Primer Parcial Mayo 2017

Lea detenidamente las siguientes instrucciones. No cumplir los requerimientos puede implicar la pérdida de los puntos del parcial.

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 según el orden de hojas.**

Dudas

- **Sólo se contestarán dudas de letra.**
- **No se aceptarán dudas en los últimos 15 minutos del parcial.**

Material

- **El examen es SIN material (no puede utilizarse ningún apunte, dispositivo móvil, 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.**
- **Al momento de finalizar el parcial 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 parcial.**

Finalización

- **El parcial dura 2 horas.**

Problema 1 (15 puntos)

1. Describa la arquitectura micronúcleo. Describa las ventajas que tiene tiene con respecto a una arquitectura monolítica.
2.
   a) ¿Qué función cumple el planificador de corto plazo del sistema operativo?
   b) ¿Qué componente controla el grado de multiprogramación del sistema?
   c) Indique que condiciones desencadenan la ejecución de un planificador de corto plazo no expropiativo.
3. ¿ Qué es una *system call* ? Describa los pasos que realiza una system call al ejecutar, indicando qué componente ejecuta cada paso (proceso de usuario, hardware o núcleo del sistema operativo)
4. Describa las tres secciones en que se divide la superficie de un disco magnético.
5. Describa el comportamiento de un monitor, las condiciones y las operaciones *wait* y *signal* sobre ellas. Explique las semánticas posibles de estas operaciones.

Problema 2 (15 puntos) (3,4,2,2,4)

Se tiene un sistema operativo multiprogramado con planificador FIFO (*First-In-First-Out*) **no expropiativo** ejecutando en un sistema monoprocesador.

En el sistema existen dos procesos. A continuación se especifica el comportamiento de los procesos y de las rutinas de interrupción de interés:

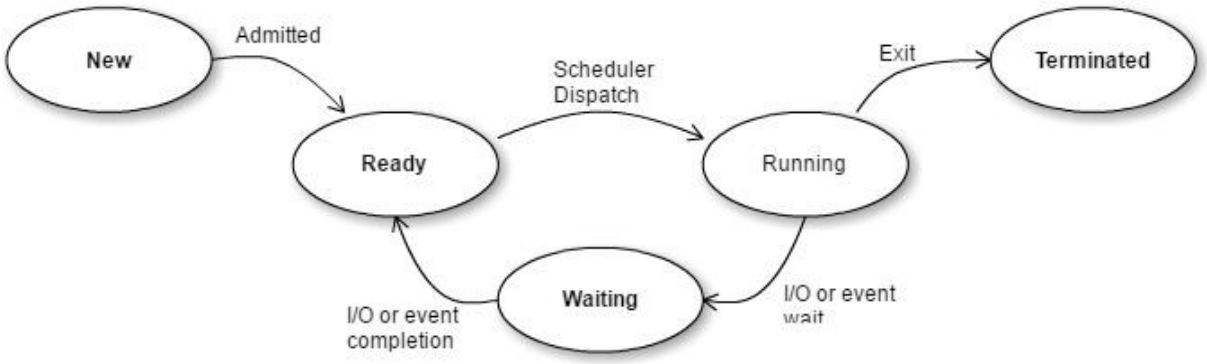
Programa 1	**Programa 2**	**Rutina Interrupción A**	**Rutina Interrupción B**
Ejecuta 300ms Se bloquea por recurso A Ejecuta 200ms	Ejecuta 400ms Se bloquea por recurso B Ejecuta 200ms	Deja disponible el recurso A.	Deja disponible el recurso B.

El proceso correspondiente al programa 1 comienza en el milisegundo 0 y el proceso correspondiente al programa 2 comienza en el milisegundo 200. La interrupción que brinda el recurso A es invocada en el milisegundo 500 y la interrupción que brinda el recurso B en el milisegundo 1.200. Se asume que las rutinas de atención a interrupciones demoran 0ms en ejecutar.

Se pide:

1. Realice un diagrama que muestre los estados y transiciones que tiene un proceso en un sistema operativo con esas características. Describa brevemente cada componente.
2. Realice un esquema en el tiempo que muestre qué proceso está asignado al procesador, la cola de procesos listos (ready queue) y el estado de los recursos A y B.
3. Calcule el tiempo de espera y el tiempo de retorno de todos los procesos.
4. Calcule el porcentaje de uso de la CPU desde el instante 0 hasta que finaliza el último proceso.
5. Realice el mismo diagrama de la parte 2 pero ahora suponiendo que se cuenta con un sistema operativo con un planificador **expropiativo** basado en prioridades. Considere que el programa 2 tiene mayor prioridad que el programa 1.

1. Esquema de estados y transiciones:



Estados:

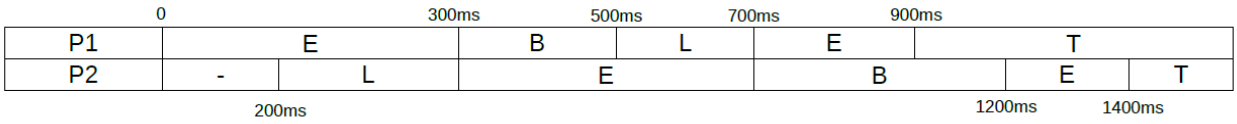
- New: Cuando el proceso es creado.
- Ready: El proceso está listo para ejecutar, sólo necesita que se le asigne el recurso procesador
- Running: El proceso tiene asignado el recurso procesador y está ejecutando sus instrucciones.
- Waiting: El proceso está esperando por un evento/recurso.
- Terminated: El proceso finalizó su ejecución.

Transiciones:

- New -> Ready
  Al crearse un proceso pasa inmediatamete al estado Listo
- Ready -> Running
  En el estado listo los procesos esperan a que se le asigne el recurso procesador. Al liberarse un procesador el scheduler selecciona el próximo proceso a ejecutar.
- Running -> Waiting
  Mientras el proceso está ejecutando puede solicitar pedidos ya sean de entrada/salida o esperar algún evento. Al ser un sistema multiprogramado, el proceso es puesto en una cola a la espera de que se complete su pedido.
- Waiting -> Ready
  Una vez que ocurre el evento que estaba esperando el proceso, este es colocado en la cola de procesos listos.
- Running -> Terminated
  Cuando el proceso ejecuta su ultima instrucción el proceso pasa al estado terminado. El sistema libera las estructuras que representan al proceso.

Nota: No se coloca transición de estado Running a Ready debido a que se tiene un planificador no expropiativo, por tal razón un proceso que está ejecutando solo puede bloquearse a la espera de un evento o finalizar su ejecución.

2.



Recurso A	No Disponible	Asignado a P1
Recurso B	No Disponible	Asignado a P2

500ms 1200ms

3.

Tiempo de Espera:

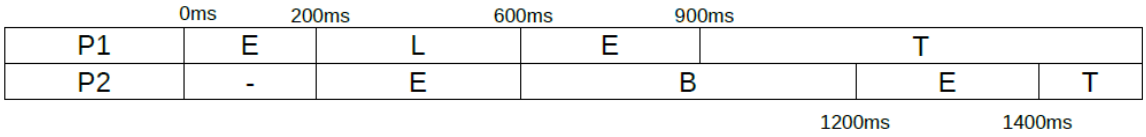
- Proceso 1: 200 ms
- Proceso 2: 100 ms

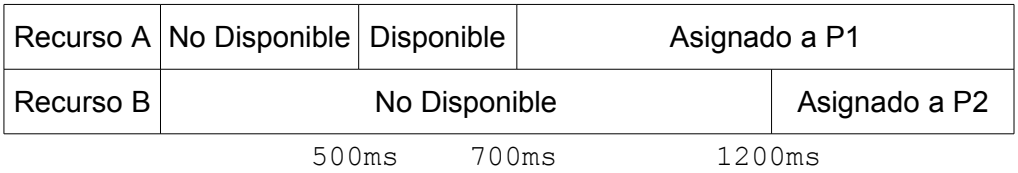
Tiempo de Retorno:

- Proceso 1: Entró en el sistema en t=0 y finalizó en t=900ms -> 900 – 0 = 900 ms
- Proceso 2: Entró en el sistema en t=200 y finalizó en t=1400ms -> 1400 – 200 = 1200 ms

4. Uso de CPU:
   intervalos [0 - 900) y [1200 – 1400) -> (900 – 0) + (1400 – 1200) = 1100 ms
   Entonces: 1100 * 100 / 1400 = 78.57%

5.





Problema 3 (7 puntos)

Sea el siguiente código:

```
Semaphore X, Y, Z;
int valor;

procedure A()
{
  V(X);
  P(Y);
  P(Z);
  cobegin
    begin
      print "Dato1:" + valor;
      if(valor < 10)
        A();
      endif
      valor = valor + 10;
      V(Z);
    end;
    begin
      P(X);
      valor = valor + 5;
      print "Dato2:" + valor;
      V(Z);
    end;
  coend
}

procedure B()
{
  P(X);
  V(Y);
  P(Z);
  valor = valor + 1;
  print "Dato3:" + valor;
  V(Z);
}

begin
  init(X,0);
  init(Y,0);
  init(Z,1);
  valor = 5;
  cobegin
     A();
     B();
  coend
  print "Fin: " + valor;
end
```

Se pide:

Describir los posibles comportamientos del programa indicando las posibles salidas del mismo..

Luego de inicializar los semáforos y valor en 5 se ejecutan concurrentemente los procedimientos A y B.

El procedimiento B espera por el V de X que realiza el procedimiento A y el procedimiento A espera por el V de Y que realiza el procedimiento B.

Luego ambos intentan obtener el semáforo Z (que actúa como mutex), por lo que hay 2 casos.

Caso 1:

Si lo toma el procedimiento A el mismo se separa en dos hilos concurrentes.

El segundo hilo (que hace un P(X)) se va a bloquear hasta que se realice un V(X). El primer hilo imprime "**Dato1: 5**" y como valor es menor a 10 llama nuevamente al procedimiento A. En este punto se hace el V(X) que va a liberar al segundo hilo y luego el hilo se bloquea esperando por el semáforo Y (y quedará en deadlock pues nadie más hace un V de dicho semáforo).

El primer hilo eventualmente toma el semáforo X, suma 5 a valor por lo que imprime "**Dato2: 10**", luego libera el mutex Z. Como el primer hilo quedó bloqueado, el procedimiento A no termina.

Al liberar el mutex Z, lo toma el procedimiento B, suma 1 a valor e imprime "**Dato3: 11**" y luego libera el mutex y termina.

Como el hilo del procedimiento A quedó en deadlock no se llega a imprimir **"Fin: 11"**

Caso 2:

Si lo toma el procedimiento B, suma 1 a valor e imprime "**Dato3: 6**", y luego libera el mutex y termina.

Ahora se repite el comportamiento indicado en Caso 1 para el procedimiento A pero teniendo en cuenta que dato tiene el valor 6.

Por lo que se imprime "**Dato1: 6**" y luego "**Dato2: 11**".

Nuevamente, como el hilo de A quedó en deadlock no se llega a imprimir **"Fin: 11"**