Solución Examen Marzo de 2011

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 3 horas y 30 minutos.**

Problema 1 (65 puntos)

Parte teórica (30 puntos, 6/8/8/8)

1. Defina y describa ventajas y desventajas entre los sistemas simétricos y asimétricos.
2. a. Describa el planificador de colas multinivel con retroalimentación (multilevel feedback queue).
   b. Los procesos son generalmente clasificados como *CPU-bound* o *IO-bound*. En un planificador que maneja prioridades, a qué clase de proceso le daría mayor prioridad ?
3. Describa lo que entiende por memoria virtual.
4. Realice un diagrama de una traducción de direcciones de un sistema de paginación de dos niveles. Mencione los registros, componentes de hardware y cache que participan en la traducción.

Parte práctica (35 puntos, 5/10/20)

Sea un sistema de archivos (filesystem) simplificado que cuenta con la siguiente estructura de datos:

```
type sector = array [0..511] of byte;
type fs = Array [0..(max_dirs-1)] of dir;
type dir = array [0..(max_files_on_dir-1)] of file;
type file = Record
               used      : boolean;                  // entrada usada o no
               file_name : array [0..7] of char;   // nombre del archivo
               file_ext  : array [0..2] of char;   // extensión del archivo
               start     : integer;              // direccion de comienzo
               type      : {archivo, directorio};  // tipo de archivo
               dir_index : 1..(max_dirs-1);          // indice en fs
               size      : integer;            // tamaño del archivo en bytes
               info      : array [0..(max_info-1)] of char;// más informacion
            end;
type fat = array [0..(sectors_in_disk-1)] of -1..(sectors_in_disk-1);
type disk = Array [0..(max_sectors_on_disk-1)] of sector;
Var
    SA   : fs;    // sistema de archivos
    D    : disk;  // disco
```

Se pide:

1. Realizar una función que encuentre un archivo en un directorio.
   `Procedure searchFile(dir_entry: dir; name : Array [0..7] of char; ext : Array [0..2] of char; Var entry : 0..(max_files_on_dir-1); Var ok : boolean);`
   Donde `dir_entry` es el directorio donde se está buscando, `name` y `ext` son el nombre y extensión del archivo a buscar, `entry` es el número de entrada donde está el archivo y `ok` determina si la operación finalizó con éxito o no.

2. Implementar una función que remueva una archivo de un directorio.
   `Procedure rmFile(Var dir_entry : dir; name : Array [0..7] of char; ext : Array [0..2] of char; Var ok : boolean);`
   `dir_entry` es el directorio donde se está buscando, `name` y `ext` son el nombre y extensión del archivo a remover, `ok` determina si la función finalizó con éxito.
   Si el nombre del archivo recibido es un directorio, la operación debe finalizar sin éxito.
3. Implementar una función que copie un archivo.
   `Procedure cpFile(d_source : dir; name_s : Array [0..7] of char; ext_s : Array [0..2] of char; Var d_target : dir;`

```
            name_t : Array [0..7] of char; ext_t : Array [0..2] of char;
            Var ok : boolean);
```

`d_source` es el directorio origen, `name_s` y `ext_s` es el nombre del archivo origen, `d_target` es el directorio destino, `name_t` y `ext_t` es el nombre del archivo destino y `ok` determina si la operación finalizó con éxito o no.
En caso de utilizar alguna estructura auxiliar, mencionar y dar semántica de operaciones.
Esta operación no es aplicable a directorios.
Para operar sobre el disco utilice las operaciones: `readSector y writeSector`, dar parámetros y semántica.

Notas generales:

- Las variables D y SA son globales.
- La información del directorio raíz está en `SA[0]`.
- En la fat el valor 0 representa "fin de archivo" y el -1 "sector libre".
- El campo `dir_index` en file es solo usado cuando el archivo es un directorio.
- Tener en cuenta que toda operación debe dejar el sistema de archivos en un estado consistente.

Solución

1.

```
void searchFile ( ... ) {
 int i = 0;
 for (; i < max_files_on_dir; i++)
      if (dir[i].used &&
          dir[i].f_name == name &&
          dir[i].f_ext == ext &&
          dir[i].type = 'archivo') {
            ok = true;
            entry := i;
            return;
      }
 ok = false;
}
```

2.

```
void rmFile ( ... ) {
 searchFile(dir_entry, name, ext, entry, ok);
 if (!ok) return;
 if (dir_entry[entry].size > 0) {
       int i := entry.start;
       while ( i != 0) {
            int j := FAT[i];
            FAT[i] := -1;
            i = j;
       }
       FAT[i] := -1; // Ultimo sector
 }
 entry.used = false;
 ok := true;
}
```

3.

```
bool hayEspacioFAT(int size, int &start) {
      bool ok, primero;
      integer start, i, anterior;
      primero = true;
      start = 0;
      i = 0;

      while ((size > 0) && (i < sectors_in_disk-1)) {
            if (FAT[i] == -1) {
                  if (primero) {
                        primero = false;
                        FAT[i] = 0;
                        start = i;
                        anterior = i;
                  } else {
                        // Seteo el anterior y seteo en 0 el actual por
                        // las dudas que sea el último sector del archivo
                        FAT[anterior] = i;
                        FAT[i] = 0;
                        anterior = i;
                  }
                  size = size - 512;
            }
            i++;
      }
      if (size > 0) {
            // Libero los sectores tomados
            i = start;
            while (i != 0) {
                  start = FAT[i];
                  FAT[i] = -1;
                  i = start;
            }
            return false;
      }
      return true;
}

void cpFile( ... ) {
 int entry, entry_s;
 int start;
 ok = true;

 // Verifico destino
 searchFile(d_target, name_t, ext_t, entry, ok);
 if (ok) {
      ok := false;
      return;
 }
 // Verifico origen
 searchFile(d_source, name_s, ext_s, entry_s, ok);
 if (!ok) return;

 // Busco entry libre
 int i = 0;
```

```
 while (dir_entry[i].used && (i < max_files_on_dir)) {
      i++;
 }
 if (i < max_files_on_dir) {
      entry = i;
 } else {
      ok = false;
      return;
 }
 if (hayEspacioFAT(d_source[entry_s]), start) {
      d_target[entry].used = true;
      d_target[entry].file_name = name_t;
      d_target[entry].file_ext = ext_t;
      d_target[entry].size = d_source[entry_s].size;
      d_target[entry].info = d_source[entry_s].info;
      d_target[entry].type = d_source[entry_s].type;
      d_target[entry].start = start;

      // Copio los sectores
      i = d_source[entry_s].start;
      while (start != 0) {
           writeSector(start,readSector(i));
           start = FAT[start];
           i = FAT[i];
      }
 } else {
      // No hay lugar en disco
      ok = false;
      return;
 }
}
/*
 * Escribe en disco D, en el indice numero 'sector'
 * void writeSector(int sector, sector s);
 *
 * Lee el sector en el indice 'sector' del disco 'D'
 * sector readSector(int sector);
 */
```

Problema 2 (35 puntos)

Se desea modelar utilizando semáforos una fábrica de bombones, la cual tiene 5 máquinas que elaboran bombones de chocolate y 2 que elaboran bombones de licor. La fábrica produce cajas de 20 bombones.
Las máquinas colocan los bombones directamente en la caja y las mismas deberán contener al menos 1 bombón de licor.
Se deberán usar al máximo posible todas las máquinas. Las máquinas podrán poner bombones en las cajas a la vez.

Para esto se cuenta con las siguientes funciones:

- **Producirbombon(): bombon** ejecutada por la máquina. Esta producirá un bombón según el tipo de máquina.
- **Ponerbombon(bombon)** ejecutada por la máquina. Pone un bombón en la caja.
- **NuevaCaja()** ejecutada por la máquina que llena la caja para sustituir la caja llena por una vacía.

Nota:

- No se podrán utilizar tareas auxiliares.

Solución:

```
Semaphore datos;
Semaphore espera;
Semaphore enCajaSem;

bool tieneLicor;
int enCaja;
int cant;
int enEspera;

void Maquina(soyLicor : bool) {
   bool entro;
   while(true) {
       Bombon b := ProducirBombon();
       P(datos);
       if (soyLicor)
            entro := cant < 20;
       else
            entro := (cant < 20 && tienelicor) || (cant < 19);

       if(!entro){ // Hay 19 bombones sin licor y soy maquina sin licor
           enEspera++;
           V(datos);
           P(espera);
       } else {
           cant++;
           tieneLicor |= soyLicor;
           if(cant < 20)
                V(datos); // Si caja no se llena dejo entrar a otro
       }

       PonerBombon(b); // Concurrente
```

```
       P(enCajaSem);
       enCaja++;
       if(enCaja == 20) { // El que pone el 20avo bombon pone caja nueva
            NuevaCaja();
            tieneLicor = false;
            enCaja = 0;
            cant = enEspera;
            while(enEspera > 0) { // Estos son todos sin licor
                 enEspera--;
                 V(espera);
            }
            V(datos); // Ahora pueden volver a entrar mas bombones
       }
       V(enCajaSem);
   }
}

begin
 init(datos, 1);
 init(espera,0);
 init(enCajaSem,1);

 tieneLicor := false;
 enCaja := 0;
 cant := 0;
 enEspera := 0;

 cobegin
  Maquina(true);
  Maquina(true);

  Maquina(false);
  Maquina(false);
  Maquina(false);
  Maquina(false);
  Maquina(false);
 coend
end.
```