

Escuela de Ingeniería en Computación

Principio de Sistemas Operativos

**Programa sincronizador de archivos en C**

**Estudiante:**

José Mario Naranjo Leiva

Gabriel Ramírez Ramírez

**Carnet:**

2013034348

2010

**Profesor:**

Armando Arce Orozco

Primer Semestre, 2017

**Tabla de contenido**

[Tabla de contenido 2](#_Toc485046805)

[1. Introducción 3](#_Toc485046806)

[2. Descripción del problema 3](#_Toc485046807)

[2.1. Programa Sincronizador 3](#_Toc485046808)

[3. Definición de estructuras de datos 4](#_Toc485046809)

[4. Descripción detallada 4](#_Toc485046810)

[4.1. Manejo de sockets 4](#_Toc485046811)

[4.2. Manejo de solicitudes 5](#_Toc485046812)

[4.3. Rutina de comparación 5](#_Toc485046813)

[4.3.1. Identificación de archivos 5](#_Toc485046814)

[4.3.2. Comparación 5](#_Toc485046815)

[5. Descripción del protocolos y formatos 5](#_Toc485046816)

[6. Análisis de resultados de pruebas 6](#_Toc485046817)

[7. Conclusiones sobre rendimiento y funcionamiento 6](#_Toc485046818)

1. Introducción
2. Descripción del problema

Un problema que presentan los diferentes sistemas de sincronización de archivos (como Dropbox, LiveDrive y SugarSync) es que solo funcionan conectados al Internet. Desde antes existían aplicaciones sencillas que permitían sincronizar máquinas en una red sin conectarse a la Web (por ejemplo, una laptop y una computadora de escritorio).

* 1. Programa Sincronizador

El programa sincronizador se debe ejecutar en dos máquinas al mismo tiempo. En la primera máquina el programa recibirá como parámetro únicamente el nombre del directorio a sincronizar y quedará esperando conexiones en el puerto 8889. En la otra máquina se debe pasar como parámetros el nombre del directorio local a sincronizar y el IP de la primera máquina.

Al iniciarse el programa deberá identificar los archivos en su directorio local, almacenando su tamaño, hora y fecha de la última actualización. Debe comparar dicha información con un listado de su corrida anterior, y determinar cuáles archivos han cambiado. Si algún archivo ha cambiado, se eliminó, o se creó, se debe enviar esa información a la otra máquina para que realice los cambios adecuados. Dentro de dichos cambios está solicitar el archivo actualizado a la otra máquina.

* Una vez que se realiza la sincronización, los programas que corren en las máquinas deben terminar. La siguiente vez que se desee sincronizar se debe volver a ejecutar el programa en ambas máquinas.
* Tome en cuenta que los relojes de las máquinas involucradas pueden estar de sincronizados. Para determinar cuáles archivos son más recientes se debe realizar un cálculo de diferencias entre las horas de ambos relojes.
* NO se deben escribir dos programas diferentes. Es el mismo programa que actúa como cliente o servidor dependiendo de los parámetros que se le pasen.
* Debe utilizar sockets para que se comuniquen los programas. Además, debe definir un formato y protocolo adecuado para enviar y recibir las solicitudes de archivos.
* Debe tomar en cuenta la posibilidad que un archivo haya sido modificado en ambas máquinas, provocando el caso de “copias en conflicto”. Aquí lo mejor es dejar las dos copias, pero con nombres ligeramente diferentes.

1. Definición de estructuras de datos

Para la implementación adecuada de este programa sincronizador, se implementaron las siguientes estructuras de datos.

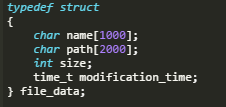


Ilustración 1: Estructura para almacenar los datos del archivo

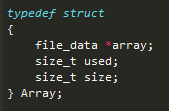


Ilustración 2: Estructura para un arreglo dinámico de archivos

* **Lista**

Para manejar los archivos contenidos en el directorio, se implementó una lista dinámica enlazada, para almacenar la estructura *file\_data* (ver Ilustración 1) que contiene los datos relevantes de un archivo: nombre, carpeta, tamaño y fecha de modificación.

El nodo de la lista enlazada está definido en la estructura *Array* (ver Ilustración 2). Un diagrama general de esta estructura está en Ilustración 3.

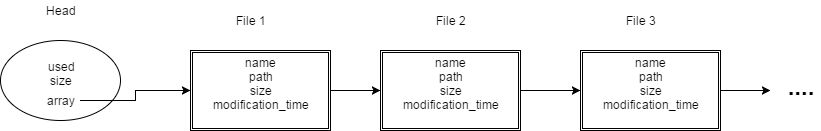


Ilustración Vista general de la estructura

* **Estructuras de sincronización**

Las estructuras de sincronización son definidas para el uso adecuado de la comunicación de las computadoras y la sincronización de los archivos. Estas se ven con más detalle en la sección

1. Descripción detallada

A continuación, se muestra la explicación de los componentes principales del servidor y el cliente del sistema sincronizador.

* 1. Manejo de sockets

Los sockets son parte importante para la implementación del proyecto. Los sockets Berkeley son usados para la comunicación de ambos programas.

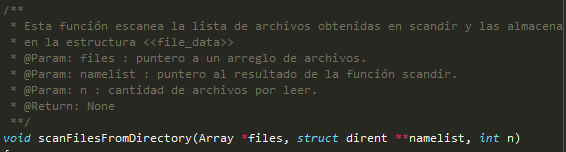
Estos sockets son un API para el uso de sockets de internet y de dominio UNIX, usados para la intercomunicación de procesos. Por sus orígenes, estos usan el estándar de componentes POSIX.

* 1. Manejo de solicitudes
  2. Rutina de comparación

A continuación, se explica el algoritmo de comparación implementado para determinar las diferencias en los archivos anteriores y actuales del directorio por sincronizar.

### Identificación de archivos

Para la lectura de los archivos del directorio se usa la funcionalidad “*scandir*”, definida en el encabezado **#include <dirent.h>**. Esta permite escanear el directorio y cada archivo es almacenado en una estructura dinámica, para el procesamiento posterior de los archivos.

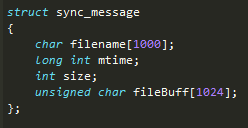


Luego, se usa la función para obtener los datos de los archivos del directorio y almacenarlos en una lista dinámica. Posteriormente, esta lista de archivos actuales es almacenada en un fichero en el directorio <<*.meta>>*, con el objetivo de realizar la comparación con los archivos locales actuales y los archivos de la corrida anterior.

### Comparación

1. Descripción del protocolos y formatos

A continuación, se explican el formato y protocolo definidos para el envió y recibimiento de solicitudes de archivos.



1. Análisis de resultados de pruebas
2. Conclusiones sobre rendimiento y funcionamiento