Examen Febrero 2008

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.**

Justificación

- **Toda solución de un problema deberá estar debidamente justificada**
- **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

Justifique brevemente cada respuesta.

1. En un equipo multiprocesador, ¿qué beneficio se obtiene el tener un sistema operativo simétrico frente a uno asimétrico?
2. Realice y describa un esquema de los estados y las transiciones entre los estados de los procesos en un sistema.
3. ¿Qué entiende por planificador expropiativo (*preemptive scheduling*)?
4. Un sistema cuenta con tres tipos de procesos: de tiempo real, procesamiento por lotes (*batch*) y de usuario. Desarrolle un algoritmo de planificación adecuado para este sistema, mencionando las características más relevantes.
5. Describa las cuatro condiciones que, si ocurren simultáneamente, pueden generar un *deadlock*.
6.
    a. ¿Para qué es utilizada la TLB (*Translation Look-aside Buffer*)?
    b. ¿La TLB es un dispositivo de hardware o software?
2. En un sistema que cuenta con el esquema de memoria virtual con paginación, cómo se asegura que no se acceda a la memoria de otro proceso?
3. ¿Cuáles son las ventajas de utilizar segmentación con respecto a paginación?
4. ¿Qué entiende por paginación bajo demanda (*Demand Paging*)?
5. Describa dos técnicas utilizadas para administrar el espacio libre (bloques libres) que hay en los discos.
6. Sea un disco con 32 cilindros numerados del 0 al 31, y la siguiente cola de pedidos:

    12, 18, 1, 15, 2, 13, 17, 6, 22 y 16.

    Haga una tabla que muestre la secuencia de planificación de atención de los pedidos para los algoritmos First Come First Served, Shortest Seek Time First, SCAN, C-SCAN, LOOK y C-LOOK. Suponga que la cabeza del disco está ubicada sobre el cilindro 30.

    Nota: En los algoritmos que pueda haber más de una opción, elija una y aclare el criterio.

Problema 2

Un computador utiliza gestión de memoria virtual *segmentada*. La memoria disponible para los procesos es de 64 bytes y el máximo tamaño de segmento es 32 bytes. En este computador vamos a ejecutar un proceso cuya imagen de memoria tiene 4 segmentos. En un determinado instante, la tabla de segmentos indica la siguiente información:

Nº de Segmento	**Dirección Base**	**Tamaño (bytes)**
1	56 = 38b16	8
3	16 = 10b16	32 = 20b16

El segmento 0 tiene un tamaño de 16 bytes y el segmento 2 un tamaño de 24 bytes.

Se realiza la siguiente secuencia de referencias virtuales a memoria: 30b16, 0Cb16, 6Eb16, 4Fb16 y 27b16.

Teniendo en cuenta que **no hay más procesos en el sistema**, se pide contestar a las siguientes preguntas, **justificando siempre adecuadamente**:

a) ¿Podemos asegurar un formato de dirección virtual y de dirección física? Indicar una propuesta para cada caso.
b) Indicar (con la mayor claridad posible) el contenido de las tablas que se utilizan para realizar la traducción dinámica de direcciones mediante traducción directa, indicando además cuándo hay que actualizar cada una de ellas, y el contenido de las mismas antes de realizar los accesos a memoria.
c) Calcular las direcciones reales correspondientes. Indicar el contenido de las tablas y estructuras de datos utilizadas en el punto anterior mientras se ejecuta el proceso. En caso de ser necesario el uso de alguna política de reemplazo / asignación de espacio; especifique cuál eligió.

Solución:

a) Por un lado, como la memoria es de 64 bytes, para direccionarla necesitaremos de 64 = 2^6, 6 bits para la dirección física de la memoria.

Dirección física de memoria
6 bits

Como el tamaño de segmento máximo es de 32 bytes, necesitaremos 5 bits para direccionarlo (como mucho). Como en el problema el proceso ocupa 4 segmentos, como mínimo necesitaremos 2 bits para direccionar el segmento.

Dirección virtual de memoria	
Mínimo 2 bits	Máximo 5 bits
Número segmento	desplazamiento

De todas formas, con la información del ejercicio, sólo podemos asegurar un mínimo de bits para el segmento, que sería 2, pero no una cantidad fija, pues no nos proporciona en ningún lado del enunciado el número de segmentos máximos a direccionar por los procesos.

b) Tablas para la traducción directa de direcciones:

Segmento	¿Está en memoria?	Tamaño (bytes)	Dirección memoria
0	no	16	----
1	si	8	111000
2	no	24	----
3	si	32	010000

Tabla de segmentos del proceso principal

Memoria principal	Rango direcciones
Libre	0 – 15
Seg. 3	16 – 47
Libre	48 – 55
Seg. 1	56 – 63

La tabla de segmentos cambiará cuando carguemos un segmento en memoria.

c) Al traducir las direcciones es necesario comprobar que el desplazamiento es menor que el tamaño del segmento, pues en caso contrario producirá un error por desbordamiento de segmento. Además, puesto que en el enunciado no se especifica ningún dato en relación a la política de asignación de espacio y reemplazo, dicha elección queda a criterio del alumno. En este caso hemos optado por ubicar el segmento 0 en memoria en un hueco libre y en el caso del segmento 2 se ha reemplazado por el 3.

Dir Virt Hex	Segmento 2bits	Desplazamiento 5bits	Dir. Física memoria 6bits	Observaciones
30b16	01 (1)	10000 (16)	----	ERROR
0Cb16	00 (0)	01100 (12)	001100 (12)	FALLO (sin sustitución)
6Eb16	11 (3)	01110 (14)	011110 (30)	ACIERTO
4Fb16	10 (2)	01111 (15)	011111 (31)	FALLO (con sustitución)
27b16	01 (1)	00111 (7)	111111 (63)	ACIERTO

Tablas para la traducción directa de direcciones:

Segmento	¿está en memoria?	Tamaño (bytes)	Dirección memoria
0	si	16	000000
1	si	8	111000
2	si	24	010000
3	no	32	------

Tabla de segmentos del proceso principal:

Memoria principal	Rango direcciones
Seg. 0	0 – 15
Seg. 2	16 – 39
Libre	40 – 55
Seg. 1	56 – 63

Problema 3

Se desea modelar con monitores el siguiente juego de mesa:

Del juego forman parte N Jugadores y un Jefe de Mesa.

Los jugadores pueden mover sus fichas todos al mismo tiempo siempre y cuando el Jefe de Mesa no este reordenando el tablero, lo cual hará cada vez que termine de pensar sus cambios. Esto implica que el Jefe de Mesa debe esperar a que la mesa quede libre de jugadores antes de reordenar el tablero.

El Jefe de Mesa además de la función de reordenar el tablero, dejará cartas con instrucciones respecto al comportamiento que debe tomar el jugador en caso que el jefe de mesa este esperando para reordenar el tablero. Hay 2 tipos de cartas, una que indica que se debe esperar a que el Jefe de Mesa reordene el tablero para poder jugar, y otra que le permite jugar antes del reordenamiento de fichas.

El Jugador debe sacar una carta cada vez que quiera jugar y el Jefe de Mesa este esperando para reordenar el tablero. Solamente una persona (Jugador/Jefe de Mesa) podrá tener acceso al mazo de cartas en cada momento.

En caso que no queden cartas en la mesa el jugador deberá esperar que el Jefe de Mesa reordene el tablero. Asimismo, el mazo nunca puede tener más de 10 cartas apiladas.

El Jefe de Mesa deberá poder hacer las 2 funciones a la vez (pensar reordenamiento y reordenar con elegir carta y colocarla en mazo).

Se dispone de las siguiente funciones que son ejecutadas por los jugadores:

- pensar_jugada()
- sacar_carta_de_mazo():Carta
- jugar()

Se dispone de las siguientes funciones que serán ejecutadas por el Jefe de Mesa:

- pensar_reordenamiento()
- reordenar_tablero()
- elegir_proxima_carta():Carta
- colocar_carta_en_mazo(Carta)

Nota:

- Carta se modela con un enumerado [ESPERAR,JUGAR]
- No se pueden usar tareas auxiliares pero si se puede representar a los actores con más de un proceso

Solución:

```
type Mazo = Monitor
        cartas: integer;
        mazo_lleno: condition;

        Procedure agregar(c: Carta)
        begin
           if (cartas = 10) then
                 mazo_lleno.wait();
           colocar_carta_en_mazo(c)
           cartas++;
        end

        Procedure sacar(): Carta
           c: Carta;
        begin
           if (cartas = 0) then
                 c := ESPERAR; // tiene que esperar
           else
           begin
                 c := sacar_carta_de_mazo()
                 cartas--;
                 mazo_lleno.signal();
           end
           return c;
        end

        begin
           cartas := 0;
        end
end Mazo

type Tablero = Monitor
        jefeEspera: boolean;
        esperoPorJefe, esperoPorJugador: condition;
        jugando: integer;

        Procedure jefeEsperando(): boolean
        begin
           return jefeEspera;
        end

        Procedure esperarPorJefe()
        begin
// Puede ocurrir que un jugador saque carta porque el Jefe esta reordenando,
// pero luego cuando tiene que esperar el jefe ya se haya ido, en ese caso
// no espero
           if jefeEspera
                 esperoPorJefe.wait();
                 esperoPorJefe.signal(); // Despierto al que sigue
```

```
            endif
        end

        Procedure empiezoAJugar()
        begin
            jugando++;
        end

        Procedure terminoDeJugar()
        begin
            jugando--;
            if (jugando = 0) then
                  esperoPorJugador.signal();
        end

        Procedure reordenar()
        begin
            jefeEspera := true;
            if (jugando > 0) then
                  esperoPorJugador.wait();
            reordenar_tablero();
            jefeEspera := false;
            esperoPorJefe.signal();
        end

        begin
            jefeEspera := false;
            jugando := 0;
        end
end Tablero

Procedure Jugador ()
        c: Carta;
begin
        while(true) do
        begin
            pensarJugada();
            if (Tablero.jefeEsperando()) then
            begin
                  c := Mazo.sacar();
                  if (c = ESPERAR) then
                        Tablero.esperarPorJefe()
            end
            Tablero.empiezoAJugar();
            jugar();
            Tablero.terminoDeJugar();
        end
end

Procedure JefeReordena ()
begin
        while(true) do
```

```
        begin
           pensar_reordenamiento();
           Tablero.reordenar();
        end
end

Procedure JefeCarta()
        c: Carta;
begin
        while(true) do
        begin
           c := elegir_proxima_carta();
           Mazo.agregar(c);
        end
end

main()
begin
   cobegin
        JefeCarta();
        JefeReordena();
        Jugador() ... Jugador()
   Coend
end
```