Temas 2 y 3

1) ¿Qué transiciones de estado puede iniciar un proceso por sí mismo?

2) Interprete el siguiente párrafo en el contexto de los ordenadores y los sistemas operativos. Relacione las actuaciones y los protagonistas de esta historia con distintas tareas y entes de un sistema informático.

> *"Juan es un aficionado al aeromodelismo. Se ha comprado una maqueta del modelo X. Tiene la tarde libre y la va a dedicar a la construcción de su avión. Saca las piezas, el manual de instrucciones y el plano, y se enfrasca en la tarea. Comienza a construir la cabina (siguiendo las instrucciones), pero llegado a un punto tiene que esperar a que las piezas se peguen, mientras tanto decide construir las alas. Justo cuando se encuentra en su realización suena el teléfono. Deja lo que está haciendo colocando en posición segura el ala y señalando por dónde se quedó; y atiende el teléfono. Es su mujer, llegará tarde y no quiere perderse la película. Se va al salón, toma el manual del vídeo (no recuerda cómo programarlo) y realiza el encargo. Después vuelve a su pasatiempo preferido."*

3) Cuando se produce una interrupción, ¿se produce necesariamente un cambio de proceso?

4) Cuando se produce una llamada al sistema, ¿se produce necesariamente un cambio de proceso?

5) Cuando un proceso desea bloquearse, ¿deberá encargarse él directamente de cambiar el valor de su estado en su descriptor de proceso?

6) ¿Tiene el *dispatcher* un descriptor de proceso y lucha por la CPU como el resto de procesos?

7) Dado un proceso que cambia su estado de en ejecución a bloqueado, ¿puede esto provocar un cambio de estado en otro proceso?, ¿entre qué estados?

8) Idem para cambio entre bloqueado y listo.

9) Idem para cambio entre listo y bloqueado.

10) Sin reloj en la computadora, ¿qué algoritmos de planificación podríamos implementar?

11) ¿Es posible que en algún momento la CPU esté ejecutando código que no pertenece a ningún proceso de usuario?

12) Puesto que un controlador es un procesador dedicado a controlar el o los dispositivos asignados, ¿existirá un descriptor de proceso para cada uno de dichos controladores?

13) Respecto a un proceso, ¿dónde se podría almacenar la información sobre el proceso que lo creó (su proceso padre), o sobre la de sus hijos?

14) Comente si es totalmente cierta la siguiente afirmación: *En un sistema de tiempo compartido en el que los procesos se planifican mediante el algoritmo de turno rotatorio con cuanto de tiempo n unidades, cada proceso se ejecuta sin interrupción durante n unidades de tiempo*. Justifique su respuesta.

15) Jacinto acaba de terminar la carrera y su proyecto consiste en construir un sistema operativo que permita niveles altos de multiprogramación. Para ello deduce que es necesario disponer de mayor cantidad de memoria principal disponible y plantea como solución el mantener los descriptores de proceso en disco. ¿Es razonable?, ¿por qué?

16) Supongamos un algoritmo de planificación que favorece a aquellos programas cuya última ráfaga ha sido corta. ¿Por qué favorecerá este algoritmo a los programas intensivos en E/S sin provocar inanición en los programas intensivos en el uso de la CPU?

17) Se aplica el término "tiempo compartido" a los sistemas operativos que permiten que varios usuarios interactivos trabajen simultáneamente, para lo cual será necesario asegurar que cada proceso disfrute de cierto tiempo de CPU a intervalos más o menos regulares. ¿Qué algoritmos de planificación quedan descartados para ser usados en este tipo de entornos?

18) ¿En qué grado discriminan a los procesos cortos los siguientes algoritmos de asignación de la CPU: FIFO, turno rotatorio, colas de niveles múltiples con retroalimentación?

19) ¿Es cierta la siguiente afirmación: *La política de planificación turno rotatorio es equivalente a la política de planificación FIFO si el cuantum de tiempo es infinitamente largo*?

20) Muchos algoritmos de planificación son parametrizados. Por ejemplo, el algoritmo de turno rotatorio requiere un parámetro: el cuantum. Esto implica que estos algoritmos son en realidad grupos de algoritmos (por ejemplo, el conjunto de todos los algoritmos *round robin* para cada valor posible del cuantum). Puede ocurrir que un grupo de algoritmos incluya a otro, ¿qué relación de este tipo existe entre las siguientes parejas de algoritmos?

a) Prioridad / El más corto primero
b) Colas de niveles múltiples con retroalimentación / FIFO
c) Prioridad / FIFO
d) Turno rotatorio / El más corto primero

21) Estamos escribiendo el código de un programa de usuario. ¿Qué operaciones podemos realizar para provocar la activación del *dispatcher*?

22) Un sistema tiene los siguientes recursos: una CPU, dos discos y una impresora. En el sistema se van a ejecutar dos procesos con las siguientes características:

- **P_1**: 13 unidades de tiempo en el orden siguiente: 1 para CPU, 3 para disco1, 2 para CPU, 6 para impresora y 1 para CPU.
- **P_2**: 16 unidades de tiempo en el orden siguiente: 6 para CPU, 1 para disco1, 3 para CPU, 2 para disco2, 1 para CPU, 1 para impresora y 2 para CPU.

Calcule el tiempo de finalización medio de los procesos, y la utilización de la CPU y de cada uno de los dispositivos durante el tiempo que dura la ejecución de ambos procesos en los siguientes supuestos:

a) el sistema es de monoprogramación
b) es de multiprogramación y se utiliza el algoritmo FIFO de planificación.

Nota: En el sistema de monoprogramación P_1 se crea antes que P_2. En el de multiprogramación los 2 procesos se crean "a la vez", siendo asignada la CPU a P_1 en el instante 0. En el sistema de multiprogramación debe despreciar el tiempo empleado en los cambios de proceso, así como en el servicio de interrupciones.

23) Se van a ejecutar dos procesos con los siguientes patrones de comportamiento:

- **P_1**: 88 unidades de tiempo en el orden siguiente: 1 de CPU, 21 de impresora, 1 de CPU, 21 de impresora, 1 de CPU, 21 de impresora, 1 de CPU, 21 de impresora.
- **P_2**: programa sin E/S que consta de una sóla ráfaga de 80 unidades de tiempo de duración.

Calcule el tiempo de finalización medio de los procesos, y la utilización de la CPU y de la impresora durante el tiempo que dura la ejecución de ambos procesos en los siguientes sistemas:

a) de monoprogramación.
b) de multiprogramación con planificación FIFO.
c) de tiempo compartido planificándose con *round robin* con cuanto de 10 unidades de tiempo.
d) de tiempo compartido planificándose con colas multinivel. Se utilizan 2 colas y un cuanto de 10 unidades de tiempo en ambas colas.

Nota: En el sistema de monoprogramación P_1 se crea antes que P_2. En los restantes los 2 procesos se crean "a la vez", siendo asignada la CPU a P_1 en el instante 0. En los cálculos que haga debe despreciar el tiempo empleado en los cambios de proceso y en la gestión de interrupciones.

Soluciones

1) La única transición que puede iniciar (lo hace indirectamente) un proceso es la de en ejecución a bloqueado. Para ello debe realizar una llamada al sistema que provoque el bloqueo del proceso.

2) En esta historia podemos equiparar a Juan con un procesador que ejecuta tareas (procesos). El manual de instrucciones que interpreta equivale a un programa que se ejecuta. La operación de pegar la cabina la podemos ver como la realización de una operación con un periférico que se puede simultanear con otras acciones como construir las alas. El timbre del teléfono se puede equiparar a la ocurrencia de una interrupción, que implica salvar el contexto de ejecución (señalar por dónde va), y darle servicio (ejecutar la rutina de servicio a la interrupción, en la parábola, el manual del vídeo).

3) No, cualquier interrupción no tiene por qué provocar un cambio de proceso. Por ejemplo, una interrupción del reloj que no coincide con el agotamiento del cuanto del proceso en ejecución no tiene por qué implicar un cambio de proceso.

4) No, existen muchas llamadas al sistema que no provocan un cambio de proceso. Las llamadas al sistema que implican algún tipo de E/S suelen conllevar un cambio de proceso para que se utilice mejor el procesador. Sin embargo, hay llamadas al sistema, como la de liberación de memoria dinámica, que no implican ningún cambio de proceso.

5) No. El descriptor de proceso es una estructura de datos que gestiona el sistema operativo y a la que no tienen acceso los procesos gracias a los mecanismos de protección de la memoria.

6) El *dispatcher* no lucha por la CPU como el resto de procesos de usuario. Por ejemplo, no tiene sentido que se sitúe en la cola de procesos en estado listo. Al *dispatcher* lo llama el sistema operativo cuando es preciso asignar la CPU, y se ejecuta sin entremezclarse con la ejecución de otros procesos de usuario. No compite por la CPU por tanto como éstos. Por otro lado no tiene descriptor de proceso. Esto es debido a que no tiene estado de ejecución y no es preciso guardar su contexto, pues siempre se ejecuta en su totalidad. La zona de memoria que ocupa siempre es la misma, y se decide al inicializarse el sistema.

7) Sí, cuando un proceso pasa de en ejecución a bloqueado la CPU queda ociosa. Un nuevo proceso de los que están en estado listo debe ocupar en este instante la CPU y pasar por tanto a en ejecución.

8) Dependerá de la política de planificación del sistema operativo. Por ejemplo, si la planificación es por turno rotatorio no puede provocar cambio. Si la planificación es la de colas de niveles múltiples con retroalimentación y el proceso que ocupa la CPU pertenece a una cola de menor prioridad que el proceso que pasa a estado listo, entonces el proceso en ejecución se verá obligado a dejar la CPU, y pasará por tanto de en ejecución a listo.

9) Esta transición de estado nunca se da.

10) FIFO y prioridad al más corto (SJF). Este último si la estimación de lo que dura una ráfaga no se calcula como el tiempo que duró la última ráfaga del proceso.

11) Sí, cuando realiza cualquier tarea del sistema como: conmutar entre procesos, planificación de procesos, gestión de interrupciones, etc.

12) No, no tiene sentido.

13) En el PCB, que es la estructura de datos que utiliza el sistema operativo para guardar toda la información relativa a un proceso y así poder gestionarlo.

14) No, porque durante el intervalo que dura un cuanto es muy posible que la CPU ejecute rutinas de servico a interrupción originadas por eventos como la pulsación de una tecla en un terminal, el fin de una operación de E/S o la interrupción del reloj.

15) No. Los descriptores de proceso son las estructuras de datos que mantienen toda la información relativa a los procesos, esta información es consultada y modificada con gran frecuencia por distintas unidades funcionales del sistema operativo. Por lo tanto, es imprescindible que estén en memoria principal para un acceso eficiente.

17) Todos aquellos que no son apropiativos. Como por ejemplo: FIFO, primero el más corto (SJF) o prioridad a la tasa de respuesta más alta (HRN).

18) FIFO: establece una asignación no apropiativa del procesador por orden de entrada en el sistema en estado listo; un proceso que pase a listo con una ráfaga corta ha de esperar a que se bloqueen o terminen todas las ráfagas largas de los procesos que le preceden en la lista preparados, por lo que puede verse muy discriminado.

Turno rotatorio: es un tipo de estrategia apropiativa. Los procesos no tienen que esperar necesariamente a que se bloqueen o terminen las ráfagas de los procesos que les preceden en la lista

de preparados. Si las ráfagas de los procesos que les preceden son largas sólo se debe esperar a que agoten su cuanto. Las procesos cortos se ven más favorecidos con este algoritmo que con FIFO.

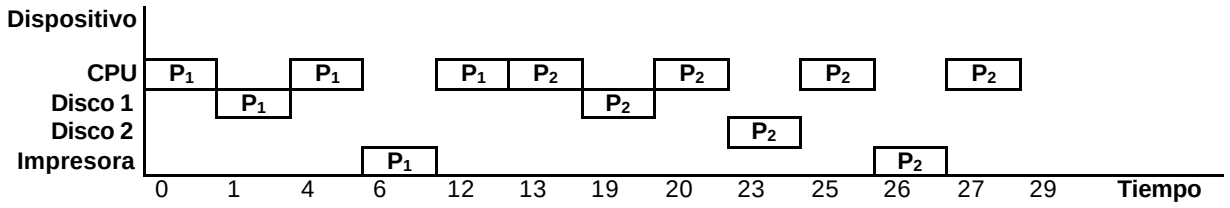
Colas de niveles múltiples con retroalimentación: Muy poco. Los procesos cortos se sitúan (al iniciarse) en la cola de mayor prioridad, con lo que sólo tienen que esperar a que se bloqueen, terminen o agoten cuanto las ráfagas de los procesos de su cola, y no deben esperar a las ráfagas de los procesos de las colas inferiores. Si el proceso es efectivamente corto, no precisará agotar el cuanto de la cola de mayor prioridad para terminar, con lo que habrá sido atendido muy rápidamente.

19) Sí.

20) a) El algoritmo primero el más corto elige como criterio de prioridad la menor utilización de CPU por parte de la ráfaga del proceso.
b) Son coincidentes cuando el número de colas es 1 y el cuantum infinito.
c) FIFO aplica el siguiente criterio de prioridad: primero el que lleva más tiempo en estado listo.
d) Ninguna.

21) Queremos activar el *dispatcher*, esto implica que el proceso en ejecución debe bloquearse o terminar. Esto se puede realizar haciendo una E/S (por ejemplo una llamada a *scanf* en C), una operación *wait* sobre un semáforo a cero o una instrucción *exit* en C.

22)
a) Al ser un sistema de monoprogramación los procesos se ejecutan uno detrás de otro, disfrutando en todo momento de todos los recursos del ordenador. El siguiente gráfico muestra la ocupación de los dispositivos a lo largo del tiempo:

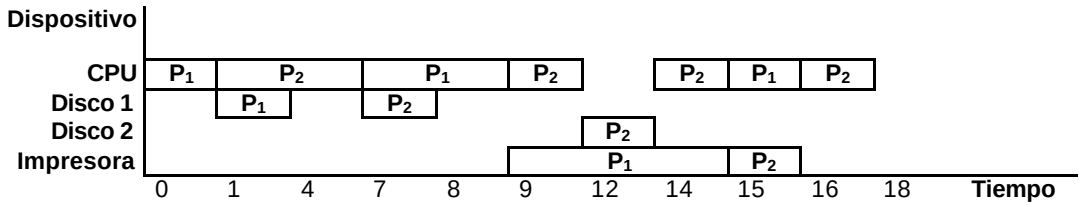


Del gráfico se deduce que P_1 tarda 13 segundos en ejecutarse, y P_2 29 segundos, luego el tiempo medio de finalización es: (13 u.t. + 29 u.t.) / 2 = 21 u.t.

La utilización de los dispositivos en el tiempo que dura la ejecución de los dos procesos es:

Utilización de la CPU = (16 u.t. / 29 u.t.) * 100 = 55.2 %
Utilización del disco 1 = (4 u.t. / 29 u.t.) * 100 = 13.8 %
Utilización del disco 2 = (2 u.t. / 29 u.t.) * 100 = 6.9 %
Utilización de la impresora = (7 u.t. / 29 u.t.) * 100 = 24.1 %

b) Ahora el sistema es multiprogramado, por lo que mientras un proceso realiza una E/S el otro puede ocupar la CPU. El siguiente gráfico muestra la ocupación de los dispositivos a lo largo del tiempo:



Del gráfico se deduce que P_1 tarda 16 segundos en ejecutarse, y P_2 18 segundos, luego el tiempo medio de finalización es: (16 u.t. + 18 u.t.) / 2 = 17 u.t.

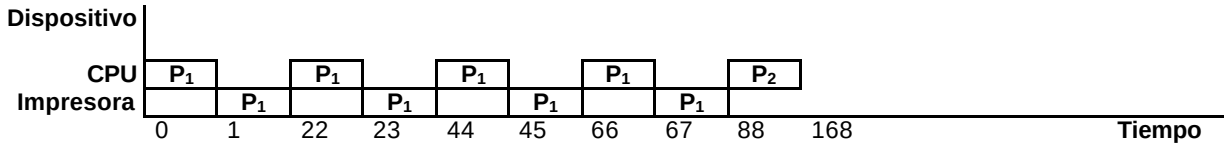
La utilización de los dispositivos en el tiempo que dura la ejecución de los dos procesos es:

Utilización de la CPU = (16 u.t. / 18 u.t.) * 100 = 88.9 %
Utilización del disco 1 = (4 u.t. / 18 u.t.) * 100 = 22.2 %
Utilización del disco 2 = (2 u.t. / 18 u.t.) * 100 = 11.1 %
Utilización de la impresora = (7 u.t. / 18 u.t.) * 100 = 38.9 %

Observe cómo se ha incrementado la utilización de la CPU y demás recursos en el sistema de multiprogramación. Un hecho notable es que en el sistema multiprogramado los dos procesos se ejecutan en 18 unidades de tiempo frente a las 27 que necesita el sistema monoprogramado.

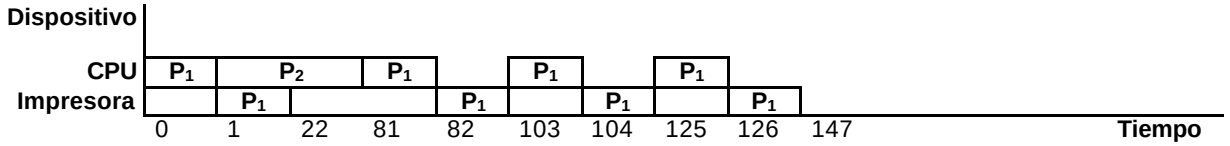
23)

a)



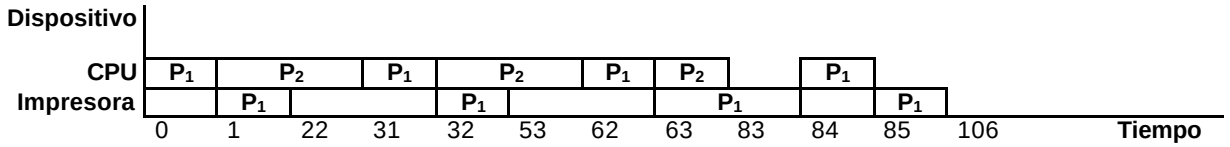
Tiempo medio de finalización: (88 u.t. + 168 u.t.) / 2 = 128 u.t.
Utilización de la CPU = (84 u.t. / 168 u.t.) * 100 = 50 %
Utilización de la impresora = (84 u.t. / 168 u.t.) * 100 = 50 %

b)



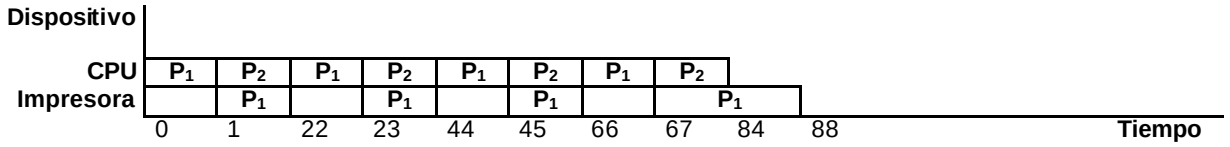
Tiempo medio de finalización: (81 u.t. + 147 u.t.) / 2 = 114 u.t.
Utilización de la CPU = (84 u.t. / 147 u.t.) * 100 = 57.1 %
Utilización de la impresora = (84 u.t. / 147 u.t.) * 100 = 57.1 %

c)



Tiempo medio de finalización: (83 u.t. + 106 u.t.) / 2 = 94.5 u.t.
Utilización de la CPU = (84 u.t. / 106 u.t.) * 100 = 79.2 %
Utilización de la impresora = (84 u.t. / 106 u.t.) * 100 = 79.2 %

d)



Tiempo medio de finalización: (84 u.t. + 88 u.t.) / 2 = 86 u.t.
Utilización de la CPU = (84 u.t. / 86 u.t.) * 100 = 97.7 %
Utilización de la impresora = (84 u.t. / 86 u.t.) * 100 = 97.7 %