RELACIÓN DE EJERCICIOS TEMAS 1 Y 2

1. Cuestiones sobre procesos, y asignación de CPU:

- ***¿Es necesario que lo último que haga todo proceso antes de finalizar sea una llamada al sistema para finalizar? ¿Sigue siendo esto cierto en sistemas monoprogramados?***

  Sí, es necesario, puesto que al finalizar un proceso, este debe informar al SO de que su ejecución ha acabado con el fin de que libere el espacio reservado y el PCB de dicho proceso para posteriormente llamar a otro proceso en estado preparado.

  Y en sistemas monoprogramados tambien es necesario puesto que igualmente (y con mas razón) el SO necesita conocer cuándo termina el proceso para liberar su espacio de memoria e introducir un nuevo proceso en espera.

- ***Cuando un proceso se bloquea, ¿deberá encargarse él directamente de cambiar el valor de su estado en el descriptor de proceso o PCB?***

  Cuando un proceso se bloquea, el sistema operativo es el encargado de cambiar su estado en el PCB. Al ser datos del sistema operativo, se debe hacer desde el modo kernel, por eso el proceso no es el que se encarga directamente de esta tarea.

- ***¿Qué debería hacer el planificador a corto plazo cuando es invocado pero no hay ningún proceso en la cola de ejecutables?***

  Este problema es resuelto en muchos sistemas operativos con el proceso NULO que es creado por el sistema en el momento de arranque. El proceso nulo nunca termina, no tiene E/S y tiene la prioridad más baja en el sistema. En consecuencia la cola de listos nunca está vacía, además la ejecución del planificador puede hacerse más rápida al eliminar la necesidad de comprobar si la cola de listos está vacía o no. Algunas de las tareas que se le pueden dar al proceso nulo, por ejemplo, es realizar estadísticas de uso de procesador, o asistencia de vigilancia de la integridad del sistema, etc.

- ***¿Qué algoritmos de planificación quedan descartados para ser utilizados en sistemas de tiempo compartido?***

  Tenemos que usar aquellos algoritmos que favorezcan procesos cortos (ya que estamos en tiempo compartido) generalmente los que no estan basados en quamtum de tiempo. Descartamos FCFS Y el más corto primero no apropiativo.

2. La representación gráfica del cociente [(tiempo_en_cola_ejecutables + tiempo_de_CPU) / tiempo_de_CPU] frente a tiempo_de_CPU

suele mostrar valores muy altos para ráfagas muy cortas en casi todos los algoritmos de asignación de CPU. ¿Por qué?

Los procesos que usan ráfagas cortas son los procesos cortos, y estos pasan mucho tiempo en cola de ejecutables y muy poco en CPU, entonces como el cociente es (tiempo en cola + tiempo de cpu)/tiempo de CPU pues aumenta mucho para tiempos de CPU muy cortos, en especial si tiempo en cola es grande de por sí. A este cociente se le conoce como penalización

3. Para cada una de las llamadas al sistema siguientes, especificar y explicar si su procesamiento por el sistema operativo requiere la invocación del planificador a corto plazo:

1. **Crear un proceso**
   Depende de si el planificador que estamos usando es apropiativo o no. Si es apropiativo el úlitmo paso al crear un proceso es llamar al planificador de corto plazo, ya que en este caso podría ocurrir que el proceso tuviera que entrar a ejecutarse nada más llegar a la cola de preparados. En caso de que el planificador no sea apropiativo, no se llama a este, pues no vamos a tener este problema, no se podría retirar la CPU de un proceso que este ejecutándose.

- **Abortar un proceso, es decir, terminarlo forzosamente**
  Sí se requiere, ya que cuando un proceso sale del procesador el planificador a corto plazo debe decidir qué proceso de la cola de preparados debe entrar a ejecutarse, sea apropiativo o no apropiativo.

- **Suspender o bloquear un proceso**
  El planificador es el que decide cuando suspender o bloquear un proceso. Posteriormente, vuelve a ser necesario para decidir que proceso pasa a ejecutarse.

- **Reanudar un proceso (inversa al caso interior)**
  Al reanudar un proceso, este vuelve a la cola de preparados, luego ocurre lo mismo que en el caso anterior (el primer caso), si es apropiativo habría que llamar al planificador, y en caso de no apropiativo no.

- **Modificar la prioridad de un proceso**
  Lo mismo que en el caso anterior (el primer caso), ya que si estamos en una política apropiativa, podría darse el caso de que el proceso deba pasar a ejecutarse por tener mayor prioridad, luego hay que llamar al planificador. En caso de que no fuera apropiativo no hace falta.

4. Sea un sistema multiprogramado que utiliza el algoritmo Por Turnos (Round Robin). Sea S el tiempo que tarda el despachador

en cada cambio de contexto. ¿Cuál debe ser el valor de quantum Q para que el porcentaje de uso de la CPU por los procesos de usuario sea del 80%?

Tiempo total de CPU = S+Q

```
T = S+Q
0.8T = Q
--> 5Q/4 - Q = S --> Q = 4S
```

5. Sea un sistema multiprogramado que utiliza el algoritmo Por Turnos (Round-Robin). Sea S el tiempo que tarda el despachador en cada cambio de contexto, y N el número de procesos existente.¿Cuál debe ser el valor de quantum Q para que se asegure que cada proceso "ve" la CPU al menos cada T segundos?

Se ponen las soluciones de ambos grupos porque las dos son válidas

S: tiempo que tarda el despachador en cada cambio de contexto

N: número de procesos existente

Q: valor de quantum

T: segundos

Para N > 1:

$T = N \cdot S + (N-1) \cdot Q \Rightarrow Q = \frac{T - N \cdot S}{N-1}$

Esto es así ya que suponemos que la última vez que el proceso ve la CPU es antes de gastar el último quantum.

Forma alternativa de expresar la frase anterior: consideramos el tiempo T desde que un proceso acaba hasta que vuelve a ser llamado

---

S+Q es el tiempo que tarda en hacer el cambio de contexto y en ejecutarlo, esto se repite para cada proceso. Por lo tanto, para que todo los N procesos pasen por el procesador se tienen N cambios de contextos y ejecuciones N(S+Q) y esto tienen que ocurrir en menos de T segundos para que todos los procesos sean vistos N(S+Q) <= T. Despejando Q tenemos Q <= (T - (N * S))/N

6 ¿Tiene sentido mantener ordenada por prioridades la cola de procesos bloqueados? Si lo tuviera, ¿en qué casos sería útil hacerlo?

No, normalmente no tiene sentido. Puede tener sentido en el caso de que haya varios procesos bloqueados con distintas prioridades esperando al mismo evento.

7. ¿Puede el procesador manejar una interrupción mientras está ejecutando un proceso si la política de planificación que utilizamos es no apropiativa?

Una interrupción debería de ser procesada independientemente del algoritmo de planificación que se esté utilizando. Si solo tenemos un procesador no puede estar haciendo las dos cosas a la vez. Si llega una interrupción, se le quitaría la CPU al proceso y se le daría a dicha interrupción. Cuando se despache la interrupción, y volvemos al proceso.

8. Suponga que es responsable de diseñar e implementar un sistema operativo que va a utilizar una política de planificación apropiativa. Suponiendo que tenemos desarrollado el algoritmo de planificación a tal efecto, ¿qué otras partes del sistema operativo habría que modificar para implementar tal sistema? y ¿cuáles serían tales modificaciones?

SO apropiativo: el SO puede apropiarse del procesador cuando lo decida. La planificación apropiativa es útil en los sistemas en los cuales los procesos de alta prioridad requieren una atención rápida.

Para políticas de apropiación basadas en quantum necesitaría un temporizador encargado de llevar los quantum que maneje las interrupciones que cambian de bloqueado a preparado y después llaman al planificador, para tareas no basadas en tiempos de quantum tenemos que cambiar otras partes del sistema operativo que se ven implicadas, pues el despachador no se modifica. Cambia el código de la llamada al sistema de creación de procesos y la rutina de desbloqueo de procesos.

9. En el algoritmo de planificación FCFS, la penalización ((t + to de espera) / t), ¿es creciente, decreciente o constante respecto a t (tiempo de servicio de CPU requerido por un proceso)? Justifique su respuesta.

La constante es decreciente, lo ilustraremos con un ejemplo: Si tenemos 3 procesos, P1,P2 y P3 que tardan todos 10 segundos en ejecutarse, vemos que los tiempos de espera

$$\frac{t + te}{t} = 1 + \frac{te}{t}$$

Pero te siempre va aumentando para cada proceso que se aleje más del primer proceso que se va a ejecuta, lo que implica que $\frac{te}{t}$ va decreciendo al aumentar t, por tanto es decreciente.

10. En la tabla siguiente se describen cinco procesos:**

Proceso	Tiempo de creación	Tiempo de CPU
A	4	1
B	0	5
C	1	4
D	8	3
E	12	2

Si suponemos que tenemos un algoritmo de planificación que utiliza una política FIFO (primero en llegar, primero en ser servido), calcula:**

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A										x					
B	x	x	x	x	x										
C						x	x	x	x						
D											x	x	x		
E														x	x

- Tiempo medio de respuesta: 27/5 = 5'4

A = 5 + 1 = 6

B = 0 + 5 = 5

C = 4 + 4 = 8

D = 2 + 3 = 5

E = 1 + 2 = 3

- Tiempo medio de espera: 12/5 = 2'4 A = 9 - 4 = 5

B = 0

C = 5 - 1 = 4

D = 10 - 8 = 2

E = 13 - 12 = 1

- La penalización, es decir, el cociente entre el tiempo de respuesta y el tiempo de CPU. A = 6/1 = 6

B = 5/5 = 1

C = 8/4 = 2

D = 5/3

E = 3/2 = 1'5

11. Utilizando los valores de la tabla del problema anterior, calcula los tiempos medios de espera y respuesta para los siguientes algoritmos:

1. **Por Turnos con quantum q=1**

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A						X									
B	X		X		X			X			X				
C		X		X			X		X						
D										X		X		X	
E													X		X

Cola:

T0 B, T1 CB, T2 BC, T3 CB, T4 BAC , T5 ACB, T6 CB, T7 BC, T8 CDB, T9 DB, T10 BD, T11 D, T12 ED, T13 DE, T14 E

Tiempo de espera:

MA = 1
MB = 6
MC = 4
MD = 3
ME = 1

M= 15/5 = 3

Tiempo de respuesta:

TA = 1
TB = 0
TC = 0
TD = 1
TE = 0

T = 2/5 = 0,4

- **b) Por Turnos con quantum q=4**

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A									X						
B	X	X	X	X						X					
C					X	X	X	X							
D											X	X	X		
E														X	X

Cola:

T0 B, T1 C, T2 C, T3 C, T4 CAB, T5 AB, T6 AB, T7 AB, T8 ABD, T9 BD, T10 D, T11 , T12 E, T13 E, T14

Tiempo de espera:

MA = 4
MB = 5
MC = 3
MD = 2
ME = 1

M = 15/5 = 3

Tiempo de respuesta:

TA =
TB =
TC =
TD =
TE =

T = ? / 5

- **c) El más corto primero (SJF). Suponga que se estima una ráfaga igual a la real.**

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A						X									
B	X	X	X	X	X										
C							X	X	X	X					
D											X	X	X		
E														X	X

Cola:

T0 B, T1 C, T2 C, T3 C, T4 AC, T5 AC, T6 C, T7 , T8 D, T9 D, T10 D, T11 , T12 E, T13 E, T14

Tiempo de espera:

MA = 1
MB = 0
MC = 5
MD = 2
ME = 1

M = 9/5 = 1.8

Tiempo de respuesta:

TA =
TB =
TC =
TD =
TE =

T = ?/5

12 Calcula el tiempo de espera medio para los procesos de la tabla utilizando el algoritmo: el primero más corto apropiativo (o primero el de tiempo restante menor, SRTF).

0	1	2	3	4	...	8	9	...	13	14	...	19	20	...	26
A	B	A	A	C	...	C	D	...	D	E	...	E	C	...	C

El tiempo medio de espera es el tiempo que el proceso pasa en la cola esperando a volver a ejecutarse.

M(A) = 1

M(B) = 0

M(C) = 12

M(D) = 0

M(E) = 2

Luego M_t = 15/5 = 3

13. Utilizando la tabla del ejercicio anterior,

Proceso	Tiempo de creación	Tiempo de CPU
A	0	3
B	1	1
C	3	12
D	9	5
E	12	5

dibuja el diagrama de ocupación de CPU para el caso de un sistema que utiliza un algoritmo de colas múltiples con realimentación con las

siguientes colas:

Cola	Prioridad	Quantum
1	1	1
2	2	2
3	3	4

y suponiendo que:

- ***Todos los procesos inicialmente entran en la cola de mayor prioridad (menor valor numérico). Cada cola se gestiona mediante la política Por Turnos.***
- ***la política de planificación entre colas es por prioridades no apropiativo.***
- ***un proceso en la cola i pasa a la cola i+1 si consume un quantum completo sin bloquearse.***
- ***cuando un proceso llega a la cola de menor prioridad, permanece en ella hasta que finalice.***

14. Suponga que debe maximizar la eficiencia de un sistema multiusuario y que está recibiendo quejas de muchos usuarios sobre los pobres tiempos de respuesta (o tiempos de vuelta) de sus procesos. Los resultados obtenidos con una herramienta de monitorización del sistema nos muestran que la CPU se utiliza al 99'9% de su tiempo y que los procesadores de E/S están activos sólo un 10% de su tiempo. ¿Cuales pueden ser las razones de estos tiempos de respuesta pobres y por qué?

1. **El quantum en la planificación Round-Robin es muy pequeño.**

   Un cuantum pequeño genera sobrecarga pero no malos tiempos de respuesta.

- **La memoria principal es insuficiente.**

  Si la memoria principal fuese insuficiente los procesadores de dispositivos estarían más ocupados por ejemplo accediendo a disco

- **El sistema operativo tiene que manejar mucha memoria principal por lo que las rutinas de gestión de memoria están consumiendo todos los ciclos de CPU.**

  La complejidad de la gestión de memoria no tiene que ver el tamaño de la misma.

- **La CPU es muy lenta.**

  Si la CPU fuese muy lenta, hubiésemos obtenido un 100% de ocupación.

- **El quantum en la planificación Round-Robin es muy grande.**

  Por último, un cuantum grande si nos da unos tiempo de respuesta malos y es compatible con los valores mostrados por la herramienta de vigilancia, pues en con un quantum grande todos los procesos largos se ejecutarían ocupando todo el tiempo disponible en el procesador, sin dar cabida a los procesos cortos. Por lo tanto la CPU pasa mucho tiempo ocupada con estos procesos mientras que apenas produce respuesta en la E/S.

15. Compare el rendimiento ofrecido al planificar el conjunto de tareas multi-hebras descrito en la tabla y bajo las siguientes configuraciones:

a) Sistema operativo multiprogramado con hebras de usuario. En este sistema se dispone de una biblioteca para la programación con hebras en el espacio de usuario. El algoritmo de planificación de CPU utilizado por el SO es Round-Robin con un quantum de 50 u.t. (unidades de tiempo). El planificador de la biblioteca de hebras reparte el quantum del proceso (tarea) entre las hebras utilizando Round-Robin con un quantum para cada hebra de 10 u.t. Suponga que no existe coste en el cambio de contexto entre hebras ni entre procesos.

b) Sistema operativo multiprogramado con hebras kernel. El SO planifica las hebras usando Round-Robin con un quantum de 10 u.t. Como en el apartado anterior,suponga que no existe coste en la operación de cambio de contexto. Considere además que las operaciones de E/S de un proceso únicamente bloquean a la hebra que las solicita. Suponga en ambos casos que los dos procesos están disponibles y que el planificador entrega la CPU al proceso P1. Para realizar la comparación represente en cada caso el diagrama de ocupación de CPU y calcule el grado de ocupación de la CPU (tiempo CPU ocupada tiempo total).

Proceso	Hebras	Ráfaga de CPU	Tiempo de E/S	Ráfaga de CPU
	Hebra1	20	30	10
P1	Hebra2	30	-	-
	Hebra3	10	-	-
P 2	Hebra1	30	30	10
	Hebra1	40	-	-

Tiempo	10	20	30	40	50	60	70	80	90	100	110	120	130	140
A	P1.1	P1.2	P1.3	P1.1	P2.1	P2.2	P2.1	P2.2	P2.1	P1.2	P1.1	P1.2	P2.2	P2.1
B	P1.1	P1.2	P1.3	P2.1	P2.2	P1.1	P1.2	P2.1	P2.2	P1.2	P2.1	P1.1	P2.2	P2.2

%CPU de A = 1

%CPU de B = 1

16. ¿EL planificador CFS de Linux favorece a los procesos limitados por E/S (cortos) frente a los limitados por CPU (largos)? Explique cómo lo hace.

Sí. Los procesos cuyo tiempo consumido de CPU es más largo tienen un vruntime superior. El vruntime depende del tiempo real que el proceso ha consumido de CPU, su prioridad y su peso. Como el planificador ejecutará los procesos de menor vruntime se verán beneficiados los procesos más cortos (en este caso E/S).

17 ¿Cuál es el problema que se plantea en Linux cuando un proceso no realiza la llamada al sistema wait para cada uno de sus procesos hijos que han terminado su ejecución? ¿Qué efecto puede producir esto en el sistema?

Si no se realiza la llamada wait para cada uno de sus hijos aunque acabe el proceso padre, el proceso hijo se mantiene en estado zombie, ocupando una entrada en la tabla de procesos y su contexto es descargado de memoria, existiendo aunque no se esté ejecutando. Si un proceso no es esperado por su padre, continúa consumiendo recursos (tabla de entrada, memoria, el planificador lo tiene en cuenta,...), provocando la ineficiencia del sistema. No se podrían crear nuevos procesos por estar la memoria ocupada, se eliminarán únicamente al apagar el ordenador

---

Zombie

EXIT_ZOMBIE

El proceso ya no existe pero mantiene la entrada de la tabla de procesos hasta que el padre haga un wait

(EXIT_DEAD)

18 La orden clone sirve tanto para crear un proceso en Linux como una hebra.

a) Escriba los argumentos que debería tener clone para crear un proceso y una hebra.

b) Dibuje las principales estructuras de datos del kernel que reflejan las diferencias entre ambas.

c)

Para crear un proceso:

```
clone(SIGCHILD, 0)
```

Para crear una hebra:

```
clone(..., CLONE_VM|CLONE_FS|CLONE_FILES|CLONE_SIGHAND|CLONE_THREAD, ...)
```

b)

Veamos un dibujo sobre las hebras:

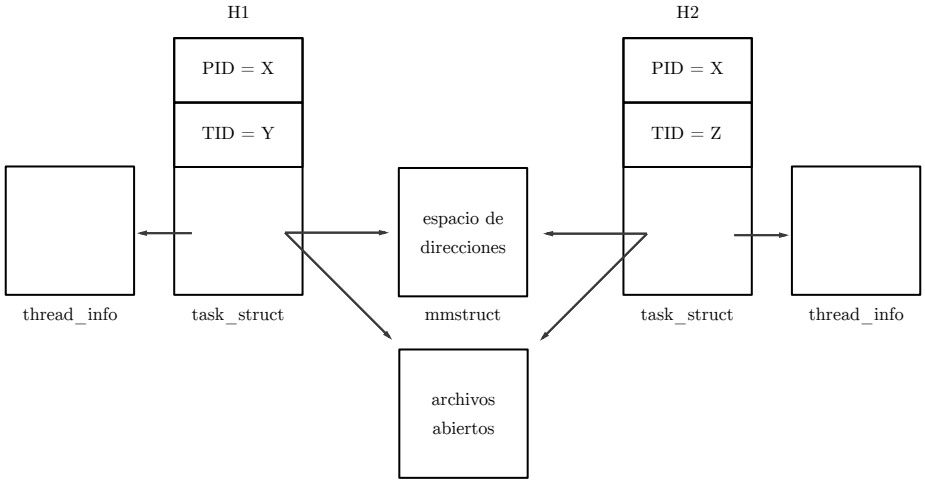

Figure 1: hebras

El dibujo sobre los procesos hijo sería similar, excepto que no se compartiría el espacio de direcciones, ni los archivos abiertos(*), y cada uno tendría su propio PID.

Al comienzo, los archivos abiertos y el espacio de direcciones es el mismo en el hijo que en el padre, pero a partir de ahí cada uno trabaja en su propia copia, sin que los cambios de uno afecten al otro.