Solución - Examen Marzo de 2003

Problema 1

Se debe responder a las cuestiones aquí planteadas de manera concisa (no más de 15 renglones por pregunta. En todos los casos debe justificar su respuesta.

Pregunta1

a) Comente y explique diferencias entre las formas de planificación de disco siguientes: FCFS, SSTF, SCAN y C-SCAN.

==========

FCFS (First Come First Served)

- servicio por orden de llegada
- hay un mayor movimiento total de la cabeza que da menor desempeño.

SSTF (Shortest Seek Time First)

- se busca aquel que implique un tiempo más corto dada la posición actual de la cabeza.
- hay que tomar alguna política adicional para evitar posposición indefinida.
- mejor tiempo que FCFS, pero no es óptimo.

SCAN

- el brazo del disco parte de un extremo del disco, y se mueve hacia el otro, atendiendo las solicitudes correspondientes a cada cilindro. Cuando llega al otro extremo del disco, invierte la búsqueda
- desventaja: cuando la cabeza "da la vuelta" tiene pocas solicitudes para atender, pues sigue su recorrida en los cilindros que visitó más recientemente. La mayoría de las solicitudes pendientes estarán en los útimos cilindros a recorrer.
- Redunda en un tiempo de espera no uniforme.

C-SCAN

- Variante del anterior. Soluciona el problema de posicionamiento de éste, moviéndose siempre al cilindro inicial luego de visitarlos a todos.

==========

b) ¿Utilizaría el método de asignación enlazada en un sistema donde la mayoría de los accesos no son secuenciales? Justifique su respuesta.

==========

No, el acceso al i-ésimo bloque de un archivo puede requerir i lecturas de disco, siguiendo los punteros hasta llegar al bloque destino.

==========

c) En el contexto de la administración del espacio libre de disco, comente que ventajas y desventajas tienen los mapas de bits.

==========

Ventaja:

- resulta sencillo y eficiente encontrar n bloques libres consecutivos en el disco, sobretodo en computadoras que tienen instrucciones de manipulación de bits, que pueden usarse eficazmente para este fin: por ejemplo una instrucción que dado un registro devuelva la posición del primer bit que tenga 1.

Desventaja:

- no son eficientes si para la mayoría de los accesos no se conserva todo el mapa en memoria real. Esto es razonable sólo si el disco no es de gran tamaño.

==========

Pregunta 2

a) ¿Que es y para que sirve la TLB y que optimización podría utilizarse a nivel de hardware para optimizar su utilización?

==========

La TLB (Translation Look-aside Buffer) es un conjunto de registros construidos con memoria de alta velocidad que, dado un número de página se compara simultáneamente con todas las entradas que contiene y si se encuentra el número de marco correspondiente, entonces se devuelve. Es usado a modo de caché para mantener algunas de las entradas de la tabla de páginas.

A diferencia de la tabla de páginas tradicional, en gral. se evita tener dos accesos a memoria (el acceso a la tabla y el acceso efectivo) para acceder a una localidad.

A nivel de hardware se puede optimizar haciendo que la MMU cargue en la TLB una entrada, luego de resolverla (ej: SPARC, 80x86), en vez de delegarlo al sistema para que lo haga a nivel de software (ej: MIPS).

==========

b)

1. Explique los requerimientos de hardware y software necesarios para soportar un mecanismo de segmentación. ¿Por qué seria bueno utilizar este mecanismo?

   ==========

   Se requiere una tabla de segmentos o descriptores usada para mapear las direcciones virtuales (segmento, offset) en direcciones físicas. La tabla debe mapear en sí números de segmento en el límite y base de cada uno. Si se mantiene en memoria, debe mantenerse registros base y límite de la tabla. Debe verificarse que el offset esté entre ambos, o generar una trap.

   Este mecanismo es bueno porque permite la protección de segmentos, si se utilizan bits de protección asociados a cada entrada de la tabla. El hardware los consultará para evitar accesos ilegales a memoria (ej: escribir en un segmento de sólo lectura). Habitualmente en el segmento se agrega un puntero inverso hacia la tabla.

   ==========

2. ¿Cómo resolvería ud., utilizando el entorno de (1), la necesidad de compartir segmentos entre procesos?

   ==========

   La segmentación permite el compartimiento de código o datos entre distintos procesos. Para los segmentos de código se comparte el diccionario, para los segmentos de datos las alternativas son:

   i. Sólo un descriptor apunta la segmento y otros procesos cuentan con copias que apuntan al principal (dando un nivel de indirección)

   ii. Que hayan descriptores adicionales que apunten al original.

   ==========

3. ¿Cómo implementaría la reorganización de la memoria contemplando (2) y en un ambiente de multiprocesadores?

   ==========

   Ante la necesidad de reorganizar memoria, con la alternativa (i), se debe bloquear el descriptor original, se desplaza el segmento y luego se actualiza el descriptor, pues en el momento de hacerlo puede haber otro proceso ejecutándose. Con la alternativa (ii) se debe hacer lo mismo, pero para todo descriptor, debiendo realizar un scan de los diccionarios que potencialmente lo referencien, buscando las copias.

   ==========

c) ¿Por que es necesario contar con dos modos de ejecución distintos como ser modo supervisor y modo usuario?. Explique en detalle los procedimientos que permiten garantizar la integridad de un sistema operativo multiusuario.

==========

Para asegurar la ausencia de inteferencias entre procesos, en particular la memoria en un ambiente de multiprogramación, donde es compartida entre distintos programas y el mismo sistema. Es posible definiendo un conjunto de instrucciones privilegiadas que el hardware permite ejecutar sólo en modo supervisor, y un protocolo que para aumentar el nivel de ejecución transfiere el control a código autenticado del kernel.

==========

Pregunta 3

a) ¿Qué ventaja o desventaja tiene soportar threads a nivel de usuario en un multiprocesador?

==========

La desventaja es que el sistema sólo gestiona a nivel de tareas o procesos, no de hilos dentro de tareas, y por tanto no les asigna distintos procesadores a hilos de una misma tarea.

==========

b) Dé una desventaja de soportar threads a nivel de núcleo, en comparación a hacerlo a nivel de usuario.

==========

Para el cambio de contexto es necesario llamar al sistema, lo que la hace más lento que la conmutación entre hilos a nivel de usuario.

==========

c) Suponga que usted debe procesar una matriz de información, de N x N, donde N es muy grande. Discuta la conveniencia de utilizar tareas multihiladas o procesos para realizar esta acción.

==========

Es más conveniente el uso de threads colocando la matriz en el espacio común de direcciones, como una variable global. Es razonable esperar que la aplicación que procesa la matriz no requiera proteger sus hilos entre sí, y que en cambio cada thread deba acceder a toda la matriz.

Si se resuelve mediante procesos, debería emplearse un mecanismo de comunicación del sistema, con la pérdida de desempeño que cada llamada implica.

==========

Pregunta 4

a) ¿Qué es un planificador preemptivo?

==========

Es un planificador que asegura que el proceso en ejecución es el de mayor prioridad en el sistema. Esto implica que cuando un proceso A con prioridad p1 está listo para ejecutar, y un proceso B con prioridad p2 (con p1 > p2) está siendo ejecutado, el planificador debe quitarle el procesador a B y despachar A.

==========

b) Discuta sobre la adecuación de un planificador no-preemptivo cuando se consideran aplicaciones con restricciones de tiempo real.

==========

Un planificador no-preemptivo no es adecuado cuando se consideran aplicaciones de tiempo real, porque estas requieren garantías de tiempo de servicio, lo cual sólo es posible si se usa un planificador preemptivo.

==========

c) Describa el funcionamiento de los planificadores a corto, mediano y largo plazo.

==========

Planificador a largo plazo:

- Común en un sistema por lotes, donde no es posible ejecutar inmediatamente todos los trabajos pendientes.
- Elige entre los procesos almacenados en un dispositivo de almacenamiento para pasarlo a la ready queue.
- Controla el nivel de multiprogramación.

Planificador a corto plazo:

- Escoge entre los procesos listos uno para asignarle CPU.
- Ejecuta con frecuencia mucho mayor a la del planificador a largo plazo.

Planificador a mediano plazo:

- Común en los sistemas de tiempo compartido.
- Toma las decisiones de swapping.

==========

Pregunta 5

a) En el contexto de deadlock, explique la diferencia entre:

- prevención
- detección con recuperación del mismo

==========

La prevención consiste en asegurarse de que al menos una de las condiciones necesarias de deadlock no se cumpla. Un ejemplo es prevenir una espera circular imponiendo una ordenación total de todos los tipos de recursos, siguiendo un orden de numeración ascendente, y forzando que para obtener unidades de un mismo recurso sea emitida una única solicitud.

La detección y recuperación no evita el deadlock, sino que examina el sistema para determinar si ha ocurrido, para recuperarse del mismo, ya sea terminando uno o más procesos involucrados, o expropiando recursos.

==========

b) ¿Que medidas de prevención de deadlock conoce?

==========

- Evitar retención y espera,
  - Forzando a que los procesos soliciten todos sus recursos y le sean asignados antes de que comience su ejecución.
  - O evitando que los procesos soliciten recursos sólo cuando no tienen ninguno asignado.
- Evitar no expropiación,
  - Forzando a que si un proceso que retiene algún recurso solicita otro que no se le puede asignar de inmediato, entonces se le expropian todos los que ya tiene.
  - O si un proceso (A) solicita un recurso y no está disponible, sino que está asignado a otro proceso (B) que está esperando recursos adicionales, entonces se le quita y se le asigna a (A).
- Evitar la condición de mutua exclusión,
  - por ejemplo utilizando un spooler para el acceso a una impresora.
- Evitar espera circular,
  - La mencionada en (a).
  - O exigir que cuando un proceso solicite un ejemplar de recurso libere todo otro recurso que sea mayor en el ordenamiento definido en (a).

==========

c) Explique las dificultades asociadas a la expropiación de recursos.

==========

- Debe haber un critero apropiado para seleccionar el proceso al que se le expropiará.
- Debe determinarse el orden de expropiación para minimizar el costo.
- Es difícil hacer retroceder el proceso al que se le expropia hasta un estado seguro y reiniciarlo. Casi siempre debe abortarse, o el el sistema debe conservar más información de estado de los procesos.
- Debe asegurarse que los recursos que se expropien no sean siempre los de un mismo proceso, posponiendo su terminación en forma indefinida.

==========

d) Que problemas presenta el algoritmo de Dekker para mutuo exclusión de procesos.

==========

Satisface los requisitos para el problema de la sección crítica, pero presenta un busy-wait, pues espera el cambio de una flag que indica qué proceso puede acceder a su sección crítica.

Sólo es aplicable para 2 procesos y no es facilmente generalizable para N procesos.

==========

Problema 2

Sea un consultorio médico, el cual posee una sala de espera y tres médicos. En el consultorio pueden haber como máximo 5 pacientes. Los tres médicos existentes atienden a una sola persona de la sala de espera por vez cada uno, teniendo prioridad los niños frente a los adultos.

Cuando un médico termina de atender al paciente, se deja que el próximo entre a la sala de espera. Si no hay ninguno, el medico lee la ultima revista de neurología pediátrica durante un rato y vuelve a ver si hay alguien. Si no lo hay vuelve a leer y así sucesivamente.

Se pide:

Modelar este problema en Ada implementando tareas para los pacientes, los médicos y la sala de espera.

Notas:

Se dispone de los procedimientos:

- **Atender()** que atiende un paciente y debe ser ejecutado en el encuentro entre el paciente y el medico.
- **Leer_Revista()** que hace que el medico lea la revista por un rato.

Se dispone de la función **que_soy()** que devuelve si un paciente es niño o adulto.

Las tareas para los pacientes son creadas dinámicamente por otra tarea que no es necesario implementar.

```
=====================================================================
=====================================================================

procedure Atender is
begin
...
end Atender;

procedure Leer_Revista is
begin
...
end Leer_Revista;

task MEDICO is
  entry niño;
  entry adulto;
end MEDICO;

task body MEDICO is
  integer id;
  boolean hay_paciente;
begin
  accept identidad(i :in integer)
     id = i;
  end;

  loop
    sala.libremedico(id, hay_paciente);
    if (hay_paciente) then
       accept atención() do
```

```
          Atender();
        end;
    else
       Leer_Revista();
    end if;
  end loop;
end MEDICO;

task SALA is
  entry entrar;
  entry salir;
  entry libremedico;
end SALAR;

task body SALA is
  n, idm :integer;
  libre: array[1..5] of char;   // A adulto, N niño, '' vacio
  id: char;
begin
  n := 0;
  loop
    select
    when (n < 5) =>
      accept entrar(ident : in char ; i : out integer) do
        id := ident;
        i  := dar_indice_libre_array();
      end ;
      libre[i] := id;
      n := n+1;
    or
      accept salir(i : in integer) do
        libre[i] = '';
      end ;
      n := n-1;
    or
      accept libremedico(id_medico: in integer, hp: out boolean) do
         if (n = 0) then
            hp = false;
         else
            hp = true;
         end if;

         idm = id_medico;
      end accept;

      i = busco_paciente_para_atender_por_prioridad_niño();

      if (i <> 0) then
         accept atención[i](id_medico: out integer) do
            id_medico := idm;
         end accept;
      end if

    end select;
  end loop;
```

```
end SALA;

task type PACIENTE;

task body PACIENTE is
   i, id_medico: integer;
   mi_tipo:  char;
begin
  mi_tipo = que_soy();    // niño o adulto

  sala.entrar(mi_tipo, i);
  sala.atención[i](id_medico);
  medicos(id_medico).atencion();
  sala.salir(i);
end PACIENTE;
```

==========

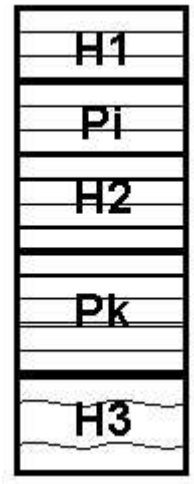
==========

Problema 3

Se desea implementar un sistema operativo con multiprogramación (gestionando un máximo de 128 procesos) y memoria virtual paginada. Se debe tener en cuenta lo siguiente:

- CPU con 32 bits para su conexión al bus de direcciones y un Registro de Reubicación.
- Un disco de 2GB y sectores de 1KB, para utilizarlo en exclusiva como soporte de la memoria virtual.
- Páginas de 4KB y Tabla de Páginas completa residiendo en Memoria Principal.
- Un único espacio de direcciones virtuales para todos los procesos y lo más grande que se pueda.

La memoria virtual, en un momento dado, tiene el aspecto de la figura siguiente:



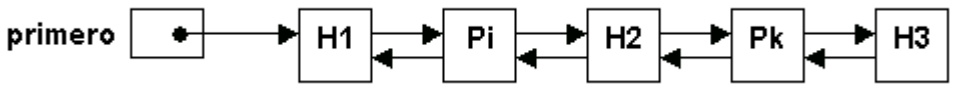
Puede observarse que las tres primeras páginas están vacías (primer hueco **H1**), las tres páginas siguientes están ocupadas por el proceso **Pi**, las cuatro siguientes están libres y las cinco siguientes están ocupadas por el proceso **Pk**, y el resto de páginas se supone que están libres.

Se trata, entre otras cuestiones, de analizar si es más eficiente en ocupación de memoria el gestionar la memoria virtual libre y ocupada mediante un **mapa de bits** o una única **lista doblemente encadenada** tal que:

Los elementos de la lista (que incluye tanto la descripción de zonas de memoria libre "huecos" como las ocupadas por procesos), están ordenados por dirección

Se usa el algoritmo NEXT FIT (siguiente que sirva a partir del ultimo asignado) para asignar memoria a un proceso.

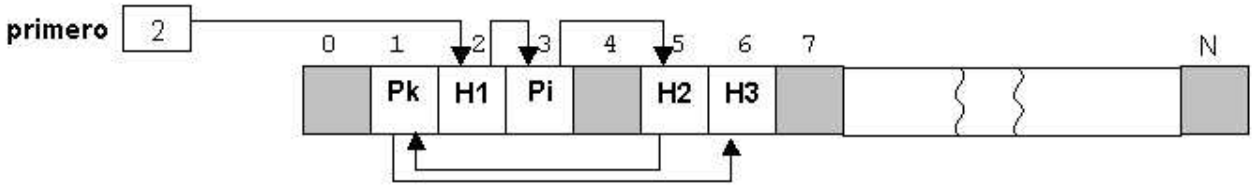
Una representación lógica de la lista doblemente encadenada sería la siguiente:



Donde el elemento “H1” describe el primer trozo de la memoria (primer hueco que abarca las páginas 0, 1 y 2), el elemento “Pi” describe el trozo de memoria virtual ocupado por un proceso (el Pi que abarca las páginas 3, 4 y 5) y así sucesivamente.

Como no se desea trabajar con memoria dinámica, se decide implementar la lista anterior con un array de un tamaño predeterminado (en principio el de la máxima fragmentación “número de trozos” en que puede estar dividida la memoria virtual).

Para completar la implementación, se dispondrá de la variable “primero” que indicará cuál es el primer elemento de la lista, de forma que la representación del estado de la memoria virtual de la figura mediante el array podría ser la siguiente:



Donde sólo se indica el puntero al siguiente para no complicar el dibujo. Las posiciones del array no utilizadas para describir los **trozos** actuales en que está fragmentada la memoria, se han expresado con rectángulos sombreados.

Se pide:

1. Comparar la eficiencia de memoria del mapa de bits frente a la lista encadenada:
   a. Indicar cuál sería el tamaño del mapa de bits expresado en bytes.
   b. Recordando que el sistema puede gestionar un máximo de 128 procesos concurrentemente, razonar si el número máximo de **trozos** en que puede estar dividida la memoria virtual es 256 ó 257.
   c. Indicar los campos (con su significado) que tendría cada elemento del array de la lista.
   d. Supuesto que los campos anteriores tienen que ocupar cada uno de ellos un número de bytes entero (1, 2, 3, etc.) y eligiendo siempre el valor menor para poder representar cada campo, indicar el tamaño total (en bytes) que ocuparía el array donde se implementaría la lista encadenada.
2. Indicar qué información debería guardarse en el descriptor de cada proceso (tanto para el caso del mapa de bits como el de lista encadenada) para que cuando un proceso termine, pueda liberarse la zona de memoria virtual ocupada por el proceso. Poner como ejemplo el caso del proceso **Pk** de la figura.

==========
==========

1.a) Tamaño del Mapa de Bits expresado en bytes

El tamaño de la Memoria Virtual será de 2GB al ser el tamaño del disco y posible de direccionar con 32 bits (que permite direccionar hasta 4GB). Como las páginas son de 4KB, habrá un total de 2GB/4KB = 0,5Mpáginas = 524.288 páginas.

Por cada página se necesita un bit en el Mapa de Bits, siendo su tamaño total: 524.288 / 8 = 65.536B = 64KB

1.b) Número máximo de trozos en que puede estar fragmentada la Memoria Virtual

Para que la fragmentación máxima fuese 257, el aspecto de la memoria debería ser:

Donde cada proceso (rectángulo sombreado) tiene hueco a su izquierda y a su derecha

Esta situación no es alcanzable ya que debe darse al crear el último proceso (el 128) y significaría que, encontrado un hueco, el proceso ocupa un trozo intermedio del mismo, dejando dos nuevos huecos, uno a su izquierda y otro a su derecha. La realidad es una fragmentación máxima de 256 con un aspecto como el siguiente:

Donde el último proceso creado se ubicó en el primer hueco que quedaba al principio y era mayor que él.

1.c) Campos de cada nodo de la lista encadenada

Nombre	Significado del campo y valores que puede tomar
Tipo	Enumerado que puede tener tres valores posibles: **Libre** => Nodo sin utilizar **Proceso** => Nodo que describe un trozo de MV ocupada por un proceso **Hueco** => Nodo que describe un trozo de MV libre (un hueco)
DirEnDisco	Dónde empieza el trozo, bastará con indicar página inicial del trozo
Tamaño	Puede expresarse en número de páginas que es la unidad de asignación
Siguiente	Índice del array donde se encuentra el siguiente nodo de la lista
Anterior	Índice del array donde se encuentra el nodo anterior de la lista

1.d) Tamaño total (en bytes) ocupado por el array que implementa la lista encadenada

Tipo requiere un byte.

DirEnDisco y Tamaño deben poder expresar un valor cualquiera dentro del rango de páginas (524.288 = 219) y por lo tanto requieren 3 bytes cada uno.

Sig y Ant necesitan 2 bytes cada uno: (rango de 0..255 en el array) más indicación de NULL.

Luego cada nodo ocuparía 11Bytes.

El tamaño total sería 256 * 11 = 2.816Bytes

2.a) Información en el descriptor para el caso de Mapa de Bits

Bastaría con indicar página inicial y número de páginas, para saber a partir de qué bit y cuántos hay que cambiar de valor. En el caso de Pk sería PáginaInicial = 10 y NúmeroPáginas = 5

2.b) Información en el descriptor para el caso de Lista Encadenada

Bastaría con indicar el índice del array de la lista encadenada donde se describe el trozo ocupado por el proceso, pues el tamaño ya se indica dentro de la propia lista encadenada.

En el caso de Pk, sería un índice que, en principio, desconocemos, pero si nos fijamos en la figura de la lista, en ese caso concreto sería un "1".

==========
==========