Examen 2016, preguntas y respuestas

Señales Y Sistemas (Universidad de Alicante)

Ejercicios de Técnicas básicas de gestión de memoria.

1. ¿Cuáles son los requisitos que debe intentar satisfacer la gestión de memoria?

- **Reubicación**
- **Protección**
- **Compartición**
- **Organización (física y lógica)**

Todo ello para solucionar la escasez.

2. ¿Por que es deseable la capacidad de reubicación?

   No se sabe a priori donde se va a ubicar un proceso en un sistema multiprogramado.

3. ¿Por qué no se puede implantar la protección de memoria en tiempo de compilación?

   Se desconoce la ubicación del programa en memoria principal y por ello no se pueden comprobar la validez de las direcciones absolutas en tiempo de compilación. Solo se pueden comprobar en tiempo de ejecución y lo debe hacer el procesador (hardware).

4. ¿Por qué se debe de permitir que dos o más procesos accedan a la misma región de memoria?

   Si varios procesos ejecutan el mismo programa sería absurdo que no pudieran compartir el código. Este soporte lo debe proporcionar el gestor de memoria del SO.

5. Supongamos que un proceso emite la dirección lógica (2,18004) utilizando un modelo de gestión de memoria basado en segmentación y el espacio de memoria física es de 64Kbytes.

   a. ¿A que direcciones físicas de las siguientes (11084, 33270, 22112), sería posible traducir dicha dirección lógica?
      La dirección base del segmento debe ser inferior a la dirección física obtenida por lo tanto base+desplazamiento= dirección física
      Base+18004<>11084
      Base+ 18004=33270 (el segmento empieza en la dir. 15266),
      Base +18004=22112 (si empieza en la dir. 4108)

   b. ¿Cuál sería el resultado de traducir la dirección lógica (0,65536) en dicho sistema? Justificar la respuesta.
      Error por exceso en el tamaño máximo del segmento (64K= 65535)

   c. Si ahora se utiliza un modelo de memoria basado en segmentación paginada (tamaño de página=512 palabras), la memoria se encuentra vacía y el criterio de asignación de memoria es por direcciones crecientes. ¿Cuál sería la dirección física correspondiente a la dirección lógica (0, 9701)?
      9701 MOD 512 = 485

d. En el caso de utilizar un modelo de memoria basado en particiones fijas (tamaño de las particiones 4K, 12K, 16K, 32K bytes respectivamente y ubicadas en orden de direcciones crecientes) ¿A qué direcciones físicas de las siguientes (9701, 26085, 32768) se puede corresponder la dirección lógica 9701? (suponiendo que pudiera aplicarse un esquema de traducción)
Dirección física inicial de cada partición.
0 (4K)
4096 (12K)
16384 (16K)
32768 (32K)
65535 (dirección final)

9701 En caso de la primera partición no es posible por que excede su tamaño, en el resto tampoco
26085 = 4+12=16K (base de tercera partición)=16384 + 9701
32768 no se obtiene este valor sumando 9701 a ninguno de los comienzos partición para obtener la dirección física

6. Sea un sistema gestionado por particiones múltiples de tamaño variable con compactación. En un instante dado, se tiene la siguiente ocupación de la memoria:

0 984k

S.O. (80K)	P1 (180K)	Libre (400K)	P2 (100K)	Libre (150K)	P3 (75K)

Se utiliza la técnica del mejor ajuste. En la cola de trabajos tenemos en este orden: P4(120K), P5(200K) y p6(80K), los cuales deben ser atendidos en orden FIFO. Suponiendo que no finaliza ningún proceso y tras intentar cargar en memoria todos los procesos que están en la cola…

a. Indicar cuantas particiones quedan libre y de qué tamaño son.
Quedan dos particiones de tamaños 120K y 30K respectivamente

b. Si en esta situación se aplica compactación, indicar qué proceso o procesos deberían moverse para que el número de Kbytes manejados fuese el menor posible y quede un único hueco.
Debería moverse el proceso P4 al hueco de 120K

c. Si los registros base de cada proceso son, respectivamente, B1, B2, B3, B4, B5 y B6, indicar cómo han cambiado los registros base correspondientes al proceso o procesos que se han movido debido a la compactación.
Todos quedan igual, salvo B4 que queda (B4-540K)

7. Un proceso genera las siguientes direcciones lógicas (612, 38 y 3,62). Indica las direcciones físicas correspondientes según cada esquema de gestión de memoria. Si no es posible indicar ERROR.

a. Particiones variables. Registro base: 150 Registro límite: 220
612 (Error) 38 (150+38=188) 3,62 (Error)

b. Paginación. Tamaño página: 128 Tabla páginas

0	1
1	4
2	2
3	5

612 (Error 4marcosx128=512) **38 (128+38=166)** **3,62 (Error)**

c. Segmentación (no paginada). Tabla de segmentos

	Base	Límite
0	200	20
1	50	10
2	105	49
3	320	70

612 (Error) **38 (Error<bases)** **3,62 (320+62=382)**

8. Supongamos un sistema de gestión de memoria con segmentación paginada, con páginas de 1Kb. Un proceso emite las siguientes direcciones lógicas: (1, 2487) y (0,635). A continuación se muestra la tabla de páginas del segmento 1. ¿Cuáles serían las direcciones físicas correspondientes?

Nº página	marco
0	3
1	6
2	8

1,2487 -> (1024*6)+2487%1024=8631
0,635 -> (1024*3)+635=(3707)

9. En un esquema de segmentación paginada con páginas de 1Kb, ¿Es posible que la dirección lógica (2,1333) se pudiera traducir a la dirección física 3654.
No es posible a 3654:(3654 mod 1024) =309; (1024x3)+309<>3654

¿Y a la dirección física 2357?
Si es posible a 2357: (2357 mod 1024)=309; (1024x2)+309=2357

10. Sea un sistema gestionado por particiones múltiples de tamaño variable sin compactación. En un instante dado, se tiene la siguiente ocupación de la memoria:

0

S.O. (80K)	P1 (180K)	Libre (400K)	P2 (100K)	Libre (150K)	P3 (90K)	Lil (200K)

1200k

En la cola de trabajos tenemos en este orden: P4(120K), P5(200K) y P6(300K), los cuales deben ser atendidos en orden FIFO. Suponiendo que no finaliza ningún proceso y tras intentar cargar en memoria todos los procesos que están en la cola, evaluar cual de las técnicas entre las de mejor ajuste y peor ajuste es conveniente utilizar y por qué.

Mejor ajuste. En el caso del peor ajuste el proceso P6 no puede entrar.

11. Clasificar los esquemas de gestión de memoria en función del tipo de fragmentación que presentan.

Fragmentación interna	**Fragmentación externa**
Partición única Particiones múltiples de tamaño fijo Paginación Paginación segmentada	Particiones múltiples de tamaño variable Segmentación

Gestión de memoria: Memoria virtual

Soluciones problemas clase

1. Un determinado S.O. gestiona la memoria virtual mediante la paginación por demanda. La dirección lógica tiene 24 bits, de los cuales 14 indican el número de página.

La memoria física tiene 5 marcos.

El algoritmo de reemplazo de páginas es el LRU LOCAL, y se ha implementado mediante un contador asociado a cada página que indica el instante de tiempo en que se referenció la página por última vez.

Las tablas de páginas en el instante 16 son:

Proceso	Página	Bit válido	Marco	Contador
A	0	v	1	10
	1	v	2	15
	2	i	-	6
	3	i	-	5
B	0	v	0	7
	1	i	-	2
	2	i	-	3
	3	v	3	4
	4	v	4	11

Indicar las direcciones físicas generadas para la siguiente secuencia de direcciones lógicas (de izquierda a derecha y de arriba abajo):

(A, 2900) (B, 1200) (A, 1850) (A, 3072) (B, 527)

(B, 2987) (A, 27) (A, 2000) (B, 4800) (B, 1500)

Solución: 10 bits de desplazamiento -> P = 2^{10} = 1024 palabras.

MARCO	INICIAL	(A,2)	(B,1)	(A,1)	(A,3)	(B,0)	(B,2)	(A,0)	(A,1)	(B,4)	(B,1)
0	B,0	B,0	B,0	B,0	B,0	**B,0**	B,0	B,0	B,0	B,0	**B,1**
1	A,0	**A,2**	A,2	A,2	**A,3**	A,3	A,3	A,3	**A,1**	A,1	A,1
2	A,1	A,1	A,1	**A,1**	A,1	A,1	A,1	**A,0**	A,0	A,0	A,0
3	B,3	B,3	**B,1**	B,1	B,1	B,1	B,1	B,1	B,1	**B,4**	B,4
4	B,4	B,4	B,4	B,4	B,4	B,4	**B,2**	B,2	B,2	B,2	B,2

DIR. LÓGICA = u	p = u DIV P	d = u MOD P	DIR. FÍSICA = marco * P + d
(A, 2900)	2	852	1*1024+852 = 1876
(B, 1200)	1	176	3*1024+176 = 3248
(A, 1850)	1	826	2*1024+826 = 2874
(A, 3072)	3	0	1*1024+0 = 1024
(B, 527)	0	527	0*1024+527 = 527
(B, 2987)	2	939	4*1024+939 = 5035
(A, 27)	0	27	2*1024+27 = 2075
(A, 2000)	1	976	1*1024+976 = 2000
(B, 4800)	4	704	3*1024+704 = 3776
(B, 1500)	1	476	0*1024+476 = 476

2. Disponemos de un sistema de M.V. por Paginación por Demanda. Las direcciones lógicas tienen 11 bits, de los cuales 2 se interpretan internamente como número de página. La memoria está organizada en 3 marcos. En este momento únicamente tenemos 2 procesos: A y B. La situación inicial de las páginas es:

PROCESO A		PROCESO B	
Pág	Marco	Pág	Marco
0	0	0	2
1	1	1	i
2	i	2	i
3	i	3	i

Del enunciado se deduce que:

(a) El tamaño de página es: $P^{11-2}=2^9=512$

(b) La tabla de marcos en la situación inicial es:

0	A0
1	A1
2	B0

(a) Si se obtuviera de la situación inicial cada una de las siguientes direcciones físicas (sin relación de orden entre sí), calcular las direcciones lógicas de las que proceden: 845,623,1024,1603

DIR. FÍSICA	CÁLCULO	MARCO	PÁG.	DIR. LÓGICA
845	512+333	1	A,1	*A,845*
623	512+111	1	A,1	*A,623*
1024	1024+0	2	B,0	*B,0*
1603	1536+67	ERROR	ERROR	ERROR

NOTA: La última dirección (1603), excede del tamaño de la memoria física.

(b) Si se utiliza un algoritmo de reemplazo LRU global, y a partir de la situación inicial se generan las siguientes direcciones lógicas (orden izquierda a derecha):

(A, 632), (A, 1130), (B, 555), (B, 28), (A, 1333), (B, 446), ...

..., (A, 501), (A, 1422), (B, 111), (A, 999), (A, 1222), (A, 888)

completar el mapa de ocupación de la memoria e indicar el número de fallos de página producido.

NOTA IMPORTANTE: Las últimas referencias a páginas antes de la situación inicial han sido: B0, A1, A0 (en este orden).

	REFERENCIAS EN FORMATO (PROCESO, PÁGINA)												
	INICIAL	A1	A2	B1	B0	A2	B0	A0	A2	B0	A1	A2	A1
0	A0			B1				A0			A1		=
1	A1	=			B0		=			=			
2	B0		A2			=			=			=	

NÚMERO DE FALLOS	5

3. Suponga un sistema de memoria virtual que utiliza paginación por demanda, donde la memoria física tiene tres marcos. En este sistema se ejecutan concurrentemente tres procesos A, B, C que provocan la siguiente serie de referencias a páginas:

A1, B0, C0, A2, A1, B0, C1, A1, B0, C0, A2, C1,B0

Para los siguientes algoritmos de reemplazo de páginas, indique el estado final de la memoria (indique qué página está ubicada en cada uno de los marcos). Suponga que la memoria está inicialmente vacía y que los marcos se asignan en orden creciente. En caso de haber varias víctimas posibles, siempre se elegirá la que ocupe el marco cuya numeración sea más baja..

Óptimo global

A1	**B0**	**C0**	**A2**	**A1**	**B0**	**C1**	**A1**	**B0**	**C0**	**A2**	**C1**	**B0**
A1	A1	A1	A1	A1	A1	A1	A1	A1	**C0**	**A2**	A2	**A2**
	B0	B0	B0	B0	B0	B0	B0	B0	B0	B0	B0	**B0**
		C0	**A2**	A2	A2	**C1**	C1	C1	C1	C1	C1	**C1**

LRU global

A1	B0	C0	A2	A1	B0	C1	A1	B0	C0	A2	C1	B0
A1	A1	A1	**A2**	A2	A2	**C1**	C1	C1	**C0**	C0	C0	**B0**
	B0	B0	B0	**A1**	A1	A1	A1	A1	A1	**A2**	A2	**A2**
		C0	C0	C0	**B0**	B0	B0	B0	B0	B0	**C1**	**C1**

4. Sea un sistema de Memoria Virtual con Segmentación Paginada y Algoritmo de reemplazo LRU Global. Sea 512 el tamaño de página, y en memoria física hay en total 3 tramas, inicialmente vacías. Suponiendo que hay dos procesos en el sistema (A y B) con las siguientes Tablas de Segmentos (especificando únicamente la longitud de cada uno):

PROCESO A	PROCESO B
160	300
200	640
1000	

Se generan las siguientes direcciones lógicas (en orden IZQUIERDA-DERECHA y luego ARRIBA-ABAJO:

(B,0,209), (A,0,27), (A,1,171), (B,0,180), (A,1,25), (A,2,638)

(A,2,815), (B,0,200), (A,0,155), (B,0,193), (B,1,608), (A,2,715)

Supóngase que las tramas se asignan de menor a mayor (primero la 0). Calcular las direcciones físicas a que dan lugar, y el número de fallos de página provocados.

Solución: número de fallos=7

Ocupación de la Memoria Física:

	B00	A00	A10	B00	A10	A21	A21	B00	A00	B00	B11	A21
0	B00	—	—	—	—	—	—	—	—	—	—	—
1	###	A00	—	—	—	A21	—	—	—	—	B11	—
2	###	###	A10	—	—	—	—	—	A00	—	—	A21

Cálculo de las direcciones físicas.

LÓGICA	FÍSICA	LÓGICA	FÍSICA	LÓGICA	FÍSICA
(B,0,209)	*209*	(A,0,27)	*539*	(A,1,171)	*1195*
(B,0,180)	*180*	(A,1,25)	*1049*	(A,2,638)	*638*
(A,2,815)	*815*	(B,0,200)	*200*	(A,0,155)	*1179*
(B,0,193)	*193*	(B,1,608)	*608*	(A,2,715)	*1227*

5. Un determinado S.O. gestiona la memoria virtual mediante segmentación paginada. Una dirección lógica tiene 24 bits, de los cuales 5 indican el número de segmento. Una dirección física tiene 12 bits, y el tamaño de la trama es de 1024 octetos. Supongamos que hay 2 procesos en el sistema, con la siguiente ocupación de memoria virtual:

Proceso	Segmento	Tamaño
A	0	100
	1	2100
	2	1120
	3	3450
B	0	950
	1	4120
	2	512

Las tablas de segmentos y páginas no consumen memoria física. Inicialmente la memoria está vacía.

Si se utiliza el algoritmo de reemplazo de páginas LRU GLOBAL, indicar las direcciones físicas generadas para la siguiente secuencia de direcciones lógicas (de izquierda a derecha y de arriba abajo):

(A, 3, 2100) (A, 2, 1100) (B, 0, 800) (B, 2, 300) (A, 0, 50) (B, 0, 300)

(A, 2, 1024) (A, 1, 2000) (B, 1, 3120) (A, 2, 1050) (B, 0, 800) (A, 1, 2100)

Solución:

Como cada dirección física tiene 12 bits y de ellos 10 se emplean para desplazamiento dentro de la trama ($\log_2 1024=10.$), nos quedan 2 para especificar el número de trama: $2^2=4$ tramas.

Direcciones lógicas intermedias:

A partir de las direcciones lógicas, que especifican (proceso, segmento, desplazamiento en segto.) se pueden calcular otras en las que se aplique la paginación: (proceso, segmento, página, desplazamiento en pag.). Además, comprobaremos si las direcciones son correctas comparando previamente si el desplazamiento en el segmento es inferior al tamaño de ese segmento:

LÓGICA	INTERMEDIA	LÓGICA	INTERMEDIA
(A, 3, 2100)	(A, 3, 2, 52)	(A, 2, 1024)	(A, 2, 1, 0)
(A, 2, 1100)	(A, 2, 1, 76)	(A, 1, 2000)	(A, 1, 1, 976)
(B, 0, 800)	(B, 0, 0, 800)	(B, 1, 3120)	(B, 1, 3, 48)
(B, 2, 300)	(B, 2, 0, 300)	(A, 2, 1050)	(A, 2, 1, 26)
(A, 0, 50)	(A, 0, 0, 50)	(B, 0, 800)	(B, 0, 0, 800)
(B, 0, 300)	(B, 0, 0, 300)	(A, 1, 2100)	**ERROR**

Algoritmo de reemplazo:

Usando las direcciones intermedias (exceptuando el desplazamiento y prescindiendo de la última dirección lógica por ser errónea), y aplicando el algoritmo LRU GLOBAL partiendo de la memoria vacía, obtenemos la siguiente distribución:

	A,3,2	A,2,1	B,0,0	B,2,0	A,0,0	B,0,0	A,2,1	A,1,1	B,1,3	A,2,1	B,0,0
0	A,3,2	—	—	—	A,0,0	—	—	—	B,1,3	—	—
1		A,2,1	—	—	—	—	*A,2,1*	—	—	*A,2,1*	—
2			B,0,0	—	—	*B,0,0*	—	—	—	—	*B,0,0*
3				B,2,0	—	—	—	A,1,1	—	—	—

NOTA: Las páginas en itálica estaban ya en memoria física, por lo que no han provocado fallo.

Direcciones físicas: Sumando el desplazamiento de las direcciones intermedias con la dirección de inicio de la trama donde se encuentra la página, obtenemos las direcciones físicas equivalentes:

LÓGICA	FÍSICA	LÓGICA	FÍSICA
(A, 3, 2100)	**52**	(A, 2, 1024)	**1024**
(A, 2, 1100)	**1100**	(A, 1, 2000)	**4048**
(B, 0, 800)	**2848**	(B, 1, 3120)	**48**
(B, 2, 300)	**3372**	(A, 2, 1050)	**1050**
(A, 0, 50)	**50**	(B, 0, 800)	**2848**
(B, 0, 300)	**2348**	(A, 1, 2100)	**ERROR**

6. En un sistema de gestión de memoria virtual se decide utilizar paginación y un algoritmo de reemplazo GLOBAL de SEGUNDA OPORTUNIDAD. En el sistema se ejecutan actualmente 2 procesos A y B. La memoria física tiene 4 marcos. La estructura de datos del algoritmo de segunda oportunidad en un momento dado es la siguiente cola FIFO:

*(A1,2,1) -> (B2,0,0) -> (A0,1,0) -> (B1,3,1)

donde cada tripleta representa (Proceso-página, marco-de-memoria-física, bit-de-referencia). En el instante considerado, la última página referenciada e insertada en la lista ha sido la B1 y el puntero a la siguiente víctima apunta a la página A1 (se indica con un asterisco). Las peticiones son: B2,B0,A2,B3,B2,A0,A3

Indique cómo evolucionará la estructura de datos del algoritmo de segunda oportunidad a partir de este instante, al producirse las siguientes referencias (se supone que todas las páginas accedidas están dentro del espacio lógico del proceso).

Referencia	Cola segunda oportunidad
B1	*(A1,2,1) -> (B2,0,0) -> (A0,1,0) -> (B1,3,1)
B2	*(A1,2,1) -> (B2,0,1) -> (A0,1,0) -> (B1,3,1)
B0	(A1,2,0) -> (B2,0,0) -> (B0,1,1) -> *(B1,3,1)
A2	(A2,2,1) -> *(B2,0,0) -> (B0,1,1) -> (B1,3,0)
B3	(A2,2,1) -> (B3,0,1) -> *(B0,1,1) -> (B1,3,0)
B2	*(A2,2,1) -> (B3,0,1) -> (B0,1,0) -> (B2,3,1)
A0	(A2,2,0) -> (B3,0,0) -> (A0,1,1) -> *(B2,3,1)
A3	(A3,2,1) -> *(B3,0,0) -> (A0,1,1) ->(B2,3,0)

Problemas resueltos en clase de sistema de ficheros y entrada/salida

Problemas sobre conversiones de direcciones de disco

Ejercicio 5a

Dada una estructura de disco de doble cara con 80 cilindros y 18 sectores por pista y un tamaño de sector de 512 bytes, se pide:
a) Determinar la capacidad del disco.
b) Dada la correspondencia entre sectores físicos y lógicos, se trata de determinar la dirección física (cilindro, cabeza, sector) del número de sector lógico 1234.
c) De forma análoga se desea determinar cual es el número de sector lógico asociado a la dirección física (47, 1, 15).

Solución:

a) **Capacidad del disco**. Tendremos que averiguar cuantos sectores tiene el disco y después multiplicar la cifra obtenida por el tamaño del sector. Así:
Núm. sectores = Núm. caras * Núm. cilindros * Sectores / pista
2.880 = 2 * 80 * 18
La capacidad del disco en Kb sería: 2.880 * 0.5 = 1.440 Kb ya que el tamaño de sector es de medio Kb.

b). **Dirección física del bloque lógico 1234**. Suponemos un solo sector por bloque. Las direcciones físicas tienen la siguiente estructura:
(núm. cilindro, núm. cara, núm. sector)
Además, la numeración se sigue del siguiente modo: hasta que no se han numerado todos los sectores de una cara no se pasa a la siguiente cara del mismo cilindro y hasta que no se hayan numerado todas las caras de un cilindro no se pasa al siguiente.
Por tanto, para obtener la dirección física efectuaremos lo siguiente:
(a) Dividiremos la dirección lógica por el número de sectores por pista. O sea:

1234 div 18 = 68
1234 mod 18 = 10

Esto quiere decir que con la dirección lógica 1234 se han podido "llenar" 68 caras y se está referenciando el undécimo sector (ya que empezamos a numerarlos a partir de cero) de la sexagésima novena cara.

(b) Dividiremos el número de caras "completas" obtenido en el punto anterior por el número de caras por cilindro que admite el disco. Así:

68 div 2 = 34
68 mod 2 = 0

Esto se interpreta como que hemos podido "completar" 34 cilindros y estamos en la primera cara del trigésimo-quinto. Por tanto, la dirección física pedida es: (34, 0, 10) Es decir, cilindro 34, cara 0, sector 10.

c) **Dirección lógica para la dirección física (47, 1, 15)**.
Suponiendo direcciones físicas de la forma (i, j, k), basta con aplicar la fórmula:

$$nb = k + sec/pista * (j + caras/cilindro * i)$$

El resultado obtenido es:

$$nb = 15 + 18 * (1 + 2 * 47) = 15 + 18 * 95 = 15 + 1710 = 1725$$

Con lo que el número de bloque lógico es el 1725

Problema 5b

Se dispone de un disco de 1024 cilindros, 16 cabezas, 12 sectores por pista y 512 bytes por sector, donde cada bloque lógico ocupa 2 sectores.

a) Indicar los sectores en formato (cilindro, cabeza, sector) que se corresponden al bloque lógico 30773.
b) Indique el bloque lógico al que corresponde la dirección (3,3,4) que sigue el formato del apartado a.

Solución:

a) 30773 div (12/2) =5128
30773 (mod (12/2))*2=10

5128 div 16= 320
5128 mod 16= 8

cilindro 320, cabezal 8 y sectores 10 y 11

b) ((3x192)+(3x12)+4)/2= bloque lógico 308

Problema 6

Dado un disco con 500 cilindros, 32 sectores/pista y donde cada bloque lógico agrupa 4 sectores de 512 bytes, se pide:

a) Calcular el número de caras que debe tener este disco para que la dirección en el formato (cilindro, cabeza, sector) (390, 10, 10) pueda corresponderse con el número de bloque 50002.
b) Indique el resto de direcciones de el citado formato que se corresponden con el mismo número de bloque

Solución:

a) Hay 32 sectores= 8 bloques por pista
50002= (390 * caras * 8)
caras= 50002 DIV (8 bloques por pista * 390) = 16 caras.

b) (390, 10, [8-11]

Ejercicio 1

Sea un sistema con los procesos que se muestran en la tabla .Calcular el tiempo de retorno, de espera y de respuesta de cada uno de los trabajos y representar su evolución temporal para los siguientes algoritmos de planificación.

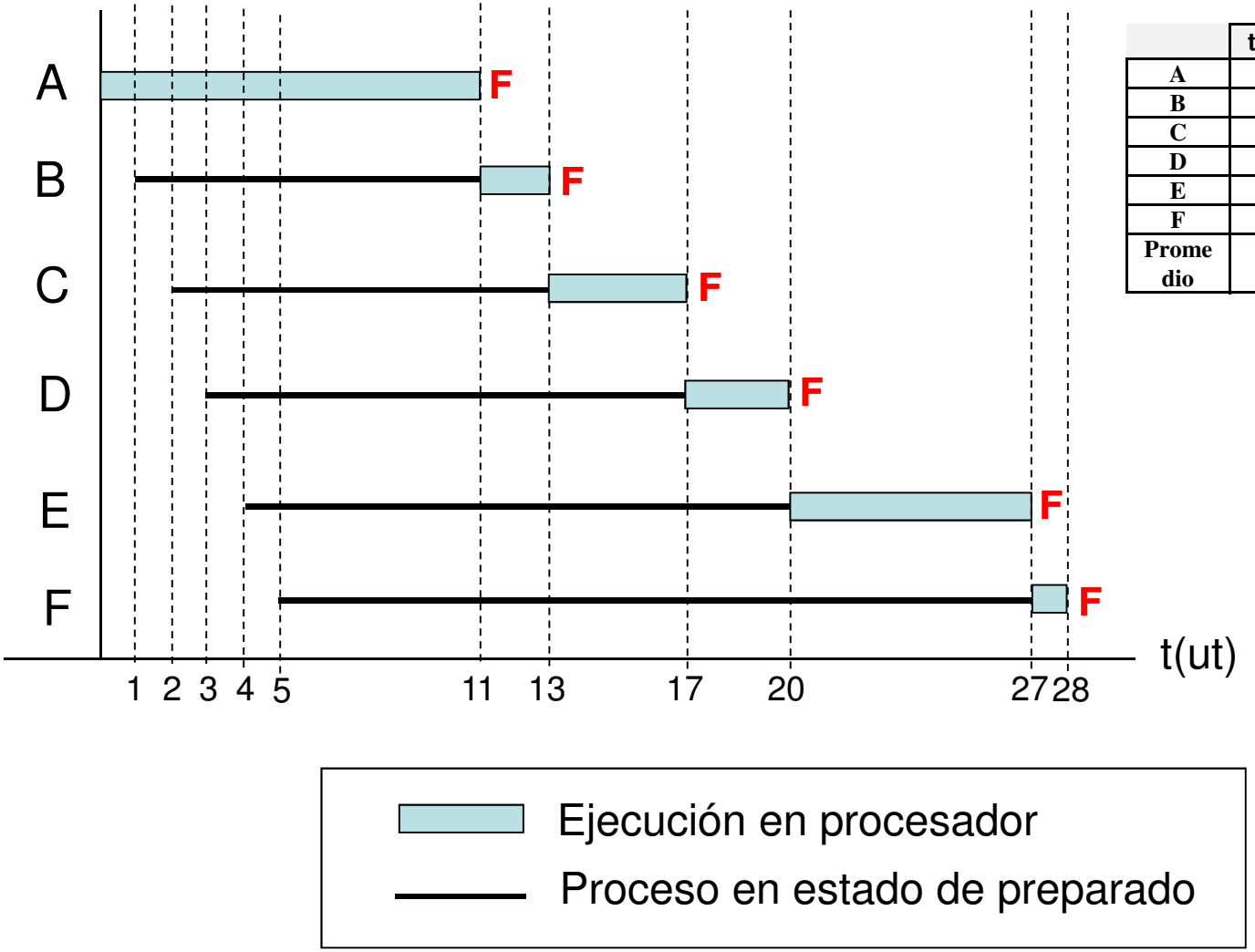
a) FCFS

b) SJF

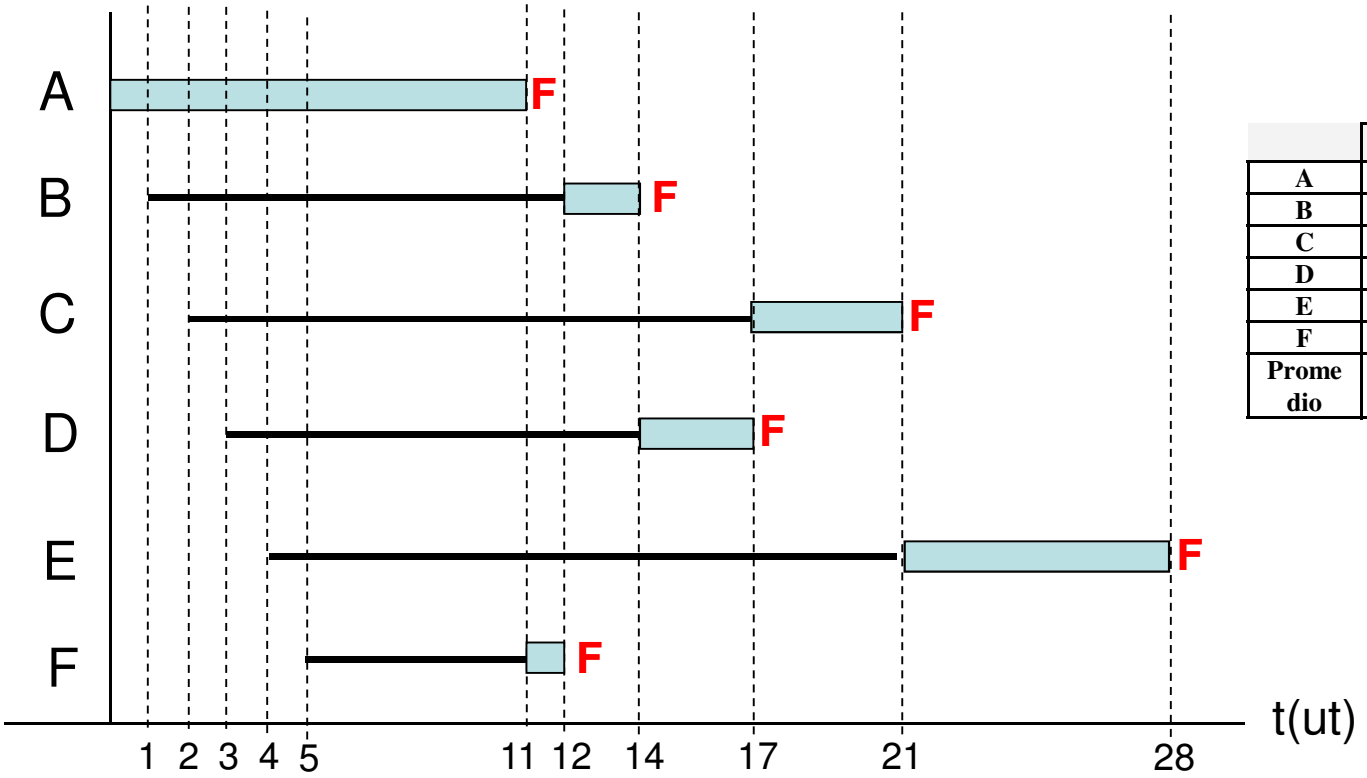
c) SRT

Proceso	Tiempo de llegada (ms)	Tiempo de ejecución (ms)
A	0	11
B	1	2
C	2	4
D	3	3
E	4	7
F	5	1

a) FCFS



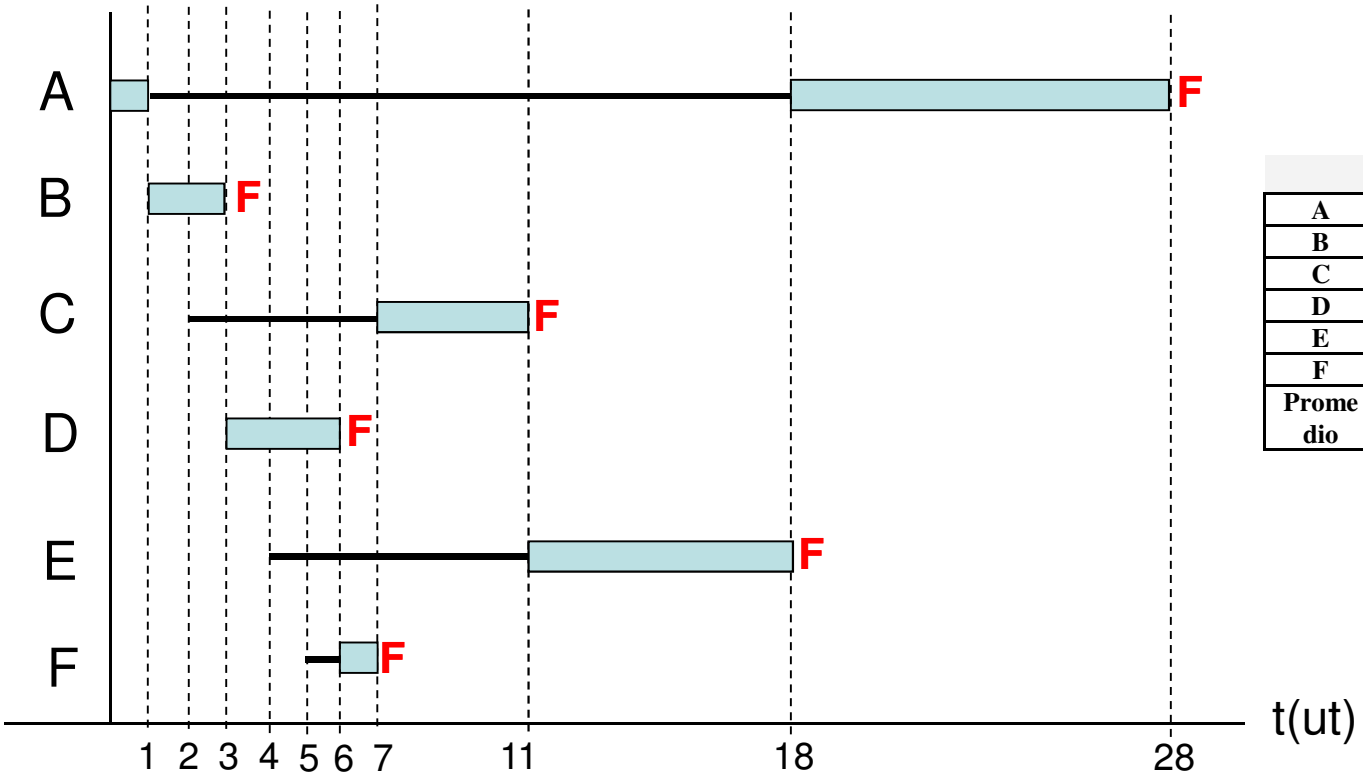
	t. respuesta	t. retorno	t.espera
A	0	11	0
B	10	12	10
C	11	15	11
D	14	17	14
E	16	23	16
F	22	23	22
Promedio	12,16	16,83	12,16



b) SJF

	t. respuesta	t. retorno	t.espera
A	0	11	0
B	11	13	11
C	15	19	15
D	11	14	11
E	17	24	17
F	6	7	6
Promedio	10,00	14,6	10

c) SRT

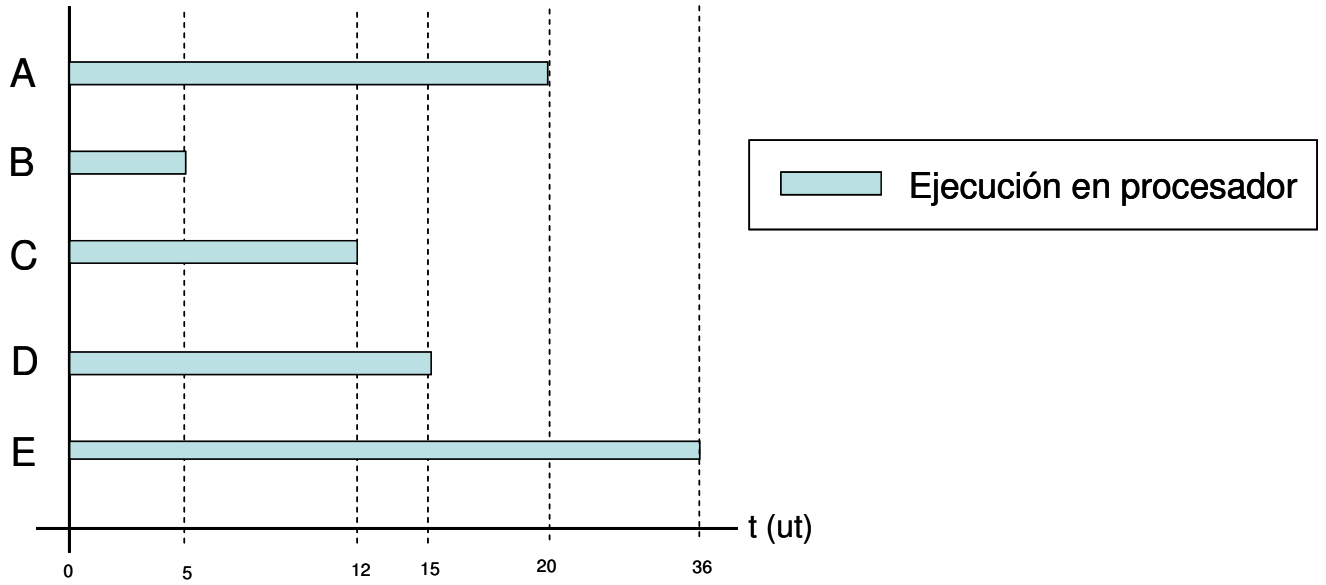


	t. respuesta	t. retorno	t. espera
A	0	28	17
B	0	2	0
C	5	9	5
D	0	3	0
E	7	14	7
F	1	2	1
Promedio	2,16	9,66	5,00

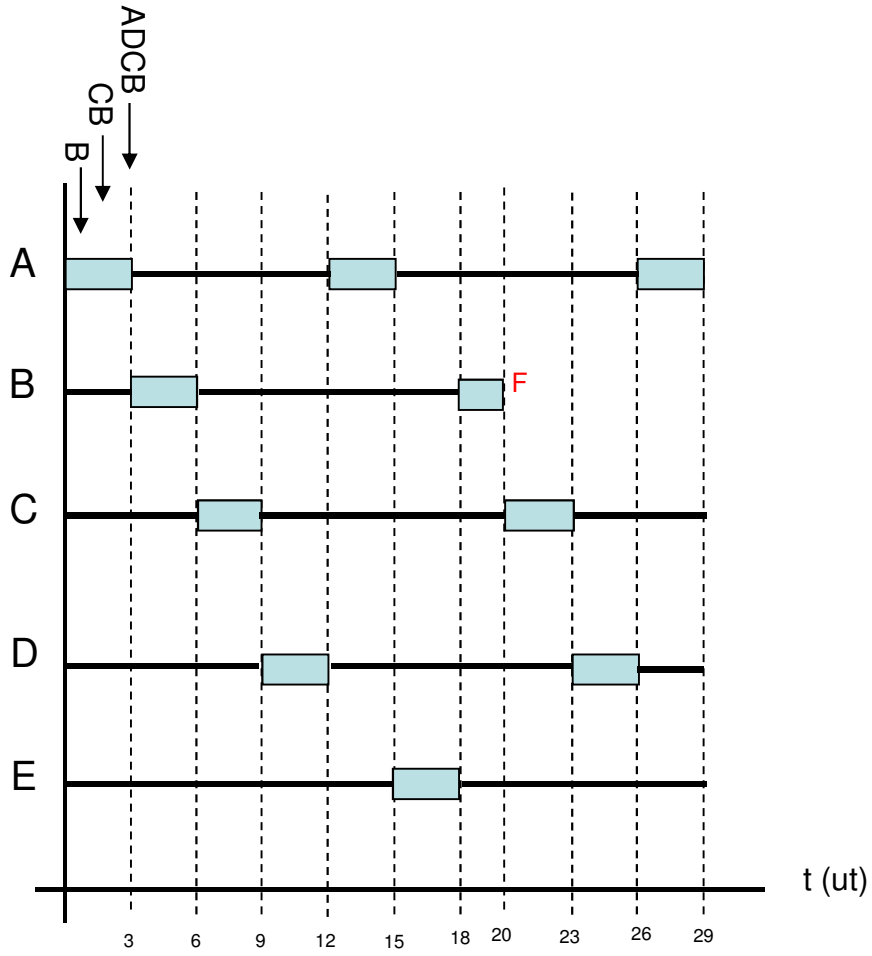
Existe expropiación en t=1 pero no en t=5

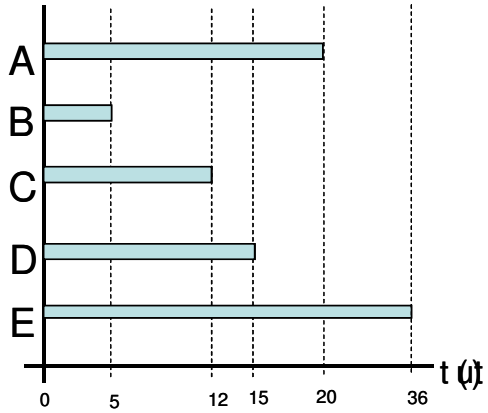
Ejercicio 2

Se dispone de un sistema monoprocesador con política de planificación del procesador RR con q = 3 ut. La ejecución de los procesos al sistema sigue el esquema descrito en la figura. Si el quantum de un proceso en ejecución expira a la vez que la llegada de un nuevo proceso, entonces el nuevo proceso se añade a la cola de procesos en espera de ejecutarse antes que el proceso que termina. El proceso A llega al sistema en el instante 0 ut., el proceso B en el instante 1 ut., el proceso C en 2 ut., el proceso D en 3 ut. y el proceso E en 4 ut.

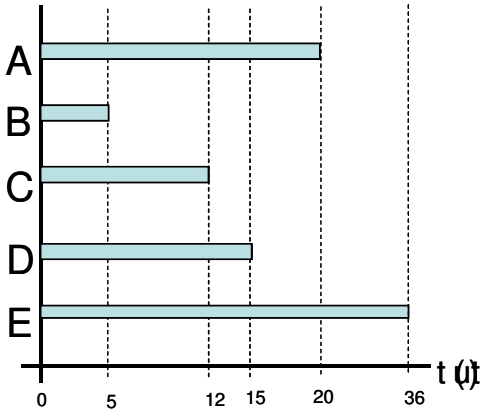


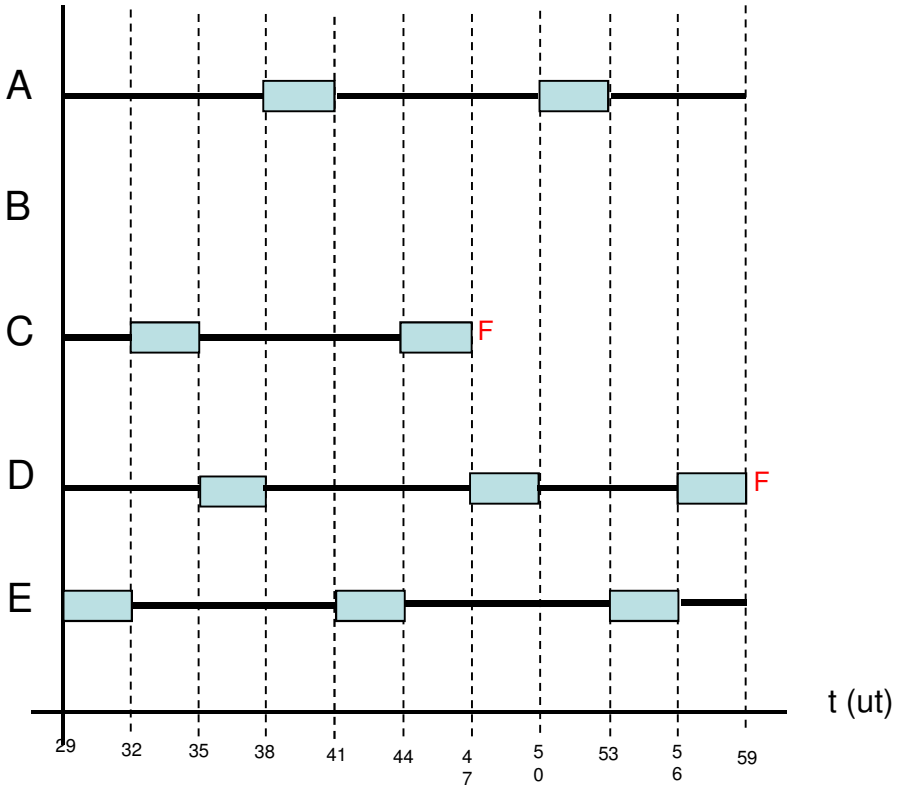
Solución:



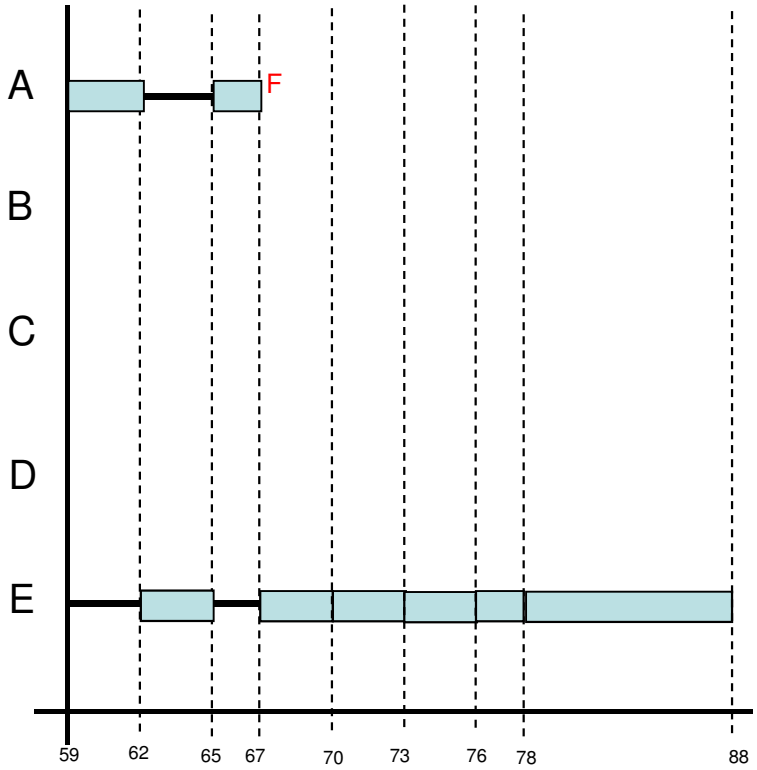


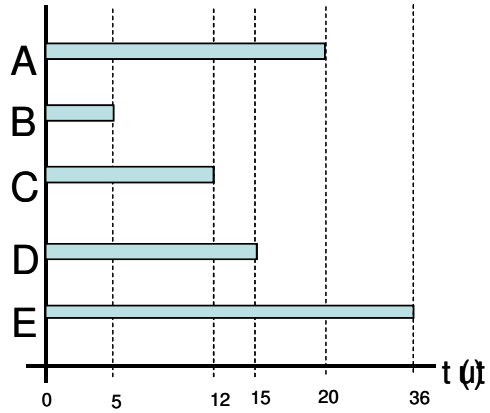
Solución:



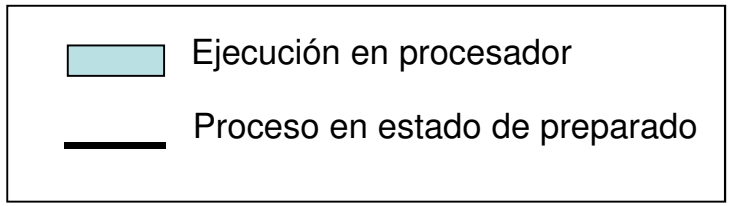


Solución:



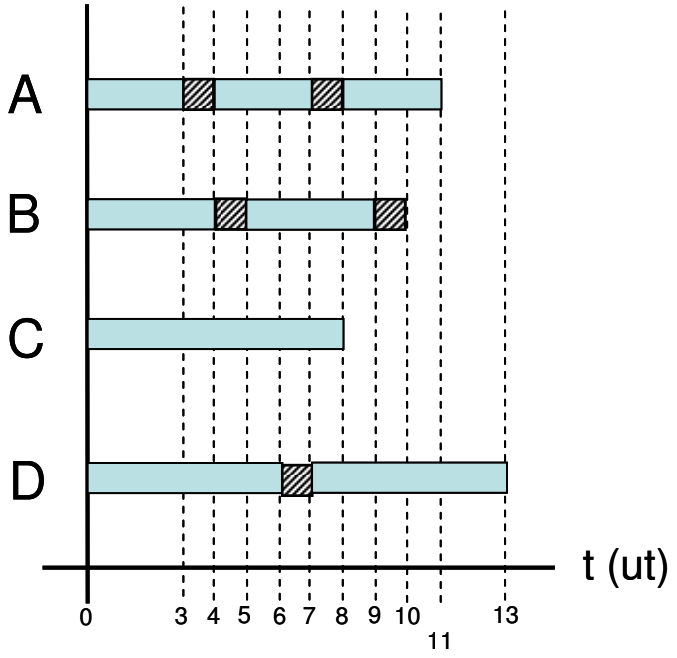


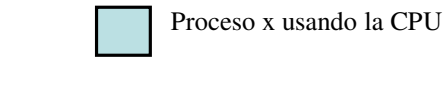
	t. retorno	t. respuesta	t. espera
A	67	0	47
B	19	2	14
C	45	4	33
D	56	6	41
E	84	11	48

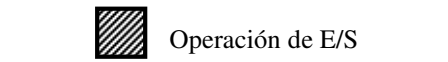


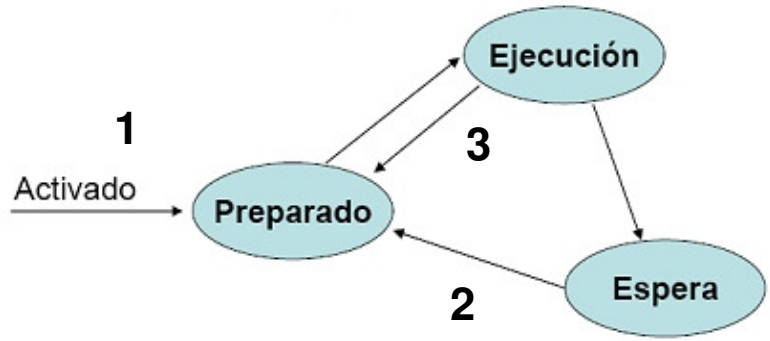
Ejercicio 3

Se dispone de un sistema monoprocesador con política de planificación del procesador RR con q = 1 ut y con gestión de los dispositivos de E/S FCFS. La ejecución de los procesos al sistema sigue el esquema descrito en la figura. Si el quantum de un proceso en ejecución expira a la vez que la llegada de un nuevo proceso, entonces el nuevo proceso se añade a la cola de procesos en espera de ejecutarse antes que el proceso que termina. El proceso A llega al sistema en el instante 0 ut., el proceso B en el instante 2 ut., el proceso C en 4 ut. y el proceso D en 6 ut.



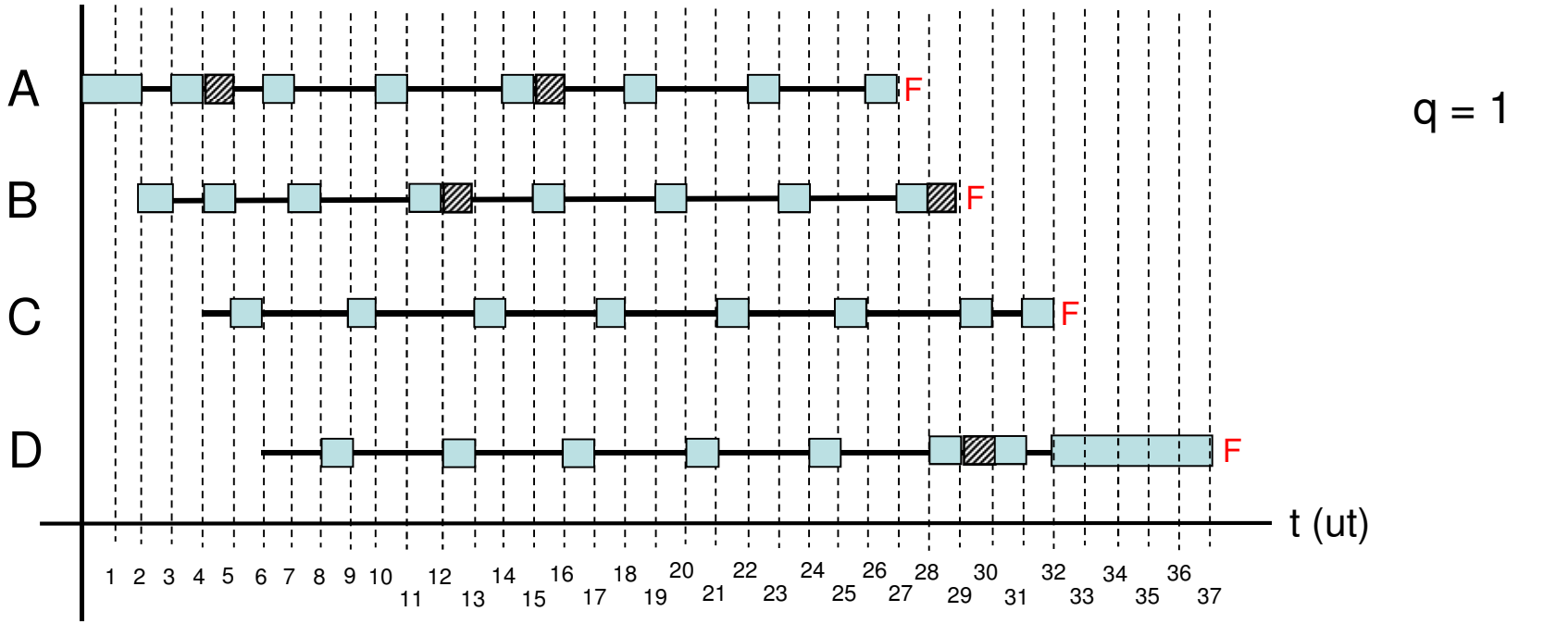




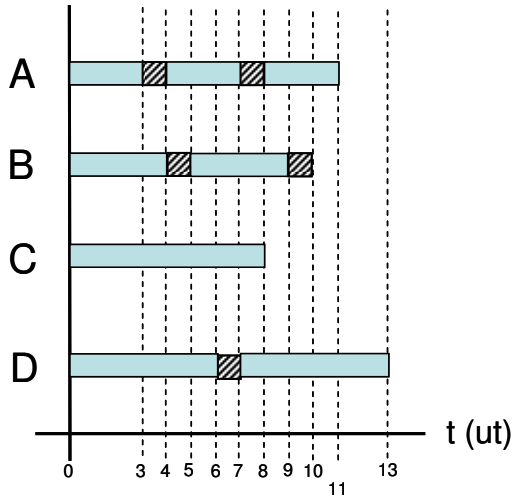


Nota prioridades

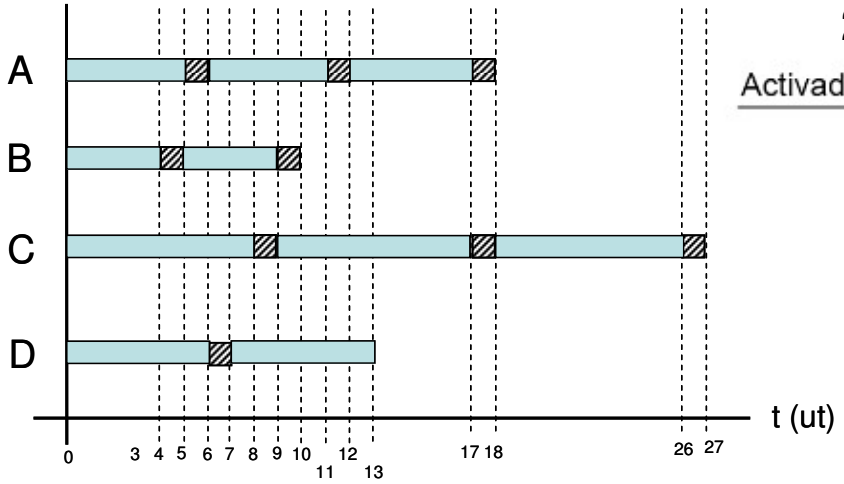
Solución:



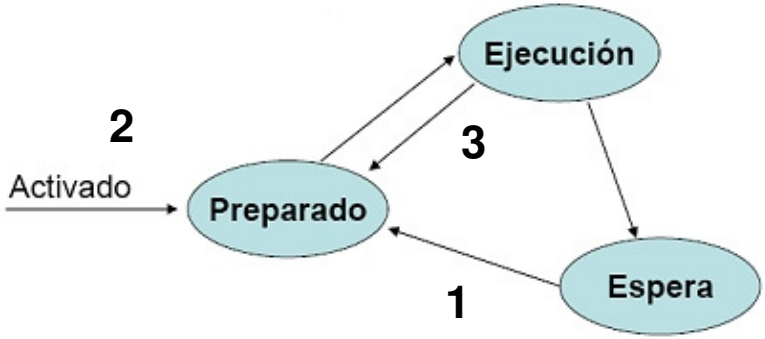
	t. retorno	t. respuesta	t. espera
A	27	0	16
B	27	0	17
C	28	1	20
D	31	2	18



Ejercicio 4

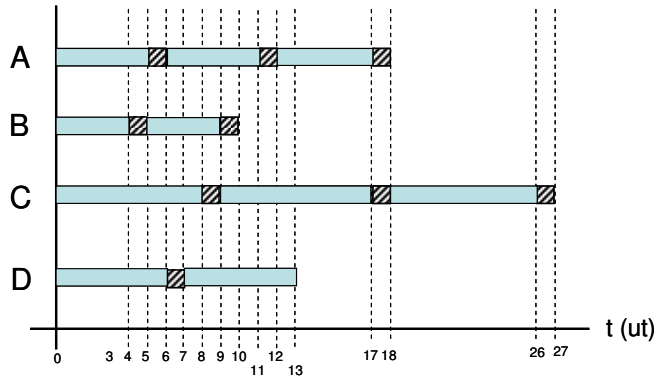


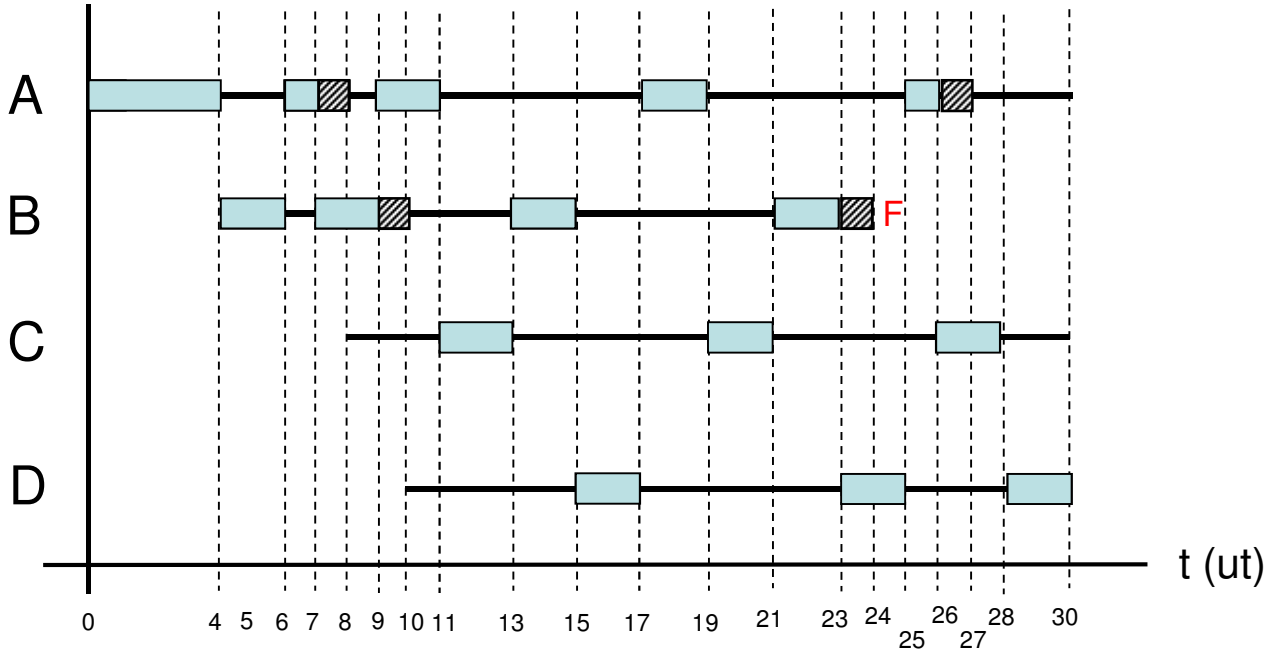
Nota prioridades



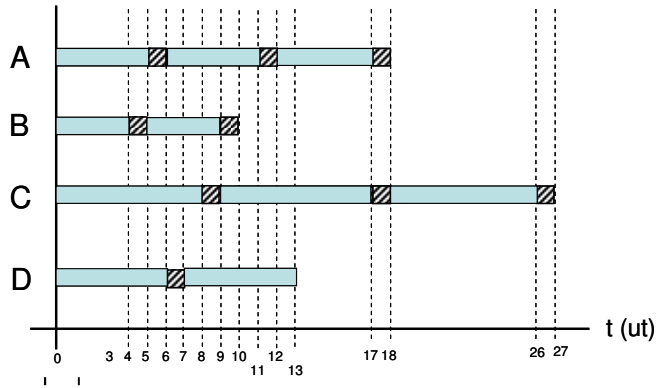
Se dispone de un sistema monoprocesador con política de planificación del procesador RR con q = 2 ut y con gestión de los dispositivos de E/S FCFS. La ejecución de los procesos al sistema sigue el esquema descrito en la figura. Si el quantum de un proceso en ejecución expira a la vez que la llegada de un nuevo proceso, entonces el nuevo proceso se añade a la cola de procesos en espera de ejecutarse antes que el proceso que termina. El proceso A llega al sistema en el instante 0 ut., el proceso B en el instante 4 ut., el proceso C en 8 ut. y el proceso D en 10 ut.

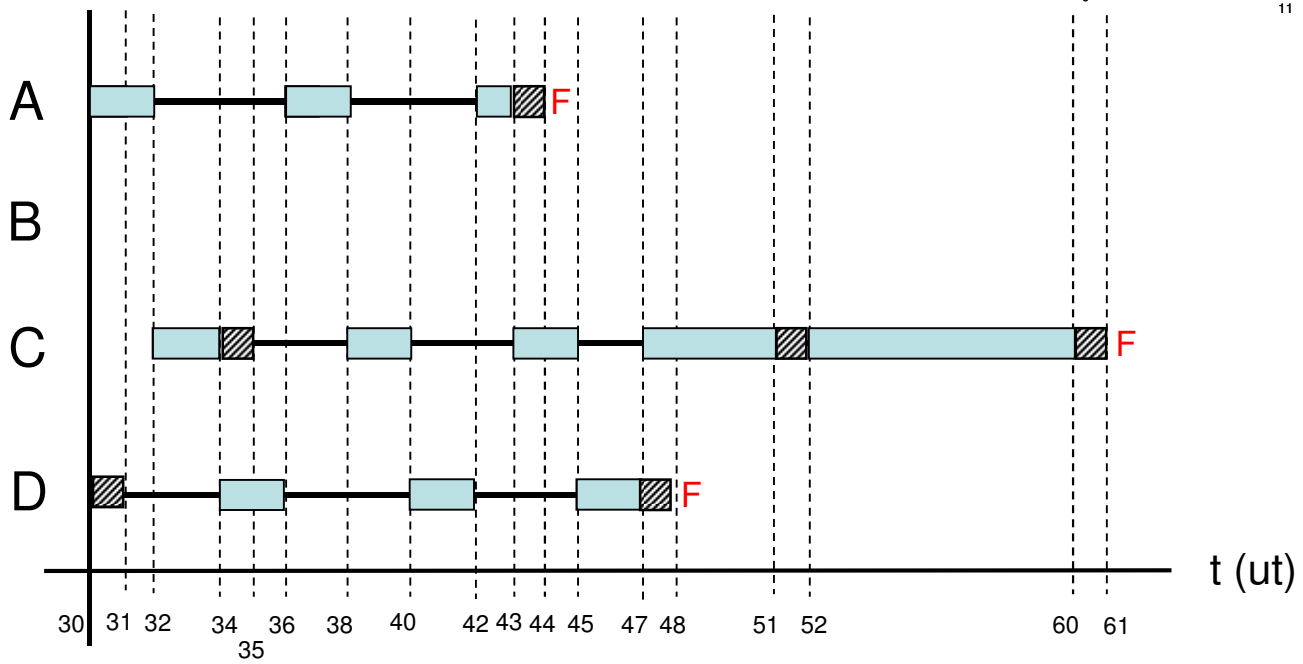
Solución:





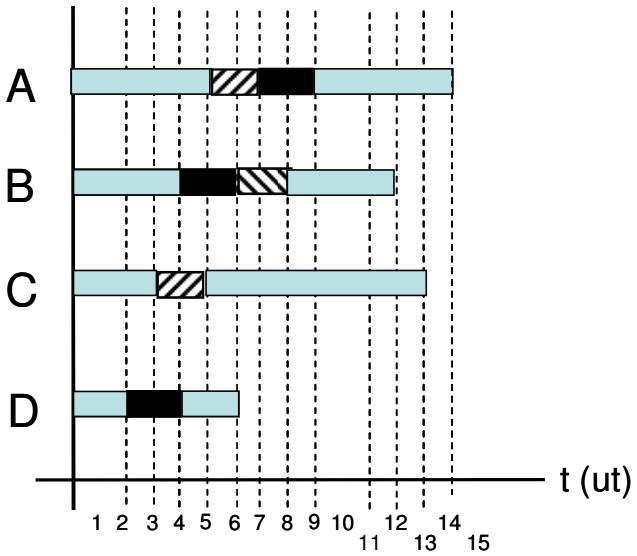
Solución:





	t. retorno	t. respuesta	t. espera
A	44	0	26
B	20	0	10
C	53	3	26
D	38	5	25

Ejercicio 5



El instante de llegada de los procesos y la cola a la que pertenecen

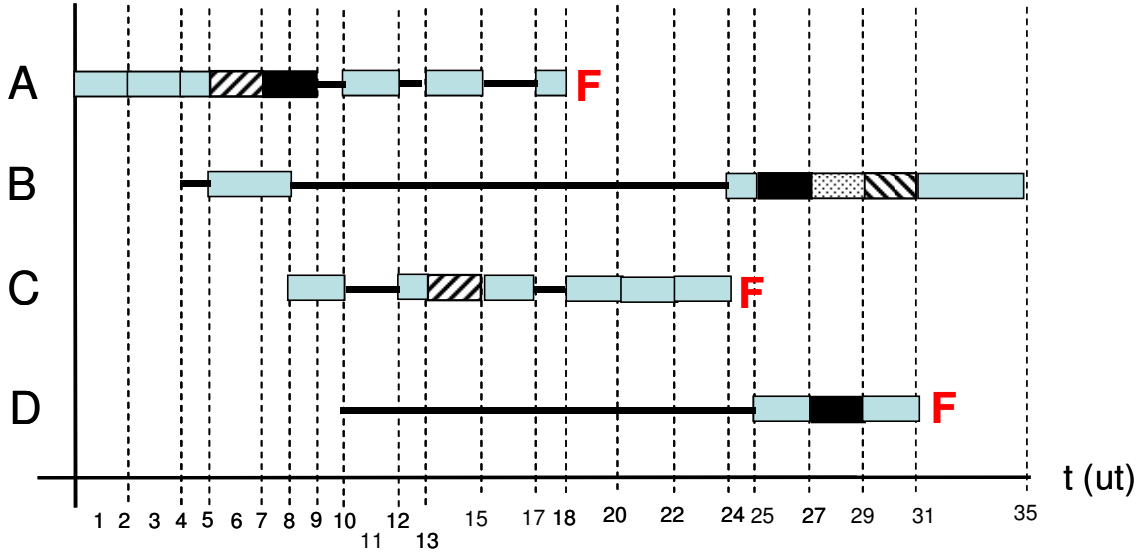
Proceso	Tª ini	Cola
A	0	1
B	4	2
C	8	1
D	10	2

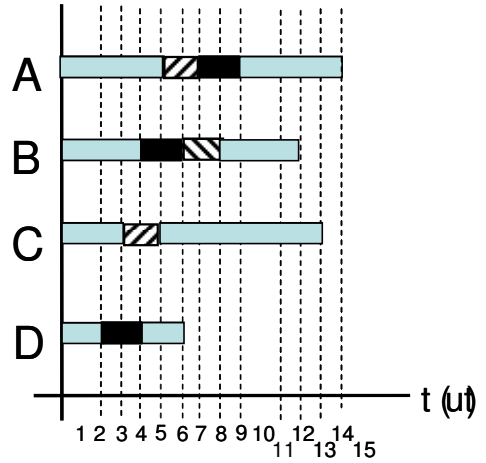
Dispositivo 1
Dispositivo 2
Dispositivos 1 y 2 simultáneamente

Se dispone de un sistema monoprocesador con política de planificación del procesador MLQ con dos colas, la cola 1 es más prioritaria que la 2 y es expropiativo entre colas. La gestión de la cola 1 es RR con q = 2 ut, mientras que la gestión de la cola 2 es FCFS. Además, existe gestión de los dispositivos de E/S FCFS. La ejecución de los procesos al sistema sigue el esquema descrito en la figura. Si el quantum de un proceso en ejecución expira a la vez que la llegada de otro a la cola de preparados (nuevo o desde operación de E/S), entonces el proceso que llega se añade antes que el proceso que termina.

Solución:

Gestión multicolas sin realimentación en el que tenemos dos colas con distinta prioridad existe expropiatividad entre colas,
los procesos tienen requerimientos de acceso a los dispositivos acceso se debe realizar en exclusión mútua.





Proceso	Tª ini	Cola
A	0	1
B	4	2
C	8	1
D	10	2

t = 8,en ejecución el proceso B (cola 2) y llega el proceso C (cola 1) cola más prioritaria → el algoritmo es expropiativo, le quita la CPU al proceso B y entra directamente a ejecución el proceso C.

t = 15, el proceso A termina un quantum de ejecución, el proceso C termina una operación en el dispositivo 1 obtiene la CPU el proceso C por la restricción de la definición del problema Nuevo proceso = proceso que termina con dispositivo Aunque pertenezcan a la misma cola

t = 27, D está en ejecución y solicita una operación sobre los dispositivos 1 y 2, B termina una operación sobre los dispositivos y solicita otra. Se le conceden los dispositivos a D por ser el proceso que estab

Ejercicio 6

Se dispone de un sistema monoprocesador con política de planificación MLFQ. La configuración de las colas es la siguiente:

Cola 0: Planificación FCFS.
Cola 1: Planificación RR con quantum = 100 ut.
Cola 2: Planificación RR con quantum = 200 ut.
Cola 3: Planificación RR con quantum = 300 ut.

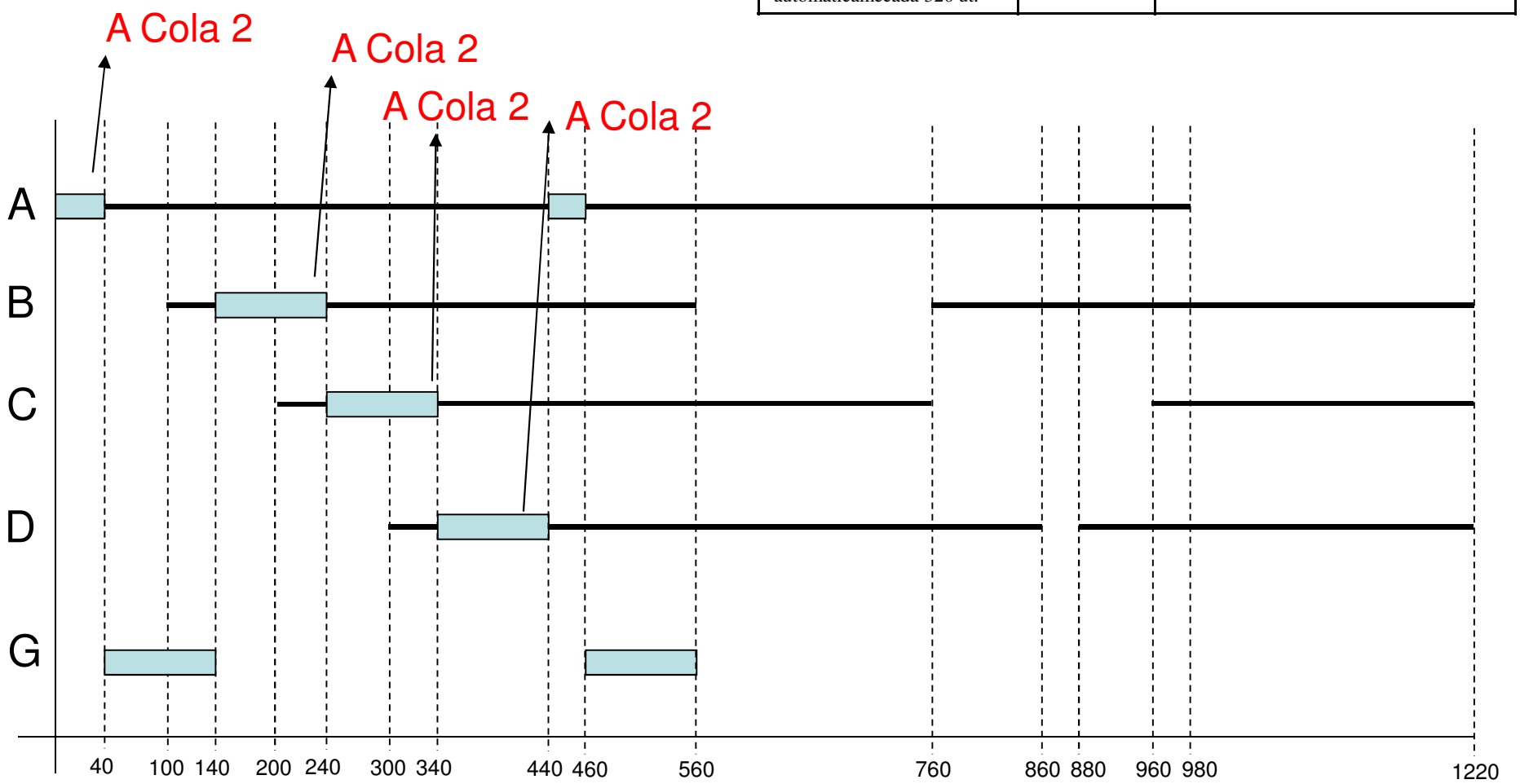
La prioridad de los procesos es decreciente con el número de cola. La llegada de los procesos al sistema sigue el siguiente esquema:

Instante de llegada	Tipo	Duración
0 ut.	A de usuario	700 ut. cada 300 ut. hace E/S de 100 ut.
100 ut.	B de usuario	500 ut. cada 500 ut. hace E/S de 100 ut.
200 ut.	C de usuario	600 ut. cada 200 ut. hace E/S de 100 ut.
300 ut.	D de usuario	700 ut. cada 600 ut. hace E/S de 100 ut.
40 ut. Este proceso se invoca automáticamente cada 320 ut. después de finalizar su ejecución.	G de sistema	100 ut..

Los procesos de sistema entran directamente a la cola 0 (cola más prioritaria). Los procesos de usuario entran a la cola 1. Cuando abandonan el procesador pasan a la cola siguiente de menor prioridad.

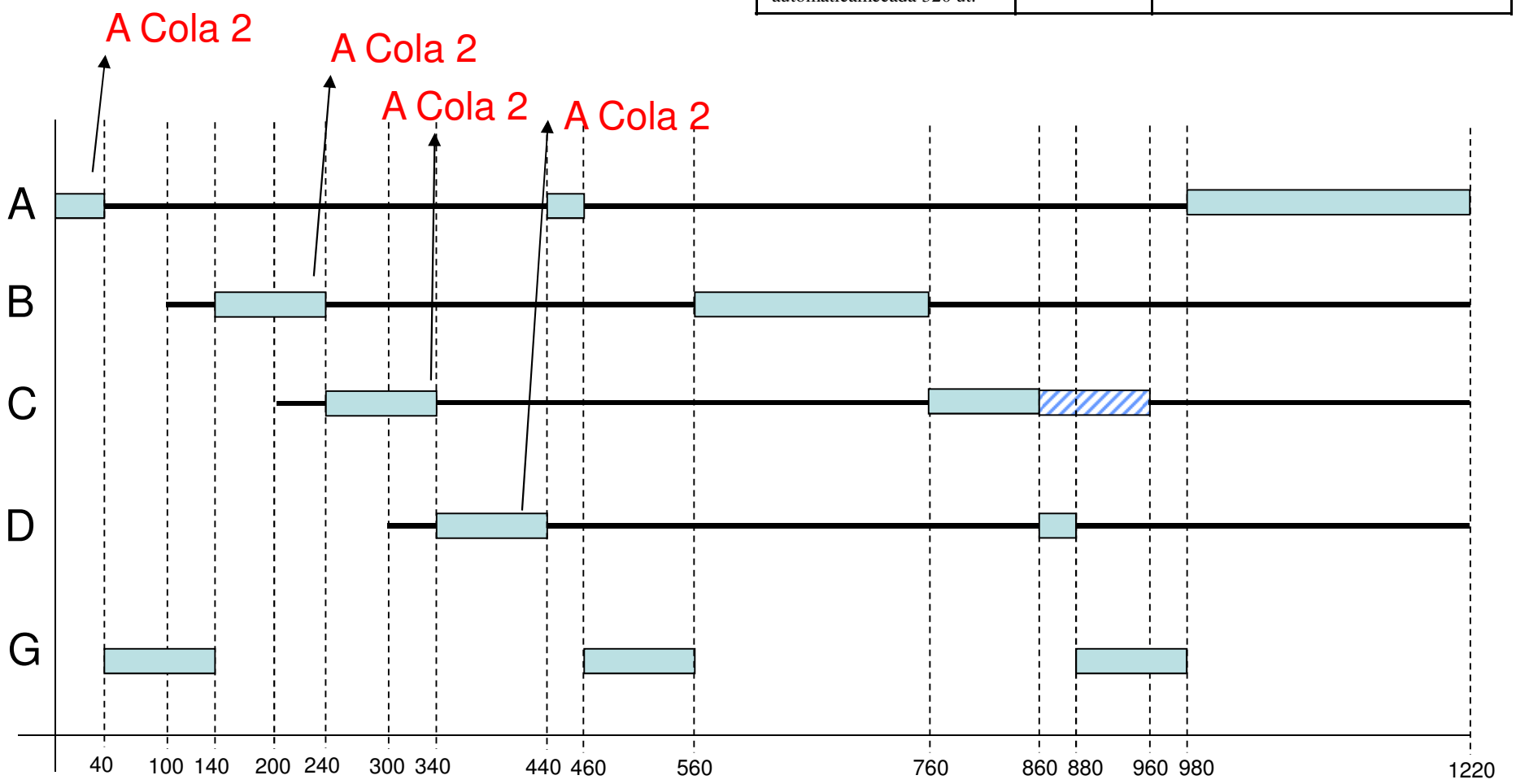
Mostrar la evolución temporal de los procesos del sistema señalando el estado en el que se encuentra cada proceso, así como la ocupación temporal de la CPU y de los dispositivos de E/S. Calcular los tiempos de respuesta, de retorno y de espera para cada uno de los procesos.

llegada	Tipo	Duración
0 ut.	A de usuario	700 ut. cada 300 ut. hace E/S de 100 ut.
100 ut.	B de usuario	500 ut. cada 500 ut. hace E/S de 100 ut.
200 ut.	C de usuario	600 ut. cada 200 ut. hace E/S de 100 ut.
300 ut.	D de usuario	700 ut. cada 600 ut. hace E/S de 100 ut.
40 ut. automáticamecada 320 ut.	G de sistema	100 ut..



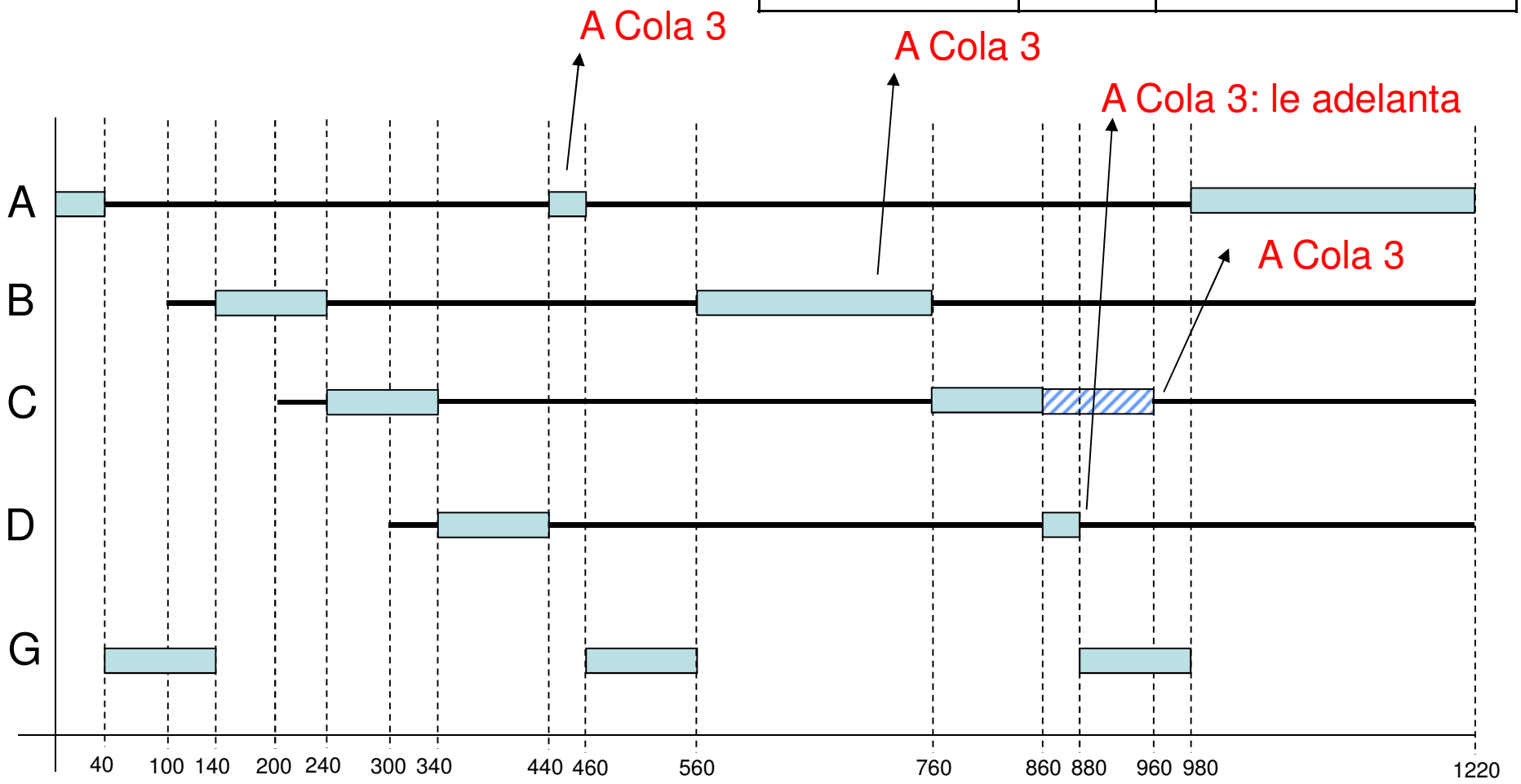
Primero dibujar los del sistema

llegada	Tipo	Duración
0 ut.	A de usuario	700 ut. cada 300 ut. hace E/S de 100 ut.
100 ut.	B de usuario	500 ut. cada 500 ut. hace E/S de 100 ut.
200 ut.	C de usuario	600 ut. cada 200 ut. hace E/S de 100 ut.
300 ut.	D de usuario	700 ut. cada 600 ut. hace E/S de 100 ut.
40 ut. automáticamecada 320 ut.	G de sistema	100 ut..



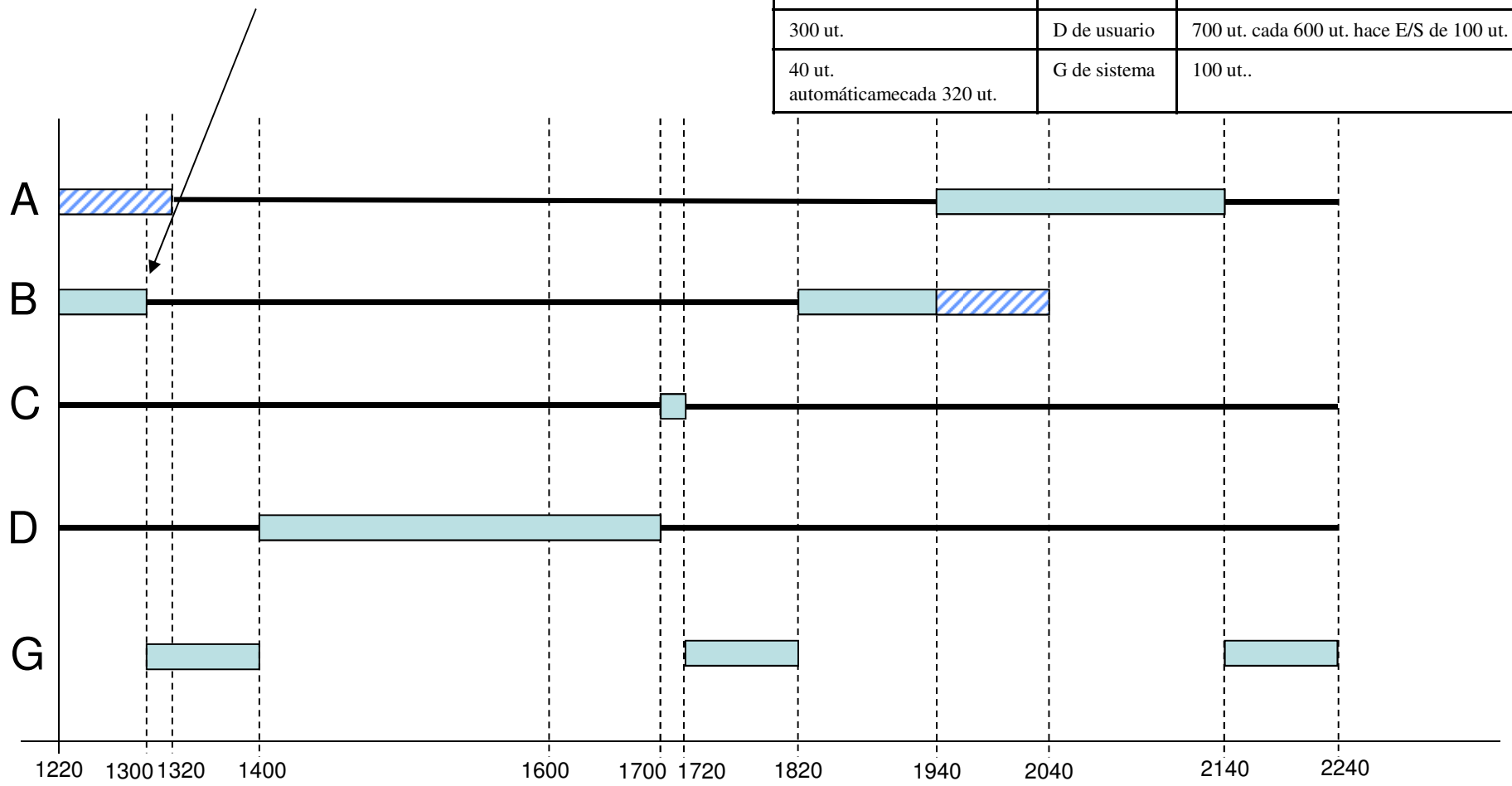
Primero dibujar los del sistema

llegada	Tipo	Duración
0 ut.	A de usuario	700 ut. cada 300 ut. hace E/S de 100 ut.
100 ut.	B de usuario	500 ut. cada 500 ut. hace E/S de 100 ut.
200 ut.	C de usuario	600 ut. cada 200 ut. hace E/S de 100 ut.
300 ut.	D de usuario	700 ut. cada 600 ut. hace E/S de 100 ut.
40 ut. automáticamecada 320 ut.	G de sistema	100 ut..

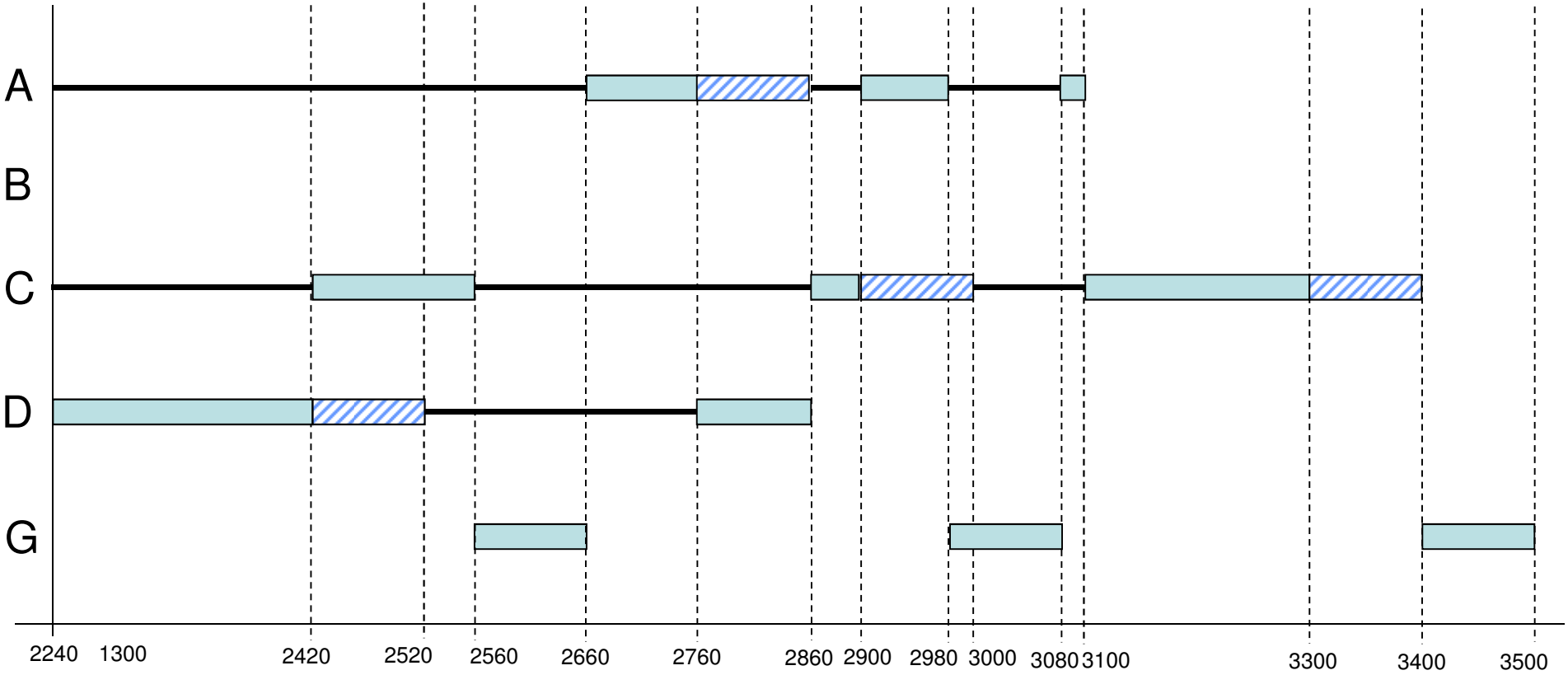


B adelanta a A (en procesos preparados)

llegada	Tipo	Duración
0 ut.	A de usuario	700 ut. cada 300 ut. hace E/S de 100 ut.
100 ut.	B de usuario	500 ut. cada 500 ut. hace E/S de 100 ut.
200 ut.	C de usuario	600 ut. cada 200 ut. hace E/S de 100 ut.
300 ut.	D de usuario	700 ut. cada 600 ut. hace E/S de 100 ut.
40 ut. automáticamecada 320 ut.	G de sistema	100 ut..



llegada	Tipo	Duración
0 ut.	A de usuario	700 ut. cada 300 ut. hace E/S de 100 ut.
100 ut.	B de usuario	500 ut. cada 500 ut. hace E/S de 100 ut.
200 ut.	C de usuario	600 ut. cada 200 ut. hace E/S de 100 ut.
300 ut.	D de usuario	700 ut. cada 600 ut. hace E/S de 100 ut.
40 ut. automáticamecada 320 ut.	G de sistema	100 ut..



Problemas resueltos en clase de sistema de ficheros y entrada/salida

Problemas de sistema de ficheros

Problema 1

Determinar el tamaño máximo de un fichero UNIX, especificando los bloques direccionados con cada tipo de puntero. Los parámetros corresponden a los siguientes valores:
Punteros a zonas de datos de 32 bits.
Tamaño de bloque 1K (1 zona = 1 bloque).
Estructura de nodo-i:

- 10 punteros directos.
- 1 puntero indirecto.
- 1 puntero indirecto doble.
- 1 puntero indirecto triple.

Realizar el mismo cálculo para un fichero en MINIX con los siguientes parámetros:
Punteros a zonas de datos de 16 bits.
Tamaño de bloque 1K (1 zona = 1 bloque).
Estructura de nodo-i:

- 7 punteros directos.
- 1 puntero indirecto.
- 1 puntero indirecto doble.

Solución:
Para determinar el tamaño máximo de un archivo en un sistema UNIX debemos tener en cuenta dos cifras:

- El número máximo de bloques que puede llegarse a mantener. Esto se determina a partir del tamaño de los punteros a zona, ya que no podremos tener una partición de disco mayor que este límite y un archivo no puede extenderse por más de una partición.
- Cuántos punteros a zona pueden llegarse a guardar a partir de un nodo-i, teniendo en cuenta los niveles de indirección que pueden mantenerse.

El tamaño máximo del archivo será el menor de estos dos valores.

UNIX.
Según el límite debido al tamaño de puntero, tenemos que el tamaño máximo sería:
tam. particióon = 2^{32} bloques = 4 Gbloques => **4 Tbytes**
Según el número de punteros tendremos:
En una misma zona podría llegar a haber 1024 /4 = 256 punteros a zona, ya que una zona tiene 1024 bytes y cada puntero ocupa 4.
Por tanto, el número de punteros a zona sería:

Tipo de puntero	Punteros directos
Directos en nodo-i.	10
Indirecto simple.	256
Indirecto doble.	$256^2 = 65.536$
Indirecto triple.	$256^3 = 16.777.216$
TOTAL:	16.843.018

Por tanto, esta cifra obtenida es menor que la anterior y el tamaño máximo del archivo sería de:

16.843.018 bloques = 16.842.018 Kb => **16'064 Gbytes**

MINIX.
Debido al tamaño de puntero, el tamaño máximo es en este caso:
tam. partición = 2^{16} bloques = 64 Kbloques => **64 Mbytes**
Según el número de punteros tendremos:
En una zona podría haber 1024 / 2 = 512 punteros a zona, ya que una zona tiene 1024 bytes y cada puntero ocupa 2.

Tipo de puntero	Punteros directos
Directos en nodo-i.	7
Indirecto simple.	512
Indirecto doble.	$512^2 = 262.144$
TOTAL:	262.663

Esto hace un total de 262.663 bloques =>**256'5 Mbytes**.
En este caso, el límite lo proporciona el tamaño de la partición. Con esto el tamaño máximo queda en **64 Mbytes**.

Problema 2

Determinar la estructura de un disco de 20 Mb en MINIX con los siguientes parámetros:

- Punteros a zonas de datos de 16 bits.
- Tamaño de bloque 1K (1 zona = 1 bloque).
- Tamaño de nodo-i de 32 bytes.
- Número máximo de nodos-i: 511.

Se pide:
a) Especifique claramente todas las estructuras de datos que forman el sistema de archivos y los bloques que ocupan.
b) En caso de resultar dañada la estructura del mapa de bits de zonas, piense la forma de reconstruirla con la información de la que se dispone en el resto de estructuras del sistema de archivos (se supone que estas se conservan inalteradas).

Solución:

a) Estructura del disco.
Un disco MINIX tiene las siguientes partes:
(a) Bloque de arranque: **1**.
(b) Superbloque: **1**.
(c) Mapa de bits para nodos-i: **1**.
Necesitamos un bit por archivo. Como el disco únicamente va a permitir 511 archivos, basta con 511 bits.
En este sistema un bloque tiene 1024 bytes (8192 bits), con lo que el mapa cabe en un solo bloque.
(d) Mapa de bits para zonas: **3**.
Un disco de 20 Mb tiene 20480 bloques de 1 Kb. En cada bloque caben 8 * 1024 bits.
Por tanto, se tendría:

$$nb = \frac{20 * 1024}{8 * 1024} = \frac{20}{8} = 2.5 \rightarrow 3$$

(e) Bloques para nodos-i: **16**.
Como cada nodo-i ocupa 32 bytes, en un bloque cabrían 1024 / 32 = 32 nodos-i.
Si vamos a tener 511 nodos-i, necesitaremos:

$$nb = \left\lceil \frac{511}{32} \right\rceil = 16$$

(f) Bloques para datos: **20458**.
Únicamente hay que descontar los bloques vistos en los puntos anteriores de la capacidad total del disco.

$$nb = 20480 - (1 + 1 + 1 + 3 + 16) = 20480 - 22 = 20458$$

b) Reconstrucción del mapa de bits de zonas.
Suponiendo que el resto del disco no ha resultado dañado, se podría realizar lo siguiente:
(a) Reinicializar todo el mapa de bits de zonas de manera que todos las zonas de disco aparezcan como libres.
(b) Siguiendo el mapa de bits para nodos-i, recorrer todos los nodos-i utilizados. Para cada uno de ellos habría que consultar sus punteros directos a zona y marcar tales zonas como ocupadas en el mapa de bits que se dañó.
Lo mismo para los punteros indirecto simple e indirecto doble, bajando hasta el nivel de zona de punteros directos. De esta manera todas las zonas utilizadas por todos los archivos son marcadas en el mapa de bits como "en uso".
Una vez terminados estos dos pasos, el mapa de bits para zonas estará recuperado.

Problema 3
Se tiene un dispositivo de almacenamiento de 32 Kb, con bloques de 512 bytes (1 zona = 1 bloque). Este dispositivo va a ser inicializado para poder utilizar un sistema de archivo similar al de UNIX con entradas de directorio de 16 bytes, un máximo de 31 archivos y nodos-i de 16 bytes de los cuales 60 bits pueden dedicarse a punteros directos a zona. Se supone que el tamaño de los punteros a zona es el mínimo posible.
Se pide:

a) Decidir cómo quedaría estructurado el disco. Es decir, cuántos bloques se dedican a cada una de las estructuras que mantiene el sistema de archivo.
b) ¿Cuál sería el tamaño máximo de un archivo?

Solución:

a) El disco tendrá (32*1024) / 512 = 64 bloques
1 bloque de arranque
1 superbloque
1 bloque para el mapa de nodos-i
Hay 32 nodos-i, por tanto hacen falta 31 bits para mapearlos (1bloque)
1 bloque para mapa de zonas de datos
Como máximo habrá 64bits (algunos no hacen falta por estar ya ocupados por bloques de arranque, superbloque…)
1 bloque para nodos-i
Hay 31 nodos-i con 16 bytes por cada uno. Luego 496 bytes
59 bloques para datos
De 64 bloques, hay 5 ya ocupados.

b) Tamaño máximo de archivo: Hay que saber cuántos bloques podemos referenciar con 60 bits que mantiene un nodo-i para punteros directos.
El tamaño de puntero a bloque será:
64 bloques -> 2^6 bloques -> punteros 6 bits
Por tanto: 60/6=10 punteros directos y cada bloque de 512 bytes-> **5Kb**

Problema 4

Sea un sistema de ficheros tipo MINIX con las siguientes características:

- Tamaño de bloque de 1Kbyte
- 1 zona = 2 bloques

- El tamaño del nodo-i es de 64 bytes y contiene las siguientes referencias a zonas: 10 referencias directas, 1 referencia indirecta y una referencia doblemente indirecta
- El tamaño de las referencias a zonas en disco es de 32 bits

Calcular el tamaño máximo que puede tener un fichero en este sistema de ficheros

Solución:

Es el mínimo de las siguientes cantidades:

a) $2^{32} * 2Kb = 2^{43} = 8$ Tbytes

b) $(10 + (2048/4) + /2048/4)^2) * 2Kb = 525.332$ Kb = 513 Mb (aprox)

Por tanto **513Mb**

Problemas planificación brazo disco

Problema 7

Dado un disco de cabeza móvil con 200 cilindros, numerados de 0 a 199 se considera que:

- Actualmente sirve una solicitud en el cilindro 143.
- Previamente se solicitó el acceso al cilindro 125.
- La cola de solicitudes se mantiene en orden FIFO: 86, 147, 91, 177, 94, 150, 102, 175, 130.

Se pide:

Determinar el movimiento total de la cabeza necesario para satisfacer estas solicitudes con los siguientes algoritmos de planificación de disco:

- FCFS.
- SSTF.
- SCAN.
- LOOK.
- C-SCAN.

Solución:

Algoritmo FCFS: Las peticiones se sirven en orden de llegada.

Cil. actual	Cil. a servir	Desplaz.	(cil.)
143	86	143 - 86 =	57
86	147	147 - 86 =	61
147	91	147 - 91 =	56
91	177	177 - 91 =	86
177	94	177 - 94 =	83
94	150	150 - 94 =	56
150	102	150 - 102 =	48
102	175	175 - 102 =	73
175	130	175 - 130 =	45
		TOTAL:	565

Algoritmo SSTF: Se sirve primero aquella petición que implique un menor desplazamiento. El asterisco (*) simboliza la posición actual de la cabeza de lectura/escritura. El cilindro a servir aparece subrayado.

Cil. actual	Cilindros pendientes	Desplaz.	(cil.)
143	86, 91, 94, 102, 130, *, <u>147</u>, 150, 175, 177	147 - 143 =	4
147	86, 91, 94, 102, 130, *, <u>150</u>, 175, 177	150 - 147 =	3
150	86, 91, 94, 102, <u>130</u>, *, 175, 177	150 - 130 =	20
130	86, 91, 94, <u>102</u>, *, 175, 177	130 - 102 =	28
102	86, 91, <u>94</u>, *, 175, 177	102 - 94 =	8
94	86, <u>91</u>, *, 175, 177	94 - 91 =	3
91	<u>86</u>, *, 175, 177	91 - 86 =	5
86	*, <u>175</u>, 177	175 - 86 =	89
175	*, <u>177</u>	177 - 175 =	2
		TOTAL:	162

Algoritmo SCAN: Ya que el último cilindro servido fue el 125 y nos encontramos en el 143, supondremos que la dirección de servicio es ascendente. Se realizan dos pasadas, la primera ascendente en la que servimos todos los cilindros entre la posición actual y

el final del disco (cilindro 199). Tras esto partimos del cilindro 199 en dirección descendente y servimos todos los cilindros pendientes.

Direc. servicio	Cilindros servidos	Desplaz.	(cil.)
Ascendente	147, 150, 175 y 177	199 - 143 =	56
Descendente	130, 102, 94, 91 y 86	199 - 86 =	113
		TOTAL:	169

Algoritmo LOOK: Igual que el anterior, pero al servir en orden ascendente no llegamos al final del disco sino que invertimos el sentido de servicio al llegar al cilindro 177.

Direc. servicio	Cilindros servidos	Desplaz.	(cil.)
Ascendente	147, 150, 175 y 177	177 - 143 =	34
Descendente	130, 102, 94, 91 y 86	177 - 86 =	91
		TOTAL:	125

Algoritmo SCAN circular (C-SCAN): Suponemos sentido de servicio ascendente. Al llegar al último cilindro del disco (el 199), volvemos al cilindro 0 sin servir ninguna petición y desde allí iniciamos de nuevo el servicio en orden ascendente.

Direc. servicio	Cilindros servidos	Desplaz.	(cil.)
Ascendente	147, 150, 175 y 177	199 - 143 =	56
Descendente	Ninguno	199 - 0 =	199
Ascendente	86, 91, 94, 102 y 130	130 - 0 =	130
		TOTAL:	385

Problema 8

Dado un disco de cabeza móvil con 200 cilindros, numerados de 0 a 199 y con un tiempo promedio de acceso (rotación+ transferencia) de 20 unidades de tiempo se trata de determinar el tiempo total de servicio que se requiere para atender las siguientes peticiones:
(0,30), (40,10), (45,40), (60, 60), (100,50), (120, 5), (140, 100), (160, 120)
donde la primera componente de cada petición se refiere al instante de tiempo en el que se efectúa dicha petición y la segunda componente indica el cilindro al que se pretende acceder. Se considera que el tiempo de posicionamiento entre cilindros contiguos es igual a 1 unidad de tiempo.
Aplicar para el cálculo del tiempo total de servicio los siguientes algoritmos:

- FCFS.
- SSTF.
- SCAN.
- LOOK.
- C-SCAN.
-

Nota: El cabezal del disco se encuentra inicialmente posicionado en el cilindro 0 y el servicio de las peticiones se realiza en el sentido de números de cilindro crecientes.

Solución:

Emplearemos las siguientes abreviaturas:

CA: Cilindro actual.
CS: Cilindro a servir.
Desp: Desplazamiento en cilindros desde el actual al que debe ser servido.
TS: Tiempo de servicio del cilindro a servir. Incluye posicionamiento, rotación y transferencia.

Su valor será igual, en este problema, a:

Algoritmo FCFS:

Inst.	CA	Cola petic.	CS	Desp	TS
0	0	30	30	30	50
50	30	10, 40	10	20	40
90	10	40, 60	40	30	50
140	40	60, 50, 5, 100	60	20	40
180	60	50, 5, 100, 120	50	10	30
210	50	5, 100, 120	5	45	65
275	5	100, 120	100	95	115
390	100	120	120	20	40
430	120	Vacía	-	-	-

Algoritmo SSTF:

Inst.	CA	Cola petic.	CS	Desp	TS
0	0	*, 30	30	30	50
50	30	10, *, 40	40	10	30
80	40	10, *, 60	60	20	40
120	60	5, 10, 50, *	50	10	30
150	50	5, 10, *, 100	10	40	60
210	10	5, *, 100, 120	5	5	25
235	5	*, 100, 120	100	95	115
350	100	*, 120	120	20	40
390	120	Vacía	-	-	-

Algoritmo SCAN: Suponemos que una vez se ha programado una operación SEEK sobre el controlador de disco no hay manera de abortarla. Además, cuando no queden peticiones en el sentido actual, se programa un SEEK hasta
el final del disco en ese sentido.

Inst.	CA	Cola petic.	CS	Desp	TS
0	0	→, 30	30	30	50
50	30	10, →, 40	40	10	30
80	40	10, →, 60	60	20	40
120	60	5, 10, 50, →	199	139	139
259	199	5, 10, 50, 100, 120, ←	120	79	99
358	120	5, 10, 50, 100, ←	100	20	40
398	100	5, 10, 50, ←	50	50	70
468	50	5, 10, ←	10	40	60
528	10	5, ←	5	5	25
553	5	Vacía	-	-	-

Algoritmo LOOK:

Inst.	CA	Cola petic.	CS	Desp	TS
0	0	→, 30	30	30	50
50	30	10, →, 40	40	10	30
80	40	10, →, 60	60	20	40
120	60	5, 10, 50, ←	50	10	30
150	50	5, 10, ←, 100	10	40	60
210	10	5, ←, 100, 120	5	5	25
235	5	→, 100, 120	100	95	115
350	100	→, 120	120	20	40

Algoritmo C-SCAN: Suponemos que una vez se ha programado una operación SEEK sobre el controlador de disco no hay manera de abortarla. Además, cuando no queden peticiones en el sentido actual, se programa un SEEK hasta el final del disco en ese sentido.

Inst.	CA	Cola petic.	CS	Desp	TS
0	0	→, 30	30	30	50
50	30	10, →, 40	40	10	30
80	40	10, →, 60	60	20	40
120	60	5, 10, 50, →	199	139	139
259	199	5, 10, 50, 100, 120, ←	0	199	199
458	0	→, 5, 10, 50, 100, 120	5	5	25
483	5	→, 10, 50, 100, 120	10	5	25
508	10	→, 50, 100, 120	50	40	60
568	50	→, 100, 120	100	50	70
638	100	→, 120	120	20	40
678	120	Vacía	-	-	-

Problema 9

Se sabe que un disco duro posee 500 cilindros (enumerados del 0 al 499). Se sabe que el cabezal ha descrito una trayectoria que viene dada por los siguientes números de cilindro: 135, 150, 195, 330, 410, 450, 10, 25, 30 y 50.
Indique si es posible que el manejador de dispositivo haya utilizado cada uno de los algoritmos que se muestran a continuación, en caso afirmativo indique bajo que condiciones se ha podido realizar esta: posición inicial del cabezal, dirección del cabezal (Ascendente y/o Descendente) y cola inicial de peticiones. Se supone que no llegan más peticiones después del instante inicial.

Solución:

Algoritmo	¿Posible? (Si/No)	Cilindro Inicial	Dirección Inicial	Cola Inicial de peticiones
FCFS	SI	Indiferente	Indiferente	135, 150, 195, 330, 410, 450, 10, 25, 30 y 50
SSTF	NO	Si se hubiera utilizado SSTF, tras la petición 450 se habría servido la 50, en lugar de la 10.		
LOOK	NO	Si se hubiera utilizado LOOK, tras la petición 450 se habría servido la 50, en lugar de la 10.		
C-LOOK	SI	51-134	Ascendente	Es indiferente el orden de los números de cilindros citados en el enunciado

Problema 10

Se tiene un disco de cabeza móvil con 100 cilindros, numerados de 0 a 99 y, un tiempo promedio de acceso (latencia+transferencia) de 10 unidades de tiempo y un tiempo de posicionamiento entre pistas contiguas de 1 unidad de tiempo. Indique el recorrido del cabezal del disco para las siguientes peticiones:

(0,40) (0,80) (5,60) (5,80) (10,35) (40,20) (50,60) (69,90)

con los algoritmos:

a) SSTF

b) SCAN
c) LOOK

Suponga que el cabezal se encuentra inicialmente sobre el cilindro 50 y el servicio de las peticiones se realiza en sentido ascendente.

Solución:

SSTF	50,40,35,60,60,80,80,90,20
SCAN	50,80,80,99,90,60,60,40,35,20
LOOK	50,80,80,60,60,40,35,20,90

Solución Ejercicios T4. Semáforos

Ejercicio 1

En un comedor de un colegio hay n alumnos. Cuando un alumno, que normalmente está estudiando en su clase, tiene hambre va al comedor y se acerca a alguna de las tres barras para recoger el menú. En estos menús, que varían cada día, sólo se puede elegir o cambiar el postre o café. Para el postre hay dos mostradores mientras que para el café hay un único mostrador. Realizar un programa que, usando semáforos, coordine las peticiones de los alumnos en el comedor.

Solución

Este es un problema en el que el número de recursos es limitado y definido, tres barras, dos de postre y una de café.

Como podemos observar, se utilizan cuatro semáforos: el semáforo *comedor* asociado a la sala y que se inicializa a n, indicando que tiene una capacidad de n comensales; *menu* inicializado a 3, representando a las tres barras donde se sirven menús; *postre* inicializado a 2, indicando las dos barras donde se sirven postres; y el semáforo *cafe* inicilizado a 1, simulando la única barra donde existe café.

```
#define alumnos n
TSemaforo comedor, menu, postre, cafe;
bool postre_cafe;

void alumno()
{
  while (true)
  {
    P(comedor);
    entrar_en_comedor();
    P(menu);
    coger_menu();
    V(menu);
    decidir_postre_cafe();
    if (postre_cafe == postre)
    {
      P(postre);
      coger_postre();
      V(postre);
    }
    else
    {
      P(cafe);
      coger_cafe();
      V(cafe);
    }
    comer();
    V(comedor);
  }
}

Void main()
{
  inicializar(comedor, n);  inicializar(menu, 3);
  inicializar(postre, 2);  inicializar(cafe, 1);
  cobegin
    alumno1();alumno2(); … alumnom();
  coend;
}
```

Observemos como en los semáforos *menú*, *postre* y *cafe* se ejecuta la operación P justa antes de llevar a cabo la acción correspondiente y la operación V inmediatamente después, para liberar la barra y que otro compañero pueda acceder. En cambio, en el semáforo *comedor* se ejecuta la operación P al entrar en el comedor y la operación V al salir del mismo, asegurando de este modo que no se superará la capacidad del comedor.

Ejercicio 2

Un parque de atracciones decide usar semáforos en sus actividades de mayor afluencia de público. En concreto en las pistas blandas (con 5 pistas) y el kamikaze(con dos pistas). Además, por exigencias de seguridad, se obliga a que no haya más de 2000 personas en el parque, todos deben de ducharse antes de entrar en estas actividades que se realizarán en el orden que se han descrito. Realizar un programa con semáforos que cumpla las exigencias descritas del parque acuático.

Solución

Como es fácilmente observable, este problema guarda muchas similitudes con el anterior, también es un problema de recursos limitados y definidos, 2000 personas de capacidad máxima en el parque, cinco pistas blandas, dos kamikaze y número de duchas indefinido.

Como en el caso anterior, asociaremos un semáforo a cada uno de los recursos anteriores y trabajaremos con la inicialización de mismos. También como en el problema anterior, se puede ver como en la utilización de las atracciones el semáforo correspondiente se cierra y se libera (operaciones P y V) antes y después del uso de la atracción. En cambio, en el semáforo parque la operación P se ejecuta al entrar y la operación V al salir, garantizando en todo momento no exceder la capacidad del parque.

Observemos la estructura secuencial del código que garantiza que las actividades se realizan en el orden en que se han descrito, en cumplimiento de la definición del problema.

```
#define duchas n
TSemaforo blandas, kamikaze, parque, duchas;

void clientei()
{
  while (true)
  {
    P(parque);
    entrar_en_el_parque();
    P(duchas);
    ducharse();
    V(duchas);
    P(blandas);
    atraccion_pistasblandas();
    V(blandas);
    P(kamikaze);
    Atraccion_kamikaze();
    V(kamikaze);
    V(parque);
  }
}

Void main()
{
  inicializar(parque, 2000);
  inicializar(duchas, n);
  inicializar(blandas, 5);
  inicializar(kamikaze, 2);
  cobegin
    cliente1(); cliente2(); … clientem();
  coend;
}
```

Ejercicio 3

En el almacén de una empresa de logística existen distintos toritos que pueden ser utilizados por personal del almacén que mueve los bultos internamente y por personal de los muelles que carga los camiones de transporte. Supongamos que la gestión de los toritos se implementa de la siguiente forma:

```
TSemáforo sc, mutex;
int numero;
```

```
void muelle_i()
{
  . . .
  P(sc);
  coger_torito();
  V(sc);
  . . .
}
```

```
void almacen_j()
{
  . . .
  P(mutex);
    numero++;
    if (numero==1) P(sc);
  V(mutex)
  coger_toriro();
  P(mutex);
    numero--;
    if (numero==0) V(sc);
  V(mutex)
  . . .
}
```

```
void main()
{
  inicializar(mutex, 1);
  inicializar(sc, 1);
  numero=0;
  cobegin
    muelle_1();…; muelle_n(); almacen_1(); … ; almacen_m();
  coend;
}
```

1) ¿Pueden utilizar a la vez los toritos el personal del almacén y del muelle?
2) Si hay un señor del almacén utilizando un torito, ¿puede otro del almacén utilizar algún torito simultáneamente?
3) Si una persona de los muelles está cargando, ¿puede otra persona de los muelles cargar al mismo tiempo?
4) Indicar si existe algún tipo de prioridad. En caso afirmativo, realizar las modificaciones necesarias para que no exista prioridad.

Solución

En el apartado 1 la respuesta es negativa, no pueden utilizar ambos tipos de personas los toritos simultáneamente. Simplemente tenemos que observar el código que ejecuta el personal del muelle, antes de coger un torito todos ejecutan una operación P sobre el semáforo asociado a la sección crítica sc y el semáforo se ha inicializado a 1.

En el apartado 2 la respuesta es afirmativa, varias personas del almacén pueden estar utilizando toritos al mismo tiempo. Si observamos el código correspondiente al almacén, podemos observar que tan sólo el primer señor del almacén que quiere coger un torito ejecuta la operación P sobre el semáforo de la sección crítica, mientras que el resto no realizarán la citada operación mientras haya alguna persona del almacén utilizando los toritos. Es la técnica utilizada para permitir que varios procesos del mismo tipo estén simultáneamente en la sección crítica.

No ocurre lo mismo en el apartado 3, no pueden haber dos personas de los muelles usando toritos al mismo tiempo. Todas las personas del muelle tienen que ejecutar obligatoriamente la operación P sobre el semáforo de la sección crítica inicializado a 1, por lo que sólo podrán coger un torito en un momento dado.

Finalmente, indicar que sí existe prioridad a favor en este caso del personal del almacén. Observemos que la prioridad está implícita en el código y no existe ningún semáforo asociado al tratamiento de ésta. La prioridad se puede eliminar simplemente añadiendo un nuevo semáforo para el tratamiento de la prioridad y utilizándolo de la misma manera que se usó en el problema de lectores y escritores sin prioridad.

```
TSemáforo sc, mutex, fifo;
int numero;
```

```
void muelle_i()
{
  . . .
  P(fifo);
  P(sc);
  V(fifo);
  coger_torito();
  V(sc);
  . . .
}
```

```
void almacen_j()
{
  . . .
  P(fifo);
  P(mutex);
  numero++;
    if (numero==1) P(sc);
  V(mutex);
  V(fifo);
  coger_toriro();
  P(mutex);
    numero--;
    if (numero==0) V(sc);
  V(mutex)
  . . .
}
```

```
void main()
{
  inicializar(mutex, 1);
  inicializar(sc, 1);
  inicializar(fifo, 1);
  numero=0;
  cobegin
    muelle_1();…; muelle_n(); almacen_1(); … ; almacen_m();
  coend;
}
```

Ejercicio 4

En una nave industrial existe una máquina de inyectar que deja cada pieza producida en una cinta transportadora de tamaño limitado. Existe un robot que recoge las piezas de la cinta (una cada vez) y las deja en las cajas de embalaje. Finalmente, tenemos un operario que, de vez en cuando, recoge 3 piezas para realizar el control de calidad, si no hay tres piezas en la cinta lo intentará más tarde. Resuelve el escenario anterior mediante semáforos.

```
#define tamaño_cinta N
TSemáforo e, s, n;        // s exclusión mutua para la variable piezas
```

```
void Inyector()
{
  while (true)
  {
    Producer_pieza();
    P(e);
    P(s);
    piezas++;
    añadir_cinta();
    V(s);
    V(n);
  }
}
```

```
void Robot()
{
  while (true)
  {
    P(n);
    P(s);
    coger_ cinta();
    piezas--;
    V(s);
    V(e);
    Robot_coge_pieza_y_embala();
  }
}
```

```
void Operario()
{
  while (true)
  {
   P(s);
   If (piezas>=3)
   {
```

```
        Piezas=piezas-3;
        P(n); P(n), P(n);
        Operario_coge_piezas();
        V(e); V(e); V(e);
         };
         V(s);
       }
    }
```

```
void main()
{
  inicializar(s, 1);
  inicializar(n, 0);
  inicializar(e, N);
  piezas=0;
  cobegin
     Inyector ();
     Robot ();
      Operario();
  coend;
}
```

Ejercicios T4. Concurrencia

Ejercicio 1

En un comedor de un colegio hay n alumnos. Cuando un alumno, que normalmente está estudiando en su clase, tiene hambre va al comedor y se acerca a alguna de las tres barras para recoger el menú. En estos menús, que varían cada día, sólo se puede elegir o cambiar el postre o café. Para el postre hay dos mostradores mientras que para el café hay un único mostrador. Realizar un programa que, usando semáforos, coordine las peticiones de los alumnos en el comedor.

Ejercicio 2

Un parque de atracciones decide usar semáforos en sus actividades de mayor afluencia de público. En concreto en las pistas blandas (con 5 pistas) y el kamikaze(con dos pistas). Además, por exigencias de seguridad, se obliga a que no haya más de 2000 personas en el parque, todos deben de ducharse antes de entrar en estas actividades que se realizarán en el orden que se han descrito. Realizar un programa con semáforos que cumpla las exigencias descritas del parque acuático.

Ejercicio 3

En el almacén de una empresa de logística existen distintos toritos que pueden ser utilizados por personal del almacén que mueve los bultos internamente y por personal de los muelles que carga los camiones de transporte. Supongamos que la gestión de los toritos se implementa de la siguiente forma:

```
TSemáforo sc, mutex;
int numero;
```

`void muelle_i()`
`{`
`  . . .`
`  P(sc);`
`  coger_torito();`
`  V(sc);`
`  . . .`
`}`
	`void almacen_j()`
	`{`
	`  . . .`
	`  P(mutex);`
	`  numero++;`
	`  if (numero==1) P(sc);`
	`  V(mutex)`
	`  coger_toriro();`
	`  P(mutex);`
	`  numero--;`
	`  if (numero==0) V(sc);`
	`  V(mutex)`
	`  . . .`
	`}`

```
void main()
{
  inicializar(mutex, 1);
  inicializar(sc, 1);
  numero=0;
  cobegin
    muelle1();…; muellen(); almacen1(); … ; almacenm();
  coend;
}
```

1) ¿Pueden utilizar a la vez los toritos el personal del almacén y del muelle?
2) Si hay un señor del almacén utilizando un torito, ¿puede otro del almacén utilizar algún torito simultáneamente?
3) Si una persona de los muelles está cargando, ¿puede otra persona de los muelles cargar al mismo tiempo?
4) Indicar si existe algún tipo de prioridad. En caso afirmativo, realizar las modificaciones necesarias para que no exista prioridad.