Solución Examen Febrero 2006

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

Para las siguientes partes, conteste justificando brevemente, cada una de las preguntas.

1. Describa dos métodos a través de los cuales el sistema operativo se entera de que un dispositivo de E/S ha finalizado un pedido generado por el sistema operativo, mencionando las principales características de cada uno y compare los mismos.
2. Describa los estados en los que puede permanecer un proceso y realice un diagrama mostrando las transiciones de un estado a otro mencionando el, o los eventos que se producen para el cambio de estado.
3. ¿Qué entiende por planificador expropiativo?
4. Considere el siguiente conjunto de procesos, donde la duración de la ráfaga de CPU se mide en milisegundos:

Proceso	Tiempo de ráfaga	Prioridad
P1	8	3
P2	1	1
P3	4	3
P4	2	4
P5	7	2

   Se dispone de un equipo monoprocesador y se supone que los procesos llegaron en el orden P1, P2, P3, P4, P5, todos en el instante 0.

   a. Dibuje un diagrama que ilustre la ejecución de estos procesos utilizando planificación: FCFS, SJF, una técnica por prioridad no expropiativa (a menor número, mayor prioridad), y RR con cuanto = 1.

   b. Calcule el tiempo de retorno y espera de los procesos para el algoritmo RR de la parte a.

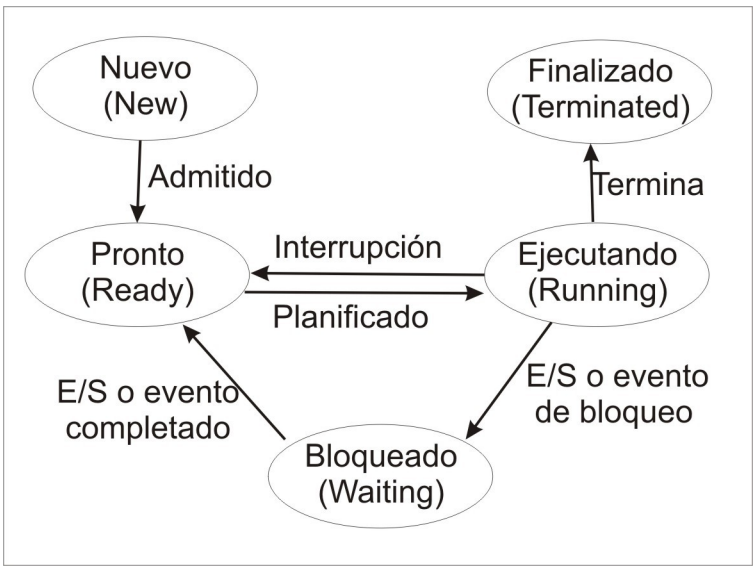
5. Aplicando el algoritmo del banquero conteste si el siguiente sistema está o no en un estado seguro.

	Asignados	Máximo	Disponible
P0	1 0 0 3	2 0 1 3	3 1 1 0
P1	1 0 0 0	1 7 5 0	
P2	1 3 5 4	2 3 5 6	
P3	0 6 3 2	0 6 5 2	
P4	0 0 1 4	0 6 5 6	

6. Una vez ocurrido un deadlock, describa un proceso que permita recuperarse del mismo. Mencione sus ventajas y desventajas.
7. ¿Qué se entiende por direccionamiento virtual? Explique que es una dirección lógica o virtual y qué es una dirección física.
8. Describa dos implementaciones posibles para saber el espacio libre en disco. Compárelos mencionando sus ventajas y desventajas entre ellos.
9. ¿Qué son y para qué sirven los semáforos binarios? Implemente semáforos binarios con monitores.
10. Describa el modelo de Working-set. Mencione los conceptos en que se basa, las propiedades que cumple y el fin que persigue.
11.

SOLUCIÓN:

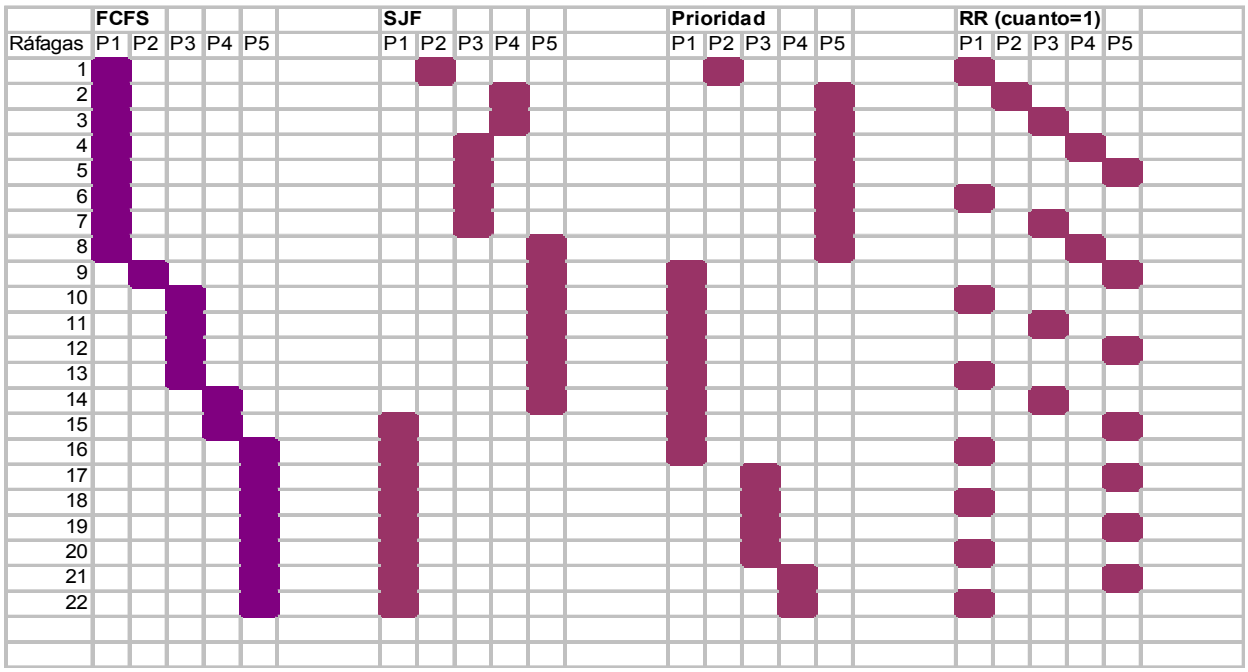
1) Polling e Interrupciones.
   Polling: El sistema realiza el pedido y queda en una iteración consultando el 'busy bit' del controlador del dispositivo hasta que este quede limpio (señal que finalizó).
   Interrupciones: El sistema realiza el pedido y será avisado de la finalización a través de una interrupción.
   El polling tiene como desventaja el hacer 'busy waiting'. Ciclos de CPU serán desperdiciados. En cambio el método de interrupciones permite ejecutar otros procesos mientras el dispositivo realiza el pedido.
   El polling sería útil en casos en que el dispositivo este libre y que tenga mayor velocidad que la del procesador.
   Las interrupciones permiten un manejo asincrónico y además permiten ser atendidas cuando el sistema operativo lo crea conveniente.
2)



3) Planificador expropiativo es aquel que ante los siguientes eventos es ejecutado para asignar un proceso a la CPU:
   a. Cuando un proceso se cambia del estado de ejecución al estado de pronto (cuando ocurre una interrupción).
   b. Cuando un proceso que estaba bloqueado pasa al estado Pronto (ante una E/S finalizada o un evento completado).

   Ante estos casos el planificador elige de la lista de procesos listos el proceso que tenga mayor prioridad según su algoritmo de planificación.
   Se puede interpretar como que el planificador asegura que siempre está ejecutando el proceso (dentro de los procesos listos) con mayor prioridad.

4)



RR (Cuanto=1)

Cs = cuantos de espera

Ce = cuantos de ejecución

tiempo de P1 = 14 Cs + 8 Ce (espera = 14ms)

tiempo de P2 = 1 Cs + 1 Ce (espera = 1ms)

tiempo de P3 = 10 Cs + 4 Ce (espera = 10ms)

tiempo de P4 = 6 Cs + 2 Ce (espera = 6ms)

tiempo de P5 = 14 Cs + 7 Ce (espera = 14ms)

Tiempo de espera: Tiempo total en la cola de listos.

Tiempo de espera total = 14 + 1 + 10 + 6 + 14 = 45ms

Tiempo de espera promedio = (14 + 1 + 10 + 6 + 14) / 5 = 9 ms

Tiempo de retorno: Tiempo transcurrido desde que se envió el proceso hasta que termina.

Tiempo de retorno total = 22 + 2 + 14 + 8 + 21 = 67ms

Tiempo de retorno promedio = (22 + 2 + 14 + 8 + 21) / 5 = 13,4 ms

5) Un estado es seguro si el sistema puede asignar recursos a cada proceso (hasta su máximo) en algún orden y aún así evitar los bloqueos mutuos.
Más formalmente, un sistema está en un estado seguro solo si existe una secuencia segura. Una secuencia de procesos [P0, P1, ..., Pn] es una secuencia segura para el estado de asignación actual si, para cada Pi, los recursos que Pi todavía puede solicitar se pueden satisfacer con los recursos que actualmente están disponibles más los recursos que tienen todos los Pj, donde j < i. En esta situación, si los recursos que Pi necesita todavía no están disponibles, Pi podrá esperar hasta que todos los Pj hayan terminado. En ese momento, Pi podrá obtener todos los recursos que necesita, llevar a cabo su tarea designada, liberar los recursos que adquirió y terminar.

	Asignados	Máximo	Disponible
P0	1 0 0 3	2 0 1 3	3 1 1 0
P1	1 0 0 0	1 7 5 0	
P2	1 3 5 4	2 3 5 6	
P3	0 6 3 2	0 6 5 2	
P4	0 0 1 4	0 6 5 6	

Vamos a ver si el sistema está en estado seguro. Aplicando el Algoritmo del banquero.

trabajo0 = (3 1 1 0)
fin0 = (False False False False False)

	Necesidades (Máximo – Asignados)
P0	1 0 1 0
P1	0 7 5 0
P2	1 0 0 2
P3	0 0 2 0
P4	0 6 4 2

Elijo P0 (Necesidad(0) ≤ trabajo0 : (1 0 1 0) ≤ (3 1 1 0)):
trabajo1 = trabajo0 + Asignados(0) =
(3 1 1 0) + (1 0 0 3) = (4 1 1 3)
fin1 = (True False False False False)

Elijo P2 (Necesidad(2) ≤ trabajo1 : (1 0 0 2) ≤ (4 1 1 3)):
trabajo2 = trabajo1 + Asignados(2) =
(4 1 1 3) + (1 3 5 4) = (5 4 6 7)
fin2 = (True False True False False)

Elijo P3 (Necesidad(3) ≤ trabajo2 : (0 0 2 0)≤ (5 4 6 7)):
trabajo3 = trabajo2 + Asignados(3) =
(5 4 6 7) + (0 6 3 2) = (5 10 9 9)
fin3 = (True False True True False)

Elijo P4: (Necesidad(4) ≤ trabajo3 : (0 6 4 2) ≤ (5 10 9 9)):
trabajo4 = trabajo3 + Asignados(4) =
(5 10 9 9) + (0 0 1 4) = (5 10 10 13)
fin4 = (True False True True True)

Elijo P1 (Necesidad(1) ≤ trabajo4 : (0 7 5 0) ≤ (5 10 10 13)):
trabajo5 = trabajo4 + Asignados(1) =
(5 10 10 13) + (1 0 0 0) = (6 10 10 13)
fin5 = (True True True True True)

Verificación:

Asignadosini + Disponiblesini = Trabajofin
(3 9 9 13) + (3 1 1 0) = (6 10 10 13)

Entonces existe una secuencia [P0, P3, P1, P4, P2] que es segura.
Entonces es estado seguro y por lo tanto no tiene que haber deadlock.

6)

- Matar todos los procesos. Es seguro pero costoso. Se pueden matar procesos que llevan mucho tiempo de cómputo. En este caso no es necesario ejecutar el algoritmo de detección de deadlock, ya que se matan todos los procesos involucrados.
- Matar un proceso a la vez hasta eliminar el ciclo que genera el deadlock. Tiene un costo elevado debido a que luego de matar cada proceso debe ejecutarse nuevamente el algoritmo de detención de deadlock para determinar si aún se está en deadlock.

7) En un sistema que maneje memoria virtual, a cada proceso se le asigna un espacio virtual (o lógico) de direccionamiento. Este espacio virtual permite abstraerse del direccionamiento físico (real).
Por lo general el direccionamiento esta restringido por los bits de la arquitectura. Por ejemplo, en una arquitectura de 32 bits, un proceso puede generar direcciones de hasta 2^{32} = 4GB. Por lo que podría acceder a una memoria virtual de 4GB.
Una dirección lógica o virtual es la direcciones generadas por el proceso (se puede decir que son las que se utilizan a nivel de CPU), mientras que una dirección física es una dirección real de la memoria principal del sistema. En la mayoría de los sistemas actuales la unidad de administración de memoria (MMU-dispositivo de hardware) es la encargada de hacer la conversión de direcciones lógicas a físicas.

8) Mapa de bits: Es un arreglo de bits en donde cada entrada se mapea a un bloque del sistema de archivo. Si el bloque está libre, el bit está indicado con un 1; si el bloque está asignado, el bits indica 0 (puede ser al revés).
Lista encadenada: Los bloques libres están en una lista encadenada. Existe un puntero al primer bloque libre y cada bloque libre tiene una referencia al siguiente bloque.
Agrupamiento: Es igual que la lista encadenada, pero además de tener el puntero al siguiente (dado que sobra mucho espacio) se tiene un grupo de bloques libres.
Contador: Al igual que los dos últimos se tiene una lista encadenada de bloques libres. Pero como a veces se tienen varios bloques libres que están contiguos, se hace un agrupamiento en el primer bloque libre de los contiguos agregando un contador que informa cuantos bloques libres contiguos ahí.

9) Los semáforos son una herramienta para la sincronización de procesos en el sistema.

```
type bin_semaforo: monitor;

var   ocupado : boolean;
      no_ocupado : condition;

procedure wait;
begin
  if (ocupado) then
    no_ocupado.wait;
  ocupado := true;
end procedure;
```

```
procedure signal;
begin
  ocupado := false;
  no_ocupado.signal;
end procedure;

begin
  ocupado := false;
end bin_semaforo;
```

Ver secuencia :

P1	P2
Bin_semaforo.wait; Bin_semaforo.signal;	bin_semaforo.wait;

a. P1 hace wait, como no esta ocupado pone ocupado en true y sigue.
b. P2 hace wait, como esta ocupado hace wait sobre la variable no_ocupado y queda esperando.
c. P1 hace signal, pone ocupado en false y hace un signal sobre no_ocupado. Entonces P2 que estaba esperando en la condition no_ocupado se despierta y pone ocupado en true y entra;

10)

El modelo de working-set es una estrategia que permite defenderse ante el problema de hiperpaginación (trashing). Se basa en el principio de localidad: Un proceso al ejecutar utiliza un conjunto limitado de páginas a la vez. A este conjunto de páginas se les denomina páginas activas. Al tamaño del conjunto se le denomina ventana.
El modelo utiliza un parámetro (A) que define el tamaño de la ventana del working-set. Si una página está siendo usada en el momento, entonces pertenece al working-set. Cuando una página hace un tiempo, mayor que la ventana definida, no es accedida deja de pertenecer del working-set.
La efectividad del modelo se logra con una buena selección del parámetro A. La propiedad más importante del working-set es el tamaño de la ventana para cada proceso. La propiedad que se debe cumplir es que la suma del tamaño de todas las ventanas del sistema debe ser a lo sumo igual a la cantidad de páginas disponibles en el sistema.

Problema 2

Sea un sistema operativo que utiliza páginas de 4KB, tablas de páginas de 2 niveles y regiones de texto compartidas. Existen simultáneamente dos procesos A y B que ejecutan el mismo programa y de los que sabemos que en un instante de tiempo de terminado Ta tienen la siguiente situación:

Proceso	Región	Nro. Paginas	Nro. Paginas Presentes	Dirección de Comienzo
A	Texto	24	7	0
	Datos	13	5	2^{23}
	Texto biblioteca dinámica Z	45	12	2^{24}
	Datos biblioteca dinámica Z	5	3	$2^{24}+2^{23}$
	Archivo M compartido	27	7	2^{25}
	Pila	11	3	2^{30}
B	Texto	24	7	0
	Datos	15	5	2^{23}
	Archivo M compartido	27	7	2^{24}
	Texto biblioteca dinámica Z	45	12	2^{25}
	Datos biblioteca dinámica Z	5	3	$2^{25}+2^{23}$
	Pila	16	7	2^{30}

Se pide resolver las siguientes cuestiones, justificando adecuadamente cada respuesta:

a) Calcular el número total de marcos de páginas que tienen asignados entre los dos procesos en ese instante.

b) Seguidamente, A ejecuta un bucle de lectura que recorre todo el archivo M compartido por ambos procesos. Suponiendo que no se reemplaza ninguna página de los procesos A y B, indicar el nº de fallos de página que se producen así como el total de marcos de página que tienen ahora asignados entre los dos procesos.

c) Tomando como referencia el instante de tiempo Ta, indicar si se produce un error de ejecución y en su caso, el valor de la variable v en cada uno de los procesos en los supuestos 1, 2, 3, y 4 siguientes.

1. El proceso A ejecuta: v = *p (donde p vale 24) e inmediatamente el proceso B ejecuta el mismo trozo de código, pero ahora p vale 25. En este caso, indicar, además, si se puede producir un fallo de página considerando que las variables v de cada proceso están cada una en su correspondiente marco de página.

2. El proceso A ejecuta: p = (2^{24}) + 1; *p = 2^{34}; v = *p; e inmediatamente el proceso B ejecuta v = p.
3. El proceso A ejecuta: p = 2^{24} + 2^{23} + 12; *p = 2^{34}; e inmediatamente el proceso B ejecuta p = 2^{24} + 2^{23} + 12
4. El proceso A ejecuta: p = 2^{25} + 10012; *p = 2^{34}; e inmediatamente el proceso B ejecuta p = 2^{24} +10012; v = *p.

SOLUCIÓN:

a) Las regiones privadas del problema son las de datos, las de datos de las bibliotecas dinámicas y las pilas. Sumamos todos los marcos de página del proceso A: 7 + 5 + 12 + 3 + 7 + 3 = 37 más los marcos de página del proceso B que no comparte con A, es decir, 5 + 3 + 7 = 15. Esto da un total de 52 marcos de página.

b) Como la región del fichero compartido tiene 27 páginas, de las que 7 ya están en marcos, el recorrer todo el archivo hará que se traigan a marcos las 27 – 7 = 20 páginas restantes, lo que produce 20 fallos de página. Después de este bucle, entre los dos procesos se tendrán 52 + 20 = 72 marcos de página, dado que no se produce ningún reemplazo.

c) No se produce error de ejecución vA = 24 y vB = 25. No es posible un fallo de página. Justo después de ejecutar A, B ejecuta el mismo código, por lo que ya tiene que estar en un marco. Por otro lado v = *p con p = 24 implica un acceso a la primera página de la región de código, al igual que v = *p con p = 25, página que es compartida por los dos procesos.

d) Se produce un error al ejecutar A puesto que se intenta escribir en una región (texto de biblioteca) que no tiene derechos de escritura. Del valor vB no podemos decir su valor, puesto que no conocemos el valor de pB.

e) Se produce un error al ejecutar B, dado que la dirección 2^{24} + 2^{23} + 12 = 2^{24} + 2^{12}•2^{11} + 12 no está en el mapa de memoria de ese proceso.

f) En este caso no se produce error de ejecución, siendo vB = 2^{34}.

Problema 3

Una tribu de N caníbales come de una gran marmita común con capacidad para 6 comensales simultáneos.

Cuando un comensal quiere comer, come de la marmita, a menos que no haya suficiente comida para él. Si no hay suficiente comida en la marmita, el caníbal despierta al cocinero y espera a que el cocinero haya rellenado la marmita con la carne de los misioneros capturados (no debe haber notificaciones repetidas). Para rellenar la marmita el cocinero debe esperar a que todos los comensales que se encuentran actualmente comiendo terminen. El comensal que avisó debe ser el primero en comer.

El cocinero, por su parte, vuelve a dormir cuando ha rellenado la marmita.

Cada cierto tiempo llega el jefe de la tribu el cual debe esperar que todos los comensales terminen para comer solo, teniendo éste prioridad sobre los nuevos comensales.

Consideraciones:

- No podrán entrar nuevos comensales a la marmita en las siguientes situaciones:
  - El jefe está comiendo o esperando para comer.
  - El cocinero está rellenando o esperando para rellenar.
- El jefe se comporta como un comensal más respecto al cocinero y a quien fue el primero en avisarle.
- Se supone que la marmita llena dispone de comida para más de seis caníbales.

Se dispone de las siguientes funciones:

- Hay_suficiente_comida (): boolean

  Esta función es ejecutada por los comensales (caníbales y el jefe) retorna si hay suficiente comida en la marmita. No puede ser ejecutada por dos o más comensales a la vez.

- Rellenar ()

  Esta función es ejecutada por el cocinero.

- Comer ()

  Es ejecutado por los comensales.

- Ocio ()

  Ejecutada por los caníbales y el jefe cuando no están comiendo.

Se pide:

Implementar los procedimientos caníbal, jefe y cocinero utilizando monitores. Especifique la semántica de los mismos en caso de ser necesario.

SOLUCIÓN:

```
Monitor Marmita
Begin

      var cant_canibales: integer
      var jefe, rellenar: boolean
      var cnd_jefe, cnd_relleno, cnd_cocinero, cnd_canibal: condition

      Procedure comer (soy_jefe: boolean)
      begin
            if soy_jefe then
            begin
                  jefe = true;
                  if cant_canibales > 0 or rellenar do
                        cnd_jefe.wait();
            end
            else
                  if cant_canibales = 6 or rellenar or jefe do
                        cnd_canibal.wait();

            if not hay_suficiente_comida() then
            begin
                  rellenar = true;
                  if cant_canibales = 0 then
                        cnd_cocinero.signal();
                  cnd_relleno.wait();
            end

            if not jefe and not rellenar and cant_canibales < 6 then
                  cnd_canibal.signal();

            cant_canibales++;
      end


      Procedure termine_comer (soy_jefe: boolean)
      begin
            if soy_jefe then
                  jefe = false;
            cant_canibales--;
            despertar();
      end


      Procedure cocinero_inicio()
      begin
            cnd_cocinero.wait();
      end


      Procedure cocinero_fin()
      var i: integer;
      begin
            rellenar = false;
            cnd_relleno.signal(); // prioridad máxima al que avisó
            despertar();
      end
```

```
      Procedure despertar()
      begin
            if cant_canibales = 0 then
            begin
                  if rellenar then
                        cnd_cocinero.signal();
                  else if jefe then
                        cnd_jefe.signal();
            end
            if not jefe and not rellenar then
                  cond_canibal.signal();
      end

      begin
            cant_canibales = 0;
            jefe = false;
            rellenar = false;
      end
end

Procedure canibal ()
begin
      while true do
      begin
            marmita.comer(false);
            comer();
            marmita.termine_comer(false);
            ocio();
      end
end

Procedure jefe ()
begin
      while true do
      begin
            marmita.comer(true);
            comer();
            marmita.termine_comer(true);
            ocio();
      end
end

Procedure cocinero ()
begin
      while true do
      begin
            marmita.cocinero_inicio();
            rellenar();
            marmita.cocinero_fin();
      end
end
```
