Universidad de Las Palmas de Gran Canaria
Escuela Universitaria de Informática

Calificación
1
2
3
4
5
6

Sistemas Operativos
Convocatoria de junio, año 2002
21 de junio de 2002

SOLUCIONES

Nombre	Titulación

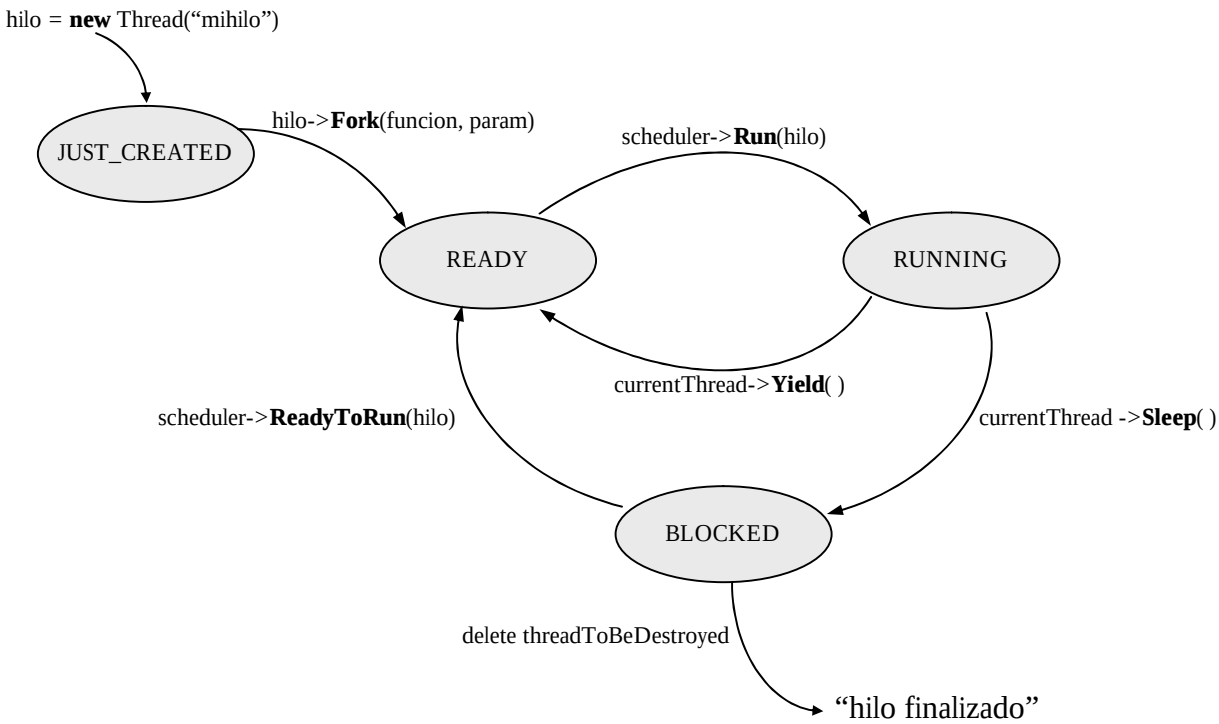
IMPORTANTE: Las seis preguntas suman 12 puntos. Usted deberá descartar al menos UNA de las preguntas 3, 4, 5 y 6. Es decir, usted puede optar por contestar estos grupos de preguntas: 12345, 12346, 12356, 12456. No se admitirá ninguna otra combinación.

Dispone de tres horas y media para completar el examen.

---

1 (2 puntos) Describa mediante un esquema todos los estados por los que atraviesa un hilo en Nachos y las transiciones entre ellos, indicando cuál es el tipo de evento que produce cada transición.

Diagrama de estados



1. El primer estado por el que pasa un hilo es el de "recién creado" (JUST_CREATED):

hilo = **new** Thread(“mihilo”);

En este punto, al hilo aún no se le ha creado una pila ni se sabe el código que va a ejecutar.

2. En el momento en que se realiza la operación:

   hilo->**Fork**(*funcion*, *param*)

El hilo pasa a la cola de preparados (estado “READY”), pendiente únicamente de que se le asigne la CPU para su ejecución (ya se le ha creado la pila y se ha indicado el código que ejecutará - *funcion*).

3. El hilo creado (en adelante “mihilo”) pasará a estado “RUNNING”, es decir, “en ejecución” cuando el planificador le ceda la CPU:

   scheduler->**Run**(*hilo*)

situación que puede venir provocada bien porque otro hilo ceda voluntariamente la CPU (recuerden que la política de planificación de hilos del Nachos es no expulsiva) o bien porque otro hilo en ejecución termine o quede bloqueado a la espera de algún evento, situaciones ambas en las que el planificador toma el siguiente de la cola de preparados. En ambos casos “mihilo” debe ser evidentemente el primero de la cola de preparados.

4. Estando en ejecución (RUNNING), un hilo puede pasar a diferentes estados:

   - “READY”: un hilo estando en ejecución pasará a la cola de preparados si voluntariamente cede la CPU (currentThread->Yield) y siempre que existan otros hilos en la cola de preparados, ya que de lo contrario permanecerá en el mismo estado, es decir, en ejecución.
   - “BLOCKED”: se pasará a estado “bloqueado” cuando se invoca:

     currentThread->**Sleep**()

   Esto se puede deber a dos razones: la primera es que el hilo necesite esperar a que ocurra algún evento o condición (por ejemplo, pasará a este estado un hilo que ejecute una operación *wait* sobre una variable condición). La segunda situación por la que un hilo pasa a estado bloqueado se debe a la forma en que un hilo finaliza. En Nachos, la terminación de un hilo provoca que este pase a estado bloqueado, de forma que sea el siguiente hilo que entre en ejecución el encargado de liberar la memoria del hilo finalizado (una variable global, threadToBeDestroyed, apunta al hilo finalizado).

5. Por último señalar que para pasar del estado “bloqueado” al estado “preparado” se debe producir el evento o condición de desbloqueo (para el caso comentado en el que el hilo quedó bloqueado al realizar una operación *wait* sobre una variable condición, el desbloqueo y consecuentemente el paso a estado preparado se producirá cuando otro hilo ejecute una operación *signal* o *broadcast* sobre dicha variable condición)

Nombre

2 (2 puntos) Implemente las operaciones de un semáforo general utilizando un monitor.

```
type Semaforo (Valor_Inicial) = monitor
  var
    bloqueado: condition;
    s: entero;

  procedure entry P()
  begin
      while s=0 then bloqueado.Wait;
      s:=s-1;
  end;

  procedure entry V()
  begin
      s:=s+1;
      bloqueado.Signal;
  end;

  begin
    s := Valor_Inicial;
  end;

end.
```

3 (2 puntos) La tabla inferior muestra una carga de procesos. Represente con un diagrama de Gantt la planificación de estos procesos y obtenga los tiempos de retorno y de espera al aplicar las siguientes políticas:

- Primero el más corto con expropiación
- Round-Robin con cuanto de 2 u.t.

TL: tiempo de llegada
D: duración

Proceso	TL	D
P0	0	4
P1	1	1
P2	1	2
P3	2	2

Primero el más corto (SJF) con expropiación

P0	P1	P2	P2	P3	P3	P0	P0	P0

Proceso	Tiempo de espera	Tiempo de retorno
P0	5	9
P1	0	1
P2	1	3
P3	2	4
Media	**2**	**4,25**

Round-Robin (Q = 2 u.t.): admite dos posibilidades (en realidad cuatro, puesto que al llegar P1 y P2 en el mismo instante, es posible colocar primero en la cola cualquiera de los dos)

Solución A:

P0	P0	P1	P2	P2	P3	P3	P0	P0

Proceso	Tiempo de espera	Tiempo de retorno
P0	5	9
P1	1	2
P2	2	4
P3	3	5
Media	**2,75**	**5**

Solución B:

P0	P0	P1	P2	P2	P0	P0	P3	P3

Proceso	Tiempo de espera	Tiempo de retorno
P0	3	7
P1	1	2
P2	2	4
P3	5	7
Media	**2,75**	**5**

Nombre

4 (2 puntos) Considere un sistema de gestión de memoria paginada de un solo nivel, sin memoria virtual. Especifique de forma algorítmica los pasos que se deben ejecutar en la traducción de una dirección lógica a una dirección física. En su especificación deberá explicar el uso de los distintos recursos físicos y lógicos requeridos.

El formato de la dirección contiene dos campos: página (PAG) y desplazamiento en la página (DES). Al no disponer de memoria virtual se asume que todo el espacio direccionable está en memoria.

1. Traducción mediante el hardware especial de traducción (TLB): si la página está representada en dicha tabla, entonces se obtiene la dirección sumando la dirección base del marco de página que la contiene con el desplazamiento.

2. Traducción mediante la tabla de páginas.

2.1. Obtener el comienzo de la tabla de páginas (registro origen de la tabla de páginas)

2.2. Verificar que se trata de una página válida. Esto se comprueba comparando PAG con el número de entradas de la tabla, si PAG lo supera entonces error por dirección inválida.

2.3. Acceder a la entrada PAG de la tabla de páginas y obtener la dirección base del marco de página que la contiene. El resultado de la traducción será la suma de esta dirección base con el campo DES.

5 (2 puntos) Se tiene un sistema con 5 procesos, P0 a P4, y 3 tipos de recursos A, B, y C. El tipo de recurso A tiene 10 ejemplares, B tiene 6 ejemplares y C tiene 2 ejemplares. Suponga que en el instante T0 el sistema se encuentra en el siguiente estado:

	Asignados			Solicitudes		
	A	B	C	A	B	C
P0	0	0	1	0	0	0
P1	2	0	0	2	2	0
P2	3	3	0	0	0	0
P3	2	1	1	1	0	0
P4	0	2	0	0	2	0

¿El sistema se encuentra en interbloqueo? Justifique la respuesta.

El sistema no está en interbloqueo

El sistema no está en un estado de interbloqueo. Para demostrarlo aplicamos el algoritmo de detección visto en clase y vemos que podemos encontrar una secuencia ordenada de procesos <p0,p2,p3,p4,p1> en la que lo que pide cada proceso puede ser satisfecho por lo disponible más lo asignado a los procesos que están antes que él en la secuencia.

6 (2 puntos) Un sistema de archivos utiliza una política de asignación de espacio indexada. En este sistema, los bloques de datos de un archivo, ¿se pueden ubicar de forma contigua? ¿Es *necesario* que los bloques estén ubicados de forma contigua? ¿Da igual que lo estén? Justifique con mucha claridad su respuesta.

¿Cambiaría en algo su contestación si la asignación de espacio fuera enlazada?

La asignación indexada permite que los bloques de datos de un archivo se encuentren dispersos por el disco, pero evidentemente no obliga a ello. Así que los bloques de un archivo indexado pueden estar contiguos.

Sin embargo, no es indiferente el que los bloques estén contiguos. Los accesos secuenciales al archivo serán más eficientes en ese caso, porque el desplazamiento de los cabezales del disco será menor, comparado con el recorrido que tendría que hacer el cabezal si los bloques estuvieran desperdigados por la superficie del disco. La contigüidad ocasiona un menor tiempo de acceso. En el mejor caso, todos los bloques del archivo estarían en un mismo cilindro y la cabeza lectora no tendría que hacer ningún movimiento para leer el archivo completo.

Si la asignación de espacio fuera enlazada, la respuesta sería similar. Incluso tendría más impacto en el rendimiento el hecho de que los bloques estén contiguos, ya que en la asignación enlazada cualquier acceso al archivo, ya sea secuencial o ya sea directo, exige recorrer los enlaces de los bloques. Si los bloques están contiguos, el cabezal del disco hará un recorrido más pequeño.