Documento anónimo

Ejercicios SO Tema1 y 2.pdf

Ejercicios resueltos T1 y T2 con enunciados

2º Sistemas Operativos

Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
UGR - Universidad de Granada

Relación de problemas - Sistemas Operativos Temas 1 y 2: Estructuras de sistemas operativos, y Procesos y hebras

1. Cuestiones sobre procesos, y asignación de CPU:

a) ¿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?

Si es necesario, ya que en caso contrario nos e mata el proceso. Por su monoprogramado esta respuesta es idéntica.

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

No. En un tema del sistema operativo, el proceso llama al sistema y este lo bloquea y cambia su estado en el PCB mediante el despachador.

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

Tendría que llamar al planificador de medio y largo plazo para que se pasara algo con lo que trabajar

d) ¿Qué algoritmos de planificación quedan descartados para ser utilizados en sistemas de tiempo compartido?

Descartaríamos los algoritmos no apropiativos debido que los procesos no podrían salirse de la CPU con la asistencia del sistema operativo por lo que no podrían coordinarse entre ellos.

Por otro lado, los algoritmos por prioridades (apropiativo o no apropiativo) también se descartarían debido a que su PC ocuparía la mayor parte de la CPU y no dejaría a los demás sistemas que ulizaran la CPU.

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?

Esto es debido a que como tiene ráfagas cortas de tiempo para la asignación de la CPU. Esto conlleva a que haya muchos cambios de contexto en poco tiempo lo cual provoca que el ---- de $\frac{\text{Tiempo\_en\_cola\_ejecutables+Tiempo\_de\_cpu}}{Tiempo\_de\_CPU}$ es tan alto.

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:
a) Crear un proceso. → puede ser necesario el planificador de corto plazo, cuando el planificado apropiado y nada más llegar el SO decide que debe ejecutarse inmediatamente

b**) Abortar un proceso, es decir, terminarlo forzosamente.** →Si

c) Suspender o bloquear un proceso. → Como estaba en ejecución, sí.

d) Reanudar un proceso (inversa al caso anterior). → depende de nuevo del algoritmo de planificación, si es apropiativo puede ser que al desbloquearlo se quiera ejecutar inmediatamente si la circunstancia es favorable para ello, por lo que intervendría el planificador a corto plazo

e) Modificar la prioridad de un proceso. → Si al modificar la prioridad el proceso entra o sale de la ejecución si, en caso contrario no.

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%?

Round-Robin
S→ Tiempo en cambio de contexto
Q →Cuantum de tiempo
$\%Uso\ de\ CPU = \frac{Q}{Q+S}$ Tiempo que trabajo (Q) / Tiempo total de ejecución (Q+S)

$$\frac{Q}{Q+S} = 80\% = 0.8 \rightarrow Q = 0.8Q + 0.8S \rightarrow 0.2Q = 0.8S \rightarrow \boldsymbol{Q = 4S}$$

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?

Round-Robin

$$(N-1) * (Q+S) + Q \leq T$$
$$NQ + (N-1)S \leq T$$

6. ¿Tiene sentido mantener ordenada por prioridades la cola de procesos bloqueados? Si lo tuviera, ¿en qué casos sería útil hacerlo?
Depende, si están bloqueados a la espera de que se cumpla una condición, por ejemplo, no tiene sentido de que estén en baja prioridad si por ejemplo, el bloqueo se debe a cuantums de tiempo (Round-Robin) si tiene sentido a la hora de desbloquear los procesos en cierto orden.

7. ¿Puede el procesador manejar una interrupción mientras está ejecutando un proceso si la política de planificación que utilizamos es no apropiativa?
Si, ya que si el proceso requiere de una E/S por ejemplo ha de ser satisfecha (con su correspondiente cambio de modo).

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?

Dependería del sistema de planificación que quisiéramos. Si queremos un sistema por prioridades tendríamos que cambiar los estados en función de ellos, si fuera Round-Robin tendríamos que ir modificando el reloj y haciendo el cambio de contexto en función de este.

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.

Penalización → $\frac{T+T^{o}\ de\ espera}{T}$ t= tiempo que el proceso está en la CPU

A no ser que el t1 de espera sea igual 0 (seria constante) decreciente. Cuanto mayor fuera el tiempo de espera la penalización sería más grande, aunque iría decreciendo en función del aumento de T.

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	15	16	17
A											■							■
B	■	■	■	■	■	■												
C							■	■	■	■								
D												■	■	■				
E															■	■	■	

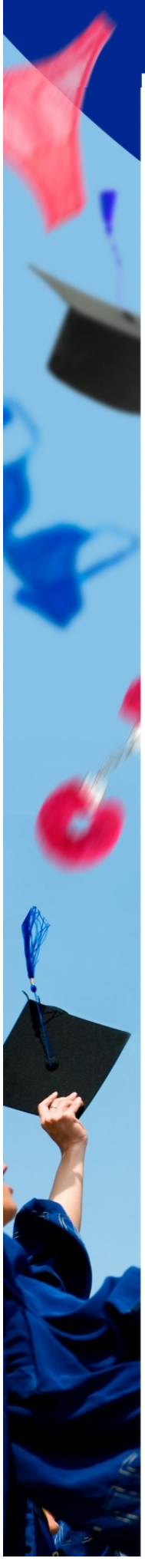
a) Tiempo medio de respuesta = $\frac{5+0+4+2+1}{5} = 2.4$

b) Tiempo medio de espera =En este caso es el mismo que el de respuesta → 2.4

c) La penalización, es decir, el cociente entre el tiempo de respuesta y el tiempo de CPU.

La penalización = $\frac{Tiempo\ respuesta=T}{Tiempo\ de\ cpu=t} = \frac{12}{15} = 0.8$

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



a) Por Turnos con quantum q=1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A																
B																
C																
D																
E																

Tiempo Medio de respuesta A=1 B=0 C=0 D=1 E=0 → 2/5=0.2
Tiempo medio de espera A=1 B=6 C=4 D=3 E=1 → 15/5=3
Penalizacion = T/t = 2/15 = 0.13

b) Por Turnos con quantum q=4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A																
B																
C																
D																
E																

Tiempo Medio de respuesta A=4 B=0 C=3 D=2 E=1 → 10/5 = 2
Tiempo medio de espera A=4 B=5 C=3 D=2 E=1 → 15/5 =3
Penalizacion 10/15=2/3= 0.6

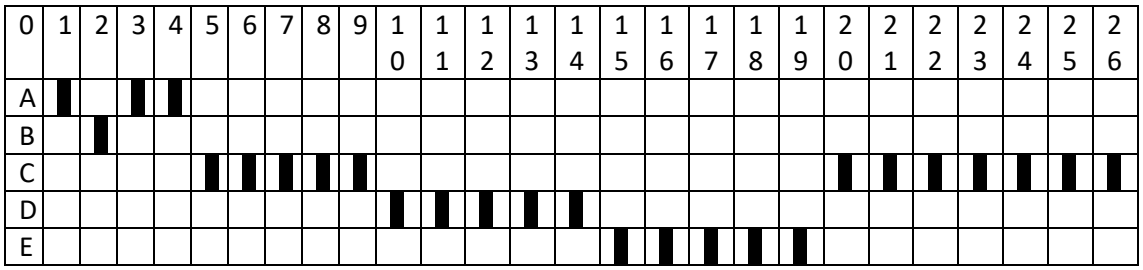
c) El más corto primero (SJF). Suponga que se estima una ráfaga igual a la real.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A																
B																
C																
D																
E																

Tiempo Medio de respuesta A=1 B=0 C=5 D=2 E=1 → 9/5 = 1.8
Tiempo medio de espera → Igual que a la espera
Penalización T/t = 9/15 =0.6

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).

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

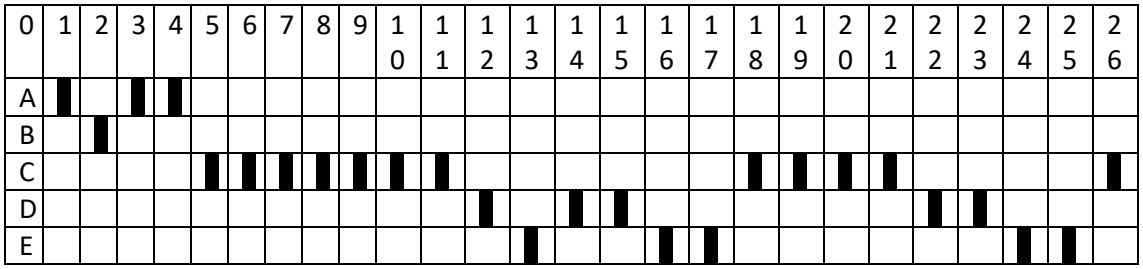


Tiempo medio de espera A=1 B=0 C=1 D=0 E=2 →14/5 = 2.8

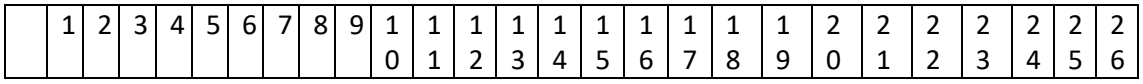
13. Utilizando la tabla del ejercicio anterior, dibuja el diagrama de ocupación de CPU para el caso de un sistema que utiliza un algoritmo de colas múltiples con realimentación con las siguientes colas:

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

y suponiendo que:



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



C1	A	B		C	C					D	D	D	E												
C2		A	A	A		C	C						D	DE	DE	E	E								
C3								C	C	C	C	C	C	C	C	CD	CD	CDE	CDE	CDE	CDE	CDE	CDE	EC	EC	C

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 solo un 10% de su tiempo. ¿Cuáles pueden ser las razones de estos tiempos de respuesta pobres y por qué?
a) El quantum en la planificación Round-Robin es muy pequeño.

b) La memoria principal es insuficiente.

c) 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.

d) La CPU es muy lenta.

e) El quantum en la planificación Round-Robin es muy grande. Ya que la E/S esta poco ocupada → el cuantum

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	Comportamiento		
		Ráfaga de CPU	Tiempo de E/S	Ráfaga de CPU
P1	Hebra 1	20	30	10
	Hebra 2	30	-	-
	Hebra 3	10	-	-
P2	Hebra 1	30	30	10
	Hebra 2	40	-	-

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

Lo hace con la runtime → es un valor que se calcula a partir de la prioridad estática, el peso y el tiempo que ha estado utilizando la CPU
Los procesos largos conforme se ejecutan van aumentando el tiempo que ha estado en la CPU, lo cual hace que aumente su y esto hace que vaya perdiendo prioridad y acabe metiendo procesos cortos.

RESUMEN → Cuanto más tiempo lleva en ejecución, más disminuye su prioridad

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 hace wait, cuando el proceso hijo termina en vez de desaparecer se queda en estado "zombie" consumiendo recursos del sistema.
Ocupa recursos del sistema.

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.

Int clone (int(fn*) (void*), void *child_stack, int**(CLONE_SIGHAND)** (lags, void, *arg);
Esto sería en un proceso hijo, si queremos una hebra tenemos que añadir en flags:
CLONE_FILES→ Copiar los archivos abiertos
CLONE_FS→ Copiar la información del sistema sobre archivos
CLONE_THREAD→ Copiar la información del proceso
CLONE_UN→ Copiar el espacio de direcciones

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