Primer Parcial Mayo 2014

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.**

Finalización

- **El parcial dura 2 horas.**
- **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.**

Problema 1 (15 puntos)

1. Indique ¿qué es el PCB? Enumere y describa 5 campos del mismo.
2. ¿Qué tareas realizan un despachador y un planificador?
3.
   a) Nombre dos problemas de seguridad que aparecen al implementar un sistema operativo multiprogramado.
   b) Explique una forma de solucionar cada uno.
4. ¿Cuál es la diferencia entre los planificadores de largo plazo, de mediano plazo y de corto plazo?
5. Cuando se hace un fork() ¿los procesos que se crean comparten el stack? ¿y el heap? Justifique.

Problema 2 (16 puntos) (2,10,4)

Sea un sistema operativo simétrico multiprogramado en el cual se dispone de dos procesadores. El planificador del sistema utiliza una estrategia de planificación basada en una cola de procesos listos con dos niveles de prioridad (alta/baja) y retroalimentación (multi-level feedback queue con dos niveles de prioridad).

Se sabe que el sistema cumple con las siguientes características:

- En el nivel de alta prioridad se utiliza un algoritmo de planificación SJF (Shortest Job First) expropiativo.
- En el nivel de baja prioridad se utiliza un algoritmo de planificación Round Robin con un quantum de 50ms.
- Los procesos al ser creados se les asigna la prioridad baja.
- Un proceso sube al nivel de mayor prioridad si en los últimos 100ms no ha usado el recurso procesador.
- Un proceso baja al nivel de menor prioridad si al momento de librar el recurso procesador ha utilizado el mismo por al menos 50ms.
- Nunca debe detenerse o retrasarse la ejecución de un proceso con alta prioridad si solo existen en el sistema procesos con baja prioridad.
- Cuando un proceso pasa a estado listo, su quantum es reiniciado.
- Se asume que el tiempo empleado por el sistema operativo para acciones como realizar un cambio de contexto o ejecutar el planificador es despreciable.
- Se asume una única cola multinivel de procesos listos.

En el sistema existen dos programas que al ejecutar tienen el siguiente comportamiento:

Programa 1	Programa 2
`Ejecuta durante 40ms.`	`Ejecuta durante 70ms.`
`Se bloquea durante 120 ms.`	`Se bloquea durante 70 ms.`
`Ejecuta durante 150ms.`	`Ejecuta durante 100ms.`
`Termina.`	`Termina.`

Los procesos P1 y P2 son instancias del Programa 1 y son creados a los 0ms y 20ms respectivamente. Los procesos P3 y P4 son instancias del Programa 2 y son creados a los 10ms y a los 100ms respectivamente.

Se pide:

1. Enumere al menos una desventaja/falencia que puede existir al utilizar la estrategia de planificación planteada en este ejercicio.
2. Realice un esquema o diagrama de planificación (tiempo vs. procesos), en que se indique el estado de cada proceso (listo/ejecutando/bloqueado/terminado), su prioridad y a qué procesador está asignado el proceso en cada intervalo de tiempo.
3. Defina la métrica Utilización de CPU, calcularla para la planificación realizada en la parte anterior.

Solución Problema 2:

1. La estrategia de planificación planteada en este ejercicio sufre de posposición indefinida ya que un proceso puede quedar por tiempo infinito sin utilizar el procesador. Un ejemplo de esto es: Un instancia del programa A que ejecuta durante 60 ms y 2 procesos de un programa B que tiene la siguiente ejecución en loop: ejecutan durante 40ms, se bloquean 10 ms. Si los tres procesos se encuentran en la cola de procesos listos de alta prioridad (Shortest Job First), siempre se van a ir alternando el uso de la CPU los procesos B. Teniendo como resultado la postergación del uso del CPU por parte del proceso A indefinidamente.
2. A continuación se muestran dos posibles planificaciones que cumplen con la característica del sistema planteado.

Tiempo (ms)	***Procesos***			
	P1	***P2***	***P3***	***P4***
0	EJEC CPU1 BAJA	-	-	-
10	EJEC CPU1 BAJA	-	EJEC CPU2 BAJA	-
20	EJEC CPU1 BAJA	LISTO BAJA	EJEC CPU2 BAJA	-
40	BLOQUEADO BAJA	EJEC CPU1 BAJA	EJEC CPU2 BAJA	-
80	BLOQUEDO BAJA	BLOQUEADO BAJA	BLOQUEADO BAJA	-
100	BLOQUEADO BAJA	BLOQUEADO BAJA	BLOQUEADO BAJA	EJEC CPU1 BAJA
140	BLOQUEADO ALTA	BLOQUEADO BAJA	BLOQUEADO BAJA	EJEC CPU1 BAJA
150	BLOQUEADO ALTA	BLOQUEADO BAJA	EJEC CPU2 BAJA	EJEC CPU1 BAJA

160	EJEC CPU1 ALTA	BLOQUEADO BAJA	EJEC CPU2 BAJA	LISTO BAJA
180	EJEC CPU1 ALTA	BLOQUEADO ALTA	EJEC CPU2 BAJA	LISTO BAJA
200	EJEC CPU1 ALTA	EJEC CPU2 ALTA	LISTO BAJA	LISTO BAJA
260	EJEC CPU1 ALTA	LISTO BJA	LISTO BAJA	EJEC CPU2
270	EJEC CPU1 ALTA	LISTO BAJA	EJEC CPU2 BAJA	BLOQUEADO ALTA
310	TERMINADO	EJEC CPU1 BAJA	EJEC CPU2 BAJA	BLOQUEADO ALTA
320	TERMINADO	EJEC CPU1 BAJA	TERMINADO	BLOQUEADO ALTA
340	TERMINADO	EJEC CPU1 BAJA	TERMINADO	EJEC CPU2 ALTA
400	TERMINADO	TERMINADO	TERMINADO	EJEC CPU2 ALTA
440	TERMINADO	TERMINADO	TERMINADO	TERMINADO

Tiempo (ms)	***Procesos***			
	P1	***P2***	***P3***	***P4***
0	EJEC CPU1 BAJA	-	-	-
10	EJEC CPU1 BAJA	-	EJEC CPU2 BAJA	-
20	EJEC CPU1 BAJA	LISTO BAJA	EJEC CPU2 BAJA	-
40	BLOQUEADO BAJA	EJEC CPU1 BAJA	EJEC CPU2 BAJA	-
80	BLOQUEDO BAJA	BLOQUEADO BAJA	BLOQUEADO BAJA	-
100	BLOQUEADO BAJA	BLOQUEADO BAJA	BLOQUEADO BAJA	EJEC CPU1 BAJA
140	BLOQUEADO ALTA	BLOQUEADO BAJA	BLOQUEADO BAJA	EJEC CPU1 BAJA
150	BLOQUEADO ALTA	BLOQUEADO BAJA	EJEC CPU2 BAJA	EJEC CPU1 BAJA
160	EJEC CPU2 ALTA	BLOQUEADO BAJA	LISTO BAJA	EJEC CPU1 BAJA
170	EJEC CPU2 ALTA	BLOQUEADO BAJA	EJEC CPU1 BAJA	BLOQUEADO BAJA
180	EJEC CPU2 ALTA	BLOQUEADO ALTA	EJEC CPU1 BAJA	BLOQUEADO BAJA
200	EJEC	EJEC	LISTO	BLOQUEADO

	CPU2 ALTA	CPU1 ALTA	BAJA	BAJA
240	EJEC CPU2 ALTA	EJEC CPU1 ALTA	LISTO BAJA	LISTO BAJA
270	EJEC CPU2 ALTA	EJEC CPU1 ALTA	LISTO BAJA	LISTO ALTA
300	EJEC CPU2 ALTA	EJEC CPU1 ALTA	LISTO BAJA	LISTO ALTA
310	TERMINADO	EJEC CPU1 ALTA	EJEC CPU2 ALTA	LISTO ALTA
350	TERMINADO	TERMINADO	EJEC CPU2 ALTA	EJEC CPU1 ALTA
370	TERMINADO	TERMINADO	TERMINADO	EJEC CPU1 ALTA
450	TERMINADO	TERMINADO	TERMINADO	TERMINADO

3.

Utilización de CPU (CPU utilization): Es el porcentaje de uso (en cuanto a ejecución de tareas de usuario o del sistema que son consideradas útiles) que tiene un procesador.

Calculo de la métrica para la primera planificación.
Total tiempo: 440 ms. Tiempo de uso de CPU 1 = 380 ms.
Utilización CPU 1 = 86,3%
Total tiempo: 440 ms. Tiempo de uso de CPU 2 = 340 ms .
Utilización CPU 2 = 77,2%
Tiempo total de ambas CPUs 880 y tiempo de uso 720 ms.
Utilización de ambas CPUs = 81,8%

Calculo de la métrica para la segunda planificación
Total tiempo: 450 ms. Tiempo de uso de CPU 1 = 430 ms.
Utilización CPU 1 = 95,5%
Total tiempo: 450 ms. Tiempo de uso de CPU 2 = 290 ms.
Utilización CPU 2 = 64,4%
Tiempo total de ambas CPUs 900 y tiempo de uso 720 ms.
Utilización de ambas CPUs = 80,0%

Problema 3 (7 puntos) (2,5)

1) Incluyendo el proceso inicial, ¿cuantos procesos genera este programa? Justifique

```
int main() {
      fork();
      fork();
      fork();
      return 0;
}
```

2)
Sea el siguiente programa

```
program Problema3()

var: G : dato;

    procedure procesar_datos()
    var aux: Integer;
    begin
        aux = 0;
        while aux >= 0
            aux := rand();
            actualizar_dato(G, aux);
        end
    end;

    procedure maestro()
    begin
        G := inicilizarDatos();
        procesar_datos();
        print(G);
    end;

    procedute esclavo()
    begin
        procesar_datos();
    end;

begin
    cobegin
        maestro();
        esclavo();
        esclavo();
        esclavo();
    coend;
end;
```

Se quiere que el proceso maestro inicialice G previo al procesamiento y despliegue el valor de G luego que finalizó completamente el procesamiento de dicha variable.

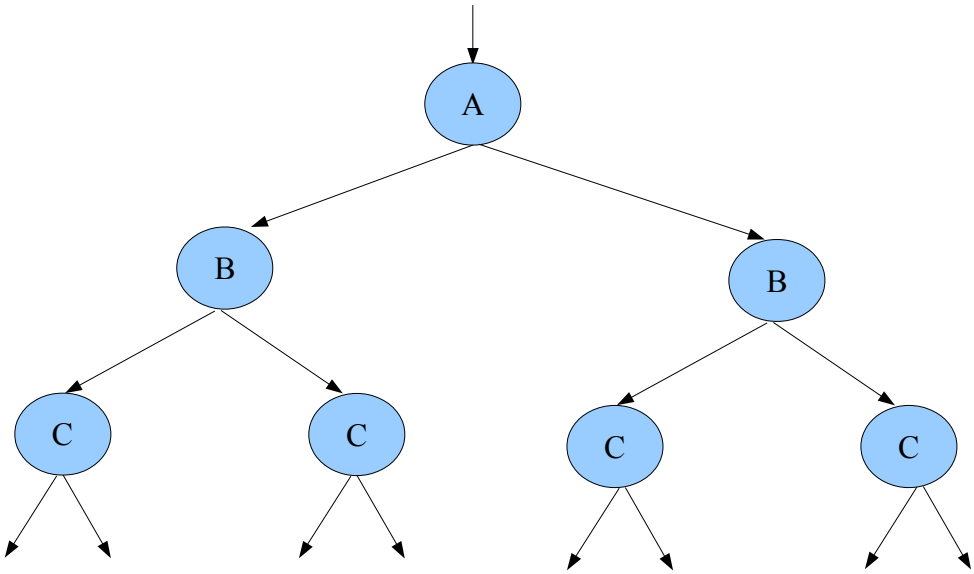
Se pide:

Corregir el programa usando semáforos para que todos los procesos procesen el dato global G de forma concurrente y segura, y que el proceso maestro imprima el valor de G una vez que finalizó completamente el procesamiento.

Solución Problema 3:

1)

El programa genera en total 8 procesos. Cada llamada a fork() crea un nuevo proceso y ambos (padre e hijo) prosiguen la ejecución a partir de la sentencia que creó el nuevo proceso. Si etiquetamos cada fork() del programa como A, B, C tenemos el siguiente grafo donde las líneas indican un proceso ejecutando y los nodos indican el fork que ejecutan:



2)

Se indica en **negrita** el código agregado. Se utilizan 2 semáforos inicializados en 0; uno para que los "esclavos" esperen a que los datos estén inicializados y otro para que el "maestro" espere a que los esclavos terminen su procesamiento. Se utiliza un tercer semáforo (inicializado en 1) para definir una región crítica al actualizar_dato.

```
program Problema3()

var: G : dato;
var: S1, S2, S3 : Semaphore;

    procedure procesar_datos()
    var aux: Integer;
    begin
        aux = 0;
        while aux >= 0
            aux := rand();
            P(S3);
            actualizar_dato(G, aux);
            V(S3);
        end
    end;
```

```
    procedure maestro()
    begin
        G := inicilizarDatos();
        V(S1); // Aviso a los esclavos
        V(S1); // que pueden iniciar
        V(S1); // su procesamiento
        procesar_datos();
        P(S2); // Espero a que los tres
        P(S2); // esclavos terminen
        P(S2); // su procesamiento
        print(G);
    end;

    procedute esclavo()
    begin
        P(S1); // Espero que el maestro habilite el procesamiento
        procesar_datos();
        V(S2); // Aviso al maestro que termine de procesar
    end;

begin
    init(S1, 0);
    init(S2, 0);
    init(S3, 1);
    cobegin
        maestro();
        esclavo();
        esclavo();
        esclavo();
    coend;
end;
```