Caracas, 1 de marzo. 2018.
Universidad Simón Bolívar
Sistemas de Operación I CI-3825
Informe técnico – Proyecto 1

# Proyecto 1

## Detector de palíndromos en el sistema de archivos

Javier Vivas. 12-11067

Daniel Francis. 12-10863

# Índice

# 1. Introducción

Para la realización del proyecto se requiere que un programa sea capaz de navegar un árbol de directorios UNIX, con el objetivo de encontrar palíndromos en el resultado de concatenar los nombres de cada carpeta. Esta tarea debe realizarse con la implementación de procesos hijo, a través de la instrucción “fork()”.

También se solicita implementar diferentes parámetros opcionales para la ejecución de la tarea. Estos parámetros permiten al programa comenzar desde un directorio específico al proveer el path absoluto del mismo (flag -d <carpeta>), que el programa descienda un altura específica en el árbol (flag -m <altura>) y finalmente, la opción de incluir nombres de archivos en el análisis (flag -f).

En el presente trabajo explicamos el funcionamiento del programa, la estrategia detrás de la creación de procesos hijo y la comunicación entre los mismos, cómo los sincronizamos y la justificación de las estructuras de datos implementadas.

Para culminar, hacemos una breve reseña de las conclusiones generadas en la realización de este proyecto.

# 2. Diseño y desarrollo

## 2.1 Ejecución:

El programa se divide en dos archivos: **main.c** y **funcionesProy1.h**. Para ejecutar el programa, es necesario compilar main.c y correr el ejecutable con las opciones (flags) deseadas. Los flags son opcionales.

## 2.2 Diseño y estructura:

Podemos dividir el diseño del programa en dos secciones: **Búsqueda de palíndromos** y **recorrido del árbol,** pero antes de explicar cada sección es preciso explicar cómo manejamos la creación de procesos hijo.

Al momento de ejecución, el proceso principal (padre) analiza los flags proporcionados y crea un **archivo de salida** desechable. Una vez hecho esto, el padre realiza la función fork, creando así a un hijo y entrando en modo de espera con la función wait hasta recibir la señal de estatus de finalización del hijo. El archivo de salida es el buzón de mensajes a través del cual los procesos transmiten datos entre sí.

El proceso hijo llena el archivo de salida con los strings correspondientes a los path de las hojas del árbol de directorios. Estos path son relativos al directorio de inicio presente en el flag “-d” o en su defecto, relativos al directorio donde se encuentra el ejecutable. Esto corresponde a la sección **recorrido del árbol.**

Cuando el hijo finaliza, el padre recibe la señal y culmina la instrucción wait, con lo cual procede a leer el archivo de salida y a extraer los palíndromos de cada string dentro de él. Esto sería la sección **búsqueda de palíndromos.**

Como resultado, en consola debería visualizarse cada string de hoja seguido de los palíndromos que contiene.

**2.2.1 Recorrido del árbol:** Hacemos uso de los i-nodo de UNIX para que el proceso pueda visitar los directorios necesarios para completar la tarea. Para implementar esta sección, decidimos hacer uso de una función recursiva condicionada por la altura máxima indicada en el flag o por su ausencia. Durante el recorrido, el proceso concatena nombres de carpetas (y/o archivos, según se requiera) hasta llegar a una hoja o hasta cumplir su altura máxima.

**2.2.2 Búsqueda de palíndromos:** Para los strings del resultantes del recorrido del arb́ol, decidimos usar la modalidad de arreglo, ya que permitía iterar con mayor facilidad sobre la palabra.

Primero, se creó una función para determinar si un subarreglo era un palíndromo (que sus caracteres sean los mismos en ambos sentidos de lectura horizontal).Vimos que la comparación de caracteres no era trivial, así que creamos también una función que permitiese considerar letras mayúsculas como equivalentes a las minúsculas, al igual que caracteres especiales.

Luego, usamos una función general que toma cualquier string y compara la primera letra con la última. Si son iguales, busca un palíndromo entre ellas. Si no son iguales, pasa a la penúltima letra. Se realiza este proceso hasta llegar a la mitad del string. De haber un palíndromo, este se imprime en pantalla.

## 2.3 Comunicación entre procesos:

Los procesos se comunicaban a través de las señales de finalización de cada uno y con el buzón de mensajes, el cual corresponde a un archivo temporal para guardar los strings a analizar. El proceso padre engendra al hijo, espera, el proceso hijo llena el archivo con los strings, finaliza, envía la señal de finalización al padre y el padre culmina al realizar el análisis de los strings.

Decidimos utilizar un buzón para evitar tener al proceso padre en espera continua (busy wait).

# 3. Conclusiones

Aprendimos que la comunicación entre procesos padre e hijo puede ser sencilla o complicada, dependiendo de la implementación. También se hace evidente que el manejo de recursos con procesos hijo no siempre es el más adecuado, ya que el proceso hijo contiene una copia de todos los recursos y la memoria utilizada del padre. Para este proyecto en particular, pudo ser conveniente hacer uso de hilos en vez de procesos padre e hijo.

Sin embargo, podemos ver que lograr que los procesos se comuniquen entre sí es una habilidad importante para incurrir en multiprogramación.

Una potencial mejora a nuestro código que podemos acotar es que el proceso hijo podría avisar al padre cada vez que se llega a una hoja (es decir, cuando se obtiene un string definitivo). Entonces el padre podría proceder a extraer los palíndromos del string mientras el hijo continúa con el recorrido.

En ese caso, surgiría la incógnita de qué ocurriría si el hijo termina la siguiente sección de recorrido y emite una señal. El padre podría ignorarla en el caso de que no hubiese aún terminado de analizar el string anterior. Tendríamos que usar una cola de interrupciones o alguna estructura similar para asegurar la sincronización entre los procesos. A pesar de esto, esta implementación podría ser más eficiente que la presente en el código entregado.