Informe de Trabajo Práctico Nº 1

7 de abril de 2016

75.42 Taller de Programación I

Cátedra Veiga

Facultad de Ingeniera - UBA

Autor: Martín Stancanelli

Padrón: 95188

Objetivo

El objetivo del trabajo práctico fue utilizar los conocimientos obtenidos en las clases de sockets y TDAs. Para ello se usaron las librerías de socket provistas por el entorno y se desarrollaron un conjunto de TDAs que se explicarán en secciones posteriores.

Configuración del entorno

El entorno de trabajo elegido para la realización del trabajo practico fue una PC con SO Linux Mint 17.3 y compilador gcc 4.8 configurado para utilizar el estándar de C99.

Explicación de enunciado

El trabajo consiste en realizar un sistema que haga una sincronización de archivos entre un cliente y un servidor. El cliente tendrá una copia local de un archivo y el servidor una nueva versión del mismo. El objetivo es que el cliente notifique al servidor el archivo que tiene a través de checksums de bloques de bytes utilizando una variante del *Adler32.*

El servidor recibe los checksums del cliente y compara contra los checksums de su archivo remoto. Para los que coincidan se le notificará al cliente el bloque para el cual el checksum coincide, evitando así reenviar todo el checksum y reutilizando el archivo local del cliente. En caso de que haya un bloque nuevo se le enviara al cliente los nuevos bytes que debe agregar a su archivo. Finalmente el cliente escribe en un nuevo archivo los datos que ya tenía y los que recibe del lado del servidor.

Desarrollo

A continuación se detallara el algoritmo más complejo del trabajo: donde el servidor compara los checksums recibidos con los de su archivo remoto y notifica al cliente.

En primer lugar el servidor abre su archivo local (*target*). Carga el primer bloque del archivo en memoria y le calcula su checksum usando la variante de *Adler32.* Seguido de esto comienza a iterar. En caso de que el checksum no este en la lista de checksums que recibió del cliente, desplaza la ventana de lectura del *target* en 1 byte y guarda el byte que quedo fuera de la ventana en un buffer. A continuación calcula el *rolling checksum*[1] del nuevo bloque y reinicia la iteración.

En el caso en que el checksum este en la lista de checksums que el cliente le envió, entonces se fija si tiene bytes que hayan sido desplazados de la ventana y guardados en el buffer. Si hay, le envía esos bytes al cliente. Si no hay bytes desplazados, envía al cliente el numero de bloque que este tiene en su archivo local para el cual el checksum del archivo *target* coincide. Para terminar mueve

[1] *Se calcula el rolling checksum ya que es mas eficiente que volver a calcular el checksum del bloque entero*

la ventana de lectura en un bloque, le calcula su checksum y reinicia la iteración.

Se incluye un diagrama de flujo (1) explicando este algoritmo

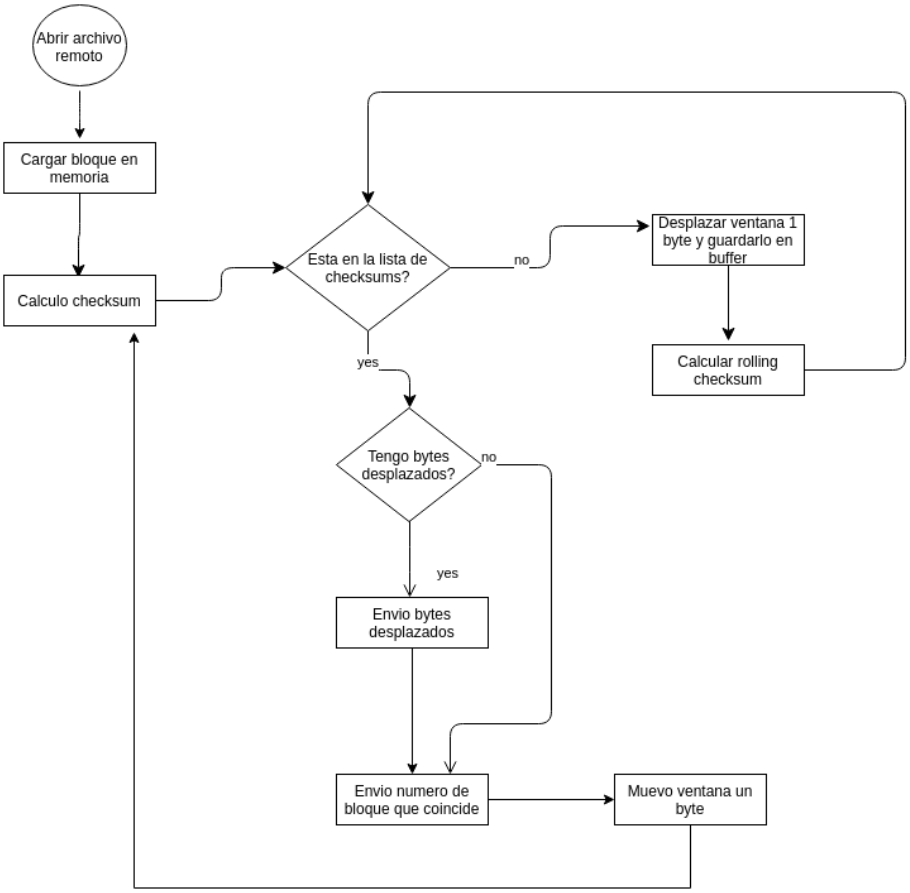

Figura 1: Diagrama de flujo

Conclusión

En este trabajo se utilizaron conocimientos nuevos, aprendidos en las clases de la materia, sobre sockets y se reutilizaron algunos como los de TDAs que habíamos incorporado de materias anteriores. Si bien al principio el problema de los sockets puede ser un poco complejo, una correcta utilización de TDAs y por ende un correcto encapsulamiento permite que nos podamos enfocar de lleno en lo mas importante del enunciado del trabajo práctico, como lo es la sincronización de archivos.