url del repositorio de Github en donde se encuentra el código y la documentación de la tarea. El asunto del correo enviado tendrá el siguiente formato: `TI-3404 TP1-nombrea1-nombre2`. Las tareas que no sean entregadas por medio de Github tendrán nota cero.

• Las tareas deberán ser revisadas con el profesor o el asistente. Todos los miembros del grupo deberán participar de la revisión, ya que de lo contrario no se les asignará el puntaje correspondiente. La nota de la revisión es individual, el resto de la nota es grupal.

• El código entregado debe ser 100% original. En caso de probarse algún tipo de fraude en la elaboración de la tarea, se aplicarán todas las medidas correspondientes, según el reglamento del TEC, incluyendo una carta al expediente.