Apartado	1	2	3	4	5	6	7
Puntuación							

☐ No Presentado

EXAMEN DE SISTEMAS OPERATIVOS (Grado en Ing. Informática) 22/1/2015.

APELLIDOS Y NOMBRE: .....

Justificar todas las respuestas. Poner apellidos y nombre en todas las hojas y graparlas a este enunciado. Tiempo total = **2 horas 30 minutos**.

1. (**1.5 puntos**) Un sistema de archivos (tipo *system V*) con tamaño de bloque de 4 Kbytes usa inodos con 10 direcciones directas de bloques, una indirecta simple, una indirecta doble y una indirecta triple. Además, utiliza direcciones de bloques de 8 bytes. Consideremos el fichero `direcciones` en el directorio `/home/usr1`, con un tamaño de 6,04883 MBytes (6 MBytes + 50 KBytes). Responder a lo siguiente (**cada respuesta sólo se puntúa si es correcta**):

   (a) **(0.4 puntos)** Calcular cuántos bloques de disco son necesarios para representar ese archivo de 6 MBytes + 50 KBytes, discriminando si son de datos o de índices.
   Núm. de bloques de datos: ☐ Núm. de bloques de índices: ☐

   (b) **(0.3 puntos)** Un proceso abre ese fichero (descriptor `fd1`) ejecutando:
   ```
   fd1=open(‘‘/home/usr1/direcciones’’, O_RDONLY)
   ```
   Seguidamente ejecuta el servicio `fd2=dup(fd1)`, duplicando el descriptor del fichero abierto. Suponiendo que el proceso ejecuta la llamada:
   ```
   lseek(fd1, 910, SEEK_SET); /*SEEK_SET referencia desde el inicio del fichero*/
   ```
   y que, posteriormente, el proceso ejecuta las llamadas:
   ```
   read(fd2, buffer, 512); /* se leen 512 bytes */
   lseek(fd2, 70, SEEK_SET);
   ```

   Tras ejecutarse la segunda llamada a `lseek`, conteste qué es verdadero (**V**) o falso (**F**)
   - El desplazamiento asociado al descriptor `fd1` queda en la posición 910. (**V**/**F**) ☐
   - El desplazamiento asociado al descriptor `fd1` queda en la posición 70. (**V**/**F**) ☐
   - El desplazamiento asociado al descriptor `fd2` queda en la posición 582. (**V**/**F**) ☐
   - El desplazamiento asociado al descriptor `fd1` queda en la posición 910, y el desplazamiento asociado al descriptor `fd2` queda en la posición 582. (**V**/**F**) ☐

   (c) **(0.4 puntos)** En ese sistema de archivos UNIX con un tamaño de inodo de 128 bytes, tamaño de bloque de 4Kbytes, la zona de inodos (lista de inodos) ocupa 2048 bloques. El superbloque mantiene un mapa de bits para determinar los inodos libres/ocupados de la lista de inodos. Calcular el núm. de bloques que ocupa ese mapa de bits = ☐

(d) **(0.4 puntos)** Supongamos la siguiente secuencia de comandos desde el directorio `/home/usr1`. Comenzamos creando un *soft link* al archivo *direcciones* (cuyo número de inodo es 22343, tamaño 6,04883 MBytes y num. de hard links=1), con el comando `ln -s direcciones slink` (es decir, el comando `ls -l` mostraría `slink` → `direcciones`).

Contestar si es verdadero (**V**) o falso (**F**) lo siguiente:

- El inodo de *slink* es el 22343 (**V**/**F**) [ ]
- Posteriormente se crea un *hard link* al fichero `slink`:
  `ln slink hardslink /* crea hard link "hardslink"*/`
  El tamaño de `hardslink` es 11 bytes (**V**/**F**) [ ]
- Una vez creados `slink` y `hardslink`, al borrar el fichero `direcciones` (`rm direcciones`) se libera su inodo (queda como libre en la lista de inodos en disco) (**V**/**F**) [ ]
- Con un sistema de ficheros Unix basado en registro (*journaling file system*), al borrar un fichero, su inodo se guarda en el registro pero no sus datos. (**V**/**F**) [ ]

2. (**total 1.2 puntos**) Considere la traslación de direcciones virtuales a físicas en el procesador Intel Core i7 estudiado en clase. Las direcciones virtuales son de 48 bits, las direcciones físicas son de 52 bits y las páginas son de 4 KB (4 kilobytes). La arquitectura es de paginación multinivel (4 niveles), las tablas de páginas en todos los niveles son de tamaño 1 página y los bits de las direcciones virtuales que componen el número de página virtual se trocean en cuatro campos del mismo tamaño que se usan para obtener la entrada en la tabla de páginas correspondiente. Conteste a los siguientes apartados:

- (**0.4 puntos**) Dibuje un esquema de esta arquitectura.
  Consulte las slides estudiadas en clase
- (**0.1 puntos**) Tamaño en bytes del espacio de direcciones virtuales: [2^{48}]
  Tamaño en bytes del espacio de direcciones físicas: [2^{52}]
- (**0.1 puntos**) Número de páginas del espacio de direcciones virtuales: [2^{36}]
  Número de páginas del espacio de direcciones físicas: [2^{40}]
- (**0.1 puntos**) Número de entradas en una tabla de páginas de segundo nivel: [2^9]
  Número de entradas en una tabla de páginas de cuarto nivel: [2^9]
- (**0.1 puntos**) Tamaño en bits de una entrada en una tabla de págs. de 2^o nivel: [64]
  Tamaño en bits de una entrada en una tabla de páginas de cuarto nivel: [64]
- (**0.1 puntos**) Número mínimo de bits necesarios en una entrada en una tabla de páginas de segundo nivel para direccionamiento: [40]
  Número mínimo de bits necesarios en una entrada en una tabla de páginas de cuarto nivel para direccionamiento: [40]
- (**0.1 puntos**) ¿Qué espacio de direcciones virtuales en bytes se puede direccionar con una entrada de una tabla de páginas de tercer nivel?: [2^9 x $2^{12} = 2^{21} = 2MB.$]
  ¿Qué espacio de direcciones virtuales en bytes se puede direccionar con una entrada de una tabla de páginas de segundo nivel?: [2^9 x $2^{21} = 2^{30} = 1GB.$]
- (**0.1 puntos**) Considere un proceso en ejecución. ¿Pueden ciertos contenidos de una tabla de páginas de segundo nivel estar almacenados (cached) en la TLB? **No** ¿Por qué? Esos contenidos son de una página de datos en memoria y como tal, pudieran estar almacenados en la caché de datos, no en la TLB.

- (**0.1 puntos**) Considere el SO Linux sobre este procesador. Linux utiliza segmentación por software (los segmentos son las vm_area). Diga si es verdadera o falsa la siguiente afirmación y por qué:

Por tanto el kernel de Linux ignora el esquema de paginación del procesador a la hora de trasladar las direcciones. Falso

¿Por qué? Las vm_area conforma un espacio de direcciones virtuales con las propiedades favorables de la segmentación y con posibles huecos y lo que hace Linux al gestionar los procesos en memoria es crear las tablas de páginas para esas vm_areas.

3. (**total 0.8 puntos**) Para cada una de las siguientes cuestiones, contestar verdadero (**V**) o falso (**F**) y **justificar** las respuestas. Marcas ambiguas o justificaciones insuficientes o erróneas invalidan la pregunta.

   - (**0.2 puntos**) En un sistema con memoria virtual y paginación, tabla de páginas (TP) de un nivel y TLB, una referencia a memoria puede producir un fallo en la TLB (TLB miss) y no haber fallo de página.
     ¿Por qué?
     Verdadero. Es debido a que una TLB normalmente no puede contener todas las entradas de la TP de un proceso. Puede producirse el fallo en la TLB pero la página está en memoria y la traslación de página lógica a página física se consigue en la TP, no produciéndose fallo de página.
   - (**0.2 puntos**) En un sistema con memoria virtual y paginación, tabla de páginas (TP) de un nivel y TLB, una referencia a memoria puede producir un acierto en la TLB (TLB hit) y un fallo de página.
     ¿Por qué?
     Falso. Si hay un acierto en la TLB, el número de página física que se corresponde con el número de página lógica se consigue en la TLB y no hay fallo de página.
   - (**0.2 puntos**) Es posible un SO que decida sobre el tamaño del conjunto residente de un proceso con una política de asignación de frames (marcos) variable y reemplazo global.
     ¿Por qué?
     Verdadero. Si es posible. Cuando un proceso requiere una frame, se le asigna. Si no hay frames libres el SO puede decidir reemplazar una página de otro proceso. Es una política sencilla pero tiene el riesgo que un proceso agresivo en el uso de memoria o con fugas de memoria (memory leaks) comprometa las prestaciones de todo el sistema.
   - (**0.2 puntos**) La política de gestión del conjunto residente de un proceso Frecuencia de Fallo de Página (Page Fault Frequency) es una política de asignación variable y reemplazo global.
     ¿Por qué?
     Falso. Esta política es una aproximación a la del Conjunto de Trabajo (Working Set) y el reemplazo no es global ya que si no hubiese memoria para contener el Conjunto de Trabajo de todos los procesos, no se reemplazarían páginas de un proceso, ya que precisamente se pretende que los procesos residentes puedan tener su conjunto de trabajo en memoria. Por tanto en ese caso se decidiría intercambiar un proceso.

4. (**0.8 puntos**) Conteste brevemente a las siguientes preguntas

   a) ¿Qué significa que el kernel de unix sea reentrante? Significa que varios procesos pueden estar ejecutando concurrentemente (la misma o distintas) funciones del kernel

   b) ¿Cuales son las condiciones necesarias para ello? Son necesarias tres condiciones:

   - Código del kernel de sólo lectura
   - Datos del kernel protegidos frente a accesos concurrentes (por ejemplo, mediante semáforos)
   - Cada proceso tiene su propia pila del kernel

5. (**0.7 puntos**) Un sistema está ejecutando la rutina de servicio de una interrupción y llega otra interrupción. Para cada una de estas afirmaciones, indica si es verdadera (**V**) o falsa (**F**) y justifica después la respuesta.

   a) Se ejecuta siempre la rutina de servicio de la interrupción que acaba de llegar y después se continua con la que está (**V**/**F**) [ ]

   b) Se completa siempre la rutina de servicio de la interrupción actual y luego se ejecuta la rutina de servicio de la interrupción que acaba de llegar (**V**/**F**) [ ]

   c) Puede continuar con la que está o atender la nueva interrupción dependiendo de la prioridad del proceso que produce la interrupción (**V**/**F**) [ ]

   d) Puede continuar con la que está o atender la nueva interrupción dependiendo de la prioridad del proceso que se está ejecutando cuando llega la interrupción (**V**/**F**) [ ]

   e) Puede continuar con la que está o atender la nueva interrupción dependiendo de la credencial efectiva del proceso que produce la interrupción (**V**/**F**) [ ]

   f) Puede continuar con la que está o atender la nueva dependiendo de la credencial efectiva del proceso que se está ejecutando cuando llega la interrupción (**V**/**F**) [ ]

   Razona la respuesta:
   Son **todas falsas**.

   Se ejecutará la rutina de servicio de una u otra interrupción dependiendo de las **IPL**s (Interrupt Priority Level) de ambas.

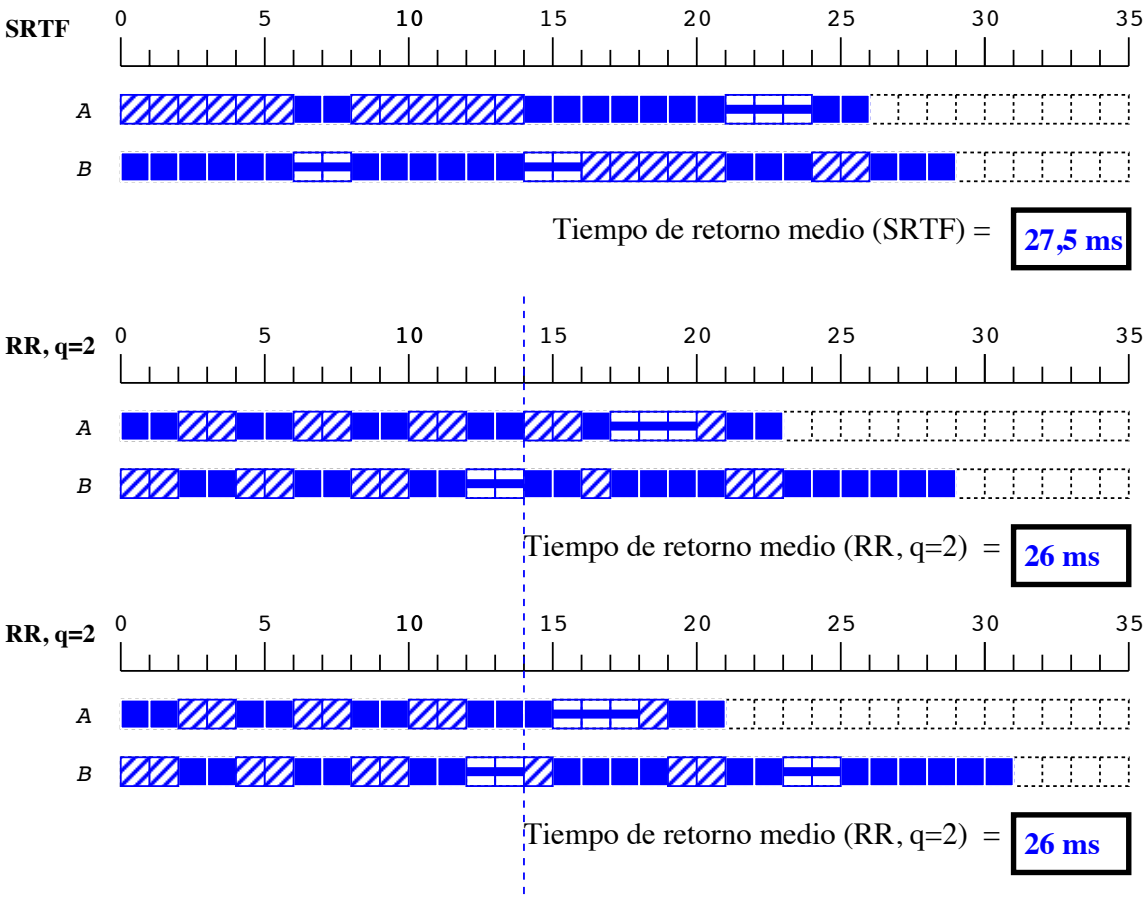
   Las opciones a) y b) son falsas porque especifican *siempre* y no tienen en cuenta las IPL de las interrupciones.

   La IPL no guarda ninguna relación ni con las prioridades de los procesos ni con sus credenciales. Por lo tanto, las demás opciones son también falsas. Las opciones c) y e) contienen además otra incorrección, ya que los procesos no producen interrupciones, siendo estas generadas por los *dispositivos* cuando requieren atención.

6. (**1 punto**) Asúmase que en un determinado instante la cola de peticiones de E/S para acceder a cilindros de un disco duro contenía las peticiones: $\langle 75, 1, 33, 66\rangle$. Desconocemos el cilindro X que estaba siendo accedido justo antes de estas peticiones, pero sabemos que X no estaba entre los cilindros de la lista de peticiones. Si sabemos que el orden de atención de dichas peticiones fue: $\langle 75, 1, 33, 66\rangle$. Indíquese si los siguientes algoritmos de planificación de accesos a disco pudieron ser utilizados (sí/no). En caso afirmativo, indíquese también UN posible número de cilindro (X) que podría estar siendo accedido antes de atender dichas peticiones. Razone brevemente su respuesta.

algoritmo	**posible** (sí/no)	**posible cilind X**	**breve justificación de la respuesta**
SSTF:	no	–	X debería ser > 70 pues se atiende 75 antes que 66, pero después debería atender 66 antes que 1 y 33.
SCAN:	no	–	Tras 75 debería atender 66.
C-SCAN:	sí	> 66	Ascensor atiende peticiones al subir. Posición tras 66.
FIFO:	si	cualquiera	se sigue el orden de llegada de las peticiones

7. (**1 punto**) Tenemos dos procesos, A y B, que inician ambos su ejecución en el instante $T = 0$, pero A llega a la cola antes que B. Las secuencias de ráfagas de cada proceso son $A = (\underline{9}, 3, \underline{2})$, $B = (\underline{6}, 2, \underline{6}, 2, \underline{6})$ donde los números subrayados representan tiempo de CPU y el resto es E/S (los tiempos se miden en ms). Rellene en las plantillas el resultado de aplicar los algoritmos SRTF y Round Robin con $q = 2\ ms$. Cada celda representa 1 ms y se debe indicar qué le sucede al proceso en cuestión en cada momento (CPU, E/S o en espera) usando la simbología indicada abajo. Calcule en ambos casos el tiempo de retorno medio.



Explicación (no necesaria en la contestación el examen):

En el RR, en el instante 14 tenemos que el proceso A acabó su cuanto de ejecución y, a la vez, el proceso B entra en la cola de preparados. Lo que suceda a continuación dependerá de cómo esté programado el ciclo de pasos a realizar en la planificación de procesos. Si se añaden a la cola de preparados los que finalizan E/S antes del chequeo del cuanto, al acabar A se encontrará a B ya listo y se produciría un cambio de contexto, como se muestra en la opción de arriba. El tiempo medio de retorno en este caso es $(23 + 29)/2 = 26\ ms$. Si, por el contrario, se comprueba primero la finalización del cuanto, y se añaden después los procesos que acabaron E/S a la cola de preparados, el resultado sería el mostrado abajo, es decir, A continúa en CPU un ms más (no llega a agotar el nuevo cuanto). El tiempo medio de retorno en esta solución es también $(21 + 31)/2 = 26\ ms$. Ambas soluciones se aceptaron como correctas.

Un error común en el algoritmo RR ha sido hacer que un proceso se quede esperando en la cola de preparados cuando la CPU está vacía. Esto no sucede *nunca* bajo ningún criterio de planificación. El motivo que conduce habitualmente a este error es pensar que el planificador debe tomar una acción siempre cada $q = 2\ ms$. Esto es incorrecto, ya que un proceso puede acabar antes de vencer su cuanto y dejar la CPU libre, en cuyo caso, el planificador toma el siguiente proceso preparado *de forma inmediata.*

En el SRTF, un error común ha sido comenzar por el proceso A considerando que entró primero en la cola. El algoritmo SRTF escoge siempre primero el proceso cuya ráfaga de ejecución restante dure menos. Como todos los algoritmos, SRTF actúa cada vez que la CPU queda vacía y hay procesos preparados. Pero además, es un algoritmo apropiativo y actúa también cada vez que entra un nuevo proceso en la cola de preparados (véanse por ejemplo, los cambios de contexto que se producen en los instantes 8 y 24). En el instante

inicial, la CPU está vacía y la ráfaga de B es menor, por lo que es el primero en pasar a CPU. El algoritmo SRTF sólo usa el criterio FIFO en la cola de preparados cuando existe un empate entre duraciones de las ráfagas.