Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3825 - Sistema de Operación I

Proyecto 3: Sincronizando Directorios

Estudiantes:
Mauricio Fragachan: 20-10265
Jesus Gutierrez: 20-10332

Profesor:
Guillermo Palma

Sartenejas, Abril de 2025

Diseño de solución del problema general

El problema a resolver es que dado dos directorios, debemos sincronizarlos de manera tal que si existen dos subdirectorios o dos archivos en ambos directorios, entonces se debe proceder a recomendarle al usuario mantener el archivo o directorio que fue modificado más recientemente. Para ello, se creó una función **`sync_dirs`** que hace uso de otras 4 funciones auxiliares:

`cp_file_to_dir:` se encarga de copiar un archivo de un directorio a otro. Recibe como parámetros la ruta de un archivo y un directorio de destino. Lógica que ejecuta la función:

1) Se construye la ruta destino para el archivo a ser copiado
2) Se abre el archivo fuente y se obtienen sus metadatos
3) Se crea el archivo de destino asignando los mismos permisos que tiene establecido el fuente
4) Se procede a copiar el contenido del archivo fuente al de destino
5) Se cierran los archivos

`cp_dir_to_dir:` se encarga de copiar un subdirectorio recursivamente de un directorio a otro. Lógica que ejecuta la función:

1) Se inicializa el struct a retornar que contendrá el peso total del directorio y los archivos totales que contiene
2) Se abre el directorio fuente y se procede a crear el directorio destino
3) Se lee el contenido del directorio fuente
    - Se construyen las rutas del directorio fuente que se usarán para crear los archivos o subdirectorios en el directorio destino
    - Se obtiene la metadata del archivo o subdirectorio
    - Se verifica si la entrada es un archivo o directorio
        - Si es un directorio, se llama recursivamente a **`cp_dir_to_dir`** y luego, se actualiza la data a ser retornada usando los valores devueltos por la llamada recursiva.
        - Si es un archivo, se incrementa la cantidad de archivos que contiene el directorio, se suma el peso del dicho archivo al campo correspondiente del struct a ser retornado y se llama a **`cp_file_to_dir`** para poder copiar dicho archivo a la ruta correspondiente del directorio destino.
    - Se cierra el directorio fuente y se retornan los datos ya mencionados.

`same_content_file:` se encarga de verificar si dos archivos contienen el mismo contenido. Lógica que ejecuta la función:

1) Se abren los archivos, cuyo contenido será comparado
2) Se inicializan variables a ser usadas para la lectura y comparación de los archivos
3) Se leen y comparan los primeros bloque
    - Si uno de los archivos está vacío y el otro no, entonces se cierran los archivos y se retorna “0” para indicar que los archivos no tienen el mismo contenido
    - Se comparan los tamaños y contenidos de estos primeros bloques, si el tamaño de los bloques es diferente o si los contenidos son distintos, entonces, se cierran los archivos y se retorna “0” indicando que los archivos no tienen el mismo contenido.

4) Se leen y comparan los siguientes bloques de los archivos de manera iterativa, tal que si en algún momento se detecta que un par de bloques tienen tamaños distintos o tienen contenidos distintos, entonces, se retorna “0”
5) Se cierran los archivos y se retorna “1” en caso de que no se haya caído en ninguno de los retornos anteriores

`rm_dir:` se encarga de eliminar un directorio de manera recursiva. Recibe como parámetro la ruta hacia el directorio. Lógica que ejecuta la función:

1) Se abre el directorio y se lee en bucle
    - Se aplica un condicional para omitir los directorios (.) y (..)
    - Se construye la ruta completa de cada entrada y se determina si dicha entrada es un archivo o subdirectorio
    - Si es un subdirectorio, se llama **`rm_dir`** de manera recursiva, sino, significa que es un archivo y por lo tanto, se procede a eliminar directamente.
2) Se cierra el directorio y se elimina dicho directorio vacío, ya que previamente ya se eliminaron los archivos que contiene.

Luego, **`sync_dirs`** ejecuta la siguiente lógica:

1) Se verifica que los dos directorios sean válidos

2) Se itera a través del contenido del primer directorio (D1) y se compara con el contenido del segundo directorio (D2)
    - Para archivos o subdirectorios que estén en D1, pero no en D2, se le solicita al usuario si quiere copiarlos de D1 a D2 o si quiere borrarlos de D1
    - Para archivos con el mismo nombre en ambos directorios, se verifica si su contenido difiere. Si es así, se compara si el tiempo de modificación del archivo en D1 es mayor que el que está en D2. Si es así, se solicita al usuario actualizar el más antiguo (el de D2) al más reciente (el de D1) o si quiere mantener es el más antiguo en ambos casos.

3) Se hace una llamada recursiva para sincronizar los archivos en los siguientes niveles de los directorios

4) Además, la función retorna un struct con los datos del peso total transferido desde D1 a D2 (y de D2 a D1 en caso de que en algún momento un archivo antiguo estuviera en D2 y se quisiera mantener), y la cantidad de archivos que se transfirieron.

Para finalizar, en la función main, se hacen dos llamadas a la función **`sync_dirs`**, de tal manera que se pueda hacer la sincronización de manera bidireccional, es decir, desde D1 hasta D2, y desde D2 a D1. Y luego, usando los datos retornados por ambas llamadas, se unifican para hacer el cálculo y mostrar por consola el peso final (en Kilobytes) transferido de D1 a D2 y de D2 a D1, más la cantidad de archivos transferida en ambos casos.