Examen Febrero 2007

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

Justifique brevemente cada respuesta.

1. ¿En un sistema computacional, qué permite la protección de CPU?
2. ¿Describa 2 actividades que son principales en la administración de la memoria de un computador?
3. Describa los 5 servicios básicos que debe brindar un sistema operativo.
4. Proponga una estructura de datos necesaria para poder representar hilos (*threads*) a nivel del sistema operativo. Mencione solamente los campos principales del bloque descriptor del proceso.
5.
    a. Mencione y realice un diagrama con los estados y transiciones de los procesos.
    b. Discuta, según las transiciones, si un núcleo (*kernel*) es expropiativo (preemptivo) o no.
6. ¿Qué quantum de CPU (mayor, menor o igual) le daría a los procesos que son catalogados como consumidores intensivos de CPU (*CPU-bound*), con respecto a los procesos consumidores de entrada-salida (*IO-bound*)?
7. Se tiene un computador con 2 procesadores y con un sistema operativo de procesamiento simétrico, cuyo planificador tiene una política *Round-Robin* de quantum 1.
   ¿Cual será el tiempo de espera promedio total de los siguientes procesos?

Proceso	Burst Time
P1	5
P2	6
P3	2
P4	3

   Notas:
    - Asuma que el sistema comienza con los 4 procesos en la cola de procesos listos con el orden P1, P2, P3 y P4.
    - Asuma que el cambio de contexto entre dos procesos es despreciable.
8. En un sistema operativo con un esquema de paginación:
    a. ¿Para qué se utiliza la tabla de páginas?
    b. ¿A qué nivel se mantiene en memoria (memoria del sistema operativo o del proceso)?
    c. ¿El *Translation Look-aside Buffer* para qué sirve?
9. En un sistema con memoria virtual, describa el algoritmo de reemplazo de segunda chance (*Second-chance algorithm*).
10.
    a. ¿Los manejadores de dispositivos (*device drivers*) son un componente de hardware o software?
    b. ¿Cuál es su función principal?

Solución:

Nota: Las respuestas se muestran como una guía de la solución. No necesariamente están completas.

1) La protección de CPU permite que los procesos no se apoderen de forma indefinida del recurso procesador. Si no se existiera en una ambiente de tiempo compartido, los procesos podrían hacer un uso excesivo del procesador, limitando a otros del recurso.

2)

- Mantener que partes de la memoria están siendo utilizadas y por quién.
- Decidir cuales procesos serán cargados a memoria cuando exista espacio de memoria disponible.
- Asignar y quitar espacio de memoria según sea necesario.

3)

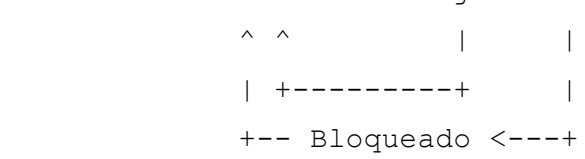
- Ejecución de programas.
- Operaciones de Entrada/Salida.
- Manipulación del sistema de archivos.
- Comunicación entre procesos.
- Manipulación de errores.

Faltan descripciones.

4) Para poder representar hilos (threads) es necesario tener estructuras para la pila (stack), el contador de programa (program counter) y los registros de cada hilo. Se debe modificar el PCB de forma tal de que cada hilo tenga estas estructuras de forma independiente.

5)

a.



Falta mencionar las transiciones.

b. Si el planificador es invocado solamente cuando un proceso finaliza su ejecución o cuando se bloquea, el núcleo no es expropiativo. Si, además, el planificador es invocado cuando un proceso pasa de estado bloqueado a listo o de ejecutando a listo, el núcleo se dice que es expropiativo.

6) La prioridad sobre un recurso es inversamente proporcional al uso del mismo. Los procesos intensivos en CPU (CPU-bound) deberán tener una prioridad baja sobre el recurso procesador, por lo que

se les asignará un quantum de tiempo menor con respecto a los procesos intensivos en entrada-salida (I/O bound).

De estar forma, los procesos que no hacen un uso desmedido del procesador podrán acceder con mayores posibilidades.

7) Ejecutarán de la siguiente forma:

Procesador	Cola listos
(P1,P2)	(P3,P4)
(P3,P4)	(P1,P2)
(P1,P2)	(P3,P4)
(P3,P4)	(P1,P2)
(P1,P2)	(P4)
(P1,P4)	(P2)
(P1,P2)	()
(P2)	()
(P2)	()

(TE(P1) + TE(P2) + TE(P3) + TE(P4)) / 4 = (2 + 3 + 2 + 3) / 4 = 2.5.

8)

a. Su función principal es para poder realizar la traducción de página virtual a frame físico.

b. La estructura es guardada a nivel del sistema operativo. De otra forma, el proceso podría modificarla. También se puede mencionar que el proceso la puede acceder, pero solo en formato lectura.

c. Cumple la función de memoria cache de las entradas de la tabla de páginas. Esto permite mejor velocidad de acceso, ya que si se logra un hit en la cache no es necesario acceder a memoria física a buscar la entrada correspondiente de la tabla de páginas.

9) El algoritmo de segunda chance conserva una cola circular con los frames según el orden de carga. Cada vez que se carga un frame se lo pone al final de la cola y se marca el bit de referencia en 0. A su vez, cada vez que el frame es accedido se marca el bit de referencia en 1. Posteriormente, al ejecutarse el algoritmo de reemplazo, se hace una recorrida desde el comienzo de la cola buscando el frame a reemplazar. Para cada frame se tienen dos posibilidades:

a. Si el frame tiene el bit de referencia en 1, entonces se le da una segunda oportunidad moviéndolo hacia el final de la cola y marcando su bit de referencia en 0.

b. Si el frame tiene el bit de referencia en 0, es el elegido para ser reemplazado y el algoritmo finaliza.

10)

a. Software.

b. Hacer la traducción de los pedidos que el sistema realiza sobre el dispositivo "dialogando" con la controladora del dispositivo. El manejador de dispositivo (device driver), por lo general, implementa un conjunto de primitivas de alto nivel (impuestas por el sistema operativo), que realizan la traducción de los pedidos y respuestas que genera el sistema. Para esto, es necesario que se pueda comunicar con el controlador del dispositivo en el lenguaje que éste propone.

Problema 2

Tenemos un sistema operativo que gestiona su memoria con memoria virtual implementada como paginación bajo demanda, de un solo nivel, y el algoritmo de segunda oportunidad como política de reemplazo, con asignación local. El sistema dispone de 2 marcos reservados para el núcleo del sistema (*kernel*) y 6 para los procesos de usuario y dedica 13 bits al desplazamiento. Las direcciones virtuales que llegan a la MMU tienen 18 bits.

En un determinado instante están ejecutando dos procesos A y B, que han leído y usado sus páginas 1 y 2, en ese mismo orden. El sistema operativo ha asignado 2 marcos al proceso A y 4 al proceso B. La fotografía de la memoria física en ese instante es la siguiente:

0	A-2
1	B-2
2	vacía
3	B-1
4	A-1
5	vacía
6	kernel
7	kernel

A partir de ese momento:

- el proceso A: lee de su pagina 6, escribe en su pagina 2, lee pagina 1
- el proceso B: lee la página 1, escribe su página la 3 y en la 0, lee de la 4, 2, 5, 2 y 1.

Se pide:

a) ¿Cuantos bits ocupa una dirección que sale de la MMU? ¿Que tamaño tiene la memoria virtual? ¿Cuantas entradas tiene cada tabla de páginas?
b) Describe el estado de las tablas de páginas de ambos procesos en el momento fotografiado y su evolución temporal a partir de entonces, indicando los fallos de página y los accesos a disco realizados, así como los bits de presente/ausente, referencia y modificación.

Solución:

a) Las direcciones que salen de la MMU son direcciones físicas. Hay 8 marcos físicos, que se direccionan con 3 bits. Si a eso unimos los 13 bits de desplazamiento, las direcciones físicas ocupan 16 bits.

Si el sistema dedica 13 bits al desplazamiento es que las paginas y los marcos son de 8KB. Las direcciones virtuales ocupan 18 bits, de los cuales 13 son de desplazamiento, de manera que hay $2^5 = 32$ paginas virtuales. Como cada pagina ocupa 8KB, la memoria virtual ocupa 32 _ 8kB = 256KB. La tabla de paginas almacena que marco físico corresponde a

cada pagina lógica. Como hay 32 paginas virtuales, la tabla de paginas tiene 32 entradas.

b) Como el sistema utiliza reemplazo local, cada proceso emplea solo los marcos que le han sido asignados y cuando ocurre un fallo de paginas, carga la nueva pagina en un marco que ocupaba otra pagina suya. Se elige la víctima a reemplazar entre las paginas del propio proceso. Por lo tanto se puede describir independientemente la evolución de las tablas de paginas de los procesos A y B.

Para el proceso A, nos centraremos en las 7 primeras entradas de la tabla, las 25 restantes estarán vacías, con el bit de Presente a 0. En el instante de la fotografía la tabla es la de la figura 1(a), con el orden FIFO 1-2.

La lectura de la pagina 6 provoca un fallo de pagina y como no hay marcos físicos libres de este proceso, se invoca al algoritmo de reemplazo. El algoritmo de segunda oportunidad elige como víctima la 1, habiendo limpiado el bit de referencia de ambas paginas para darles su segunda oportunidad. Como la pagina 1 solo había sido leída, este fallo de pagina solo implica traer de disco la pagina 6. La tabla queda en el estado 1(b) y el orden FIFO 2-6.

La escritura en la pagina 2 no provoca fallo de pagina, solo el cambio de los bits de referencia y modificación de su entrada en la tabla de paginas. La tabla queda en el estado 1(c) y el orden FIFO 2-6.

La lectura de la pagina 1 provoca un fallo de pagina y el algoritmo de segunda

oportunidad elige como víctima la pagina 2. Es justo a la que le toca siguiendo

12 el orden FIFO, tanto la 2 como la 6 tenían su bit de referencia puesto, así que le acaba tocando a la pagina 2 una vez que el algoritmo da una segunda oportunidad a ambas. Este fallo de página desemboca en dos accesos a disco: uno para llevar a disco la pagina 2 modificada y otro para traerse la pagina 1. La tabla queda en el estado 1(d) y el orden FIFO 6-1.

	#marco	P	R	M
0	X	0	0	0
1	4	1	1	0
2	0	1	1	0
3	X	0	0	0
4	X	0	0	0
5	X	0	0	0
6	X	0	0	0
...	...			
31	X	0	0	0

(a)

	#marco	P	R	M
0	X	0	0	0
1	X	0	0	0
2	0	1	0	0
3	X	0	0	0
4	X	0	0	0
5	X	0	0	0
6	4	1	1	0
...	...			
31	X	0	0	0

(b)

	#marco	P	R	M
0	X	0	0	0
1	X	0	0	0
2	0	1	1	1
3	X	0	0	0
4	X	0	0	0
5	X	0	0	0
6	4	1	1	0
...	...			
31	X	0	0	0

(c)

	#marco	P	R	M
0	X	0	0	0
1	0	1	1	0
2	X	0	0	0
3	X	0	0	0
4	X	0	0	0
5	X	0	0	0
6	4	1	0	0
...	...			
31	X	0	0	0

(d)

En cuanto al proceso B, la situación de su tabla de paginas en el instante de la

fotografía es 2(a), y el orden FIFO: 1-2. La siguiente lectura de la pagina 1 no

provoca fallo de pagina, si acaso un nuevo refresco del bit de referencia de esa

pagina, que ya estaba a 1. La escritura de la pagina 3 provoca un fallo de pagina, pero no se invoca al algoritmo de reemplazo, pues aun hay marcos físicos libres asignados a este proceso. Por ejemplo se le asigna el marco 2. Este fallo de pagina desemboca en un acceso a disco para traerse la pagina 3. Como es una escritura el bit de modificación se pone a 1.

La tabla queda en el estado 2(b) y el orden FIFO: 1-2-3.

La escritura de la pagina 0 provoca un fallo de pagina, pero no se invoca al algoritmo de reemplazo, pues aun hay marcos físicos libres asignados a este proceso. Se le asigna el marco 5. Este fallo de pagina desemboca en un acceso a disco para traerse la pagina 0. Como es una escritura el bit de modificación se pone a 1. La tabla queda en el estado 2(c) y el orden FIFO: 1-2-3-0.

La lectura de la pagina 4 provoca un fallo de pagina, y como no hay marcos físicos libres de este proceso, se invoca al algoritmo de reemplazo. El algoritmo de segunda oportunidad elige como víctima la 1, habiendo limpiado el bit de referencia de las páginas 2-3-0 para darles su segunda oportunidad. Como la víctima no había sido modificada, no hay que hacer swap-out, el fallo desemboca en un único acceso a disco para traerse la pagina 4. La tabla queda en el estado 2(d) y el orden FIFO: 2-3-0-4.

La lectura de la pagina 2 no provoca fallo de pagina, solo el cambio del bit de referencia de su entrada en la tabla de paginas. La tabla queda en el estado 2(e) y el orden FIFO 2-3-0-4.

La lectura de la pagina 5 provoca un fallo de pagina, y como no hay marcos físicos libres de este proceso, se invoca al algoritmo de reemplazo. El algoritmo de segunda oportunidad elige como víctima la 3. Siguiendo el orden FIFO le tocaba a la página 2, pero aprovecha su segunda oportunidad y como la pagina 3 tiene su bit de referencia a 0, resulta elegida. Como la página 3 había sido modificada este fallo de página desemboca en 2 operaciones de disco: swap-out de la 3 y swap-in de la 5.

La tabla queda en el estado 2(f) y el orden FIFO 2-0-4-5.

La siguiente lectura de la pagina 2 no provoca fallo de pagina, solo el cambio del

bit de referencia de su entrada en la tabla de paginas. La tabla queda en el estado 2(g) y el orden FIFO 2-0-4-5.

Finalmente, la lectura de la pagina 1 provoca un nuevo fallo de pagina. El algoritmo de segunda oportunidad elige como víctima a la pagina 0, a la cual hay que mandar a disco pues tiene su bit de modificación a 1. Por ello hay dos accesos a disco, el anterior y el necesario para traer la pagina 1 a memoria. La tabla queda en el estado 2(h) y el orden FIFO 2-4-5-1.

	#marco	P	R	M
0	X	0	0	0
1	3	1	1	0
2	1	1	1	0
3	X	0	0	0
4	X	0	0	0
5	X	0	0	0
6	X	0	0	0
...	...			
31	X	0	0	0

(a)

	#marco	P	R	M
0	X	0	0	0
1	3	1	1	0
2	1	1	1	0
3	2	1	1	1
4	X	0	0	0
5	X	0	0	0
6	X	0	0	0
...	...			
31	X	0	0	0

(b)

	#marco	P	R	M
0	5	1	1	1
1	3	1	1	0
2	1	1	1	0
3	2	1	1	1
4	X	0	0	0
5	X	0	0	0
6	X	0	0	0
...	...			
31	X	0	0	0

(c)

	#marco	P	R	M
0	5	1	0	1
1	X	0	0	0
2	1	1	0	0
3	2	1	0	1
4	3	1	1	0
5	X	0	0	0
6	X	0	0	0
...	...			
31	X	0	0	0

(d)

	#marco	P	R	M
0	5	1	0	1
1	X	0	0	0
2	1	1	1	0
3	2	1	0	1
4	3	1	1	0
5	X	0	0	0
6	X	0	0	0
...	...			
31	X	0	0	0

(e)

	#marco	P	R	M
0	5	1	0	1
1	X	0	0	0
2	1	1	0	0
3	X	0	0	0
4	3	1	1	0
5	2	1	1	0
6	X	0	0	0
...	...			
31	X	0	0	0

(f)

	#marco	P	R	M
0	5	1	0	1
1	X	0	0	0
2	1	1	1	0
3	X	0	0	0
4	3	1	1	0
5	2	1	1	0
6	X	0	0	0
...	...			
31	X	0	0	0

(g)

	#marco	P	R	M
0	X	0	0	0
1	5	1	1	0
2	1	1	0	0
3	X	0	0	0
4	3	1	1	0
5	2	1	1	0
6	X	0	0	0
...	...			
31	X	0	0	0

(h)

15

Problema 3

En un puente sobre un río entre dos ciudades existe un grupo de piqueteros que bloquean periódicamente el tránsito por la cabecera oeste.

Los automóviles que intentan cruzar el puente desde el lado no bloqueado tendrán que quedarse esperando sobre el puente hasta que se libere el bloqueo (el puente es doble vía por lo que solamente ocupan la mitad de la calzada). Los automóviles que intentan cruzar del lado bloqueado esperan fuera del puente pues no pueden ingresar al mismo.

Cuando hay 50 autos bloqueados sobre el puente es necesario liberar el mismo pues se pone en riesgo su estructura por exceso de peso. En este caso los automóviles bloqueados deberán dejar el puente en el orden inverso al que llegaron (dado que deben hacerlo marcha atrás) y deberán cruzar el río por otro puente. No podrán entrar autos nuevos al puente mientras están saliendo los anteriores.

Por otra parte, el piquete se disuelve en el momento en que desea pasar el señor Q. En este momento podrán pasar autos en ambos sentidos hasta que el señor Q termine de cruzar el puente.

Además, para que el bloqueo pueda realizarse debe haber más de 10 piqueteros. Mientras no se llegue a este número los autos podrán circular libremente y los piqueteros esperarán hasta que lleguen más. Al superar las 10 personas se establece el bloqueo el cual continuará mientras haya al menos 10 piqueteros. Los piqueteros se mantienen realizando el bloqueo un tiempo aleatorio y luego se van a su casa a descansar.

Se dispone de los siguientes procedimientos y funciones:

- **Cruzar(AvanzaORetrocede :INTEGER)** que invocada por un auto o el señor Q, lo cruza o sale del puente marcha atrás. Si se invoca con el valor 0 cruza el puente y con el valor 1 sale marcha atrás.
- **Random(OUT :INTEGER)** que devuelve un numero aleatorio.

Se pide:

- Implementar en ADA las tareas Piquetero, SenorQ, AutoHaciaEste, AutoHaciaOeste.

Notas:

- Se puede implementar hasta una tarea auxiliar.
- La cantidad de autos del sistema no es conocida ni acotada.

Solución:

```
TASK TYPE AutoHaciaEste
```

```
END TASK;

TASK BODY AutoHaciaEste
  BEGIN
  Puente.LlegaHaciaEste();
  Cruzar(0);
END TASK


TASK TYPE AutoHaciaOeste
END TASK;

TASK BODY AutoHaciaOeste
  INTEGER Posicion;
  INTEGER QueHago;
BEGIN
  Puente.LlegaHaciaOeste(Posicion);
  IF Posicion = 0
    -- Si no hay piquete
    Cruzar(0);
    Puente.PasoAutoOeste;
  ELSE
    -- Si hay piquete
    Puente.ParaDondeVoy[Posicion](QueHago);
    Cruzar(QueHago);
    IF QueHago = 0
      Puente.PasoAutoOeste;
    ELSE
      Puente.MeFui;
    ENDIF
  ENDIF
END TASK;


TASK TYPE Piquetero
END TASK;
```

```
TASK BODY Piquetero
  INTEGER TiempoEspera;
BEGIN
  Puente.LlegaPiquetero;
  Random(TiempoEspera);
  DELAY TiempoEspera;
  Puente.SeVaPiquetero;
END TASK;

TASK SenorQ
END TASK;

TASK BODY SenorQ
BEGIN
  Puente.LlegaSenorQ;
  Cruzar(0);
  Puente.SeVaSenorQ;
END TASK;

TASK Puente
  ENTRY LlegaHaciaEste;
  ENTRY LlegaHaciaOeste(Posicion :OUT INTEGER);
  ENTRY ParaDondeVoy[50](QueHago :OUT INTEGER);
  ENTRY MeFui;
  ENTRY PasoAutoOeste;
  ENTRY LlegaSenorQ;
  ENTRY SeVaSenorQ;
  ENTRY LlegaPiquetero;
  ENTRY SeVaPiquetero;
END TASK;

TASK BODY Puente
  INTEGER Piqueteros, AutosDentro, Posicion;
  BOOLEAN SenorQ;
BEGIN
  Piqueteros = 0;
  AutosDentro = 0;
```

```
  Posicion = 1;
  SenorQ = FALSE;
  LOOP
    -- No hay piquete
    SELECT
        ACCEPT LlegaHaciaEste;
    OR  ACCEPT LlegaHaciaOeste(Pos)
          Pos = 0;
        END;
        AutosDentro++;
    OR  ACCEPT PasoAutoOeste;
        AutosDentro--;
    OR  ACCEPT LlegaSenorQ;
        SenorQ = TRUE;
    OR  ACCEPT SeVaSenorQ;
        SenorQ = FALSE;
    OR  WHEN LlegaSenorQ'Count = 0 ==>
        ACCEPT LlegaPiquetero;
        Piqueteros++:
    OR  ACCEPT SeVaPiquetero;
        Piqueteros--:
    END SELECT;
  ENDLOOP

  -- Veo si se puede formar piquete
  IF Piqueteros > 10 AND NOT SenorQ
    WHILE AutosDentro > 0 AND NOT SenorQ LOOP
      -- Espero a que se vayan todos los autos que ya estaba cruzando
      SELECT
        ACCEPT PasoAutoOeste;
        AutosDentro--;
      OR ACCEPT LlegaSenorQ;
        SenorQ = TRUE;
      END SELECT;
    ENDLOOP;
    IF NOT SenorQ
      -- Hay Piquete
```

```
      WHILE Piqueteros > 10 AND NOT SenorQ LOOP
        SELECT
          ACCEPT LlegaHaciaOeste(Pos)
            Pos = Posicion
          END;
          Posicion++;
          IF Posicion = 51
            -- Si debo mandarlos marcha atras
            WHILE Posicion > 1 LOOP
              Posicion--;
              ACCEPT ParaDondeVoy[Posicion](QueHago)
                QueHago = 1; -- Voy para atras
              END;
              ACCEPT MeFui; -- Espero a que salga del puente
            ENDLOOP;
          ENDIF
        OR ACCEPT LlegaSenorQ;
          SenorQ = TRUE;
        OR ACCEPT LlegaPiquetero;
          Piqueteros++;
        OR ACCEPT SeVaPiquetero;
          Piqueteros--;
      ENDLOOP;

      -- Se acabo el piquete
      FOR Pos = 1 TO Posicion > 1 LOOP
        ACCEPT ParaDondeVoy[Pos](QueHago)
          QueHago = 0; -- Voy para adelante
        END;
        AutosDentro++;
      ENDLOOP;
      Posicion = 1;
    ENDIF
  ENDIF
END TASK;
```