Examen Diciembre de 2013

Lea detenidamente las siguientes instrucciones. No cumplir los requerimientos puede implicar la pérdida del examen.

Formato

- **Indique su nombre completo y número de cédula en cada hoja (No se corregirán las hojas sin nombre, sin excepciones). Numere todas las hojas e indique la cantidad total de hojas que entrega en la primera.**
- **Escriba las hojas de un solo lado y empiece cada problema en una hoja nueva y cada parte del problema de teórico en una hoja nueva.**
- **Si se entregan varias versiones de un problema solo se corregirá el primero de ellos.**

Dudas

- **Sólo se contestarán dudas de letra.**
- **No se aceptarán dudas en los últimos 30 minutos del examen.**

Material

- **El examen es SIN material (no puede utilizarse ningún apunte, libro ni calculadora). Sólo puede tenerse las hojas del examen, lápiz, goma y lapicera en su banco. Todas sus demás pertenencias debe colocarlas en el piso debajo de su asiento.**

Aprobación

- **Para aprobar el examen se debe tener un mínimo de 60 puntos.**

Finalización

- **El examen dura 4 horas.**
- **Al momento de finalizar el examen no se podrá escribir absolutamente nada en las hojas, debiéndose parar e ir a la fila de entrega. Identificar cada una de las hojas con nombre, cédula y numeración forma parte de la duración del examen.**

Problema 1 (32 puntos)

1.
   i. Indique dos métodos de acceso a los registros de un controlador de E/S.
   ii. Indique dos métodos para efectuar una operación de E/S
2. Mencione los servicios fundamentales del sistema operativo concernientes a administración de memoria.
3. Describa y compare los diseño de sistema operativos *monolítico* y *en capas.*
4.
   i. Describa y compare los *hipervisores* tipo *I y II*
   ii. Indique ventajas y desventajas del uso de máquinas virtuales
5.
   i. Describa las tareas que debe hacer un sistema operativo para realizar el *cambio de contexto* de un proceso a otro.
   ii. Mencione 3 ventajas en el uso de *threads*
6.
   i. Describa el algoritmo de planificacion *Multilevel-Feedback-Queue* (Cola multinivel con retroalimentación).
   ii. Mencione las tareas que realiza un planificador de largo plazo.
7. Describa el algoritmo de reemplazo de segunda oportunidad mejorado (*NRU*)
8. Describa el nivel 1 de RAID. Indique un escenario donde su uso es adecuado y otro donde no lo sea.

Problema 2 (33 puntos)

En un sistema operativo experimental se determinó que dada las características de su utilización la mejor estrategia para implementar su sistema de archivo es una que combine la asignación de bloques en forma de lista (Linked Allocation) y la asignación indexada (Indexed Allocation).

Este sistema de archivos cuenta con la siguiente estructura de datos:

```
type block = array [0..4095] of byte; // 4096 bytes

type dir_entry = Record
     name : Array [0..9] of char;     // 10 byte
     type : (file,dir);               // 1 bit
     used : Boolean;                  // 1 bit
     comienzo_archivo: Integer;       // 4 byte
     tamano_archivo: Integer;         // 4 byte
     id_dueno: Integer;               // 4 byte
     id_grupo: Integer;               // 4 byte
     i_node_num : Integer;            // 4 byte
     permisos : array [0..13] of bit; // 14 bits
     End;

type i_node = Record
     i_node_num : Integer;           // 32 bits
     used : Boolean;                 // 1 bit
     data: array [0..4] of Integer;  // 20 byte
     tope: Integer;                  // 4 byte
     size : Integer;                 // 4 byte
     id_dueno: Integer;              // 4 byte
     id_grupo: Integer;              // 4 byte
     reserved : array[0..6] of bit;  // 7 bits
     End;

type inode_table = Array [0..max_i_node_on_disk-1] of i_node;
type fat = array [0..sectors_in_disk-1] of integer;
type disk = Array [0..max_sectors_in_disk-1] of sector;

Var
     TD : inode_table;
     FAT :fat;
     D : disk;
```

Las características de este sistema de archivos son la siguientes:

- los archivos son almacenados siguiendo una estrategia de asignación en forma de lista tipo FAT (File Allocation Table), donde el atributo `comienzo_archivo` del `dir_entry` indica el comienzo del archivo.
- La información de los directoria es almacenada utilizando una estrategia de asignación indexada directa, donde el atributo `i_node_num` (del `dir_entry`) indica el inodo que almacena la información del directorio.
- Las variable TD, FAT y D son globales
- En la FAT el valor 0 representa "fin de archivo", el -1 "sector libre" y el -2 indica que es sector este siendo utilizado por un directorio.
- Los archivos de tamaño 0 no reservan ningún bloque de disco.
- El inodo 0 almacena la información del directorio raíz.

Por otra parte se dispone de los siguientes procedimientos:

- `Procedure readBlock(d: disk; block_num: 0..MAX_BLOCKS_ON_DISK; Var buff: block, Var ok: Boolean)`; Lee de disco el bloque pasado como parámetro (carga el parámetro de salida buff), En el parámetro ok se retorna el éxito de la ejecución de la operación.
- `Procedure writeBlock(d : disk; block_num : 0..MAX_BLOCKS_ON_DISK; buff : block, ok : Boolean)`; Escribe en el bloque pasado como parámetro la información que se encuentra en el parámetro buff. En el parámetro ok se retorna el éxito de la ejecución de la operación.
- `Procedure getInode(cam: array of char; Var nro_inodo: Integer; Var ok: Boolean)`; Retorna en nro_inodo el número de inodo correspondiente al camino absoluto (cam). En el parámetro ok se retorna el éxito de la ejecución de la operación.
- `Procedure dirname(cam : array of char; Var dir: array of char)`; Retorna en el parámetro dir el directorio que contiene al archivo referenciado en el parámetro cam.Ejemplo, dirname('/home/sistoper/arch.txt') = '/home/sistoper'.
- `Procedure basename(cam : array of char; Var base: array of char)`; Retorna en el parámetro base el directorio que contiene al archivo referenciado en el parámetro cam. Ejemplo, basename('/home/sistoper/arch.txt') = 'arch.txt'.

Se pide:

1. Determine cual es la cantidad máxima de elementos (archivos y/o directorios) que se pueden almacenar en este sistema de archivos. Justifique su respuesta
2. Implementar una función (`borrarFile`) que borra un archivo en el sistema. El cabezal de la función es el siguiente:

   ```
   Procedure borrarFile(cam: array of char; Var ok : Boolean);
   ```

   El parámetro cam representa el camino absoluto al archivo (ej. '/home/sistoper/arch.txt'). En el parámetro ok se retorna el éxito en la ejecución de la operación.

Solución:

1) En el caso de que max_sectors_in_disk sea mayor que 5*max_i_node_on_disk, la cantidad de elementos se vería limitada por la tabla de inodos. Por lo tanto la máxima cantidad de elementos se obtenría ocupando todos los inodos y para cada uno de ellos todos los dir_entries de sus 5 bloques.

Un bloque tiene 4096 bytes. Cada dir_entry ocupa 32 bytes (10bytes + 1bit + 1bit + 4*5bytes + 14bits). Por lo tanto en cada bloque de disco se pueden almacenar 128 dir_entries (4096/32).

Cada inodo puede ocupar hasta 5 bloques de disco, por lo tanto cada inodo podría tener 640 elementos (5*128).

La máxima cantidad de elementos es: 640*max_i_node_on_disk.

En el caso de que max_sectors_in_disk sea menor que 5*max_i_node_on_disk, la cantidad de elementos se vería limitada por la FAT. Por lo tanto la máxima cantidad de elementos sería la cantidad de dir_entries que se podrían crear.

En cada sector se pueden almacenar hasta 128 dir_entries. Por lo tanto el sistema podría almacenar 128*max_sectors_in_disk elementos.

2)

```
procedure deleteFile(cam: array of char; Var ok : Boolean)

  var base: array of char;
  var dir: array of char;
  var num_inode_dir: integer;
  var entry: integer;
  var bloque: integer;
  var sectorFAT: integer;
  var sectorAux: integer;
  var buff: array [1..128] of dir_entry;

begin
      dirname(cam, dir);
      basename(cam, base);
      getInode(dir, num_inode_dir, ok);
      if(!ok)
      {
            return;
      }

      entry = 0;
      bloque = 0;
      while((bloque <= TD[num_inode_dir].tope) && !encontre)
      {
            if(entry mod 128 == 0)
            {
                  entry = 0;
                  sectorAux = TD[num_inode_dir].data[bloque];
                  readBlock(D, sectorAux, buff, ok);
                  if(!ok)
                  {
                        return;
                  }

                  bloque++;
            }
```

```
            if((buff[entry].used) && (buff[entry].name == base) &&
                        (buff[entry].type == file))
            {
                  encontre = true;
            }

            entry++;
      }

      if(!encontre)
      {
            ok = false;
            return;
      }

      entry--;
      buff[entry].used:= false;
      sectorFAT = buff[entry].comienzo_archivo;
      writeBlock(D, sectorAux, buff, ok);
      if(!ok)
      {
            return;
      }

      while(sectorFAT != 0)
      {
            sectorAux = FAT[sectorFAT];
            FAT[sectorFAT] = -1;
            sectorFAT = sectorAux;
      }
end
```

Problema 3 (35 puntos)

Se desea modelar el proceso de asignación de claves de usuarios a los funcionarios de la corporación Segurol SA. Para ello implementa una ventanilla de recepción que recibe a los funcionarios en forma presencial y tres equipos para el ingreso de la clave de usuario.

Los funcionarios deberán entregar la cédula de identidad en la ventanilla de recepción para que ésta verifique la pertenencia de la persona a la corporación. En caso que se confirme la pertenencia se registra el nuevo usuario en el sistema y se le comunicará al funcionario su clave de activación. En caso que no se confirme la pertenencia se le avisa al funcionario para que vaya al cuarto de interrogatorios y se procede a llamar a Seguridad.

El funcionario con la clave de activación deberá dirigirse a uno de los tres equipos para elegir su clave de acceso siempre y cuando haya algún equipo libre. Sino esperará en una fila por orden de llegada. Una vez ingresada la clave de activación el equipo retorna un código de confirmación que deberá entregar en la ventanilla con la cual le devolverán su cédula de identidad.

El funcionario que desea entregar el código de confirmación en la ventanilla tendrá prioridad sobre los funcionarios que lleguen a iniciar el trámite.

Se desea modelar utilizando mailboxes los procesos Ventanilla y Funcionario. No se podrán utilizar tareas auxiliares.

Se dispone de las siguientes funciones:

- `mi_cedula(): cedula`
  Devuelve la cedula de indentidad del funcionario
- `validar_pertenencia(cedula): boolean`
  Indica si el funcionario con dicha cedula pertenece a la corporación
- `generar_clave_de_activacion(): string`
  Genera una clave dehttp://download.eclipse.org/releases/indigo/ activación
- `registrar_usuario(cedula): void`
  Registra el nuevo usuario en el sistema
- `introducir_clave(nro_equipo: (1..3), clave_activacion: string):string`
  Utilizada por los funcionarios para ingresar su clave de acceso en un equipo. Retorna el código de confirmación
- `obtener_cedula(codigo_confirmacion: string): cedula`
  Utilizada por la ventanilla para obtener la cedula a devolver
- `llamar_a_seguridad()`
  Ejecutada por la ventanilla para llamar a seguridad.
- `ir_al_cuarto()`
  Ejecutada por el funcionario para ir al cuarto de interrogatorios.

Solución:

Mailboxes infinitos con resolución FIFO, *send* no bloqueante, *receive* bloqueante

```
Mailbox mb_equipos of Integer;
Mailbox mb_fila_entrega of Integer;
Mailbox mb_requiero_atencion of Nil;
Mailbox mb_espero_atencion of Nil;
Mailbox mb_funcionario_cedula of Cedula;
Mailbox mb_ventanilla_resp of {boolean, string};
Mailbox mb_quiero_entregar_codigo_conf of Nil;
Mailbox mb_entrega_conf of String;
Mailbox mb_entrego_cedula of Cedula;
Mailbox mb_listo of Nil;

procedure Ventanilla
var
  cantEspera:integer
  string codigo;
  cedula: Cedula;
begin
  mb_equipos.send(1);
  mb_equipos.send(2);
  mb_equipos.send(3);
  mb_fila_entrega.send(0});

  while(true)
  {
    mb_requiero_atencion.receive();
    mb_fila_entrega.receive(cantEspera);
    if(cantEspera == 0)
    { // No hay nadie esperando para entregar codigo de confimacion
      mb_fila_entrega.send(cantEspera);
      mb_espero_atencion.send(NIL);
      mb_funcionario_cedula.receive(cedula);
      if(validar_pertenencia(cedula))
      {
        registrar_usuario(cedula);
        mb_ventanilla_resp.send(true, generar_clave_activacion());
      }else
      {
         mb_ventanilla_resp.send(false, NIL);
         llamar_a_seguridad();
      }
      mb_listo.receive();
    }else
    { // Hay funcionarios esperando para entregar codigo de confirmacion
       mb_fila_entrega.send(cantEspera - 1);
       mb_quiero_entregar_codigo_conf.send(NIL);
       mb_entrega_conf.receive(codigo);
       cedula = obtener_cedula(codigo);
       mb_entrego_cedula.send(cedula);
       mb_listo.receive();
    }
  }
end procedure;
```

```
procedure Funcionario
var nroEquipo: integer
  cedula: Cedula
  clave, codigo: string
  pertenece: boolean
  cantEspera: integer;
begin
  mb_requiero_atencion.send(NIL);
  mb_espero_atencion.receive();
  mb_mutex_1.receive();
  mb_funcionario_cedula.send(mi_cedula());
  mb_ventanilla_resp.receive({pertenece, clave});
  mb_listo.send(NIL);
  if(pertenece)
  {
    mb_equipos.receive(nroEquipo);
    codigo = introducir_clave(nroEquipo, clave);
    mb_equipos.send(nroEquipo);
    mb_fila_entrega.receive(cantEspera);
    mb_fila_entrega.send(cantEspera + 1); // Quiero entregar codigo confirmacion
    mb_requiero_atencion.send(NIL);
    mb_quiero_entregar_codigo_conf.receive(); // Espero atencion de ventanilla
    mb_entrega_conf.send(codigo);
    mb_entrego_cedula.receive(cedula);  // recibo mi cedula
    mb_listo.send(NIL);
}else ir_al_cuarto(); // Voy al cuarto de interrogatorios
end procedure;

main()
begin
 cobegin
   Ventanilla;
   Funcionario;
    ...
   Funcionario;
 coend
end;
```