

1. Cuestiones generales sobre procesos y asignación de CPU:

- ¿Cuáles son los motivos que pueden llevar a la creación de un proceso?
- ¿Es necesario que lo último que haga todo proceso antes de finalizar sea una llamada al sistema para finalizar de forma explícita, por ejemplo exit()?
- Cuando un proceso pasa a estado “BLOQUEADO”, ¿Quién se encarga de cambiar el valor de su estado en el descriptor de proceso o PCB?
- ¿Qué debería hacer cualquier planificador a corto plazo cuando es invocado pero no hay ningún proceso en la cola de ejecutables?
- ¿Qué algoritmos de planificación quedan descartados para ser utilizados en sistemas de tiempo compartido?

o) Los motivos principales para crear un proceso son:

- Se crea un proceso para ejecutar la petición de un servicio ocasionado por otro proceso.
- En sistemas batch, cuando se admite la ejecución de trabajos.
- Por creación de un proceso padre.
- En logros interactivos, cuando autenticamos la sesión se lanza un proceso con el intérprete correspondiente.

o) Seguro lo visto en clase, realmente la función que se realiza como finalización es la sys.exit(). Sin embargo este no es el único motivo de finalización, pero sí el único de forma explícita.

o) El planificador de CPU

o lo más conveniente sería avisar al planificador a largo plazo o al planificador a medio plazo para que proporcione procesos listos.

o) Resuelto evidentemente eliminar todos aquellos que perjudiquen los procesos cortos, es decir FCFS.

2. Cuestiones sobre el modelo de procesos extendido:

o ¿Qué pasos debe llevar a cabo un SO para poder pasar un proceso de reciente creación de estado "NUEVO" a estado "LISTO"?

o ¿Qué pasos debe llevar a cabo un SO para poder pasar un proceso ejecutándose en CPU a estado "FINALIZADO"?

o Hemos explicado en clase que la función context_switch() realiza siempre dos funcionalidades y que además es necesario que el kernel la llame siempre cuando el proceso en ejecución pasa a estado "FINALIZADO" o "BLOQUEADO". ¿Qué funcionalidades debe realizar y en qué funciones del SO se llama a esta función?

o Indique el motivo de la aparición de los estados "SUSPENDIDO-BLOQUEADO" y "SUSPENDIDO-LISTO" en el modelo de procesos extendido.

ASÍ → Cuando un proceso se bloquea pero sigue ocupando memoria principal impidiendo la ejecución de otros procesos.

o Debe cargar el código del proceso en memoria principal, guardar el PCB del proceso en la lista de listos y cambiar el estado del PCB de nuevo a listo, esto último lo hace el planificador a largo plazo.

o Debe realizar un cambio de contexto con todo lo que eso implica para pasar el proceso de ejecutándose a finalizado. Además, se deberá eliminar el contenido de los registros liberar la memoria ocupada y eliminar el PCB, lo cual es realizado por el kernel.

3. ¿Tiene sentido mantener ordenada por prioridades la cola de procesos bloqueados? Si lo tuviera, ¿en qué casos sería útil hacerlo? Piense en la cola de un planificador de E/S, por ejemplo el de HDD, y en la cola de bloqueados en espera del evento “Fin E/S HDD”.

Si tiene sentido de manera que se trate primero los que más prioridad tienen. Lo lógico es usar, en caso de E/S, algoritmos que beneficien a procesos cortos porque tienen en cuenta el tiempo de espera.

4. Explique las diferentes formas que tiene el kernel de ejecutarse en relación al contexto de un proceso y al modo de ejecución del procesador.

Si pensamos en la ejecución del kernel dentro de un proceso, este se ejecutará para tratar la ejecución de código del sistema operativo como tratar llamadas al sistema como tratamiento de peticiones de E/S o excepciones.

En el caso de ejecutarse fuera de un proceso, esto lo hará para tratar interrupciones o tareas del sistema.

5. Responda a las siguientes cuestiones relacionadas con el concepto de hebra:

- ¿Qué elementos de información es imprescindible que contenga una estructura de datos que permita gestionar hebras en un *kernel* de SO? Describa las estructuras *task_t* y la *thread_t*.
↳ Nosotros hicimos unde
- En una implementación de hebras con una biblioteca de usuario en la cual cada hebra de usuario tiene una correspondencia N:1 con una hebra kernel, ¿Qué ocurre con la tarea si se realiza una llamada al sistema bloqueante, por ejemplo *read()*?
- ¿Qué ocurriría con la llamada al sistema *read()* con respecto a la tarea de la pregunta anterior si la correspondencia entre hebras usuario y hebras kernel fuese 1:1?

o) Necesitaremos describir el TCB:

- Identificador de hebra
 - Contenido de registros
 - Estado de ejecución de la hebra
- { Cada usuario

o) Solo se bloquearán aquellas hebras que tengan asociada la misma hebra kernel, es decir, si una tarea tiene 2 hebras kernel y sus hebras usuario divididas en otras dos, solo se bloqueará un subconjunto de ellas y no todas las de la tarea.

o) Solo se bloquearía dicha hebra y no las demás, en ocasiones es muy útil realizar esta correspondencia; sobre todo en hebras con muchos eventos bloqueantes.

6. ¿Puede el procesador manejar una interrupción mientras está ejecutando un proceso sin hacer `context_switch()` si la política de planificación que utilizamos es no apropiativa? ¿Y si es apropiativa?

Usar una política no apropiativa implica que no se realiza un `context_switch()` hasta que se bloquee o finaliza un proceso por medios propios. En el caso de que la interrupción no sea bloqueante, si quisiera hacer ese manejo. Por ejemplo si otro proceso ha realizado una E/S y se trata de una interrupción, puede que se haga si se produce un `context-switch`.

Esto es más complicado en políticas apropiativas puesto que el tratamiento de la interrupción tiene más prioridad que el proceso ocasionalmente apropiado de la CPU y por consiguiente un `context-switch`.

7. Suponga que es responsable de diseñar e implementar un SO que va a utilizar una política de planificación apropiativa (*preemptive*). Suponiendo que el sistema ya funciona perfectamente con multiprogramación pura y que tenemos implementada la función `Planif_CPU()`, ¿qué otras partes del SO habría que modificar para implementar tal sistema? Escriba el código que habría que incorporar a dichas partes para implementar apropiación (*preemption*).

Habrá que modificar el algoritmo de las colas de estados, si buscamos una planificación RoundRobin es obligatorio cambiar tanto la estructura de colas como la estructura del planificador de CPU.

8. Para cada una de las siguientes llamadas al sistema explique si su procesamiento por parte del SO requiere la invocación del planificador a corto plazo (`Planif_CPU()`):
- Crear un proceso, `fork()`.
 - Abortar un proceso, es decir, terminarlo forzosamente, `abort()`.
 - Bloquear (suspender) un proceso, `read()` o `wait()`.
 - Desbloquear (reanudar) un proceso, `RSI` o `exit()` (complementarias a las del caso anterior).
 - Modificar la prioridad de un proceso.

Hecha de cabeza

9. En el algoritmo de planificación FCFS, el índice de penalización, $(M+r)/r$, ¿es creciente, decreciente o constante respecto a r (ráfaga de CPU: tiempo de servicio de CPU requerido por un proceso)? Justifique su respuesta.

$$\text{Meu FCFS: } \# \text{ processos anteriores} + \frac{u+r}{r} = u+1$$

Si r aumenta, el tiempo de espera será mayor, además proporcionalmente mayor. Aunque dependerá del $\#$ de procesos anteriores \Rightarrow es constante.

10. Sea un sistema multiprogramado que utiliza el algoritmo Por Turnos (Round-Robin, RR). 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%?

$$b \approx s + q$$

$$\text{Car A: } s = 0.2t \Rightarrow t = 5s$$

$$\$ s : s + Q \Rightarrow \boxed{Q = 4s}$$

switch

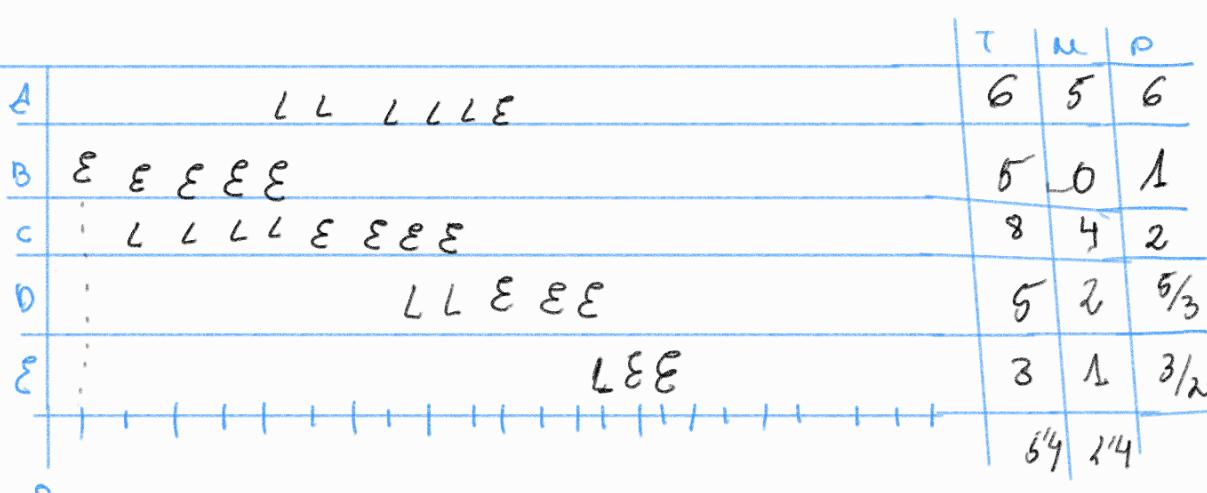
3 $\frac{4}{5}$ csel quantum.

11. Para la siguiente tabla que especifica una determinada configuración de procesos, tiempos de llegada a cola de listos y ráfagas de CPU; responda a las siguientes preguntas y analice los resultados:

Proceso	Tiempo de llegada	Ráfaga CPU	Tiempo ej.
A	4	1	
B	0	5	
C	1	4	
D	8	3	
E	12	2	

- FCFS. Tiempo medio de respuesta, tiempo medio de espera y penalización.
 - SJF (ráfaga estimada coincide con ráfaga real). Tiempo medio de respuesta, tiempo medio de espera y penalización.
 - SRTF (ráfaga estimada coincide con ráfaga real). Tiempo medio de respuesta, tiempo medio de espera y penalización.
 - RR ($q=1$). Tiempo medio de respuesta, tiempo medio de espera y penalización.
 - RR ($q=4$). Tiempo medio de respuesta, tiempo medio de espera y penalización.

\rightarrow greater
 \leftarrow less



o)

		T	u	p
A	L E	2	1	2
B	E E E E E	5	0	1
C	L L L L L E E E E	9	5	9/4
D	L L E E E	5	2	5/3
E	L E E	3	1	3/2
6		4'8	1'8	

o)

	A ↓	T	u	p
A	L E	2	1	2
B	E E E E E	5	0	1
C	L L L L L E E E E	9	5	9/4
D	L L E E E	5	2	5/3
E	L E E	3	1	3/2
6		4'8	1'8	
	↑ D	↑ E		

o) q=1

	T	u	p
A	L E	2	1 2
B	E L E L E L L E L L E	11	6 11/5
C	E L E L E L E	8	4 2
D	L E L E L E	6	3 2
E	E L E	3	1 3/2
6		6	3
	↑ D	↑ E	

o) $q=4$

										T	U	P	
0	A	C	C	C	C	E				5	4	5	
0	B	E	E	E	E	L	L	L	L	E	10	5	2
0	C	L	L	L	E	E	E	E			7	3	$\frac{7}{4}$
0	D					L	L	E	E	E	5	2	$5\frac{1}{3}$
E								L	E	E	3	1	$\frac{3}{2}$
6						1			1		6	3	
							0		E				

DE ->

12. Utilizando los datos de la tabla del ejercicio anterior dibuje 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

Tenga en cuenta las siguientes suposiciones:

- o Todos los procesos inicialmente entran en la cola de mayor prioridad (menor valor numérico).
- o Cada cola se gestiona mediante la política RR y la política de planificación entre colas es por prioridades no apropiativa.
- o Un proceso en la cola i pasa a la cola $i+1$ si consume un quantum completo sin bloquearse.
- o Cuando un proceso llega a la cola de menor prioridad, permanece en ella hasta que finaliza.

	<i>c</i>	<i>d</i>	T	u	p
A		E	1	0	1
B	E L ₂ E E L ₃ L L E E		9	4	9/5
C	E L ₂ L L E E L ₃ L L E		10	6	8/3
D		L E L ₂ E E	5	2	5/3
E		L E L ₃ E 4	2	2	
	6 0	48	92	58	28
	0	E			

p=1 q=1 Cola 1: ~~XXXXXX~~

p=2 q=2 Cola 2: ~~XX~~

p=3 q=4 Cola 3: ~~XX~~