Proyecto #1
(Buscador de archivos duplicados)

Motivación:
Ha habido un gran interés en la comunidad de investigación recientemente en la cantidad de duplicación de datos: por ejemplo, el mismo archivo puede almacenarse en varios equipos, los mismos datos pueden ser enviados a través de redes a múltiples destinatarios o varias veces al mismo destinatario, o una sola máquina puede almacenar múltiples copias independientes de los mismos datos. Funcionalmente, esta asignación es sobre el último de ellos: ¿Qué tan común es que archivos distintos en un solo equipo puedan contener los mismos datos?

Diseño del comparador:

El proyecto consiste en escribir un programa en C que recorra un sub-árbol de archivos completo, empezando por el directorio que se le indicará al programa y determina qué archivos (si los hay) son duplicados de otros. Para ello, se hará una comparación por pares de todos los archivos que encuentre.
Si usted hace una simple comparación por pares de N archivos de longitud L, el tiempo requerido es $O(N^2L)$. En la práctica, el tiempo de ejecución se puede acelerar en gran medida mediante el uso de "hashes criptográficos" de los contenidos del archivo. He aquí una cita del RFC1321 sobre MD5, la función hash criptográfica que usaremos:

“El algoritmo toma como entrada un mensaje de longitud arbitraria y produce como salida una "huella" de 128 bits o "resumen del mensaje" de entrada. Se conjetura que es computacionalmente inviable encontrar dos mensajes que tengan el mismo resumen, u obtener un mensaje que tenga un resumen de mensaje en concreto, establecido previamente como objetivo. El algoritmo MD5 está pensado para ser aplicado en firmas digitales, donde un archivo grande debe ser "comprimido" de forma segura antes de ser cifrado con una llave privada (secreta) dentro de un sistema de cifrado de llave pública, como RSA.”

Lo que esto significa para nosotros es que las probabilidades de que dos archivos con diferente contenido produzcan el mismo valor de hash es prácticamente nula, por lo que la comparación de los valores hash de archivos es (probabilísticamente) lo suficiente como para determinar si sus contenidos son los mismos. Nosotros nos basaremos en eso para acelerar el proceso de comparación de archivos.

Implementación:

Su programa debe verificar la existencia de archivos duplicados en el directorio actual y sus respectivos subdirectorios de la siguiente forma:

1. Crear dos estructuras de datos: una que contendrá los registros de archivos "a visitar" y otra que contendrá los registros de los “visitados".
2. Inicializar las estructuras: de ser una estructura de nodos “a visitar” se inicializa para contener un registro que indique el directorio que se le pasó al programa en los argumentos. La estructura que contendrá los nodos “visitados” se inicializa vacía.
3. Mientras que la estructura de nodos “a visitar” no esté vacía, se hace un ciclo de los pasos 4-8.
4. Obtener el siguiente nodo “a visitar”.
5. Determinar si el archivo es un directorio, un enlace simbólico, o un archivo de datos.
6. Si es un enlace simbólico se ignora.
7. Si se trata de un directorio, enumerar los archivos que contiene. Para todos los archivos que no sean "." y "..", guardar registros acerca de ellos en la estructura de datos “a visitar”.
8. Si se trata de un archivo de datos que no esté vacío, calcular el hash MD5 de los contenidos del archivo y comprobar la igualdad contra los hashes de todos los archivos en la estructura de datos “visitados”. Agregue el archivo que se acaba de verificar a “visitados”.
9. Al terminar de procesar los archivos imprimir las estadísticas recogidas durante la ejecución.

Hashes MD5:

Se puede escribir código para calcular hashes MD5, o hacer uso de una biblioteca ya escrita. Se le proporcionarán los archivos de esta última de dos formas:
1) Modo ejecutable: Un conjunto de archivos que se encuentran en el directorio md5-app.
Para hacer uso del ejecutable debe invocar el comando make y obtenido el ejecutable escribir en la línea de comandos:

./md5 <nombre archivo>

Y a continuación se imprimirá el hash MD5 del archivo de nombre <nombre archivo>
Para utilizar esta versión (binario), es necesario conectarse a una tubería entre

una invocación con fork del código md5 y su programa. Una vez creada la tubería y conectada al proceso md5, su programa sólo puede leer el hash de la tubería.
2) Modo biblioteca: Un conjunto de archivos que se encuentran en el directorio md5-lib.
Para hacer uso de la biblioteca debe invocar el comando make y luego de obtenido el archivo de biblioteca estática (libmd5.a) debe enlazarlo a su proyecto y utilizar la función MDFile cuya firma es:

```
int MDFile(char* filename, char hashValue[33]);
```

Donde filename es el nombre del archivo al cual se le calculará el hash MD5 y en hashValue se almacenará el respectivo hash. El valor de retorno de la función es 1 si se realizó la operación de forma exitosa o 0 si hubo un error.

El modo de funcionamiento se indicará por argumentos de la línea de comandos con la opción –m seguida de las letras e (modo ejecutable) o l (modo biblioteca).

Programación Multihilo:

Este programa debe ser implementado usando los hilos de la biblioteca pthread. Su programa debe aceptar como argumento de línea de comandos el número de hilos que deben trabajar en el problema a la vez. Así mismo debe utilizar las primitivas de bloqueo y sincronización (semáforos, variables de condición, etc.) que considere necesarias para la correcta ejecución de su programa.

Consideraciones de implementación:

El formato de ejecución de su programa debe ser el siguiente:

$> ./duplicados -t <numero de threads> -d <directorio de inicio> -m <e | l >

Donde:

<numero de threads>: cantidad de hilos que trabajaran en el problema.

<directorio de inicio>: ruta (absoluta o relativa) del directorio en el cual se buscarán archivos duplicados.

<e | l >: Modo ejecutable o modo biblioteca

El formato de las estadísticas a imprimir al finalizar el programa debe ser el siguiente:

Se han encontrado <número de duplicados> **archivos duplicados.**

<nombre archivo 1> **es duplicado de** <nombre archivo duplicado 1>

.

.

.

<nombre archivo N> **es duplicado de** <nombre archivo duplicado N>

Donde:

<número de duplicados> es el número de archivos a los que se les ha encontrado una

copia.

Entregables

Su proyecto debe contener al menos los siguientes archivos al momento de ser entregado para revisión:

– Sus archivos de código fuente en C (archivos .c) y cabeceras relacionadas (archivos .h) necesarios para compilar el proyecto.
– Un archivo Makefile que permita compilar el proyecto. El Makefile debe contener al menos las siguientes reglas:
1. all: debe permitir compilar el proyecto completo y producir el programa ejecutable correspondiente.
2. clean: debe eliminar los archivos intermedios de compilación que se produzcan (por ejemplo, los archivos .o) y el ejecutable del proyecto.
– Un informe de minimo 8 páginas en formato .PDF que contenga una descripción del programa desarrollado. Debe describir como está estructurado el código fuente e identificar los puntos de su código fuente donde se implementaron las funcionalidades que considere más relevantes. Se debe anexar cada descripcion/explicacion con sus respectivas capturas.

Consideraciones Generales:

– El proyecto deberá realizarse en grupo de 2 (dos) personas máximo.
– Para compilar el proyecto creado debe hacerse uso de un Makefile, el cual debe ser desarrollado por cada grupo, **si el codigo es enviado sin el Makefile o el Makefile no funciona, el codigo no sera evaluado.**
– Los estudiantes son libres de organizar su código en diferentes cabeceras y secciones de código, **es obligatorio el desarrollo y uso de cabeceras. De no cumplirse tendra una penalizacion de 2 puntos.**
– La implementación del proyecto será realizada utilizando el lenguaje de programación C, bajo un sistema Linux, de preferencia compatible con el sistema Ubuntu 24.04 LTS o derivados.
– Usted deberá entregar el código fuente correspondiente al proyecto totalmente funcional y sin errores. **Aquellos códigos que presenten errores de compilación no serán evaluados.**
– No se aceptarán entregas fuera de la fecha y hora establecida.
– De acuerdo con lo establecido en la Ley de Universidades, las copias serán penalizadas con la nota mínima (0) para todos los involucrados.
– Se prohibe el uso de IA generativas (LLM) para desarrollar el proyecto, de ser detectado **la nota sera severamente penalizada con la nota mínima (0).**
– El asunto del correo debe ser [SO]_[Proy1]_[Cedula1]_[Cedula2], si no cumple con el formato del asunto, **la nota sera severamente penalizada. De no cumplirse tendra una penalizacion de 5 puntos.**
– El codigo debe estar escrito de una manera clara, ordenada y bien documentada/comentada.
– Los entregables del proyecto deben ser enviados en un archivo .zip al correo laboratoriosisop@gmail.com.
– La fecha de entrega sera determinada por el GDSO, se le avisara por correo y por el grupo de Telegram de la materia.

La distribucion de puntos queda de la siguiente manera:

Informe - 4 pts
Defensa individual - 3 pts
Documetacion/comentarios - 2 pts
Funcionalidad del codigo - 11 pts