UNGS – Universidad Nacional General Sarmiento

Sistemas Operativos y Redes II

Segundo cuatrimestre de 2020

7 de octubre de 2020

FAT, o un tp muy grande

Aclaraciones

- Para aprobar la totalidad del TP es necesario tener aprobado cada uno de sus módulos.
- **Fecha de entrega: 2 de Noviembre de 2020, hasta las 23:59hs - Deberán entregar el informe por mail a la casilla ungssor2@gmail.com con el asunto:TP-<apellido_un_integrante>. En el cuerpo del mail deberán poner nombre y documento de cada uno de los integrantes.**
El código, el informe (en formato PDF) y el archivo de imagen deberán estar comprimidos en un archivo con el mismo nombre del subject del mail y de extensión .tar.gz
Deberán armar grupos de 2 personas!

Informe

Proponemos realizar una exploración en profundidad del sistema de archvos *FAT 12*.

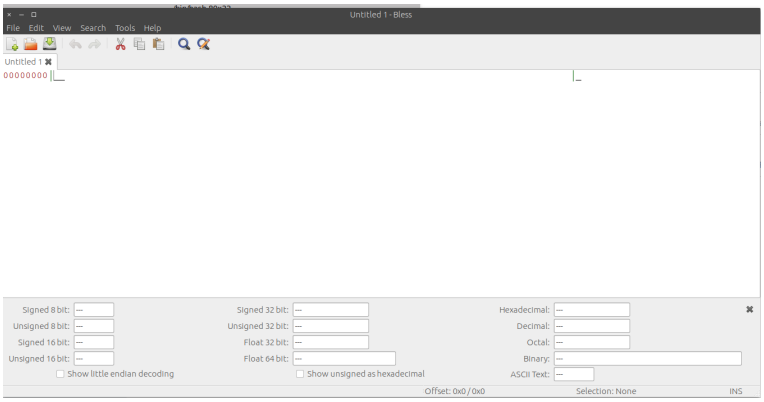
Para esto estaremos trabajando con el archivo de imágen provisto llamado *test.img*.

Lo estaremos leyendo a bajo nivel, o sea directo del iso, pero podrán montarlo para entender y comprobar lo que van mirando.

Para esto deberán:

- Montarlo: Siendo root (o mediante sudo) ejecutar:
  `mount test.img /mnt -o loop,umask=000`
- Desmontarlo: Siendo root (o mediante sudo) ejecutar:
  `umount /mnt`

Para realizar los ejercicios se deberá instalar un editor hexadecimal. Dicho editor nos permitirá leer archivos a bajo nivel. Nos permitirá ver qué hay en un byte exacto y traducirlo a varios formatos, como ser binario, hexadecimal o texto. Uno de estos editores es `bless` que se encuentra como paquete en *ubuntu*



Se deberán responder las siguientes preguntas. Todo código deberá estar correctamente documentado. A su vez deberán crear un archivo para la compilación del código completo. Genere los distintos puntos de código en archivos separados. Explicar en el informe entregado dónde se encuentra el código de la resolución de cada ejercicio.

Ejercicios:

1. Al montarlo. ¿ Para qué se ha puesto `umask=000` ?
2. Cargando el *MBR*
   a) Mostrando el *MBR* con el *Hex Editor*: Muestre los primeros bytes y la tabla de particiones. ¿Cuántas particiones hay ? Muestre claramente en qué lugar puede observarlo.
   b) Lea los datos del punto anterior utilizando código `C` y muéstrelos por pantalla.
   c) Muestre en el *Hex Editor* si la primer partición es booteable o no. ¿Lo es?
   d) Muestre, mediante un programa en `C`, para la primer partición: el flag de booteable, la dirección *Cylinder-head-sector (chs)*, el tipo de partición y su tamaño en sectores.

3. Cargando la tabla de archivos. En todos los ejemplos siguientes, cuando se pida código `C`, deberá leer la tabla de particiones e ir recorriendo las estructuras de datos adecuadamente. Es decir no podrá hardcodear direcciones vistas de alguna otra manera, salvo que se indique lo contrario.

   a) ¿Cuántos y cuáles archivos tiene el filesystem ? Muésrelos con Bless y genere el código `C` para mostrarlos.
   b) Montando el filesystem (mediante `mount`) cree un archivo en la carpeta root `/` y luego bórrelo. Búsquelo por bless y muéstrelo en con el código generado previamente.
   c) ¡Muestre medante bless el archivo que ha sido borrado. Explique cómo lo ha visto. Genere código `C` para mostrarlos.
   d) ¿Qué puede decir acerca del recupero de archivos ?

4. Leyendo archivos.

   a) Montando el filesystem (mediante `mount`) cree un archivo llamado `lapapa.txt` y póngale algún texto como contenido. Hágalo en la carpeta root `/`. . Búsquelo por bless y muéstrelo en con el código generado previamente.
   b) Muestre, mediante el `hex editor` y mediante código `C` lo que hay en el archivo no borrado.
   c) Cree código `C` para que dado un archivo (o una parte), lo busque y si lo encuentra y el mismo se encuentra borrado, lo recupere.

Referencias:

https://en.wikipedia.org/wiki/Design_of_the_FAT_file_system
http://www.c-jump.com/CIS24/Slides/FileSysDataStructs/FileSysDataStructs.html