Examen Julio de 2004

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. (No se corregirá la hoja que tenga el ejercicio compartido, sin excepciones).**
- **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 ejercicio entero bien hecho y medio más.**

Finalización

- **El examen dura 4 horas.**

Problema 1

1. A continuación se propone una solución al problema de la sección crítica para dos procesos *i* y *j* que ejecutan en un único procesador. Las instrucciones son atómicas.

```
Proceso i
while (true) do begin
      flag_i := true;
      while ( flag_j ) do begin
            flag_i := false;
            while ( flag_j ) do
                  no-op;
            flag_i := true;
      end;
      Sección Crítica
      flag_i := false;
      Sección no crítica
end;
```

```
Proceso j
while (true) do begin
      flag_j := true;
      while ( flag_i ) do begin
            flag_j := false;
            while ( flag_i ) do
                  no-op;
            flag_j := true;
      end;
      Sección Crítica
      flag_j := false;
      Sección no crítica
end;
```

   Determine si es correcta. Justifique brevemente.

2. Un sistema que implementa memoria virtual con paginación utiliza el algoritmo FIFO para la sustitución de páginas.

   Un proceso genera la siguiente secuencia de referencias a páginas de memoria:

   1 2 3 4 5 3 4 1 6 7 8 9 7 8 9 6 5

   a. Determinar cuantos fallos de página se producen según se disponga de 4 ó 5 marcos de página para el proceso. Justifique brevemente.
   b. Aumentando a 6 o más el número de marcos. ¿Se mejoraría la tasa de fallos de página? Justifique brevemente.

3. Se tiene un sistema de gestión de memoria paginada. El espacio de direccionamiento virtual es de 10 páginas de 1024 palabras, la palabra es 2 bytes de tamaño. La memoria física esta dividida en 32 marcos.
   a. ¿Cuántos bits componen una dirección virtual? Justifique brevemente.
   b. ¿Cuántos bits componen una dirección física? Justifique brevemente.

4. En la implementación de que tipo de programas resulta especialmente útil el contar con un servicio de hilos o procesos multihilados.

5. Describa los diferentes componentes involucrados en una operación de entrada-salida incluyendo el propósito específico de cada uno de ellos.

6. Describa brevemente tres métodos de autenticación de usuarios.

7. Explique brevemente para que sirve el bit de “setuid” en Unix.

Problema 2

En vista de los comicios de octubre-noviembre, la corte electoral ha decidido implementar una serie de cambios apuntando a mejorar el funcionamiento de las mesas de votación.

Una mesa de votación, esta compuesta por dos integrantes, un **presidente** de mesa y un **secretario**. Además se dispone de una urna y un conjunto de sobres de votación. Las tareas de cada integrante son:

- **Presidente**: Recibe el número de credencial del votante, verifica la identidad del mismo en el acta de inscripción y le pasa el número de credencial al secretario.
- **Secretario**: Anota en el acta de votación la identidad del votante y lo autoriza a tomar un sobre. El votante no puede tomar el sobre hasta que se lo diga el secretario. Luego de anotar en el acta de sobres elegidos, el secretario le da permiso al votante para entrar al cuarto secreto siempre y cuando el mismo esté libre.

Una vez en el cuarto secreto, el votante pondrá la hoja de votación en el sobre. Cuando esta tarea finalice, retornará a la mesa para depositar el voto. La mesa puede atender más votantes mientras el cuarto secreto este ocupado. En el momento que el votante salga del cuarto secreto a depositar su voto puede tener que esperar a que la mesa termine la atención de otro votante que se encuentra en la mesa haciendo los trámites previos a la entrada al cuarto secreto, pero tendrá prioridad frente a los demás votantes y delegados esperando que la mesa los atienda.

Cada cierto tiempo, los delegados llegan a la mesa de votación y solicitan acceder al cuarto secreto para reponer listas. Un delegado partidario debe presentar a la mesa su identificación de delegado.

Se desea modelar utilizando monitores, los procesos antes descriptos para una mesa de votación. (Presidente, Secretario, Votante y Delegado).

Puede asumir la existencia de las siguientes funciones:

- **`presentar_credencial()`**
  `Devuelve la credencial del votante que ejecuta la función`

- **`presentar_identificacion_delegado()`**
  `Devuelve el identificador del delegado que ejecuta la función`

- **`tomar_sobre(): sobre`**
  `Toma un sobre de la mesa`

- **`obtener_numero_sobre(sobre)`**
  `Obtiene el numero de sobre. Recibe un sobre como parametro`

- **depositar_voto(sobre)**
  Deposita el voto en la urna. Recibe un sobre como parametro

- **validar_numero_credencial(nro_credencial)**
  Valida la identificacion de la credencial.

- **validar_numero_delegado(nro_delegado)**
  Valida la identificacion del delegado.

- **anotar_numero_credencial(nro_credencial)**
  Anota el numero de credencial en el acta del secretario.

- **anotar_numero_sobre(nro_sobre)**
  Anota el numero de sobre en el acta del secretario.

- **sufragar(sobre)**
  Utilizada por los votantes para entrar al cuarto secreto e introducir la lista en el sobre.

- **recargar()**
  Utilizada por los delegados para entrar al cuarto secreto y recargar las listas de votación.

Problema 3

Se desea diseñar un sistema de almacenamiento de archivos para un disco duro de PC que cumpla las siguientes propiedades:

- Para cada archivo deberá almacenar el nombre, la extensión (si la tiene), el tamaño y la fecha de creación del mismo.
- Se podrán tener directorios de cualquier profundidad
- El disco utiliza sectores de 8K
- Deberá manejar asignación indizada con dos niveles.
- Los discos usados tendrán a lo sumo 32 GB de tamaño
- Los archivos deberán poder insertarse y borrarse del disco así como aumentar y disminuir de tamaño.

Para maximizar la eficiencia del almacenamiento de los archivos el sistema deberá guardar a aquellos cuyo tamaño es menor o igual a 1 MB utilizando asignación encadenada (linked) y aquellos que superan esta cota usando asignación indizada (indexed).

Como los archivos pueden crecer o achicarse a lo largo del tiempo se desea que el sistema de archivos sea capaz de cambiar de una forma de almacenamiento a la otra cuando los archivos se extiendan o se contraigan por encima o por debajo de la cota mencionada.

Los nodos a utilizarse para la asignación inidizada ocuparán un sector completo del disco y tendrán esta estructura:

```
type Indice = array [1..MAX_TAM_INDEX] of integer
```

donde cada entrada del arreglo contendrá el índice de un sector del disco. Los nodos de primer nivel referenciarán exclusivamente a nodos de segundo nivel y los de segundo nivel referenciarán a los sectores con los datos de los archivos.

Se pide:

Definir las estructuras de datos necesarias para implementar el sistema de archivos cumpliendo los requerimientos pedidos e implementar la función que permite pasar un archivo almacenado con asignación encadenada a asignación indizada.

Se asume que el disco tiene espacio libre suficiente como para que no se llene durante la transformación pedida.

Se puede disponer de una función `write (dato, sector)` que escribe el dato pasado como parámetro en el sector indicado (el dato puede ser de cualquier tipo).

==========

Soluciones

==========

Problema 1.

1.

Para comprobar que el algoritmo es correcto, hay que demostrar que cumple las tres condiciones de exclusión mutua, progreso y espera limitada.

Para comprobar que no es correcto basta con un contraejemplo.

En este caso no se cumple ya que el siguiente ejemplo muestra un caso para el cual no se cumple la condición de progreso.

```
flag_i := true;
flag_j := true;
while ( flag_j ) do begin
while ( flag_i ) do begin
flag_i := false;
flag_j := false;
while ( flag_j ) do
while ( flag_i ) do
flag_i := true;
flag_j := true;
while ( flag_j ) do begin
while ( flag_i ) do begin
…
```

Si se mantiene está secuencia de ejecución nunca accederan a la sección crítica, y por lo tanto el esquema no progresa.

2.

a. Con 4 marcos. 11 fallos. Se los representa con "*"

1*	2*	3*	4*	5*	5	5	1*	6*	7*	8*	9*	9	9	9	9	5*
X	1	2	3	4	4	4	5	1	6	7	8	8	8	8	8	9
X	X	1	2	3	3	3	4	5	1	6	7	7	7	7	7	8
X	X	X	1	2	2	2	3	4	5	1	6	6	6	6	6	7

Con 5 marcos. 9 fallos. Se los representa con "*"

1*	2*	3*	4*	5*	5	5	5	6*	7*	8*	9*	9	9	9	9	5
X	1	2	3	4	4	4	5	5	6	7	8	8	8	8	8	9
X	X	1	2	3	3	3	4	4	5	6	7	7	7	7	7	8
X	X	X	1	2	2	2	2	3	4	5	6	6	6	6	6	7
x	x	x	x	1	1	1	1	2	3	4	5	5	5	5	5	5

b. En esta secuencia hay 9 páginas distintas, luego como mínimo se tiene que producir 9 fallos de página. Con 5 marcos se producen 9, luego si mantenemos el número de marcos no se va a reducir el número de fallos de página.

3.

a)En memoria paginada las direcciones virtuales se componen de un número de página y del desplazamiento dentro de la página.

El número de páginas totales es 10, por lo que se precisan 4 bits para distiguir entre ellas. Mientras que cada página tiene 1024 palabras, si se referncia por palabra se deberán utilizar 10 bits, si es por byte es de 11.

Entonces con referenciamiento por palabra serían: 14 bits (4 + 10).

Entonces con referenciamiento por byte serían: 15 bits (4 + 11).

b)Para direccionar una dirección física, es necesario indicar su número de marco de página y su desplazamiento:

El número de marcos de páginas totales es 32, por lo tanto, para distinguir entre estos 32 marcos hacen falta 5 bits. El desplazamiento corresponde al tamaño del marco de página, igual al de la página, por lo tanto de 1024 palabras, luego es necesario 10 bits para distinguir entre las palabras u 11 bits si el desplazamiento se expresa en bytes.

En total para representar la dirección física son necesarios 15 bits si se expresa el desplazamiento en palabras o 16 bits si se expresa en bytes.

4.

En aplicaciones que son servidores de múltiples usuarios atendidos en paralelo, como servidores de bases de datos, servidores de archivos, etc. Frecuentemente estos servidores mantienen un pool de hilos, evitando el costo de crearlos bajo demanda. Hilos es especialmente útil si en memoria se mantiene información común a todos los usuarios, por ejemplo información de estado de la aplicación.

5.

* Hardware de E/S: Dispositivos diversos que generalmente comparten un medio común o bus, a donde reportan siguiendo un protocolo específico. Para cada dispositivo o puerto hay, en hardware, un controlador que implementa la parte del protocolo que corresponde a aquél.

* Manejador de interrupciones: Luego de cada instrucción, la CPU verifica una linea de solicitud de interrupciones, y en caso de que ocurra, salva el estado y llama a una rutina manejadora. Esta determina el origen de la interrupción, la procesa y retorna a la CPU, a la instrucción previa a la interrupción.

* DMA: Un controlador especializado que evita ocupar el procesador común como pasamanos para transferir información del dispositivo a memoria, o viceversa. Para eso accede directamente al BUS.

6.

Autenticación por passwords: el usuario o proceso que quiere autenticarse provee, además de su identificador, una password. Pueden asociarse distintas passwords, c/u con distintos derechos de acceso. La ventaja es que son fáciles de entender y usar, la desventaja es que puede ser intencional o accidentalmente descubierta o transferida.

Otra forma, variante de las anterior, es encriptar/desencriptar las passwords al de almacenarlas/recuperarlas, o al transmitirlas por un medio inseguro.

Finalmente, puede hacerse por passwords “de una sola vez”. En este caso el sistema genera aleatoriamente un valor cuyo resultado conoce, para ese usuario. El usuario aplica una función (que recibe ese valor e información extra que sólo el usuario y el sistema tienen) y retorna el resultado al sistema, probando, según sea el esperado o no, si es quien dice ser.

7.

Si un ejecutable tiene el SETUID encendido, y lo ejecuta un usuario distinto al que lo creó, el nuevo proceso tendrá, en efectividad, el UID del usuario dueño del ejecutable. Esto sirve, por ejemplo, para dar un acceso controlado, mediante un ejecutable, a dispositivos de E/S o archivos privilegiados. Por ejemplo, el comando de cambio de password es propiedad de un superusuario, tiene el bit SETUID encendido, y actualiza un archivo de passwords. Sin SETUID, habría que prohibir a los usuarios que cambien su propia password, o darle demasiados permisos a ese archivo.

Problema 2.

Creamos un monitor para administrar el procesamiento en la mesa de votantes, de nombre MesaVotacion. Luego, cada actor en el sistema accedera a los procesos del monitor.

```
// Procedimiento para los votantes
Procedure Votante();
      nro_credencial: integer;      // Numero de credencial
      credencial_ok: boolean;       // Numero de credencial ok?
      sobre: Sobre;                 // Sobre tomado
      nro_sobre: integer;           // Numero de sobre
Begin
   // Entro a la mesa. Espero si están atendiendo a otros..
   MesaVotacion.EntraVotante(true);
   nro_credencial = presentar_credencial();

   // Espero validación del presidente y aviso para tomar el sobre
   credencial_ok = MesaVotacion.VotantePresidente(nro_credencial, true);
   if (credencial_ok) then
         sobre = tomar_sobre();
         nro_sobre := obtener_numero_sobre(sobre);

         if (nro_sobre <> NULL) then
            // Presentar numero de sobre al secretario
            MesaVotacion.VotanteSecretario(nro_sobre);

            // Espera autorización para ir al cuarto
            MesaVotacion.VotanteIrCuarto();

            // Tomo la lista
            sufragar(sobre);
            MesaVotacion.FinalizarVotante(sobre, true);
   endifelse
 // No estoy habilitado para votarendif
   endif
End

// Procedimiento para los delegados
Procedure Delegado();
```

```
      nro_delegado: integer;
      nro_delegado _ok: boolean;
Begin
   // Entro a la mesa. Espero si están atendiendo a otros..
   MesaVotacion.EntraVotante(false);

   nro_delegado = presentar_identificacion_delegado();

   // Espera validación del presidente
   nro_delegado _ok = MesaVotacion.VotantePresidente( nro_delegado, false);

   // Espera autorización del secretario para ir al cuarto
   MesaVotacion.VotanteIrCuarto();

   // Recarga las listas del cuarto secreto
   Recargar();

   // Avisa a la mesa que se va para que entre otro
   MesaVotacion.FinalizarVotante(NULL, false);
End

// Procedimiento para el presidente
Procedure Presidente();
      nro_credencial: integer;
      credencial_ok, hay_delegado: boolean;

Begin
   While (true) do
      nro_credencial := MesaVotacion.esperaPresidente();

      // Verificamos la identidad. Seguro hay votante o delegado
      if (MesaVotacion.hay_delegado()) then
            credencial_ok := validar_numero_delegado(nro_credencial);
      else
            credencial_ok := validar_numero_credencial(nro_credencial);
      end if;

      MesaVotacion.PresidenteLibre(credencial_ok);
      // Libero al secretario para que siga con el procesamiento solo si no es
      //un delegado
```

```
      if (credencial_ok && MesaVotacion.hay_delegado()) then
            MesaVotacion.PresidenteSecretario();
      endif
   EndWhile
End

// Procedimiento para el secretario
Procedure Secretario();
      nro_credencial: integer;
      nro_sobre: integer;
Begin
   While (true) do
      // Esperamos que el presidente nos pase un votante
      nro_credencial := MesaVotacion.SecretarioPresidente();

      // Anotamos, pedimos que tome sobre y le damos la credencial
      anotar_numero_credencial(nro_credencial);

      // le indicamos y esperamos que tome un sobre
      nro_sobre := MesaVotacion.SecretarioVotante();

      // Anotamos, el numero de sobre
      anotar_numero_sobre(nro_sobre);

      // Autorizamos la entrada al cuarto secreto
      MesaVotacion.SecretarioFinVotante();
   EndWhile
End

// Al inicio hacemos un cobegin-coend para lanzar las tareas
cobegin
      Secretario();
      Presidente();
      Delegado(); // Multiples instancias de esta tarea pueden existir
      Votante();  // Multiples instancias de esta tarea pueden existir
Coend;


// Monitor para administrar el acceso a la mesa de votacion
Monitor MesaVotacion is
```

```
// Variables internas del monitor
var
   cnt_votantes_in: integer;     // Cantidad votantes en espera
   cnt_votantes_out: integer;    // Cantidad votantes en espera
   hay_votante: boolean;         // Hay votante?
   hay_delegado: boolean;        // Hay delegado?
   votante_en_cuarto: boolean;  // Hay votante en cuarto
   nro_credencial: integer;      // Numero de credencial
   nro_credencial_ok: boolean;   // Numero de credencial ok?
   nro_delegado: integer;        // Numero de delegado
   nro_delegado_ok: boolean;     // Numero de delegado ok?
   nro_sobre: integer;           // Numero de sobre

   cnd_votante_in: condition;    // Cola de votantes para entrar
   cnd_votante_out: condition;   // Cola de votantes para salir
   cnd_votante_cuarto: condition; // Cola de votantes para cuarto
   cnd_presidente_vot: condition;  // Presidente espera votante
   cnd_presidente_vot_fin : condition; // Votante espera presidente
   cnd_presidente_sec: condition;  // Secretario espera presidente
   cnd_secretario_vot: condition;  // Votante espera secretario

procedure EntraVotante(soy_votante : boolean)
begin
   // Verificar que no esten atendiendo a otro. No me puedo
   // colar en el proceso de otro votante
   if (hay_votante) then
      cnt_votante_in := cnt_votante_in + 1;
      cnd_votante_in.wait();
   endif
   hay_votante := true;
   hay_delegado := not soy_votante;
end;

function VotantePresidente(nro_cred_o_deleg: integer,
                           soy_votante: bolean) : boolean
begin
   if soy_votante
         nro_credencial = nro_cred_o_deleg;
   else
         nro_delegado = nro_cred_o_deleg;
```

```
      endif;
      votanteEsperaPresidente := true;
      // Si presidente esta esperando, lo despierto
      cnd_presidente_vot.signal();

      // Esperar validacion por parte de presidente
      cnd_presidente_vot_fin.wait();

      if (soy_votante) then
         if (nro_credencial_ok) then –-variable inicializada por PresidenteLibre
            // Esperar llamado del secretario para tomar el sobre
            cnd_secretario_vot.wait();
         end if;
      end if;
      // Retorna el resultado de la validacion
      return nro_credencial_ok;
   end;

   procedure VotanteSecretario(nro_sobre_in: integer)
   begin
      // Aviso al secretario que tengo el sobre
      nro_sobre := nro_sobre_in;
      cnd_secretario_sobre.signal();

      // Esperar autorizacion del secretario para entrar al cuarto
      cnd_secretario_cuarto.wait();
   end;

   procedure VotanteIrCuarto()
   begin
      // Antes de ir al cuarto, ya no hay votante
      hay_votante := false;

      // Nos vamos al cuarto secreto, pero...
      // Si hay un votante para despertar, entonces lo hacemos
      if (cnt_votante_out > 0) then
         cnt_votante_out := cnt_votante_out – 1;
         cnd_votante_out.signal();
      else if (cnt_votante_in > 0) then
         cnt_votante_in := cnt_votante_in – 1;
```

```
        cnd_votante_in.signal();
    endif
    // Ahora veo si hay alguien dentro del cuarto
    if (votante_en_cuarto)
        cnd_votante_cuarto := cnt_votante_cuarto+1;
        cnd_votante_cuarto.wait();
    endif;
    votante_en_cuarto := true;
end;

procedure FinalizarVotante(Sobre sobre, soy_votante boolean)
begin
    if (cnt_votante_cuarto > 0)
        cnt_votante_cuarto := cnt_votante_cuarto-1;
        cnd_votante_cuarto.signal();
    else
        votante_en_cuarto := false;
    endif;
    // Esperar que terminen de atender a otro, si ese otro existe
    if (hay_votante) then
        cnt_votante_out := cnt_votante_out + 1;
        cnd_votante_out.wait();
    endif

    // Ahora hay votante saliendo
    hay_votante := true;

    if (soy_votante) then
        // Depositar voto
        depositar_voto(sobre);
    endif

    // Conceptualmente, ya no esta el votante o delegado
    hay_votante := false;

    // Nos vamos, pero...
    // Si hay un votante para entrar, entonces lo hacemos
    if (cnt_votante_out > 0) then
        cnt_votante_out := cnt_votante_out – 1;
        cnd_votante_out.signal();
```

```
      else if (cnt_votante_in > 0) then
         cnt_votante_in := cnt_votante_in – 1;
         cnd_votante_in.signal();
      endif
   end;

   function hayDelegado(): boolean
   begin
      return hay_delegado;
   end;

   function EsperaPresidente(): integer
   begin
      // Si llego primero el votante, no me tranco
      if (not votanteEsperaPresidente) then
         cnd_presidente_vot.wait();
      endif;
      votanteEsperaPresidente = false;
      // Pasamos el numero de credencial o delegado al presidente
      if hay_delegado then
           return nro_delegado;
      else
           return nro_credencial;
   end;

   procedure PresidenteSecretario()
   begin
      // Libero al secretario para que siga con el procesamiento
      cnd_presidente_sec.signal();
   end;

   procedure PresidenteLibre(credencial_ok_in: boolean)
   begin
      // Libero al votante para que siga el proceso
      nro_credencial_ok := credencial_ok_in;
      cnd_presidente_vot_fin.signal();
   end;

   Function SecretarioPresidente(): integer
   begin
```

```
      // Espero a que me despierte el presidente
      cnd_presidente_sec.wait();
      return nro_credencial;
   end;

   Function SecretarioVotante(): integer
   begin
      // Liberar al votante para que tome un sobre
      cnd_secretario_vot.signal();

      // Esperar que nos devuelva el nro de sobre
      cnd_secretario_sobre.wait();

      // Devolvemos el nro de sobre que tomo el votante
      return nro_sobre;
   end;

   Procedure SecretarioFinVotante()
   begin
     // Liberar al votante para que entre al cuarto
      cnd_secretario_cuarto.signal();
   end;


Begin
   // Inicializaciones
   cnt_votantes_in   := 0;
   cnt_votantes_out  := 0;
   hay_votante       := false;
   hay_delegado      := false;
   nro_credencial    := 0;
   nro_credencial_ok := false;
   nro_delegado      := 0;
   nro_delegado_ok   := false;
   sobre             := NULL;
   nro_sobre         := 0;
End Monitor
```

Problema 3.

Estructuras:

```
type File = record
                nombre: array [1..MAX_FILENAME_LENGTH] of Char
                extension: array [1.._MAX_EXTENSION_LENGTH] of Char
                tamaño: Integer
                creacion: Date
                inicio: Integer
                tipo: (archivo, directorio, vacio)
            end;

type Files = array [1..MAX_NUM_FILES] of File;
```

- El arreglo Files contiene un registro para cada archivo o directorio del disco
- Inicio da la el índice del primer sector del archivo o directorio

```
type Directory = array [1..MAX_NUM_CHILDREN] of Integer
```

- Los directorios son archivos cuyos datos son un registro de este tipo. Tiene los índices en el arreglo Files para cada uno de sus archivos y directorios hijos.

```
type FAT = array [1..MAX_TAM_FAT] of Datos (MAX_TAM_FAT = 4194304)

type Datos = record
                  asignacion: (index, link)
                  sector: Integer
             end
```

- Se tiene una FAT cuyos datos se interpretan de la siguiente manera:
    - Cuando asignación = link se utuliza como una FAT normal
        - sector = 0: el sector está vacio
        - sector = 0xFFFF: marca el último sector del archivo
        - en otro caso: índice del siguiente sector del archivo
    - Cuando asignación = index se usa para saber que sectores están ocupados
        - sector > 0: ocupado
        - sector = 0: el sector está vacío

```
type Indice = [1..MAX_TAM_INDEX] of Integer (MAX_TAM_INDEX = 2978)
```

- Los nodos de índices primarios tienen los sectores con los índices secundarios de un archivo.
- Los nodos de índices secundarios tienen los sectores con los datos efectivos del archivo.
- Cuando indice[i] = 0 significa que esa entrada no está en uso.

Transformación:

La función recibe el nombre del archivo a transformar. Se asume que tiene tamaño mayor o igual a 1 MB.

```
files: Files;
fat: FAT;



procedure link2Index (nombre, extension, ruta: String)

var i, inicio, libre, pos, posIndex, posIndex2: Integer;
    end: Boolean;
    index, index2: Indice;

begin
      i := translate (nombre, extension, ruta);
      if i < 0 then
         return error;
      inicio := buscoLibre (fat);
      pos := files[i].inicio;
      files[i].inicio := inicio;
      inicializo(index); // índice primario
      inicializo(index2); // índice secundario
      posIndex := 0;
      posIndex2 := 0;

      repeat
```

```
            index2[posIndex2] := pos;
            fat[pos].asignacion := index;
            pos := fat[pos].sector; // avanzo en la fat
            posIndex2 := posIndex2 + 1;
            if posIndex2 > MAX_TAM_INDEX then
            begin
                  libre := buscoLibre(fat);
                  write(index2, libre);
                  inicializo(index2);
                  posIndex2 := 0;
                  index[posIndex] := libre;
                  posIndex := posIndex + 1;
            end;
      until pos = 0xFFFF;

      write(index, inicio);
end;

function buscoLibre (fat: FAT): integer
var i: integer;
    end: boolean;
begin
      i := 0
      end = false;
      while (i < MAX_TAM_FAT)  and (not end) do
        end := fat[i].sector = 0;
      if end then
        return i;
      else
        return -1;
end;
procedure inicializo (ind: Indice)
begin
      for i := 1 to MAX_TAM_INDEX do
        ind[i] := 0;    // marco el índice como vacio
end;
```

```
function translate (nombre, extension, ruta: String)

var pos, largo, i, j: integer;
    tokens: array of String;
    dir: Directory;
    end: Boolean;
begin
      pos := 0; // directorio raiz
      tokens := parse (ruta, largo); // separa la ruta en sus
                                     // componentes
      for i := 1 to largo
      begin
        dir := read(files[pos].inicio);
        j := 0;
        end := false;
        while (j < NUM_MAX_CHILDREN) and (not end) do
        begin
           if files[dir[j]].nombre = tokens[i] and
                 files[dir[j]].tipo = directorio then
             end := true;
           else
             j := j + 1;
        end;
        if not end then
           return -1;
         else
            pos := dir[j];
      end;

      if files[pos].nombre = nombre and
            files[pos].extension = extension and
            files[pos].tipo = archivo then
        return pos
      else
        return -1;
end;
```