Sistemas Operativos - 2c2015
Clase de Repaso 2do Parcial

Ejercicio 1)

a) Se tiene un dispositivo de almacenamiento con un `FS` de tipo `ext2` que se ha dañado. Se sabe que el **único** daño producido es que los *i-nodos* se han desordenado. Es decir que el *i-nodo* que se encuentra en la posición n de la tabla podría ser, en realidad, el m-ésimo. Se desea saber cuál de la siguiente información es recuperable y cuál no:

   I. Cantidad de archivos y directorios presentes en el `FS`
   II. Estructura de directorios del `FS`
   III. El tamaño, tipo y contenido de los archivos incluyendo el nombre original
   IV. El tamaño, tipo y contenido de los archivos sin incluir el nombre original
   V. El tamaño y tipo de los archivos pero no su contenido.
   VI. El tamaño de los archivos pero no su tipo ni su contenido.

b) Se tiene un dispositivo de almacenamiento con un `FS` de tipo `FAT` que se ha dañado. Se sabe que el **único** daño producido es que la entrada de las entradas de directorio que indica el bloque inicial de un archivo se ha corrompido. Se desea saber cuál de la siguiente información es recuperable y cuál no:

   I. Cantidad de archivos y directorios presentes en el `FS`
   II. Estructura de directorios del `FS`
   III. El tamaño, tipo y contenido de los archivos incluyendo el nombre original
   IV. El tamaño, tipo y contenido de los archivos sin incluir el nombre original
   V. El tamaño y tipo de los archivos pero no su contenido.
   VI. El tamaño de los archivos pero no su tipo ni su contenido.

Ejercicio 2)

En un sistema **monoproceso** se nos pide implementar un *driver* para una controladora de `RAID 5` por *software*. La misma debe controlar 4 `HDDs` iguales (cuyos *drivers* ya se encuentran cargados en el SO). Queremos implementar la función

```
driver_write(uint startsector, char * buf, uint len)        (1)
```

que recibe el sector donde se quiere comenzar a escribir, un *buffer* con la información a escribir y el tamaño de dicho *buffer* en *bytes* (notar que el tamaño del *buffer* podría ser mayor que 1 sector[1]). Un requerimiento adicional es que la función `write()` de esta controladora sea **NO bloqueante**. Es decir que la misma debe retornar *inmediatamente*. Para que esto sea posible nos informan que el SO invocará periódicamente a una función

```
driver_scheduled_writes()        (2)
```

sin parámetros. Esta función también debe ser implementada por nosotros.

a) Utilizando las funciones que se listan a continuación: (1) `hdd_write(uint sector, char * buf)` (2) `copy_from_user(char * from, char * to, uint size)` (3) `copy_to_user(char * from, char * to, uint size)` (4) `void * kmalloc(uint size)` (5) `kfree(void * buf)` (6) `parity(char * buf_sector_1, char * buf_sector_2, char * buf_sector_3, char * buf_parity_sector)` implemente las funciones pedidas del *driver*. No olvide que también puede implementar la función `driver_init()` y `driver_remove()` si lo considera necesario.

[1] Para simplificar puede asumir que el tamaño del *buffer* siempre será múltiplo del tamaño de un sector.

b) Si utilizó semáforos/mutexes para la implementación explique por qué los utilizó. Si no los utilizó explique por qué no son necesarios en este caso.

Ejercicio 3) Responda las siguientes preguntas justificando sus respuestas:

a) ¿Se puede implementar paginación en cualquier procesador?

b) ¿Por qué la páginación es útil en un sistema que maneja bibliotecas dinámicas (`.dll`, `.so`, `.dynlib`, etc)?

c) ¿Tiene sentido hacer *copy-on-write* sobre las páginas de código de un proceso al ejecutarse un `fork`?

d) ¿Qué debe tener en cuenta el *kernel* de un SO sobre la *TLB* (*translation lookaside buffer*)?

e) Se tiene un sistema con 8 GB de RAM que implementa paginación con páginas de 2 MB. ¿Cuántos bits se necesitan para: (I) direccionar toda la memoria, (II) direccionar las páginas y (III) direccionar el offset dentro de una página ?

f) ¿Se puede implementar segmentación en cualquier procesador?

g) ¿Qué tipo de fragmentación se puede producir en un sistema con paginación: externa o interna? ¿Y en uno con segmentación?

Ejercicio 4)

Encuentre los problemas de seguridad del siguiente código.

```
1  #include <stdio.h>
2
3  void escribir(char *archivo, char *contenido)
4  {
5      FILE    *f;
6      char    nombre[16];
7
8      strcpy(nombre, archivo);
9
10     setuid(0);
11
12     f = fopen(nombre, "w+");
13
14     fprintf(f, contenido);
15
16     fclose(f);
17 }
18
19 int main(int argc, char *argv[]) {
20     escribir(argv[1], argv[2]);
21
22     return 0;
23 }
```