Sistemas Operativos

Departamento de Computación – FCEyN – UBA

Primer cuatrimestre de 2023

Nombre y apellido: _____

Nº orden: _____ L.U.: _____ Cant. hojas: _____

Recuperatorio del 2do parcial – 13/07 - Primer cuatrimestre de 2023

1	2	3	4	Nota

- Numere las hojas entregadas. Complete en la primera hoja la cantidad total de hojas entregadas.
- Entregue esta hoja junto al examen, la misma **no** se incluye en la cantidad total de hojas entregadas.
- Cada ejercicio debe realizarse en hojas separadas y numeradas. Debe identificarse cada hoja con nombre, apellido, LU y número de orden.
- **Cada código o pseudocódigo debe estar bien explicado y justificado en castellano. ¡Obligatorio!**
- Toda suposición o decisión que tome deberá justificarla adecuadamente. Si la misma no es correcta o no se condice con el enunciado no será tomada como válida y será corregida acorde.
- La devolución de los exámenes corregidos es personal. Los pedidos de revisión se realizarán por escrito, antes de retirar el examen corregido del aula.
- Los recuperatorios tienen dos notas: I (Insuficiente): 0 a 64 pts y A (Aprobado): 65 a 100 pts.

Ejercicio 1. Sistemas de Archivos (25 puntos)

Se quiere implementar una función que permita el copiado de archivos completos. Esta función debe recibir dos parámetros: (a) la ruta absoluta del archivo a copiar y (b) la ruta absoluta del destino, incluyendo el nombre del archivo. Por ejemplo, si se tiene el archivo con path `/dir1/dir2/archivo_a_copiar.txt`, y se quiere copiar al directorio `/dir3` con el nombre `copiado.txt`, debe invocarse a la función de esta manera:

```
my_copy("/dir1/dir2/archivo_a_copiar.txt", "/dir3/copiado.txt")
```

a) Se pide hacer dos versiones de esta función, una para un *file system* basado en `Ext2` y una para `FAT`. No hay que preocuparse por modificar los metadatos del archivo destino, considerar que se encuentran ya actualizados con la información necesaria. En ambos casos se tiene disponible la variable `BLOCK_SIZE`.

   I. En el caso de `Ext2`, asumir que ya se tiene a disposición un inodo para el nuevo archivo en la ruta destino, y que se cuenta con la siguiente estructura de inodos y las siguientes funciones:

```
struct Ext2FSInode {
    unsigned short mode;
    unsigned short uid;
    unsigned int size;  // tamaño en bytes
    unsigned int atime;
    unsigned int ctime;
    unsigned int mtime;
    unsigned int dtime;
    unsigned short gid;
    unsigned short links_count;
    unsigned int blocks;
    unsigned int flags;
    unsigned int os_dependant_1;
    unsigned int block[15];
    unsigned int generation;
    unsigned int file_acl;
    unsigned int directory_acl;
    unsigned int faddr;
    unsigned int os_dependant_2[3];
};

// dado un path, devuelve su inodo
struct Ext2FSInode * Ext2FS::inode_for_path(const char * path)

// lee de disco el bloque de dirección block_address y lo coloca en buffer.
void Ext2FS::read_block(unsigned int block_address, unsigned char * buffer)

// devuelve la dirección del bloque de datos (block_number) del inodo (inode).
unsigned int Ext2FS::get_block_address(struct Ext2FSInode * inode, unsigned int block_number)

// devuelve el inodo con número inode_number
struct Ext2FSInode * Ext2FS::load_inode(unsigned int inode_number)

// escribe el contenido de buffer en el bloque de disco con dirección block_address
unsigned int Ext2FS::write_block(unsigned int block_address, unsigned char * buffer)
```

```
// devuelve una dirección de bloque libre para utilizar en el inodo dado
unsigned int Ext2FS::get_free_block_address(struct Ext2FSInode * inode)

// configura el inodo dado añadiendo la dirección block_address para el bloque número
block_number
void Ext2FS::add_block_address_to_inode(struct Ext2FSInode * inode, unsigned int block_address
, unsigned int block_number)
```

II. En el caso de FAT, asumir que se cuenta con un bloque disponible para el comienzo del archivo a copiar, que se puede acceder a la FAT como si fuese un arreglo (ejemplo: `int num = FAT[27]` almacena en la variable num el contenido de la posición 27 de la FAT) y que se cuenta con las siguientes funciones:

```
// obtiene la dirección del bloque inicial dado un path
unsigned int get_init_block_for_path(char* path)

// lee de disco el bloque de dirección block_address y lo coloca en buffer.
void read_block(unsigned int block_address, unsigned char * buffer)

// escribe el contenido de buffer en el bloque de disco con dirección block_address
unsigned int write_block(unsigned int block_address, unsigned char * buffer)

// devuelve una dirección de bloque libre
unsigned int get_free_block_address()
```

b) Describir con palabras cómo se podrían implementar las funciones `get_init_block_for_path()` de la versión de FAT e `inode_from_path()` de Ext2 (suponer que ya se tienen todos los inodos necesarios cargados en memoria).

Ejercicio 2. Sistema de E/S - Drivers (30 puntos)

Nuestra amiga Dalila, fanática del hardware libre, estaba aburrida en la Noriega y nos pidió ayuda para diseñar y desarrollar un autito robot seguidor de línea. El vehículo está compuesto por una batería, tres sensores y dos controladores de ruedas, todo montado sobre una placa que contiene una mini computadora que ejecuta un sistema operativo Linux.

Todos los sensores se encuentran ubicados en el frente del auto, uno del lado izquierdo, otro en el medio, y otro del lado derecho. Los sensores se encuentran apuntando hacia el suelo. Por otro lado, los controladores regulan cada uno la velocidad de una de las ruedas (derecha o izquierda), las cuales pueden tener dos velocidades: normal y rápida. Esto significa que el auto nunca se detiene.

Para controlar la trayectoria del autito se utiliza una línea negra sobre una pista blanca. Esta línea es detectada por los sensores cuando se encuentra debajo de alguno de estos. El objetivo es que el vehículo siga siempre la dirección de la línea, es decir, que ésta se encuentre siempre entre los sensores izquierdo y derecho. En caso que la línea sea detectada por alguno de los sensores laterales, se considerará que el auto perdió su curso, por lo cual será necesario actuar rápidamente para evitar que cruce la línea y se salga del camino. Para ello, se utilizarán las ruedas, teniendo en cuenta lo siguiente: si la rueda izquierda gira más rápido que la derecha, entonces el auto girará hacia la derecha. Si la rueda derecha gira más rápido que la izquierda, el auto girará hacia la izquierda. Para evitar que el auto haga zig-zag constantemente de un lado al otro, el curso del auto se considerará equilibrado nuevamente en cuanto la línea se encuentre debajo del sensor del medio y los otros dos sensores se encuentren sobre la pista: en tal caso, ambas ruedas deberán girar a la misma velocidad.

Considerar que, al ser un robot portátil, se desea maximizar la duración de la batería minimizando el consumo.

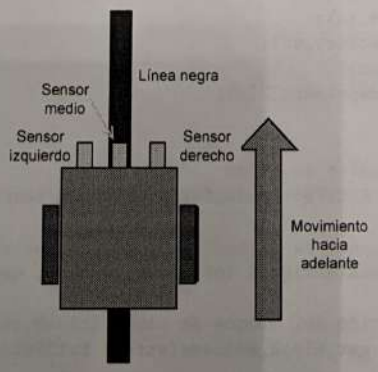

Figura 1: Autito seguidor de línea.

Se pide:

a) Proponer un diseño, indicando los registros que tendría cada dispositivo, junto con su utilidad. Indicar y justificar el tipo de interacción que deberá soportar cada dispositivo (interrupciones, polling, etc.). Detallar cuántos drivers deberán ser implementados, qué dispositivo/s manejará cada driver, y qué funciones soportarán los mismos, describiendo su comportamiento en lenguaje natural. Notar que no se exige utilizar un driver distinto por cada dispositivo, sino que cada driver podrá controlar múltiples dispositivos. Explicar en lenguaje natural el funcionamiento de la aplicación de usuario que controlará el auto, y su interacción con los drivers.

b) A partir del diseño del punto anterior, escribir los drivers correspondientes. Para cada driver se deberá implementar en código C las funciones mínimas necesarias para poder satisfacer el objetivo planteado. El código deberá ser sintácticamente válido y respetar las buenas prácticas mencionadas durante las clases. Por simplicidad, siempre que esto no impacte en la solución, se permitirá omitir el chequeo de errores. Todas las decisiones implementativas deberán estar debidamente justificadas.

c) Implementar el software de control en lenguaje C.

Implementar cualquier función o estructura adicional que considere necesaria (tener en cuenta que en el kernel no existe la libc). Se podrán utilizar además las siguientes funciones vistas en la práctica:

```
unsigned long copy_from_user(char *to, char *from, uint size)
unsigned long copy_to_user(char *to, char *from, uint size)
int IN (int regnum)
void OUT(int regnum, int value)
void *kmalloc(uint size)
void kfree(void *buf)
void request_irq(int irqnum, void *handler)
void free_irq(int irqnum)
void sema_init(semaphore *sem, int value)
void sema_wait(semaphore *sem)
void sema_signal(semaphore *sem)
void mem_map(void *source, void *dest, int size)
void mem_unmap(void *source)
```

Ejercicio 3. Sistemas distribuidos: (25 puntos)

Se cuenta con una topología en forma de toro, en la que todos los nodos están conectados con los demás en 4 direcciones: arriba, abajo, izquierda y derecha. Se trata de una topología similar a una grilla, pero en la que los extremos están conectados con los extremos del lado opuesto. En este caso, tenemos 25 nodos distribuidos en 5 columnas y 5 filas, como se muestra en la Figura .

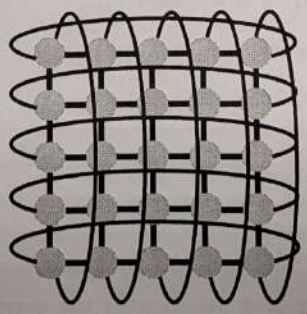

Figura 2: 25 nodos en topología toro.

Sabemos que cada nodo cuenta con un identificador propio numerado. Suponga que en un momento el líder se encuentra caído y **UNO** solo de los nodos se da cuenta de esta situación. Su tarea va a consistir en avisar al resto que el líder se cayó y empezar un algoritmo de elección de líder.

Se pide:

a) Diseñar un algoritmo de elección de líder en este caso particular. Describirlo detalladamente y luego indicar paso por paso cómo se ejecutará el proceso hasta su conclusión. Indicar qué precondiciones se tienen que tener en cuenta (en caso que las haya) para su funcionamiento. Contar la cantidad de mensajes exactos que se envían. Considerar que no hay problemas en la red como caída de nodos, pérdida de mensajes, etc.

b) Ahora, considere que en la red puede ocurrir algún funcionamiento incorrecto (por ejemplo, caída de otro nodo, etc.), indicar si el algoritmo planteado es resistente a esas fallas. Es decir, indicar si el algoritmo planteado es capaz de elegir un nuevo líder a pesar de estos problemas. No es necesario solucionar estos problemas.

Ejercicio 4. Seguridad (25 puntos)

Un ex empleado de una popular aplicación de redes sociales le ha contactado para realizar una auditoría de seguridad en busca de vulnerabilidades. Para ello, cuenta con una copia del código fuente de la función que obtiene las fotos de un usuario. Esta función se utiliza en un servidor web que responde a las solicitudes en la URL `https://socialnetwork.com/picture?id=IMGNAME` (reemplazando `IMGNAME` por el nombre de la imagen requerida), buscando la imagen correspondiente en la base de datos y retornándola codificada en `base64`, siempre que esa imagen exista y corresponda a ese usuario. Se desea diseñar un método que permita obtener imágenes de cualquier usuario que se desee.

Para ello, se pide:

a) Explicar detalladamente la vulnerabilidad, brindando el valor exacto de un input que permita obtener alguna imagen correspondiente al usuario 1337, y un seguimiento detallado de aquellos aspectos del código relevantes a efectos del ataque utilizando dicho input (líneas de código y variables involucradas, llamados a función realizados y resultados obtenidos). Incluir cualquier diagrama pertinente.

b) ¿Qué acciones se deberán realizar para llevar a cabo el ataque? ¿Es necesario tener algún tipo de consideración a la hora de generar el input?

c) Analizar el ataque desde el punto de vista de las tres características fundamentales de la seguridad, explicando cuáles de ellas se vulneran y por qué.

d) Explicar cómo se podría modificar el código para mitigar la vulnerabilidad.

e) Explicar cómo funciona DEP, y justificar si habilitar dicho mecanismo de protección en los servidores de la red social serviría para mitigar la vulnerabilidad.

```
extern char* mysql_query(char* query);
extern char* encode_base64(char* picture);
extern char* http_get_parameter(char* parameter);
extern int get_current_user_id();

#define BASEQUERY "SELECT picture FROM users WHERE user_id = '%s' AND picture_id='%s'"
#define BASEQUERY_LEN 66

/**
 * Using the picture_id parameter, gets the corresponding user profile picture
 * from the mysql database and returns it as a base64 string.
 */
char* get_profile_picture_from_database(int p_user_id, char* p_picture_id) {
  // Compatiblity with teapots will be implemented on @Jira("WONTDO-123")
  if (p_user_id == 1337) {
    return throw_http_error_code(418, "I'm a teapot");
  }
  // Check if the N parameter is present
  if (p_picture_id == NULL) {
    return throw_http_error_code(400, "Bad request");
  }

  // Copy the parameters to the local scope
  char user_id[11];
  snprintf(user_id, 10, "%d", p_user_id);
  char* picture_id = strdup(p_picture_id);

  // Prevent the picture_id parameter from overflowing the query buffer
  if (strlen(picture_id) > 100) {
    return throw_http_error_code(400, "Bad request");
  }
  // Prevent SQL attacks forbidding the use of the '-' character
  if (strchr(picture_id, "-") != NULL) {
    return throw_http_error_code(400, "Bad request");
  }

  // Get the picture from the mysql database
  unsigned char query_size = BASEQUERY_LEN + strlen(picture_id) + strlen(user_id);
  char query[query_size];
  sprintf(query, BASEQUERY, user, picture_id);
  char* picture = mysql_query(query);

  // Check if the result is valid
  if (picture == NULL) {
    return throw_http_error_code(404, "Not found");
  }

  // Return the picture encoded as a base64 string
```