Primer Parcial Mayo 2015

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.**

Problema 1 (15 puntos)

1.
   a) Describa dos maneras de cargar un manejador de dispositivo (*device driver).*
   b) ¿Cual es el beneficio de usar spooling?
2. Describa las tareas y servicios que debe proveer el SO para la administración de procesos.
3. Describa la semántica/funcionamiento de la siguiente sentencia de un programa en ADA:

```
Select
   When A > 15 =>
    accept ENTRADA1 (a:in integer) do
          v:= a
    end ENTRADA1
    v:= f(v)
OR
   When B > 0
   accept ENTRADA2;
   v:= v*2
End Select
```

4. Indique las ventajas y desventajas del uso de máquinas virtuales.
5. Indique los pasos que deben realizarse efectuar un llamado al sistema (*system call*), indicando quién realiza cada paso.

SOLUCIÓN:

A continuación se presentan respuestas mínimas requeridas a cada pregunta.

1 a) *(elegir 2)*
- Especificado en un archivo de configuración y cargado al iniciar el sistema
- Cargado por demanda en el momento que se conecta un nuevo dispositivo
- Enlazados estáticamente al núcleo del sistema

1 b)
Aumenta el uso de la CPU dado que la misma no debe esperar por el procesamiento de dispositivos lentos. En lugar de esperar a que terminen, se escriben los requerimientos en el spooler y la CPU queda libre para hacer otras tareas. Cuando el dispositivo queda libre lee el siguiente trabajo a realizar desde el spooler.

2)
- Creación y destrucción de procesos
- Suspensión y reanudación de procesos
- Comunicación entre procesos
- Sincronización entre procesos

3)
Al inicio se evalúan las dos guardas una sola vez. Las entradas que evalúan la guarda como verdadera quedan habilitadas, las otras es como si no existieran. Si las dos guardas dan falsas se produce un error.
Si hay solicitudes previas para alguna de las entradas abiertas se produce el encuentro. En caso de haber solicitudes para las dos entradas (abiertas) se atiende una entrada al azar. Para cada entrada se forma una cola de atención FIFO. Si no había solicitudes previas se bloquea hasta que aparezca una solicitud para alguna de las entradas habilitadas.
Una vez que se produjo el encuentro el proceso que lo llama se bloquea hasta que termine de ejecutarse el cuerpo del accept (para ENTRADA1) y luego se ejecuta el comando que sigue al accept (*v:= f(v)* para ENTRADA1 y *v:= v*2* para ENTRADA2).

4)
Ventajas *(se acepta que no mencionen alguna)*:
- Los procesos en cada máquina virtual son completamente independientes de los procesos en las otras máquinas virtuales *(por lo que se mejora la seguridad del sistema)*.
- En caso de falla del hardware se puede levantar otra máquina virtual rápidamente en otro hardware *(con lo cual se mejora la disponibilidad del sistema)*.
- Permite ejecutar un sistema operativo de test sin correr riesgos con el sistema real.

Desventajas:
- Las operaciones tardan más tiempo que en una máquina real *(puesto que hay que traducir algunas operaciones)*
- Los tiempos de respuesta son muy poco predecibles

5)
i) El proceso de usuario guarda los parámetros del system call y el número de system call a invocar donde corresponda
ii) El proceso de usuario invoca a una interrupción por software
iii) El hardware cambia la CPU a modo monitor e invoca al manejador de la interrupción correspondiente (cuyo código es parte del SO)
iv) El SO procesa la system call y guarda el resultado donde corresponda
v) El planificador del SO selecciona el próximo proceso a ejecutar
vi) Se pasa el CPU a modo usuario y se pasa el control al nuevo proceso

Problema 2 (16 puntos) (3,3,8)

Dado un sistema multiprogramado con un único procesador, considerando un conjunto arbitrario de 3 procesos que arriban simultáneamente al sistema. Se desean evaluar los siguientes algoritmos de planificación: First Come First Served (FCFS), Shortest Job First (SJF), Prioridad, Round Robin (RR).

Suponga que el tiempo de cambio de contexto es despreciable.

1) Si los procesos no realizan bloqueos. ¿Es posible determinar cuáles de los algoritmos de planificación presentan el menor tiempo total de retorno?. Justifique su respuesta y en caso de que no sea posible, ejemplifique utilizando diagramas de planificación.

2) Suponga ahora que los procesos realizan una determinada cantidad de bloqueos. En este nuevo caso, ¿es posible determinar cuáles de los algoritmos de planificación presentan el menor tiempo total de retorno?. Justifique su respuesta y en caso de que no sea posible, ejemplifique utilizando diagramas de planificación.

3) Dados los siguientes programas:

Programa 1	Programa 2	Programa 3
init_sem(s,0) init_sem(t,0) Ejecuta durante 20ms. p(t) p(t) Termina.	Ejecuta durante 30ms. p(s) Ejecuta durante 10ms. Se bloquea durante 5ms. Ejecuta durante 15ms. v(t) Termina.	Ejecuta durante 15ms. Se bloquea durante 80ms. v(s) Ejecuta durante 15ms. v(t) Termina.

El proceso 1 llega en el tiempo 0ms, el proceso 2 en el tiempo 15ms, y el proceso 3 en el tiempo 25ms. Asuma que el tiempo de ejecución de la instrucción init_sem es despreciable. El tiempo de ejecución para P(sem) y V(sem), siendo sem un semáforo, es de 5ms cada una.

Para los algoritmos de SJF y RR con un quantum de 20ms:

a) Realice los diagramas de planificación (tiempo vs procesos), en que se indique el estado de cada proceso (listo/ejecutando/bloqueado/terminado), en cada intervalo de tiempo.

b) Calcule el tiempo de espera total de los procesos en el sistema.

c) Calcule la utilización de la CPU.

SOLUCIÓN:

1) Siendo el tiempo total de retorno el tiempo que transcurre desde el momento que entran los procesos al sistema hasta el momento que termina de ejecutar el último proceso. Debido a que no se producen bloqueos y a que el cambio de contexto es despreciable, todos los algoritmos de planificación presentan un 100% de uso de CPU y tienen igual tiempo total de retorno (igual a la suma de los tiempos de ejecución de todos los procesos). Por lo tanto; no importa en qué orden sean ejecutados los procesos, o si lo hacen de forma continua; todos los algoritmos de planificación presentan el menor tiempo total de retorno.

2) No es posible determinar cuál es el algoritmo de planificación que presenta el menor tiempo total de retorno. Sin tener conocimiento del patrón de bloqueo de los procesos no se puede asegurar que un algoritmo será capaz de alcanzar el menor tiempo total de retorno para todos los casos. Vamos a presentar un contra ejemplo: 2 casos en los cuáles el algoritmo que optimiza el tiempo de retorno no es el mismo.

Para el primer ejemplo vamos a considerar 2 procesos tal que P2 arriba primero que P1, P2 tiene más prioridad que P1 y el quantum de RR es de 30ms.

P1:	P2:
Ejecuta 5ms Se bloquea 20ms Ejecuta 10ms	Ejecuta 20ms Se bloquea 10ms Ejecuta 20ms

Según SJF:

P1	E	B	E	T	T
P2	L	E	B	E	T
	0	***5***	***25***	***35***	***55***

Según FCFS, Prioridad y RR:

P1	L	E	B	B	L	E	T
P2	E	B	B	E	E	T	T
	0	***20***	***25***	***30***	***45***	***50***	***60***

Vemos que para este caso el menor tiempo de retorno lo tiene SJF. Para el segundo caso nos consideramos un proceso de cada uno de los siguientes programas y que P1 arriba antes que P2.

P1:	P2:
Ejecuta 20ms Se bloquea 30ms Ejecuta 10ms	Ejecuta 10ms Se bloquea 10ms Ejecuta 10ms

Según SJF:

P1	L	E	E	B	B	E	T
P2	E	B	L	E	T	T	T
	0	***10***	***20***	***30***	***40***	***60***	***70***

Según FCFS:

P1	E	B	B	B	E	T	
P2	L	E	B	E	T	T	
	0	***20***	***30***	***40***	***50***	***60***	

Vemos que para este caso el menor tiempo de retorno lo tiene FCFS. Podemos concluir con este caso que no se puede determinar de forma general cuáles algoritmos tienen un tiempo de retorno menor.

3)

a) Diagrama de planificación utilizando SJF

P1	E	E	B	B	B	B	B	B	B	E	B	T
P2	-	L	L	E	B	B	E	B	L	L	E	T
P3	-	-	E	B	B	E	L	E	E	T		
	0	**15**	**25**	**40**	**75**	**120**	**125**	**135**	**140**	**155**	**160**	**180**

Diagrama de planificación utilizando RR con quantum 20

P1	E	E	L1	L1	E	B	B	B	B	B	B	B	E	B	T
P2	-	L1	E	E	L2	L1	E	B	B	L1	E	B	L1	E	T
P3	-	-	-	L2	L1	E	B	B	E	E	L1	E	T	T	T
	0	**15**	**20**	**25**	**40**	**45**	**60**	**75**	**140**	**145**	**160**	**170**	**175**	**180**	**200**

b) Espera total con la planificación SJF

El proceso 1 no está en ningún momento en la lista de espera. Tiempo de espera 0ms.
El proceso 2 está en dos intervalos en la lista de espera: (15-40), (140-160). Tiempo de espera 45ms.
El proceso 3 está sólo un intervalo en la lista de espera: (125-135). Tiempo de espera 10ms.
Tiempo total de espera = 55ms

Espera total con la planificación RR

El proceso 1 está sólo un intervalo en la lista de espera: (20-40). Tiempo de espera 20ms
El proceso 2 está en cuatro intervalos en la lista de espera: (15-20), (20-40), (145-160), (175-180). Tiempo de espera 45ms
El proceso 3 está en dos intervalos en la lista de espera: (25-45), (160-170). Tiempo de espera 30ms
Tiempo total de espera = 95ms

c) El tiempo total de ejecución de los 3 programas para ambos casos es de 135ms (suma del tiempo de cómputo de los tres programas).
Uso de CPU con la planificación SJF es 135/180 * 100 = 75%
Uso de CPU con la planificación RR es 135/200 * 100 = 67.5%

Es posible considerar una segunda versión para la planificación del algoritmo de RR. En este segundo caso la planificación queda como sigue.

a)

P1	E	E	L1	L1	E	B	B	B	B	B	B	B	L1	E	T
P2	-	L1	E	E	L2	L1	E	B	B	L1	E	B	E	T	T
P3	-	-	-	L2	L1	E	B	B	E	E	L1	E	T	T	T
	0	**15**	**20**	**25**	**40**	**45**	**60**	**75**	**140**	**145**	**160**	**170**	**175**	**195**	**200**

b)

Tiempo de espera de p1 = 20 + 20 = 40ms
Tiempo de espera de p2 = 5 + 20 + 15 = 40ms
Tiempo de espera de p3 = 20 + 10 = 30ms
Tiempo total de espera = 110ms

Para la corrección también se consideraron como correctas respuestas que definían como tiempo total de retorno como la suma de los tiempos de retorno de cada proceso. En este caso el ejercicio 1) queda como sigue.

1) Siendo el tiempo total de retorno la suma del tiempo de retorno de cada uno de los procesos. En este caso, y dadas las condiciones de cambio de contexto sin costo y sin bloqueos, SJF computará siempre el menor tiempo total de retorno. Al ejecutar primero los procesos con mejor tiempo de ejecución, reducirá al mínimo la contribución de los tiempos de retorno de estos trabajos a la sumatoria del total.

Problema 3 (7 puntos) (4,5)

1. Se desea implementar el siguiente programa en forma concurrente de forma de maximizar la concurrencia.

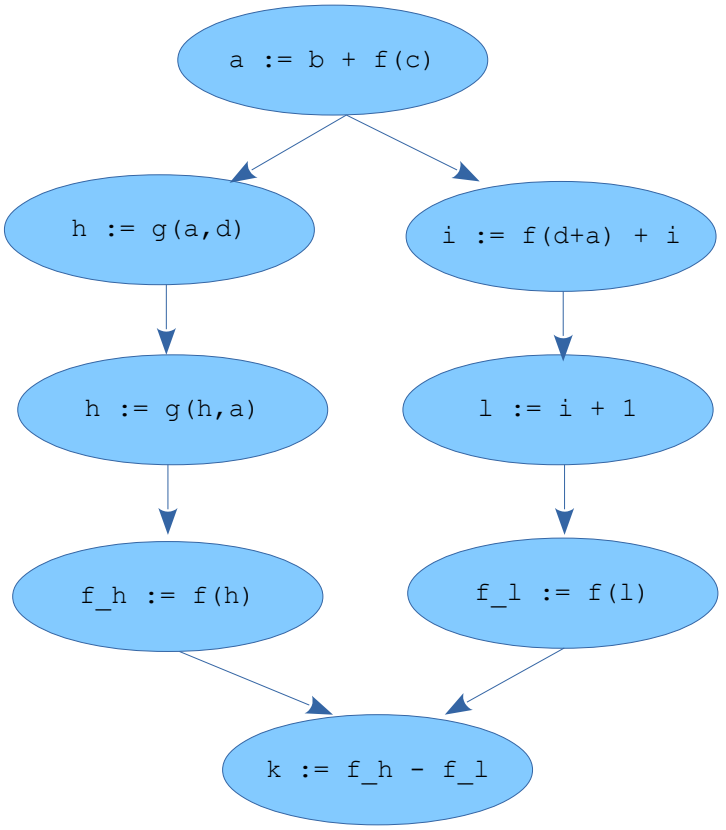
   Se pide: Dibujar el grafo de precedencia del programa concurrente e implementarlo usando cobegin-coend y semáforos usando la mínima cantidad de semáforos posible.

```
Procedure parte1
begin
      a := b + f(c)
      h := g(a,d)
      i := f(d + a) + i
      l := i + 1
      h := g(h,a)
      k := f(h) – f(l)
end
```

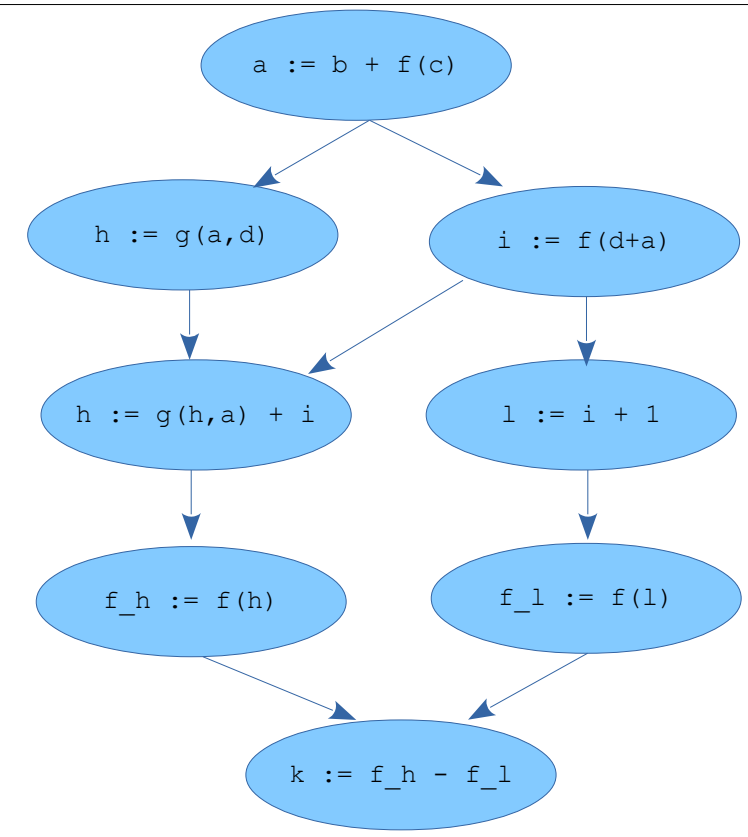
2. Ídem anterior con el programa modificado de la siguiente forma:

```
Procedure parte2
begin
      a := b + f(c)
      h := g(a,d)
      i := f(d + a)
      l := i + 1
      h := g(h,a) + i
      k := f(h) – f(l)
end
```

SOLUCIÓN:




```
Procedure parte1
begin
  a := b + f(c)
  cobegin
    begin
      h := g(a,d)
      h := g(h,a)
      f_h := f(h)
    end
    begin
      i := f(d+a) + i
      l := i + 1
      f_l := f(l)
    end
  coend
  k := f_h - f_l
end
```




```
semaphore S;
Procedure parte2
begin
  init(S, 0)
  a := b + f(c)
  cobegin
    begin
      h := g(a,d)
      P(S)
      h := g(h,a) + i
      f_h := f(h)
    end
    begin
      i := f(d+1)
      V(S)
      l := i + 1
      f_p := f(l)
    end
  coend
  k := f_h - f_l
end
```