República Bolivariana de Venezuela
Universidad Simón Bolívar
CI3825 - Sistemas de Operación I
Profesor: Fernando Torre Mora

Diseño de Buscador de Archivos

Integrantes:
Ian Goldberg 14-10406
Manuel Rodriguez 13-11223
David Segura 13-11341

Introducción

Mediante el nombre de un archivo dado se debe realizar una búsqueda del mismo, lo cual se realiza tanto a través de la búsqueda en un índice con información de archivos previamente almacenada como directamente entre los archivo alcanzables desde un directorio específico y, de encontrar algún archivo que concuerde con el nombre dado, se imprimirá en consola su ruta.

El informe cuenta con 6 secciones las cuales tratarán respectivamente los siguientes temas: cómo compilar y correr el programa, estrategia de creación de hilos, estrategia de exclusión mutua usada, función de hash implementada, el formato de archivo escogido y por último una sección de comentarios adicionales acerca de nuestra solución.

Sección 1: Como correr y compilar el programa

Para ejecutar el programa es necesario ejecutar la instrucción “make”. La misma creara todos los archivos necesarios para la ejecución del programa. Para ejecutar el buscador-indexador ejecute el archivo “buscador_indexador”. Si no se introduce ningún término de búsqueda el programa finaliza con un error.

Además se puede realizar la ejecución del archivo seguido de los flags opcionales dados en el enunciado del proyecto, si alguno de ellos requiere argumentos y el mismo no es dado por el usuario se generará un error. Por dificultades a la hora de la realización de este proyecto los flags -u y -a no están disponibles.

Después de la ejecución del programa puede limpiar los archivos creados por el make con la instrucción “make clean”.

Sección 2: Estrategia de Creación de Hilos

Para el desarrollo de nuestra solución decidimos implementar una estrategia basada en la creación de dos hilos, a través de la función pthread_create(). El primero tiene el propósito de cargar la información que se encuentra en el archivo de índices en la tabla de hash, posterior a eso busca dentro de ella y revisa si encuentra alguna ruta que tenga como clave asociada el término que se ingresó como parámetro de búsqueda y, de encontrar una, la imprime en consola. El segundo hilo se encarga de indizar los archivos alcanzables desde el directorio ingresado como parámetro o, de no haberse ingresado alguno, el directorio por defecto. Ésto consiste en navegar entre los directorios y archivos alcanzados por éste y al encontrar un archivo revisar si se encuentra en el índice (cargado en la tabla de hash) y de no estarlo ingresa su ruta en el archivo de índices tras imprimirla en consola.

El main antes de finalizar el programa hace la llamada a pthread_join para esperar que la búsqueda realizada por cada hilo concluya con éxito.

Sección 3: Estrategia de Exclusión Mútua

Al momento de implementar el par de hilos, los cuales realizan la tarea de cargar índice, buscar e indizar simultáneamente, nos encontramos con que podría ocurrir un problema de sincronización al momento de que éstos dos ingresaran a la tabla de hash. El problema encontrado es que el indizador podría estar accediendo al índice en busca de algo que sí está en el archivo pero que no ha sido cargado

todavía por el primer hilo, por lo que el indizador actuará en consecuencia de creer que el archivo no está indizado y lo indizará.

Para solucionar el problema antes mencionado implementamos mutex, el cargador bloquea el acceso del indizador a la tabla de hash desde que se comience a cargar el archivo en ella hasta que termine. Una vez ésto haya pasado dicho hilo liberará el acceso a la tabla de hash, con lo cual el indizador podrá buscar en ella los archivo que encuentre al navegar los directorios, para verificar si han sido indizados o no.

Posteriormente se espera que ambos hilos terminen su ejecución y se procede a destruir el mutex a través de la función pthread_mutex_destroy().

Sección 4: Explicación de la Función y Tabla de Hash

La función de hash escogida es la postulada por Brian Kernighan y Dennis Ritchie en el libro “The C Programming Language”, la cual consiste en utilizar una seed, o semilla, la cual toma un valor, por lo general de 31, pero puede variar entre valores compuestos por 1 y 3 (ej, 131 1313). Esta semilla multiplica el valor de hash en cada iteración y se le suma al valor al que le estamos hallando la clave de hash. Al resultado de esta suma se le aplica el módulo del tamaño de la tabla, con lo cual obtenemos la clave de hash.

La tabla de hash implementada es de hashing abierto y posee 100 casillas compuestas por celdas, las cuales poseen una ruta y un apuntador a la celda siguiente, a modo de lista enlazada.

Sección 5: Explicación del Formato de Archivo

La información de los archivos (ruta y claves) recopilada por el indizador es almacenada en un archivo en la siguiente forma: se ingresa al archivo una línea por cada clave encontrada, por lo que una ruta aparece en éste tantas veces como claves posea. El formato de la línea es la ruta seguida de un espacio y posteriormente una clave perteneciente a dicha ruta. Cabe destacar que una ruta no puede encontrarse en más de una línea con una misma clave.

El hecho de que todos los archivos alcanzados por el indizador deberán tener sus claves separadas por guiones (-) o por puntos (.), de tenerlos separados por otro caracter no serán detectadas como tales.

Sección 6: Comentarios adicionales

En el programa implementamos una variable llamada NAME_MAX y PATH_ MAX que hace referencia a la longitud máxima de un archivo y un directorio respectivamente. Esta variable se define en la librería limits.h.

Es necesario aclarar que, si bien existen dos funciones con instrucciones bastante similares, indizar y navegar_directorio, ésto es justificado, ya que la primera es exclusivamente para definir el hilo mientras que la otra es para realizar la ejecución recursiva, que con la primera no se podría.

Por el carácter opcional de algunos flags existe la posibilidad de que alguno de éstos nos sea ingresado, por lo que nos vimos en la necesidad de colocar valores por defecto a ciertas variables. Entre éstas se encuentran: maxlength, la cual inicia con un valor de 20, archivoindice, la cual posee el nombre de archivo por defecto “indice.txt” y indice[], la cual posee un director por defecto desde el cual iniciará la búsqueda del indizador “.”.

Conclusión

Con el presente informe pretendemos mostrar uno de los posibles mecanismos detrás de una herramienta de gran uso cotidiano, como lo es un buscador de archivos, dejando ver aspectos de éste como lo son el almacenamiento y clasificación de información y la búsqueda de ésta.

Para la implementación de la solución debimos involucrarnos tanto teórica como prácticamente en ámbitos como la implementación de hilos y la sincronización de éstos, como se explicó anteriormente, herramienta extremadamente útil para el problema que nos ocupa.