1. Introducción y Estructura del Informe.

Un sistema de archivos es considerado como un componente central de un sistema operativo. Este debe proveer al usuario información de almacenamiento y medios para conseguir la información almacenada. El sistema de archivos de los sistemas operativos inspirados por Unix no posee directorios, en cambio estos usan los llamados Inodes, los cuales contienen tanto el archivo como sus metadatos.

En este informe se mostrará cómo es posible recorrer un sistema de archivos Unix, mediante el uso de recursos que provee el sistema operativo y que permiten la obtención de los metadatos de los directorios y archivos almacenados en este, haciendo uso del programa suministrado.

Nuestro programa recorre, desde cierto “path”, cada directorio, tomando el nombre de cada elemento encontrado y finalmente imprimiendo los palíndromos encontrados dentro de cada sub-cadena armada con los caracteres de los nombres de estos elementos.

El contexto de este informe tendrá como base el programa “DetectorDePalindromos.c” continuación, se detallará la estructura del informe.

* En el punto **2** se explicará cómo compilar y correr el programa suministrado.
* En el punto **3** se explicará la estrategia de creación de procesos hijos usada para manejar las distintas partes de nuestro programa.
* En el punto **4** se explicará la estrategia de comunicación y sincronización de procesos que se usaron
* En el punto **5** se explicarán las estrategias de “parsing” y manejo de cadenas usadas
* En el punto **6** se explicará la estrategia de recorrido de Inodes y el uso de sus metadatos.
* En el punto **6** se detallarán las conclusiones del trabajo realizado.

1. Compilación y Corrida.

**Compilación:**

El programa puede ser compilado mediante 2 métodos. Ambos métodos necesitan el uso de un sistema operativo derivado de Unix y ubicarse con la terminal en el directorio en donde se encuentra ubicado el programa.

El método preparado es el del archivo “Makefile”, este permite la compilación automática del programa, convirtiendo el código fuente en un archivo objeto (“\*.o”). Ubicado en el directorio del programa, desde la terminal se puede usar el comando “make”, este creará el archivo ejecutable “main.out” el cual podrá ser ejecutado desde la terminal para correr el programa.

El otro método es la compilación manual desde la terminal, mediante el comando del compilador “gcc” siguiente:

*>: gcc main.c -pthreads -o main.out*

Esto igualmente creará el archivo ejecutable “main.out”

**Corrida:**

El programa puede ser ejecutado mediante el comando “main” mientras se encuentre ubicado en el mismo directorio del archivo ejecutable. Esto imprimirá en pantalla los palíndromos encontrados en la concatenación de los nombres encontrados desde el directorio actual hasta los directorios vacíos a una profundidad máxima de 20 directorios, tomando entonces a cualquier directorio a esta profundidad como un directorio vacío.

Existen un número de opciones que pueden ser agregadas al programa por medio del uso de “flags”.

Mediante el **flag “-d <directorio>”** se especificará la carpeta desde la cual se inicia el recorrido, imprimiendo así los palíndromos encontrados en la concatenación de los nombres obtenidos desde el directorio especificado hasta los directorios vacíos.

Mediante el **flag “-m <altura>”** se especificará la profundidad máxima de exploración hasta la cual se llegará.

Mediante el **flag “-f”** se especificará si los archivos que no sean directorios serán tomados en cuenta, considerando estos como directorios vacíos.

1. Creación y Manejo de Procesos.

Los procesos utilizados para la ejecución de nuestro programa son el proceso padre, el proceso con variable de nombre “proceso\_palindromos”, llamado proceso palíndromos y el proceso con variable de nombre “proceso\_parser” llamado, naturalmente, proceso parser. Estos son creados mediante la función “fork” y manejados mediante una operación de selección que discrimina las instrucciones que cada proceso realizará.

Al finalizar la ejecución de la función principal del proceso padre, este parará la ejecución de los otros procesos mediante la función “kill”. Es importante mencionar que estos procesos poseen memoria compartida, la cual almacena ciertos elementos que todos usan, lo cual será mencionado en el siguiente punto.

En particular, el proceso padre se hará cargo de el recorrido entre directorios, comunicando el path de la hoja del directorio encontrado, como una cadena de caracteres, al proceso de parser el cual modificará la cadena de tal manera que pueda ser manejada por el proceso palíndromos, el cual buscará cada palíndromo en este path e imprimiéndolos.

1. Sincronización y Comunicación entre Procesos.

**Sincronización:**

La sincronización entre procesos es realizada mediante estructuras semáforo que detienen y continúan la ejecución de los procesos. Estos están almacenados en la memoria compartida de estos procesos. Son usados de la siguiente manera:

1. El proceso padre iniciará su ejecución, buscando el primer path válido que pueda enviar al proceso parser; al encontrar este path esperará una señal del proceso parser, y enviará el path encontrado.
2. El proceso parser estará inicialmente esperando la señal del proceso palíndromo, al recibirla procesará la string recibida con la señal, preparándola entonces para ser usada en la búsqueda de palíndromos, enviándola entonces al siguiente proceso y dando la señal al proceso padre de buscar el siguiente path. Por último, quedará esperando la señal del proceso palíndromo
3. El proceso palíndromo, igualmente, estará esperando la señal del proceso padre, y de una manera similar, procesará su entrada, la imprimirá en pantalla y dará la señal al proceso parser de limpiar la siguiente string, esperando entonces por la señal del mismo.
4. Al no encontrar más paths válidos, se envía una señal de kill a los procesos hijo para detener su ejecución.

**Comunicación:**

La comunicación se basa en la necesidad de enviar a los procesos hijo cadena de caracteres conseguidas en otros procesos. El medio de comunicación utilizado es el de estructuras pipe, las cuales permiten almacenar en un buffer, entre señales, la información necesaria para ser procesada. El proceso padre y el proceso parser compartirán un pipe, y el proceso parser y el proceso palíndromo compartirán otro.

1. Manejo de Cadenas de Caracteres.

El manejo de cadenas de caracteres fue dividido en 2 partes, el “parsing” y la búsqueda de sub-cadenas palindrómicas.

El “parsing” es manejado mediante la sub-rutina “parsear”., el cuál toma la representación en cadena de caracteres de un path. Esta función se encarga de 3 cosas. Primero elimina todos los caracteres que no se consideran parte del nombre de algún directorio recorrido, i.e. los caracteres ‘/’, ‘.’, etc. Segundo, normaliza la capitalización de los caracteres, incluyendo caracteres especiales como las vocales acentuadas, cambiando los caracteres alfabéticos por minúsculas. Se decidió en el diseño del programa imprimir en pantalla cada cadena de caracteres como minúsculas. Otras decisiones de diseño fueron la diferenciación de cualquier carácter, sean estos caracteres no alfanuméricos o alfabéticos “no-comunes” encontrados en el idioma español, excepto por las mayúsculas y minúsculas de un path.

La búsqueda e impresión de sub-cadenas palindrómicas es manejada mediante la sub-rutina “palíndromo”. Esta función toma una cadena de caracteres como entrada y muestra como salida en pantalla cada sub-cadena perteneciente a esta que sea un palíndromo y que su largo sea igual o mayor a 3. La naturaleza de nuestra función imprime cada sub-cadenas encontrada, incluidas las sub-cadenas equivalentes anteriormente impresas. La búsqueda de sub-cadenas es realizada de la siguiente manera:

1. Se itera sobre cada carácter de la cadena entrante, tomando a este como el centro del palíndromo, comenzando con el segundo carácter y terminando en el penúltimo.
2. Se toma el carácter anterior y el centro, si se quiere buscar palíndromos pares, o el carácter anterior y el siguiente al centro, para impares. Estos marcadores determinaran la sub-cadena a imprimir.
3. Se comparan estos caracteres, usando la función “compare\_acentos” para revisar su equivalencia
   1. Si son iguales, se imprime si la distancia de los marcadores es mayor o igual a 3, y se siguen alejando los marcadores del centro, acotando sus posibles posiciones por el largo de la cadena misma.
   2. En caso contrario, inmediatamente se toma el paso 1, moviendo el centro hacia adelante.

Se toma siempre en cuenta el hecho de que los caracteres especiales toman un tamaño doble en la cadena de caracteres. El centro y los marcadores siempre apuntaran a la segunda posición ocupada por estos.

1. Recorrido de Directorios.

El recorrido de directorios se realizó mediante la función “ftw”, la cual permitía, al proveer el path a un directorio, conseguir los metadatos encontrados en cada inodo desde el especificado hasta cualquier directorio vacío o archivo que pudiera ser alcanzado desde este, siendo el especificado el directorio padre. Esta función permite proveer también una función más, “acción\_por\_nodo”, la cual toma como argumentos los metadatos de cada inodo recorrido.

Para identificar si el nodo tomado en cada iteración de la función “ftw” era un directorio o un archivo, se reviso el modo del inodo haciendo uso de sus metadatos.

1. Conclusiones.

El trabajo realizado nos permitió interactuar y tener a la mano 2 características importantes del manejo de sistemas operativos. La simultaneidad de corrida y el control, sincronización de procesos, y el manejo de sistemas de archivos mediante Inodes. El resultado de nuestro trabajo fue la capacidad de crear un programa en completo funcionamiento que pudiera manejar simultáneamente el recorrido de arboles de directorios y procesos que realizaban operaciones sobre los datos obtenidos, confirmando así nuestro conocimiento sobre el manejo de ambos.