Primer Parcial Mayo 2018

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) (3,3,3,3,3)

1. ¿Qué garantiza el algoritmo de Dekker? Describa al menos una ventaja y una desventaja de este algoritmo.
2. Describa brevemente el propósito de la estructura PCB. Describa 4 campos de esta estructura que usted considere los más importantes.
3. Un proceso se encuentra en estado "ready". Un tiempo después se encuentra en estado "waiting". Por cuales estados pudo haber pasado y qué posibles eventos pueden haber causado dicho cambio?
4. ¿Qué significa el término "espera activa" (busy waiting)? ¿Existe otro tipo de espera en un Sistema Operativo? En caso afirmativo, descríbalo brevemente.
5. Considere los siguientes componentes de un proceso: stack, heap, registros del CPU y variables globales. Cuáles de estos componentes son compartidos entre:

   a) Un proceso padre y su hijo luego de utilizar la operación fork()

   b) Todos los threads de un mismo proceso

Problema 2 (15 puntos) (3,6,3,3)

Se tiene un sistema operativo multiprogramado en el cual se dispone de un solo procesador. El planificador del sistema utiliza una estrategia de planificación expropiativa por prioridad, siendo 0 la máxima prioridad y 10 la mínima. En caso de procesos con la misma prioridad se utiliza la estrategia FIFO. Considerando que la cantidad de procesos en el sistema es ***n*,** la prioridad para cada proceso varía cada 5ms de la siguiente manera:

- Si está ejecutando, disminuye la prioridad aumentando su valor en *n-1.*
- Si está en cola de listo, aumenta la prioridad disminuyendo su valor en 1.
- Todos los procesos de usuario inician con prioridad 5.

Existen tres programas que tienen el siguiente comportamiento:

Programa A	**Programa B**	**Programa C**
Ejecuta 10 ms I/O por 5 ms Ejecuta 5ms V(s) Ejecuta 10 ms P(c) V(s)	Ejecuta 10 ms P(s) Ejecuta 5ms	Ejecuta 5ms I/O 5 ms Ejecuta 5ms V(c)

En el instante de tiempo 0 (t0) se crean los procesos:

- P1 el cual es una instancia del Programa A
- P2 y P3 que son instancias del Programa B

En el instante de tiempo 20 (t20) se crea el proceso:

- P4 el cual es una instancia del Programa C

Todos los semáforos fueron previamente inicializados con valor 0 y los tiempos de ejecución de las operaciones sobre ellos son despreciables.

Se pide:

a. Realice un diagrama que muestre los estados y transiciones que tiene un proceso en un sistema operativo con las características presentadas. Describa brevemente cada componente y sus transiciones.

b. Realice un esquema en el tiempo que muestre qué procesos están asignados al procesador, cola de procesos listos (*ready queue*) y bloqueados.

c. Defina y calcule la utilización de la CPU, tiempos de espera y tiempos de retornos para la planificación de la **parte b**.

d. Dados los algoritmos de planificación vistos en el curso, a cuál de ellos es similar la planificación de la **parte b**. ¿Porqué?

Solución:

Parte a. Ver teórico.

Parte b. Aclaraciones:

- Cuando ingresa P4 influye en el calculo de probabilidades a partir del ms 25.
- En caso de empate en la cola de listo, se encola en orden ascendente por número de proceso.
- En esta solución se asume que las CPU no es expropiada antes de realizar una IO.

Nota: Estas aclaraciones son arbitrarias, hay otras soluciones igual de validas.

tiempo(ms)	0	5	10	15	20	25	30
P1	E5	L7,2	L6,1	L5,1	E4	B7,IO	L7,1
P2	L5,1	E4	L6,2	L5,2	L4,1	E3	B6,P(S),2
P3	L5,2	L4,1	E3	E5,P(S)	B7,P(S)	B7,P(S)	B7,P(S),1
P4	-	-	-	-	L5,2	L4,1	E3

35	40	45	50	55	60	65
E6,V(S)	L9,2	L8,1	E7	E7,P(C),V(S)	T	T
B6,P(S),2	B6,P(S),2	B6,P(S)	B6,P(S)	B6,P(S)	E6	T
B7,P(S),1	L7,1	E6	T	T	T	T
B6,IO	E6,V(C)	T	T	T	T	T

Parte c. Utilización del CPU: La CPU no estuvo ociosa en ningún momento, por lo tanto su utilización fue del 100%.

Tiempos de espera:

P1: 20-5 + 35-30 + 50-40 = 30ms
P2: 5-0 + 25-10 = 20ms
P3: 10-0 + 45-40 = 15ms
P4: 30-20 = 10ms

Tiempos de retorno:

P1: 60-0 = 60ms
P2: 65-0 = 65ms
P3: 50-0 = 50ms
P4: 45-20= 25ms

Parte d. Se comporta como un planificador RoundRobin ya que el cpu va alternando entre todos los procesos a un intervalo fijo de tiempo, como si existiera un quantum.

Problema 3 (7 puntos) (2,5)

a) Describa las ventajas del uso de monitores frente a semáforos.

b) El siguiente programa intenta resolver el problema de productor-consumidor con buffer finito. Indique si el siguiente programa es correcto o no, y en caso de no serlo indique sus errores y especifique como podría corregirlos.

int cantidad;

semaphore sem_productor, sem_consumidor;

```
procedure productor()
{
   while (true)
   {
      elem = producir();
      if (cantidad == TAM_BUFFER)
      {
         P(sem_productor);
      }
      guardarEnBuffer(elem);
      cantidad = cantidad + 1;
      if (cantidad == 1)
      {
         V(sem_consumidor);
      }
   }
}
```

```
procedure consumidor()
{
   while (true)
   {
      if (cantidad == 0)
      {
         P(sem_consumidor);
      }
      elem = sacarDelBuffer();
      cantidad = cantidad - 1;
      if (cantidad == BUFFER_SIZE - 1)
      {
         V(sem_productor);
      }
      consumir(elem);
   }
}
```

```
main {
        init(sem_productor, 0);
        init(sem_consumidor, 1);
        cantidad = 0;
        cobegin
                productor(); productor();
                consumidor(); consumidor();
        coend
}
```

Nota: todas las operaciones son atómicas salvo producir(), consumir(), guardarEnBuffer() y sacarDelBuffer()

Solución:

Parte a)

1. Son una herramienta de más alto nivel, específicamente las estructuras de datos del monitor son accedidas únicamente mediante los procedimientos que éste ofrece. Entonces, la solución de los problemas implica una visión más abstracta del problema y, por lo tanto, una mayor facilidad y comprensión de la misma.

2. Los monitores permiten no utilizar variables globales y los procedimientos, que permiten acceder a las estructuras de datos, garantizan por definición la mutuo-exclusión.

Parte b) El progama no es correcto!

Los errores principales:

1) No se mutuo-excluye el acceso al buffer.

2) No se mutuo-excluye el acceso a la variable cantidad, puede llevar a acceder al buffer sin datos o escribir con el buffer lleno.

3) El semáforo sem_consumidor está mal inicializado.

Errores menores:

4) El problema se define con un sólo productor y un sólo consumidor. Esta formulación genera que al despertar con cantidad = 1 y con cantidad = TAM_BUFFER - 1 solo se despierta un proceso y posiblemente hay dos.

5) Las constantes TAM_BUFFER y BUFFER_SIZE deberían ser la misma.

Código corregido:

semaphore sem_zona, sem_consumidor, sem_productor;

```
procedure productor() {
   void* elem;
   while (true)  {
      elem = producir();
      P(sem_productor);
      P(sem_zona);
      guardarEnBuffer(elem);
      V(sem_zona);
      V(sem_consumidor);
   }
}
```

```
procedure consumidor() {
   void* elem1;
   while (true)  {
      P(sem_consumidor);
      P(sem_zona);
      elem1 = sacarDelBuffer();
      V(sem_zona);
      V(sem_productor);
      consumir(elem1);
   }
}
```

```
main {
        init(sem_zona, 1);
        init(sem_productor, TAM_BUFFER);
        init(sem_consumidor, 0);
        cobegin
                productor(); productor();
                consumidor(); consumidor();
        coend
}
```