Examen 28 de febrero de 2015

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.**
- **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 4 horas.**
- **Al momento de finalizar el examen no se podrá escribir absolutamente nada en las hojas, debiéndose parar e ir a la fila de entrega. Identificar cada una de las hojas con nombre, cédula y numeración forma parte de la duración del examen.**

Problema 1 (32 puntos)

1 - Describa y compare los métodos de asignación de datos de archivos contiguo e indexado.

2 - Indique qué es buffering y cuáles son las razones para utilizarlo en el subsistema de E/S.

3 -

a - Indique cómo es el funcionamiento de un RAID 1 y un RAID 3. (RAID = Redundant Array of Inexpensive Disks).

b - ¿Cuántos accesos a disco deberá realizar una operación de escritura de un bit en cada uno de los casos?

4 -

a – Describa tres estrategias de asignación de memoria para un proceso.

b - Describa los registros que componen un puerto de entrada/salida e indique para qué son utilizados estos registros.

5 -

a - Defina qué es una *system call*

b - Indique y explique cada una de las tareas que se realizan en una llamada a una *system call*.

6 - Indique las ventajas de un sistema con micronúcleo.

7 - Indique los pasos por los que pasa un programa para ser ejecutado.

8 - Realice el diagrama de estados y transiciones de los procesos. Explique cada uno de sus elementos.

Problema 2 (35 puntos)

Sea un sistema operativo con un planificador round-robin con quatum de 3 unidades de ejecución en un sistema computacional monoprocesador. El sistema utiliza memoria virtual con direcciones virtuales de 16 bits. Se usa un sistema de paginación bajo demanda de dos niveles, con páginas de 256 bytes, un algoritmo de asignación global de 8 marcos y un algoritmo de reemplazo LRU (*Least Recently Used*).

Los procesos de este sistema pueden ejecutar 3 tipos de instrucciones: accesos a memoria (M), operaciones que bloquean el proceso (B) y operaciones que manipulan registros (Op). Cada operación M y Op ocupa una unidad de ejecución. Las operaciones de memoria van seguidas de la dirección virtual de memoria accedida, donde los últimos bits de la dirección corresponden al desplazamiento. Las operaciones que bloquean el proceso van seguidas de la cantidad de unidades de ejecución que el proceso esperará por su completitud (p.ej. B(5) – el proceso que invoca la operación debe esperar por 5 unidades de ejecución para que se complete la operación).

Sean 4 procesos P1, P2, P3 y P4 con la siguiente secuencia de instrucciones:

P1	M(0xF012)	M(0xCCA4)	B(3)	M(0xCC1F)	M(0xF0A4)	M(0x9C12)	-	-
P2	M(0xFD01)	Op	Op	Op	B(2)	Op	M(0xFDD1)	-
P3	M(0xFD55)	M(0xBA1E)	B(2)	M(0xBA31)	M(0x1C44)	M(0x9C10)	M(0x2113)	M(0x1C16)
P4	M(0x1CD1)	M(0x0A00)	Op	M(0x21A1)	M(0x1CC1)	B(2)	-	-

Notas:

1. P1, P2, P3, P4 comienzan su ejecución en los instantes de tiempo t=0, t=1, t=2 y t=3 respectivamente.
2. Para ingresar a la ready queue los procesos que se les acabo el quantum tienen prioridad sobre los procesos que se encontraban bloqueados
3. Los procesos comienzan sin ninguna página cargada en memoria.
4. Asuma que el procesamiento de los cambios de contexto y fallos de página no consumen tiempo de procesador.
5. Las direcciones de memoria que aparecen asociadas a las operaciones M, son direcciones virtuales.

Se pide:

1. Realice un esquema que muestre el uso del recurso procesador así como la cola de procesos listos (*ready queue*) en cada instante del tiempo (13 puntos).

2. Calcule el tiempo de espera de P1 y el tiempo de retorno de P4 (4 puntos).

3. Sabiendo que la tabla de páginas de primer y segundo nivel tienen la misma cantidad de entradas. Cuantos bits son necesarios para direccionar cada nivel de las tablas de páginas? (2 puntos)

4. Realice un esquema que muestre el estado de la memoria al producirse cada acceso (12 puntos).

5. Determine cuántos fallos de página se producen en total (4 puntos).

Solución: 1)

Tiempo	Proceso	Operación	Ready queue	Bloqueados
0	P1	M(0xF012)	-	-
1	P1	M(0xCCA4)	P2	-
2	P1	B(3)	P2,P3	
3	P2	M(0xFD01)	P3,P4	P1
4	P2	Op	P3,P4	P1
5	P2	Op	P3,P4	P1
6	P3	M(0xFD55)	P4,P2,P1	-
7	P3	M(0xBA1E)	P4,P2,P1	-
8	P3	B(2)	P4,P2,P1	-
9	P4	M(0x1CD1)	P2,P1	P3
10	P4	M(0x0A00)	P2,P1	P3
11	P4	Op	P2,P1,P3	-
12	P2	Op	P1,P3,P4	-
13	P2	B(2)	P1,P3,P4	-
14	P1	M(0xCC1F)	P3,P4	P2
15	P1	M(0xF0A4)	P3,P4	P2
16	P1	M(0x9C12)	P3,P4,P2	-
17	P3	M(0xBA31)	P4,P2	-
18	P3	M(0x1C44)	P4,P2	-
19	P3	M(0x9C10)	P4,P2	-
20	P4	M(0x21A1)	P2,P3	-
21	P4	M(0x1CC1)	P2,P3	-
22	P4	B(2)	P2,P3	
23	P2	Op	P3	P4
24	P2	M(0xFDD1)	P3	P4
25	P3	M(0x2113)	-	-
26	P3	M(0x1C16)	-	-

2) P1 espera desde t=6 hasta t=13, por lo tanto el tiempo de espera de P1, suma de los intervalos de tiempo que el proceso estuvo en la cola de procesos listos, es 8. P4 termina en t=25 y empieza en t=3, por lo tanto el tiempo de retorno de P4 (intervalo de tiempo desde que el proceso es cargado hasta que este finaliza su ejecución) es 22.

3) Se utilizan 8 bits en el desplazamiento para direccionar los 256 bytes de cada página. Como las direcciones virtuales del sistema son de 16 bits, eso nos deja 4 bits para el direccionamiento de la tabla de primer nivel y 4 bits para la tabla de segundo nivel.

4)

Tiempo	**Proceso**	**Acceso**	**LRU**
0	P1	0xF012*	P1(F0) [1,F0][-,--],[-,--],[-,--],[-,--],[-,--],[-,--],[-,--]
1	P1	0xCCA4*	P1(CC),P1(F0) [1,F0][1,CC],[-,--],[-,--],[-,--],[-,--],[-,--],[-,--]
3	P2	0xFD01*	P2(FD),P1(CC),P1(F0) [1,F0][1,CC],[2,FD],[-,--],[-,--],[-,--],[-,--],[-,--]
6	P3	0xFD55*	P3(FD),P2(FD),P1(CC),P1(F0) [1,F0][1,CC],[2,FD],[3,FD],[-,--],[-,--],[-,--],[-,--]
7	P3	0xBA1E*	P3(BA),P3(FD),P2(FD),P1(CC),P1(F0) [1,F0][1,CC],[2,FD],[3,FD],[3,BA],[-,--],[-,--],[-,--]
9	P4	0x1CD1*	P4(1C),P3(BA),P3(FD),P2(FD),P1(CC),P1(F0) [1,F0][1,CC],[2,FD],[3,FD],[3,BA],[4,1C],[-,--],[-,--]
10	P4	0x0A00*	P4(0A),P4(1C),P3(BA),P3(FD),P2(FD),P1(CC),P1(F0) [1,F0][1,CC],[2,FD],[3,FD],[3,BA],[4,1C],[4,0A],[-,--]
14	P1	0xCC1F	P1(CC),P4(0A),P4(1C),P3(BA),P3(FD),P2(FD),P1(F0)
15	P1	0xF0A4	P1(F0),P1(CC),P4(0A),P4(1C),P3(BA),P3(FD),P2(FD)
16	P1	0x9C12*	P1(9C),P1(F0),P1(CC),P4(0A),P4(1C),P3(BA),P3(FD),P2(FD) [1,F0][1,CC],[2,FD],[3,FD],[3,BA],[4,1C],[4,0A],[1,9C]
17	P3	0xBA31	**Finaliza P1** P3(BA),P4(0A),P4(1C),P3(FD),P2(FD) [-,--][-,--],[2,FD],[3,FD],[3,BA],[4,1C],[4,0A],[-,--]
18	P3	0x1C44*	P3(1C),P3(BA),P4(0A),P4(1C),P3(FD),P2(FD) [3,1C][-,--],[2,FD],[3,FD],[3,BA],[4,1C],[4,0A],[-,--]
19	P3	0x9C10*	P3(9C),P3(1C),P3(BA),P4(0A),P4(1C),P3(FD),P2(FD) [3,1C][3,9C],[2,FD],[3,FD],[3,BA],[4,1C],[4,0A],[-,--]
20	P4	0x21A1*	P4(21),P3(9C),P3(1C),P3(BA),P4(0A),P4(1C),P3(FD),P2(FD) [3,1C][3,9C],[2,FD],[3,FD],[3,BA],[4,1C],[4,0A],[4,21]
21	P4	0x1CC1	P4(1C),P4(21),P3(9C),P3(1C),P3(BA),P4(0A),P3(FD),P2(FD)
24	P2	0xFDD1	P2(FD),P4(1C),P4(21),P3(9C),P3(1C),P3(BA),P4(0A),P3(FD)
25	P3	0x2113*	**Finaliza P2 y P4** P3(21),P3(9C),P3(1C),P3(BA),P3(FD) [3,1C][3,9C],[3,21],[3,FD],[3,BA],[-,--],[-,--],[-,--]
26	P3	0x1C16	P3(1C),P3(21),P3(9C),P3(BA),P3(FD)

5) En total se producen 12 fallos de página (marcados con * en la tabla anterior).

Problema 3 (33 puntos)

Se desea modelar un juego de azar grupal que consiste en adivinar la ciudad más cercana a la elegida por el servidor. El servidor recibirá las apuestas de los jugadores y cada cinco apuestas determina el o los ganadores y el monto a pagar (que depende de la distancia). A cada jugador se le deberá informar el monto ganado (que puede ser 0) y luego el jugador deberá registrar la ganancia de la jugada.

Se desea modelar usando mailboxes los procesos jugador y servidor. La cantidad de jugadores no está acotada. Se debe especificar la semántica de los mailboxes utilizados.

Se dispone de los siguientes procedimientos auxiliares:

- `pensarCiudad(): Ciudad`
  Ejecutada tanto por el servidor como por el jugador.

- `evaluarApuestas(c1: Ciudad, c2: Ciudad, c3: Ciudad, c4: Ciudad, c5: Ciudad, objetivo: Ciudad): array [1..5] of integer`
  Ejecutada por el servidor para obtener los montos a pagar. En este caso c_i es la ciudad pensada por el jugador i y objetivo es la ciudad pensada por el servidor.

- `registrarGanancia(valor: integer): void`
  Ejecutada por el jugador para registrar la ganancia de la jugada.

Solución:

```
Se utilizan Mailbox de tipo FIFO, infinitos, con send no bloqueante y receive
bloqueante.

Mailbox<integer> posicion;
Mailbox<Ciudad>[5] apuesta;
Mailbox<integer>[5] resultado;

void servidor() {
      while (true) {
            Ciudad objetivo = pensarCiudad();

            integer ronda_apuestas[5];
            for (int i=0; i<5; i++) {
                  ronda_apuestas[i] = apuesta[i].receive()
            }

            integer ronda_resultados[5];
            ronda_resultados = evaluarApuestas(
                  ronda_apuestas[0], ronda_apuestas[1],
                  ronda_apuestas[2], ronda_apuestas[3],
                  ronda_apuestas[4], objetivo);

            for (int i=0; i<5; i++) {
                  resultado[i].send(ronda_resultados[i]);
            }
      }
}
```

```
void jugador() {
      int ganancia  = 0;

      while (true) {
            Ciudad c = pensarCiudad();
            int pos = posicion.receive();

            apuesta[pos].send(c);
            ganancia = resultado[pos].receive();
            posicion.send(pos);
            registrarGanancia(ganancia);
      }
}

void main() {
      for (int i=0; i<5; i++) {
            posicion.send(i);
      }

      cobegin
            servidor
            jugador
            ...
            jugador
            ...
      coend
}
```