

Proyecto 2: Manejo de Archivos y de Señales (20%)

Objetivo

Familiarización con las llamadas al sistema para la creación y acceso a las estructuras del sistema de archivos; y para el envío y para el manejo de señales.

Introducción al Problema

Es de gran utilidad en redes UNIX, donde los usuarios tienen asignadas cuotas de disco, contar con una aplicación que recorra recursivamente un árbol de directorios, generando un reporte con la información más importante para cada directorio en el árbol. De esta manera el usuario podrá saber con exactitud en qué directorios está invirtiendo mayor cantidad de espacio.

Este segundo proyecto está enmarcado en el área de los sistemas de archivos y manejo de señales, utilizando las herramientas y llamadas al sistema que han aprendido en las clases de Laboratorio. Se pide desarrollar una aplicación llamada *els* (enhanced – ls) que debe cumplir con las siguientes características:

- La línea de ejecución de la aplicación será: `els <output_file>`
donde el parámetro `output_file` es opcional y representa el *nombre* del archivo donde se generará el reporte final. Si no se provee este dato, el reporte se generará por pantalla.
- A partir del directorio donde es ejecutado el programa *els* y para cada directorio contenido en ese directorio base, deberá aparecer en un archivo reporte la información que se especifica a continuación:
 - Path absoluto del directorio Ej: `/home/yudith/SOPI/`
 - permisos : todos los permisos actuales del directorio en letras, al igual que son mostrados por el comando `ls -l`.
 - user-id : user id del dueño del directorio.
 - group-id : group id asignado al directorio.
 - Fecha de la última modificación.
 - Fecha del último acceso.
 - Número total de archivos contenidos en el directorio (incluyendo subdirectorios).
 - Número total de bytes ocupados por los archivos ubicados en ese directorio (no incluye subdirectorios).
- El nombre del archivo reporte de cada proceso hijo se generará con su PID y txt como extensión (PID.txt)
- **(OPCIONAL)** En caso de encontrar un archivo `core` durante el recorrido, éste deberá ser eliminado y agregar al reporte la información del directorio donde se encontraba el core, su fecha de modificación y el tamaño.

Un archivo core dump es generado cuando un programa termina abruptamente, ya sea por un *bug*, una

violación de acceso a memoria o por mecanismos de protección de acceso al hardware. El archivo core contiene un vaciado de la memoria y del estado del CPU para el momento en que el proceso terminó abruptamente. Existen herramientas que ayudan a examinar el contenido de un core con la intención de identificar el problema que produjo el *crash* del proceso.

Implementación:

- El proceso principal deberá crear un proceso por cada subdirectorio que exista en el directorio donde se ejecute el programa *els <output_file>*. Es decir que deberá existir un proceso hijo por cada subdirectorio existente en el directorio de ejecución.
- Cada proceso hijo deberá generar un reporte con la información descrita en el punto anterior a partir del subdirectorio del cual es responsable.
- El proceso Padre generará un reporte final con la información de su directorio e información global para cada uno de los subdirectorios, considerando los datos acumulados de sus hijos referente a:
 - Número total de archivos contenidos a partir del directorio de ejecución (incluyendo subdirectorios).
 - Número total de bytes ocupados por todos los archivos ubicados a partir del directorio de ejecución (no incluye subdirectorios).
- El proceso Padre lee la información de sus hijo de sus archivos respectivos PID.txt.
- El proceso Padre deberá instalar un manejador de señales para evitar que sea interrumpido con CTRL-C desde la consola.

La siguiente figura ilustra el esquema de creación de procesos, y un ejemplo de reportes finales para la estructura de directorio mostrada (**ignoren los pipes**).

Suponiendo la siguiente estructura de directorios a partir de /home/yudith:

Dir1: Dir2: Dir2.1: Dir2.1.1: M(5)

P(4)

Q(4)

A(2)

B(2)

C(2)

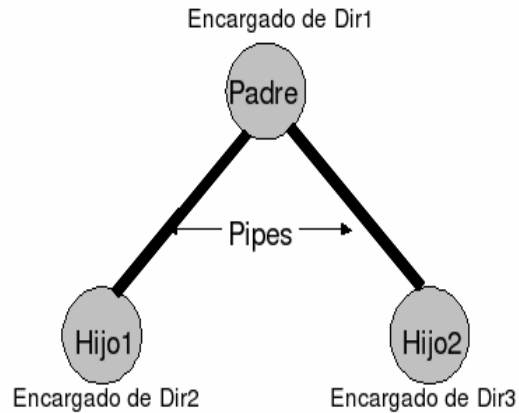
Dir3: Dir3.1: R(4)

X(2)

Y(2)

N(2)

El nro entre () representa el tamaño en bytes de los archivos. Los DirN son directorios.



Reporte de Padre:

/home/yudith/Dir1 drwxr-xr-x yudith sasl 2008-03-01 2008-03-06 3 2

/home/yudith/Dir1/Dir2 drwxr-xr-x yudith sasl 2008-03-01 2008-03-06 8 19

/home/yudith/Dir1/Dir3 drwxr-xr-x yudith sasl 2008-03-01 2008-03-06 4 8

Reporte de Hijo 1:

/home/yudith/Dir1/Dir2 drwxr-xr-x yudith sasl 2008-03-01 2008-03-06 4 6

/home/yudith/Dir1/Dir2/Dir2.1 drwxr-xr-x yudith sasl 2008-03-01 2008-03-06 3 8

/home/yudith/Dir1/Dir2/Dir2.1/Dir2.1.1 drwxr-xr-x yudith sasl 2008-03-01 2008-03-06 1 5

Reporte de Hijo 2:

/home/yudith/Dir1/Dir3 drwxr-xr-x yudith sasl 2008-03-01 2008-03-06 3 4

/home/yudith/Dir1/Dir3/Dir3.1 drwxr-xr-x yudith sasl 2008-03-01 2008-03-06 1 4

Otros aspectos importantes a tener en cuenta para la implementación:

- Tener en cuenta que en el recorrido se pueden encontrar directorios sin permiso de lectura o ejecución.
- **(OPCIONAL)** Verificar que el directorio donde se desee crear el archivo reporte puede no tener permisos de escritura, en este caso dar un mensaje de error y abortar el proceso (se aborta sólo el proceso que no puede escribir el reporte).
- Recuerden tomar en cuenta aspectos de modularidad, eficiencia y buen uso de la memoria.

Los aspectos marcados como OPCIONAL tendrán puntos extras si están correctamente implementados.