Examen Febrero de 2012

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 y cada parte del problema 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 4 horas.**

Problema 1 (35 puntos)

1. Qué es un sistema multiprogramado y cuál es la principal ventaja de desarrollar un sistema de este tipo?
2. Qué ventaja tiene un sistema simétrico frente a uno asimétrico?
3. Qué es el registro “program counter” de un proceso? Y de un hilo (*thread*)?
4. Sea un sistema con paginación, realice un diagrama de una traducción de una dirección virtual de 32bits con 2 niveles de tabla de página. Tome en cuenta que la tabla de página de primer nivel tiene 2048 entradas y las tablas de página de segundo nivel tienen 1024 entradas.
5. Cuales son las ventajas de segmentación frente a paginación.
6. Sea un sistema de archivos con inodos (tipo Unix). Describa como se obtiene el número de inodo de `archivo.txt`, siendo su path el siguiente: `/a/b/archivo.txt`.
7. Hasta cuantos discos se pueden romper en un RAID 5 (*Redundan Array of Inexpensive Disk*) sin que el sistema pierda la indesponibilidad? Se tiene degradación en esa situación?

Problema 2 (30 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 un sistema de paginación por demanda con asignación de marcos local de 3 marcos por proceso y algoritmo de reemplazo LRU (*Least Recently Used*). Los procesos de este sistema pueden ejecutar 3 tipos de instrucciones: accesos a memoria (M), acceso a Entrada/Salida (I) y operaciones con registros (Op). Cada operación M, I o Op ocupa una unidad de ejecucipon. Las operaciones de memoria van seguidas del número de página (p.ej. M3 – acceso a la página 3). Las operaciones de E/S van seguidas de la cantidad de unidades de ejecución que el proceso esperará por la completitud de la operación (I5 – el proceso debe esperar por 5 unidades de ejecución para que se complete el pedido de E/S).

Sean 3 procesos P1, P2 y P3 con la siguiente secuencia de instrucciones:

P1	M7	M6	I5	M4	M7	M8	M9	-	-	-
P2	M3	M4	I6	M7	M3	M8	Op	Op	I7	M3
P3	M1	M2	Op	I2	M3	M5	M5	M6	-	-

Notas:

1. El proceso P1 comienza a ejecutar en el tiempo t0, el proceso P2 en el momento t1 y el proceso P3 en el momento t3.
2. Los procesos que están en la ready queue tiene proridad sobre otros procesos. Los procesos que vienen de una E/S tiene priridad sobre los que se les acabo el quantum.
3. Los procesos comienzan sin ninguna página cargada en memoria.
4. Asuma que los cambios de contexto y fallos de página no consumen tiempo de procesador.

Se pide:

a) Realice un esquema que muestre el uso del recurso procesador y la cola de procesos listos (*ready queue*) en cada instante del tiempo. (10pts).

b) Realice un esquema que muestre el estado de la memoria física y swap, señalando los fallos de página que ocurren en cada instante del tiempo. (10pts).

c) Calcule la utilización del sistema. (4 pts.)

d) Calcule el tiempo de espera promedio. (6 pts.)

---

a)

Tiempo	Procesador	Cola de listos	Procesos bloqueados
T0	P1		
T1	P1	P2	
T2	P2		P1
T3	P2	P3	P1
T4	P3		P1 – P2
T5	P3		P1 – P2
T6	P3		P1 – P2
T7	P1	P3	P2
T8	P1	P3	P2
T9	P1	P3	P2
T10	P2	P1	P3
T11	P2	P1	P3
T12	P2	P1 – P3	
T13	P1	P3 – P2	
T14	P3	P2	
T15	P3	P2	
T16	P3	P2	
T17	P2	P3	
T18	P2	P3	
T19	P3		P2
T20			P2
T21			P2
T22			P2
T23			P2
T24			P2
T25			P2
T26	P2		

b)

tiempo	Proceso1						Proceso 2						Proceso 3						swap	Fallo de pagina
	Marcos			LRU			Marcos			LRU			Marcos			LRU				
	1	2	3				1	2	3				1	2	3					
T0	M7			M7																Si
T1	M7	M6		M6	M7															Si
T2	M7	M6		M6	M7		M3			M3										Si
T3	M7	M6		M6	M7		M3	M4		M4	M3									Si
T4	M7	M6		M6	M7		M3	M4		M4	M3		M1			M1				Si
T5	M7	M6		M6	M7		M3	M4		M4	M3		M1	M2		M2	M1			Si
T6	M7	M6		M6	M7		M3	M4		M4	M3		M1	M2		M2	M1			No
T7	M7	M6	M4	M4	M6	M7	M3	M4		M4	M3		M1	M2		M2	M1			Si
T8	M7	M6	M4	M7	M4	M6	M3	M4		M4	M3		M1	M2		M2	M1			No
T9	M7	M8	M4	M8	M7	M4	M3	M4		M4	M3		M1	M2		M2	M1		M6(P1)	Si
T10	M7	M8	M4	M8	M7	M4	M3	M4	M7	M7	M4	M3	M1	M2		M2	M1		M6(P1)	Si
T11	M7	M8	M4	M8	M7	M4	M3	M4	M7	M3	M7	M4	M1	M2		M2	M1		M6(P1)	No
T12	M7	M8	M4	M8	M7	M4	M3	M8	M7	M8	M3	M7	M1	M2		M2	M1		M6(P1) M4(P2)	Si
T13	M7	M8	M9	M9	M8	M7	M3	M8	M7	M8	M3	M7	M1	M2		M2	M1		M6(P1) M4(P1) M4(P2)	Si
T13	El proceso finalizo y es liberada la memoria						M3	M8	M7	M8	M3	M7	M1	M2	M3	M3	M2	M1	M4(P2)	Si
T14							M3	M8	M7	M8	M3	M7	M5	M2	M3	M5	M3	M2	M4(P2) M1(P3)	Si
T16							M3	M8	M7	M8	M3	M7	M5	M2	M3	M5	M3	M2	M4(P2) M1(P3)	No
T17							M3	M8	M7	M8	M3	M7	M5	M2	M3	M5	M3	M2		No
T18							M3	M8	M7	M8	M3	M7	M5	M2	M3	M5	M3	M2		No
T19							M3	M8	M7	M8	M3	M7	M5	M6	M3	M6	M5	M3	M4(P2) M1(P3) M2(P3)	Si
T20							M3	M8	M7	M8	M3	M7	El proceso finalizo y es liberada la memoria						M4(P2)	No
T21							M3	M8	M7	M8	M3	M7							M4(P2)	No
T22							M3	M8	M7	M8	M3	M7							M4(P2)	No
T23							M3	M8	M7	M8	M3	M7							M4(P2)	No
T24							M3	M8	M7	M8	M3	M7							M4(P2)	No

T25		M3	M8	M7	M8	M3	M7		M4(P2)	No
T26		M3	M8	M7	M3	M8	M7		M4(P2)	No

c)

total de unidades de tiempo: 27
unidades de tiempo de CPU utilizada: 21
unidades de tiempo de CPU ociosa: 6
utilización de CPU 21/27

d)

tiempo de espera de P1: 3 unidades

tiempo de espera de P2: 5 unidades

tiempo de espera de P3: 8 unidades

tiempo de espera promedio: (3+5+8)/3 = 16/3

Problema 3 (35 puntos)

Se desea modelar usando **monitores** una peluquería de damas.

Los servicios que da la peluquería son: corte, brushing y tinta. La peluquería cuenta con 3 sillones para la atención de las clientas y una sala de espera para 5 personas. Se cuenta con dos peluqueras que realizan corte y tinta, y una colorista que puede hacer tinta y brushing.

Las clientas se sientan en los sillones de atención por orden de llegada y deben indicar al llegar a los mismos los servicios que requieren. Los servicios de corte, tinta y brushing deben ser realizados serialmente en este orden. Al finalizar la atención la clienta pagará por los servicios recibidos.

Si hay tiempos de espera entre las diferentes tareas las clientas permanecerán sentadas en el sillón. Si la sala de espera de la peluquería está llena al llegar una nueva clienta esta se retira sin esperar.

Se dispone de los siguientes procedimientos auxiliares:

- queServicios(out corte: boolean, out tinta: boolean, out brushing: boolean)

  Ejecutada por las clientas al llegar a la peluquería para obtener que servicios quiere la clienta (se sabe que cada cliente requiere al menos un servicio).
- pagar()

  Ejecutada por las clientas al terminar de ser atendidas
- cortarPelo(int sillon)

  Ejecutada por la empleada correspondiente indicando el sillón sobre el que va a trabajar
- hacerBrushing(int sillon)

  Ejecutada por la empleada correspondiente indicando el sillón sobre el que va a trabajar
- hacerTinta(int sillon)

  Ejecutada por la empleada correspondiente indicando el sillón sobre el que va a trabajar

Nota:

- Se deben implementar las tareas clienta, peluquera y colorista.
- Las empleadas no deben quedar ociosas si hay clientas esperando por un servicio que ellas brindan.
- No se podrán utilizar tareas auxiliares.

Solución:

La idea es que una vez terminado un objetivo de la clienta, se despierta al siguiente en la cadena. El primero que lo tome se queda con el objetivo y el resto vuelve a dormir.

```
#define corte = 0;
#define tinta = 1;
#define brushing = 2;

Monitor Organizador {
  Condition cndEnSalaEspera;
  Condition cndPeluquera, cndColorista;
  Condition cndFinAtencion[3];
  int enSalaEspera = 0;
  bool sillones[3] = {true, true, true }; // todos los sillones libres
  bool atencion[3] = {false, false, false }; // todos los sillones libres
  bool servicios[3][3]; // matriz que indica los servicios requeridos en cada
                        // sillón

  bool AtenderClienta(bool esCorte, bool esTinta, bool esBrushing) {
      If(enSalaEspera == 5)
            return false; // la clienta se va

      enSalaEspera++;
      while((sillon = dameSillonLibre()) == -1)
            cndEnSalaEspera.wait(); // La clienta espera a ser atendida
      enSalaEspera--;

      servicios[sillon] = {esCorte, esTinta, esBrushing};
      chequeoFin(sillon);
      cndFinAtencion[sillon].wait(); // La clienta espera a que terminen
      sillones[sillon] = true;       // de antenderla
      cndEnSalaEspera.signal(); // Ahora despierta a la siguiente
      return true;
  }

  int Peluquera(int sillon, inout: bool esTinta) {
// Si vengo de atender a una clienta, marco el servicio como atendido.
// Se hace todo en la misma funcion para no perder el monitor entre medio
    If(sillon != -1) {
      atencion[sillon] = false;
      If(esTinta)
            servicios[sillon][tinta] = false;
      else servicios[sillon][corte] = false;

      chequeoFin(sillon);
    }
    else sillon = 0;

    while (true) {
      For(int i = 0; i < 3; i++) {
        int idx = (sillon + i) % 3; // Primero ve si puede seguir con la misma
        if(!atencion[idx])          // clienta, sino intenta con las otras 2
            If(servicios[idx][corte]) {
                  atencion[idx] = true;
                  esTinta = false;
                  Return idx;
            }
            Else If(servicios[idx][tinta]) {
                  atencion[idx] = true;
                  esTinta = true;
```

```
                Return idx;
            }
        }
        cndPeluquera.wait();
    }
  }

  int Colorista(int sillon, inout: bool esTinta) {
    If(sillon != -1) {
      atencion[sillon] = false;
      If(esTinta)
            servicios[sillon][tinta] = false;
      else servicios[sillon][brushing] = false;

      chequeoFin(sillon);
    }
    else sillon = 0;

    while (true) {
      For(int i = 0; i < 3; i++) {
        int idx = (sillon + i) % 3;
        if(!atencion[idx] && !servicios[idx][corte])
            If(servicios[idx][tinta]) {
                  atencion[idx] = true;
                  esTinta = true;
                  Return idx;
            }
            Else If(servicios[idx][brushing]) {
                  atencion[idx] = true;
                  esTinta = false;
                  Return idx;
            }
      }
      cndColorista.wait();
    }
  }

  void chequeoFin(int sillon) {
    int i;
    for(i = 0; (i < 3) && !servicios[sillon][i]; i++);
    if (i == 3)
      cndFinAtencion[sillon].signal();
    else if (i == corte)
      cndPeluquera.signal();
    else if (i == tinta)
      cndPeluquera.signal();
      cndColorista.signal();
    else if (i == brushing)
      cndColorista.signal();
  }

  int dameSillonLibre() {
    int sillon = -1;
    if(sillones[0]) sillon = 0;
    else if(sillones[1]) sillon = 1;
    else if(sillones[2]) sillon = 2;
    else return sillon;
    sillones[sillon] = false;
    return sillon;
  }
}
```

```
Procedure Clienta() {
    bool corte, tinta, brushing;
    queServicios(corte, tinta, brushing);

    if (Organizador.AtenderClienta(corte, tinta, brushing))
      pagar();
}

Procedure Peluquera() {
    int sillon = -1;
    bool tinta = false;

    while(true) {
      sillon = Organizador.Peluquera(sillon, tinta);
      If(tinta)
            hacerTinta(sillon);
      else cortarPelo(sillon);
    }
}

Procedure Colorista() {
    int sillon = -1;
    bool tinta = false;

    while(true) {
      sillon = Organizador.Colorista(sillon, tinta);
      If(tinta)
            hacerTinta(sillon);
      else hacerBrushing(sillon);
    }
}

Main() {
  cobegin
    Peluquera(); Peluquera(); Colorista();
    Clienta(); ... Clienta();
  coend;
}
```