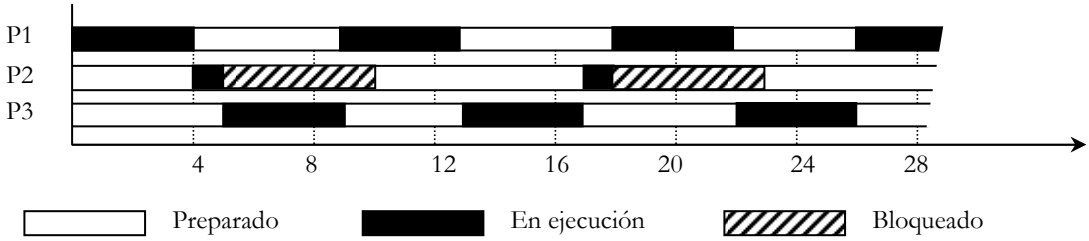
Sistemas Operativos II — Febrero 2009

Nombre:	ITIS Castellano

Ejercicio 1 [1,5 puntos]

El siguiente cronograma representa la ejecución de tres procesos en un sistema operativo durante 28 ticks. Inicialmente, los tres están en estado preparado.



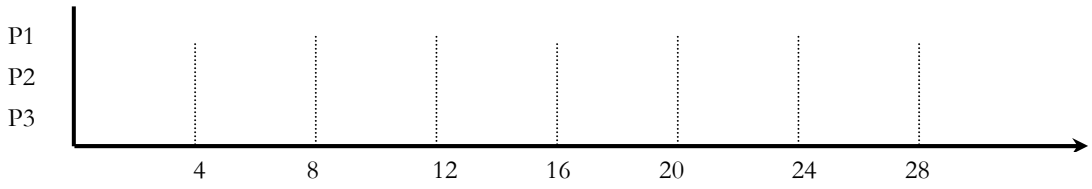
1. Indica todas las características que se puedan extraer a partir del cronograma acerca de la política de planificación de este sistema.

2. Calcula las tasas de CPU de P1, P2 y P3 durante estos 30 ticks.

Tasa CPU de P1	Tasa CPU de P2	Tasa CPU de P3

3. Calcula el tiempo de respuesta medio del proceso P2 en el tramo de ejecución mostrado.

4. Considera ahora un sistema con prioridades estáticas y expulsión por evento, donde P2 tiene prioridad A y P1 y P3 tienen prioridad B, siendo A>B. Dibuja el cronograma de ejecución durante 30 ticks. Como en el caso anterior, P1 y P3 son procesos orientados a cálculo que no se bloquean y P2 es un proceso orientado a E/S que ejecuta repetidamente intervalos de 1 tick en la CPU y se bloquea durante 5 ticks. En el instante 0 están los tres en la cola de preparados.



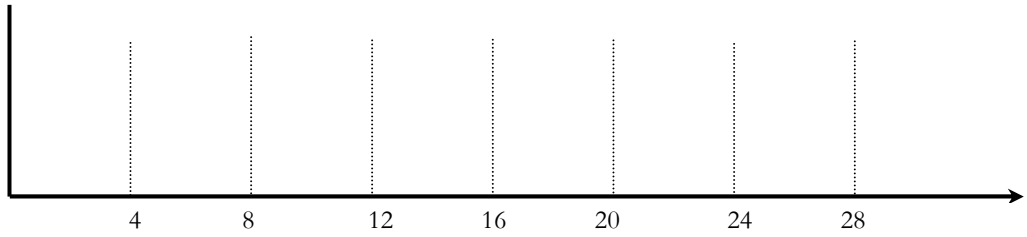
Ejercicio 2 [1 punto]

Dadas las tareas de tiempo real cuyos parámetros se especifican en la tabla:

Tarea	T. de ejecución	Periodo
T1	1	4
T2	1	6
T3	2	12
T4	3	12

1. Demuestra que su planificación es viable.

2. Dibuja un cronograma con la planificación de estas tareas de acuerdo a la política Rate Monotonic (RM).



3. Algunos sistemas Unix y Windows incluyen una clase de procesos de tiempo real. Describe brevemente en qué se basan para garantizar que las tareas de tiempo real puedan cumplir los plazos a pesar de que tienen que convivir con procesos de todo tipo.

Ejercicio 3 [1,5 puntos]

Un conjunto de procesos accede a recursos compartidos en exclusión mutua utilizando cerrojos de espera activa, mediante `lock()` y `unlock()`, y semáforos, mediante `bajar()` y `subir()`. Los procesos se están ejecutando concurrentemente en un sistema operativo con planificación por prioridades estáticas y expulsión por evento. Puede haber otros procesos en el sistema, pero no acceden a estos recursos. En la tabla adjunta se muestran las prioridades asociadas a cada proceso y el código que ejecuta cada proceso en lo que respecta a primitivas de sincronización. En un instante de tiempo dado, *T*, cada proceso está ejecutando el código que indica la flecha.

P1 (prio=4)	**P2** (prio=3)	**P3** (prio=3)
`bajar(`R_A`)`	`bajar(`R_B`)`	`bajar(`R_B`)`
→`bajar(`R_B`)`	→`lock(`R_C`)`	`bajar(`R_A`)`
`subir(`R_A`)`	`unlock(`R_C`)`	`subir(`R_A`)`
`subir(`R_B`)`	`subir(`R_B`)`	`subir(`R_B`)`
`lock(`R_C`)`		→`lock(`R_C`)`
`unlock(`R_C`)`		`unlock(`R_C`)`

1. Dibuja el grafo de asignación de recursos en el instante *T* y razona si indica alguna situación de interbloqueo. En su caso, señala cuáles son los procesos interbloqueados. Si no hubiera una situación de interbloqueo en la ejecución indicada, señala si alguna otra ejecución podría haber conducido a un interbloqueo. (Ten en cuenta la política de planificación y que los procesos han podido comenzar en cualquier orden.)

2. Para los procesos no interbloqueados, señala un posible orden de ejecución, a partir del estado indicado por las flechas, que permita que todos ellos terminen. Para ello anota sobre la tabla a la derecha de cada primitiva el número de orden en que se ejecuta. Ten en cuenta la política de planificación.

3. Indica si existe algún posible orden de ejecución que conduzca a la inanición de algún proceso por el efecto de inversión de prioridad, y comenta cuáles son las consecuencias de esta situación, cuáles son las condiciones de la planificación que conducen a ella y cuáles son las alternativas para evitarla.

Ejercicio 4 [2 puntos]

Considera un sistema de memoria paginado, con direcciones virtuales de 32 bits y páginas de 8 Kbytes. El sistema es capaz de direccionar una memoria física de hasta 1024 Gbytes. Contesta razonadamente a las siguientes cuestiones:

1. ¿Cuántos elementos tiene la tabla de páginas?

2. Calcula el tamaño de la tabla de páginas, teniendo en cuenta que hay que reservar algún bit para futuras extensiones.

3. Se pretende segmentar la tabla de páginas para que cada una ocupe exactamente un marco de página. Dibuja el esquema de la nueva dirección virtual explicando la longitud de cada campo.

4. ¿Cuánto espacio necesitará el sistema operativo para ubicar las tablas de páginas de un proceso que tiene un segmento de código, otro de datos y un tercero de pila?

5. Calcula la fragmentación externa total originada por la segmentación.

6. Calcula qué porcentaje de la memoria debe dedicar el sistema operativo para almacenar el mapa de bits.

7. Si en el sistema se están ejecutando 10.000 procesos y cada elemento de una tabla de segmentos ocupa 4 bytes, ¿qué espacio está dedicando el sistema a almacenar las tablas de segmentos de todos los procesos?

8. Verifica que esos 4 bytes asignados a un elemento de la tabla de segmentos en el apartado anterior es un tamaño razonable.

9. Este sistema se gestiona mediante memoria virtual. Tenemos instalada 4 Gbytes de memoria RAM. Por la documentación del sistema operativo sabemos que se garantiza un working-set de un tamaño mínimo para cada proceso y que la memoria virtual se combina con swapping. Hemos probado que con esa cantidad de memoria el sistema soporta razonablemente bien los 10.000 procesos de los que hablábamos antes, pero sólo la mitad ellos están en memoria, y que si cargamos más procesos en el sistema, los nuevos procesos quedan fuera de memoria. A partir de esta información, haz una estimación del tamaño mínimo de working-set que utiliza el sistema operativo.

Ejercicio 5 [1,5 puntos]

En las siguientes tablas se muestra la simulación obtenida con tres políticas de reemplazo de página sobre la misma secuencia de referencias y una memoria con tres marcos de página. Para cada política se pide que identifiques de cuál se trata; si es implementable y, en su caso, qué mecanismo(s) se puede(n) utilizar para implementarla, y, si admite aproximaciones que simplifiquen su implementación, en qué consisten éstas.

Política A:

Sec. de referencias:	**1**	**3**	**1**	**2**	**3**	**1**	**3**	**4**	**2**	**3**	**2**	**0**	**4**	**2**	**3**	**2**	**4**
Contenido de los marcos	1	1	3	3	1	2	2	1	3	4	4	3	2	0	4	4	3
		3	1	1	2	3	1	3	4	2	3	2	0	4	2	3	2
				2	3	1	3	4	2	3	2	0	4	2	3	2	4
Página víctima:								2	1			4	3		0		

Política B:

Sec. de referencias:	**1**	**3**	**1**	**2**	**3**	**1**	**3**	**4**	**2**	**3**	**2**	**0**	**4**	**2**	**3**	**2**	**4**
Contenido de los marcos	1	1	1	1	1	1	1	3	3	3	3	2	2	2	3	3	3
		3	3	3	3	3	3	2	2	2	2	4	4	4	4	2	2
				2	2	2	2	4	4	4	4	0	0	0	0	0	4
Página víctima:								1				3			2	4	0

Política C:

Sec. de referencias:	**1**	**3**	**1**	**2**	**3**	**1**	**3**	**4**	**2**	**3**	**2**	**0**	**4**	**2**	**3**	**2**	**4**
Contenido de los marcos	1	1	1	1	1	1	1	4	4	4	4	4	4	4	4	4	4
		3	3	3	3	3	3	3	3	3	3	0	0	0	3	3	3
				2	2	2	2	2	2	2	2	2	2	2	2	2	2
Página víctima:								1				3			0		

	¿A qué política corresponde?	¿Es implementable? Si lo es, mecanismos para implementarla	Aproximaciones
A			
B			
C			

Ejercicio 6 [1 punto]

El siguiente código corresponde a un driver simplificado de disco para un sistema operativo con estructura cliente-servidor.

```
1  driver_disco()
2  {
3         struct iorb *pet;
4         int cilindro, pista, sector, i, e=OK;
5
6         inicializar_cola(&drivers[DRIVER_DISCO].cola_driver);
7         while (1) {
8           bajar(drivers[DRIVER_DISCO].sema_driver);
9           if (estado_disco(PARADO) encender_motor();
10          tomar_peticion(&drivers[DRIVER_DISCO].cola_driver, pet);
11          traducir(pet->IO_dir, &cilindro, &pista, &sector);
12          if ((e=posicionar_cilindro(cilindro)) != ERROR) {
13            if (pet->operación == LEER_DISCO)
14                 e= leer_sector(pista, sector, pet->pbuf);
15            else e= escribir_sector(pista, sector, pet->pbuf);
16            }
17            pet->diagnostico= e;
18        }
19        subir(pet->evento);
20 }
```

1. Las líneas 6 y 10 hacen referencia a una estructura `drivers`. Explica cuál es el papel de esa estructura en el sistema operativo y qué aporta.

2. ¿Qué función del sistema operativo se encarga de rellenar los campos de la estructura `pet`?

3. La línea 19 ejecuta una operación de sincronización. ¿Cuál es su objetivo?

4. Las operaciones de sincronización de las líneas 8 y 19 utilizan semáforos. Discute si sería razonable el uso de primitivas de espera por ocupado en lugar de semáforos.

5. ¿En qué parte del código se implementa la política de gestión de peticiones que sigue este driver?

Ejercicio 7 [2 puntos]

Un fabricante de memorias tipo flash lanza una tarjeta con un sistema de ficheros FAT 16 destinada a cámaras fotográficas. El tamaño de bloque se fija en 32 Kbytes. El fabricante tiene la intención de ir sacando al mercado tarjetas de capacidad cada vez mayor de acuerdo al avance de la tecnología.

Contesta razonadamente a las siguientes cuestiones:

1. Calcula el límite de la capacidad de almacenamiento de estas tarjetas.

2. Para una tarjeta de 512 Mbytes, calcula el espacio que hay que asignar para almacenar la FAT en la tarjeta.

3. Se prevé que el tamaño medio de las fotografías que almacenarán estas tarjetas va a ser de aproximadamente 1 Mbyte. Calcula una estimación de la fragmentación interna que puede esperarse en estas tarjetas.

4. Hemos adquirido una de estas tarjetas, de 512 Mbytes de capacidad, para usarla en nuestro teléfono móvil, como almacenamiento de las fotografías tomadas con la cámara de baja resolución del móvil. Cada una de estas fotos ocupa unos 20 Kbytes, y son todas de tamaño parecido. ¿Cuántas fotos de este tipo podremos almacenar? ¿Cuál es la fragmentación interna en este caso?

5. Conocemos los sistemas de ficheros tipo UNIX/Linux. Estos, basados en i-nodos, en general presentan ventajas sobre los basados en FAT. Teniendo en cuenta el mercado al que están destinadas estas tarjetas, vamos a sopesar las posibles ventajas e inconvenientes que sobre la FAT-16 tendría un sistema basado en i-nodos con apuntadores de 16 bits. En concreto, y considerando también un tamaño de bloque de 32 Kbytes, ¿qué ocurriría con el límite de la capacidad de almacenamiento?

6. ¿Introduciría el sistema basado en i-nodos algún límite en el tamaño máximo de las fotografías que puede almacenar? En su caso ¿cuál sería?

7. Explica cómo se direccionaría desde el i-nodo una foto de 998 Kbytes en este sistema. Utiliza un esquema. ¿Cuántos bloques (de datos y de índices) se requerirían para almacenarla?