Examen Diciembre de 2010

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 y cada parte del ejercicio 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 (35 puntos)

Responda las siguientes preguntas justificando cada respuesta:

1. Al implementar un algoritmo de planificación (*scheduling*) round-robin, ¿qué problema se puede presentar con el tamaño del quantum?

2. Sea un sistema operativo con un algoritmo de planificación expropiativo (*preemptive*) round-robin . Un proceso en ejecución se le expropia el procesador solo si su quantum expira o si termina su ejecución. Suponiendo que el quantum es 2 y se disponde del siguiente esquema de procesos:

Proceso	Tiempo de creación	Tiempo de cómputo
1	0	4
2	1	6
3	3	4

   ¿Cuál es el tiempo de retorno (*turnaround time*)?

   ¿Cuál es el tiempo de retorno promedio?

3. Explique el método *working-set* utilizado para evitar la hiperpaginación (*thrashing*).

4. Explique cómo funciona el método de asignación indexada para los sistemas de archivos.

5. Describa dos de los principales servicios del subsistema de Entrada/Salida.

Parte práctica (30 puntos)

Sea un sistema operativo que gestiona su memoria con memoria virtual implementada como paginación bajo demanda. Se utiliza un sistema de traducción de direcciones de un solo nivel. Las páginas son de 256 Bytes de tamaño y las direcciones virtuales de 24 bits de largo. A su vez, el sistema operativo solo soportará hardware con hasta 4 KBytes de memoria física.

El sistema utiliza un algoritmo de reemplazo LRU (*Least Recently Used*) con una estrategia de asignación local.

Se pide:

a. ¿Cuántas entradas tiene la tabla de páginas?

b. ¿Cuántos Bytes son utilizados en cada entrada para referenciar el marco?

c. Si a un proceso se le asignaron 4 marcos, donde ya fueron cargadas las páginas 0, 1, 3 y 7 (en ese orden) de ese proceso y se tiene la siguiente secuencia de accesos a páginas:

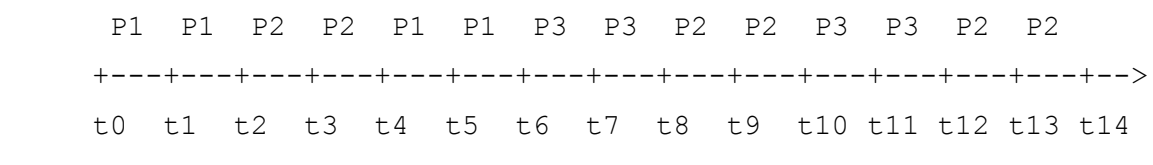
3 1 8 8 7 0 5 2 3 1 5 7 1 3 2

   Determine cuántos fallos de página se tendrán y que páginas quedan cargadas en memoria al final.

d. Debido a que el sistema cuenta con un gran número de procesos que utilizan pocas páginas, proponga un sistema de traducción que disminuya la cantidad de memoria utilizada por las tablas de traducción.

Solución Problema 1 Parte Teórica

Asumiendo que la cola de espera es FIFO:



El tiempo de retorno para cada proceso es el siguiente:

Proceso	**Inicio**	**Fin**	**Tiempo de retorno**
P1	0	6	6
P2	1	14	13
P3	3	12	9

Por lo que el tiempo de retorno promedio es: (6+13+9)/3 = 28/3

Solución Problema 1 Parte Práctica

a. Dado que el tamaño de página es de 256 Bytes, necesitamos 8 bits para poder referenciar cada byte de la página (desplazamiento). Por lo que con direcciones virtuales de 24 bits nos quedan 16 bits para referenciar la página. Al ser una tabla de páginas de un solo nivel tenemos entonces 2^16 entradas posibles.

b. El tamaño de los marcos es de 256 Bytes (igual que el tamaño de las páginas) y tenemos como máximo 4KBytes de memoria física. Por lo tanto, se tendrán como máximo 16 marcos (4KB/256B). Para referenciar 16 marcos se precisan solamente 4 bits o medio byte.

c)

Dado que las páginas son cargadas en el orden 0, 1, 3, 7 esto implica que para el algoritmo de remplazo LRU, la pagina menos recientemente utilizada es la 0 luego la página 1, posteriormente las 3, y la más recientemente utilizada es la 7, entonces el orden queda 7, 3, 1, 0

	7 3 1 0
3	3 7 1 0
1	1 3 7 0
8	8 1 3 7 F
8	8 1 3 7
7	7 8 1 3
0	0 7 8 1 F
5	5 0 7 8 F

```
2     2 5 0 7 F
3     3 2 5 0 F
1     1 3 2 5 F
5     5 1 3 2
7     7 5 1 3 F
1     1 7 5 3
3     3 1 7 5
2     2 3 1 7 F
```

La cantidad de fallos de página son 8.

Las páginas que quedan cargadas son 2, 3, 1, 7.

d.

Utilizar un sistema de paginación multinivel pudiendo así tener en memoria física tablas de paginado más pequeñas y solamente las necesarias.

Problema 2 (35 puntos)

Se desea modelar en ADA una sección de una fábrica de motores en la que trabajan seis trabajadores los cuales deben apretar tuercas. Para esto se dispone de seis llaves de tuerca ubicadas en la mesa de herramientas. El trabajador primero aprieta una tuerca y en caso que no quede bien ajustada deberá utilizar una segunda llave en conjunto con la primera para terminar el trabajo.

Luego de apretar la tuerca el trabajador debe dejar las herramientas que utilizó en su lugar y dirigirse al medidor para comprobar que la tuerca fue ajustada con la presión correcta. En caso en que esté correcta deberá presionar un botón para se le asigne otro motor mientras que en otro caso deberá volver a ajustarla.

En la fábrica hay un supervisor de herramientas que solo podrá verificar las herramientas todas a la vez. Cuando el supervisor está esperando para supervisar los trabajadores no podrán comenzar a trabajar en un motor nuevo.

Se dispone de los siguientes procedimientos:

- **presionarBoton()** para que se le asigne un nuevo motor.
- **apretarTuerca(llave):boolean** que aprieta la tuerca con una llave y retorna true si la tuerca quedo bien ajustada y false en caso contrario.
- **apretarTuercaFuerte(llave, llave)** que aprieta la tuerca con dos llaves.
- **verificarAjuste(): boolean** que devuelve falso en caso que la presión no sea la correcta.
- **verificarLlaves()** ejecutada por el supervisor para verificar todas las llaves.
- **otrasTareas()** ejecutada por el supervisor cuando no quiere verificar llaves.

Solucion:

```
procedure Fabrica is

      task Mesa is
            entry comenzarTrabajo;
            entry tomarLlave ( llave: out Llave );
            entry dejarLlave ( llave: in Llave, primera: in Boolean );
            entry comenzarSupervision;
            entry finSupervision;
      end Mesa

      task Supervisor;

      task type Trabajador;

      task body Mesa is
            cantTrabajadores : Integer;
            cantLlaves : Integer;
            llavesEnMesa: array (1..6) of Llave; // inicializo 6 llaves
            aux: Integer;
            auxLlave : Llave;
      begin
            cantTrabajadores := 0;
            loop
                  select
                        when comenzarSupervision'count = 0 =>
                        accept comenzarTrabajo;
                  or
                        when cantTrabajadores < 5 AND cantLlaves > 0 =>
                        accept tomarPrimeraLlave (llave: out Llave) do
                              aux := 0;
                              loop
                                    aux := aux + 1;
                                    llave := llavesEnMesa(aux);
                              until llave <> NIL;
                        end;
                        llavesEnMesa(aux) := NIL;
                        cantTrabajadores := cantTrabajadores + 1;
                        cantLlaves := cantLlaves - 1;
                  or
                        when cantLlaves > 0 =>
                        accept tomarSegundaLlave (llave: out Llave) do
                              aux := 0;
                              loop
                                    aux := aux + 1;
                                    llave := llavesEnMesa(aux);
                              until llave <> NIL;
                        end;
                        llavesEnMesa(aux) := NIL;
                        cantLlaves := cantLlaves - 1;
                  or
                        accept dejarLlave (llave: in Llave, primera: boolean) do
                              auxLlave := llave;
                        end;
                        if(primera)
                              cantTrabajadores := cantTrabajadores - 1;
                        end;
                        aux := 0;
```

```
                        while(llavesEnMesa(aux) <> NIL)
                              aux := aux + 1;
                        end;
                        llavesEnMesa(aux) := auxLlave;
                        cantLavesEnMesa := cantLavesEnMesa + 1;
                  or
                        when cantLlavesEnMesa = 6 =>
                        accept comenzarSupervision;
                        accept finSupervision;
                  end select;
            end loop;
      end Mesa;

      task body Supervisor is
      begin
            loop
                  otrasTareas();
                  Mesa.comenzarSupervision;
                  verificarLlaves();
                  Mesa.finSupervision;
            end loop;
      end Supervisor;

      task body Trabajador is
            llave1, llave2: Llave;
      begin
            presionarBoton();
            Mesa.comenzarTrabajo;
            loop
                  Mesa.tomarPrimeraLlave(llave1);
                  if( not (apretarTuerca(llave1)) )
                        Mesa.tomarSegundaLlave(lleve2);
                        apretarTuercaFuerte(llave1, llave2);
                        Mesa.dejarLlave(llave2, false);
                  end if;
                  Mesa.dejarLlave(llave1, true);
                  if( verificarAjuste() )
                        presionarBoton();
                        Mesa.comenzarTrabajo;
                  end if;
            end loop;
      end Trabajador;

      trabajadores: array (1..6) of Trabajador;
```