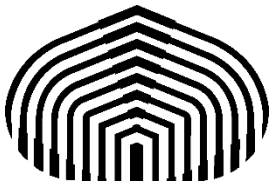

República Bolivariana de Venezuela
Universidad Simón Bolívar
CI3825 – Sistemas de Operación I
Profesor: Fernando Torres

PROYECTO 1 – Detector de Palíndromos en el Sistema de Archivos (INFORME)

Estudiantes:

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

Caracas, Marzo de 2018

Introducción

Dado un árbol de directorios que se obtiene a partir de un directorio o archivo raíz, se recorre dicho árbol recolectando los paths encontrados desde la raíz hasta cada una de las hojas. Luego se procede a revisar todos los caminos, que están formados por los nombres de los directorios que lo componen concatenado con el nombre del archivo que representa la hoja (si la opción fue habilitada), y se encuentra, si existen, todos los palíndromos ubicados dentro de ellos.

El informe estará dividido en 5 secciones en las cuales se explicará cómo compilar y correr el programa, la estructura general de la solución, la estrategia utilizada para la creación de procesos hijos, cómo comunicamos y sincronizamos los procesos utilizados y cómo manejamos la cadena de caracteres que almacena los caminos del árbol.

Sección 1: Cómo Compilar y Correr el Programa

Para realizar la compilación del programa se creo un makefile que al ejecutar la instrucción “make” crea un archivo llamado “arbol” ejecutable que realiza la solución al problema dado. Para correr el programa debe ejecutarse la instrucción “./arbol” seguida de los flags opcionales dados en el enunciado del proyecto.

Sección 2: Estructura General de la Solución

Para el problema dado creamos un grafo no dirigido implícito definido de la siguiente manera: un nodo es cada uno de los archivos o directorios que componen el árbol, y una relación es todo par (x,y) tal que x es un directorio que apunta a y, o viceversa. Una vez construido el grafo procedemos a implementar el algoritmo DFS sobre él, partiendo desde la raíz, con la variación de que no utilizamos el atributo de visitados de los nodos, ya que por la estructura de árbol dos caminos distintos no pueden interceptarse, y a medida que avanzamos en el grafo concatenamos el nombre del nodo actual en un arreglo de caracteres que contiene el camino que se ha recorrido para cada nodo.

Una vez alcanzada una hoja, o llegado al límite de altura introducido como flag, no se sigue extendiendo el camino actual y se procede a escribir en un archivo el contenido en la cadena de caracteres acumulado hasta el nodo en que nos encontramos, incluyendo este si no es un archivo o, si es un archivo, si la opción fue habilitada. Una vez se hayan visitado todas las hojas del árbol, o todos los nodos cuya altura es igual a la altura máxima, se procede a leer el archivo, línea por línea y, por cada una de ellas, se hayan todos los palíndromos que se encuentren en su interior.

Sección 3: Creación de Procesos Hijo

Para realizar el recorrido del árbol explicado en la sección anterior decidimos emplear una función de creación recursiva de procesos llamada “navegar_directorio”, para la cual uno de los parámetros ingresados es la ruta de un directorio. Dicha función crea un proceso hijo del proceso que la llama, la cual abre el directorio y, para cada uno de los elementos encontrados en él, revisa si es un directorio o un archivo. Si el elemento que se encuentra en el directorio pasado como parámetro es a su vez un directorio procedemos llamar de nuevo a la función, esta vez con el directorio actual como parámetro, con lo cual seguiremos creando procesos que explorarán el directorio.

Sección 4: Comunicación y Sincronización Entre Procesos

La comunicación entre proceso se lleva a cabo, como mencionamos con anterioridad, a través de un archivo. Éste sirve como lugar común entre los procesos para que en él puedan escribir los caminos obtenidos, facilitando así la búsqueda de los palíndromos.

Para la sincronización de procesos nuestra estrategia fue aprovechar la estructura del DFS, la cual consiste en una exploración no uniforme del grafo. Esto lo hace al ir expandiendo todos los nodos del grafo que va localizando, formando un camino definido. Cuando ya no se puede seguir expandiendo se devuelve por el camino hasta que consiga un nodo que alcance a nodos que no hayan sido visitados previamente, al cual expande para seguir formando caminos.

Gracias a la estructura antes mencionada, como la expansión de los nodos es uno a la vez, de manera ordenada, sabemos que dos procesos no se van a ejecutar en simultáneo ya que los directorios, que son los nodos, cuentan con un solo proceso como máximo cada uno y el programa no se encuentra explorando más de un directorio a la vez. El proceso de los directorios a los cuales se ha explorado al menos un elemento se le aplica la función wait(), con la cual esperamos que todos sus elementos sean explorados. Luego de que ésto ocurra procedemos a matar dicho proceso con la función kill().

Como todos los procesos asociados a directorios que no se están explorando se les aplica wait(), y sólo hay un directorio a la vez que está siendo explorado, sabemos que no pueden haber dos procesos ejecutándose al mismo tiempo, por lo que no ocurrirá un problema de sección crítica.

Sección 5: Manejo de Caracteres

Los strings utilizados para las rutas de directorios se crearon con un tamaño fijo según la variable del sistema “PATH_MAX” de esta forma se evita trabajar con arreglos dinámicos de strings.

La manera en la que se manejaron los strings para hallar los palíndromos existentes en una palabra está basado en el Algoritmo de Manacher, el cuál encuentra en tiempo lineal el número de palíndromos distintos que existen en una cadena, para ello se define un arreglo en donde cada bloque representará una letra de la palabra, en el cual se colocará un número que representara el centro del palindromo (en caso de ser impar) y dicho numero significara la longitud del palindromo desde el centro hacía la derecha e izquierda. Para hallar el centro se recorre cada letra y luego se van comparando las letras que le rodean, tanto por izquierda como por derecha al mismo paso, y en el momento en que sean diferentes se guarda en el

bloque que representa el centro del palindromo el numero de pasos el cual indica la longitud del palindromo, y así con cada letra. Luego se lee dicho arreglo y se hallan los substrings a través de una función que recibe la palabra completa, la posición en que comienza el substring y su longitud, para después imprimirse. Para el caso de los palíndromos pares el procedimiento es el mismo, pero la diferencia es que en el arreglo, como no tiene un centro de una letra sino de dos, entonces la longitud se guarda en el bloque correspondiente a la letra derecha del par que componen el medio del palindromo. Tanto los palíndromos pares y los impares se obtienen a través de dos funciones respectivamente, los cuales retornan el número de palíndromos encontrados y de esta manera, si no se encuentran palíndromos, tenemos que la suma de los resultados que retornan ambas funciones sería 0 por lo que se reporta que ningún palíndromo fue encontrado.

Conclusión

En el proceso de desarrollo de la solución al problema, se presento la interrogante de como crear procesos de manera concurrente para lograr explorar rutas de forma simultanea y así lograr un mejor rendimiento. Dada la complejidad de una solución como esta, se opto por utilizar una estructura mas sencilla que permitía la correcta ejecución del programa.

La solución propuesta ofrece una ejecución con creación de procesos y con acceso a los i-nodos de Linux, sin embargo, no utiliza mecanismos de comunicación como pipes o de sincronización, como semáforos o signals, que nos hubiese gustado explorar más a fondo.

Investigando sobre la implementación de los distintos métodos de creación/comunicación/sincronización de procesos logramos entender de manera rápida y práctica el funcionamiento de los sistemas de directorios, además de la creación y el uso de procesos y la comunicación y sincronización entre éstos. Conocimiento muy valioso que nos ayudará en incontables aspectos de nuestra vida profesional.

El realizar un proyecto con ejecución de procesos nos hizo entender la dificultad que implica crearlos y nos hizo reflexionar acerca de la posibilidad de resolver el problema con hilos, la cual seria mas interesante de aplicar.

Entre los desafíos presentados nos encontramos con el hecho de que debíamos sincronizar los procesos para que dos de ellos no abrieran el archivo a la vez, aspecto que solucionamos gracias a la estructura de DFS utilizada explicada anteriormente, pero que pensamos solventar con la aplicación de un booleano o de semáforos.

Posteriormente nos vimos en la necesidad de utilizar pipes para comunicar los procesos, para que todos utilizaran el mismo semáforo. Ésto se vio descartado cuando eliminamos la concurrencia de procesos para que se ejecutaran uno a la vez.