SISTEMAS OPERATIVOS (Código: 71902048) **Enero 2012**

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N1**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

1. Conteste **razonadamente** a los siguientes apartados:

a) *(1 p)* Explicar la diferencia entre un enlace duro y un enlace simbólico.

b) *(1 p)* Describir el algoritmo de planificación basada en múltiples colas de prioridad y realimentación.

2. *(2 p)* Explicar **razonadamente** qué es un cambio de contexto o proceso y cuáles son las principales causas que lo motivan.

3. *(2 p)* Enumerar y describir **brevemente** las capas de software de E/S del núcleo de un sistema operativo.

4. *(2 p)* Una persona tiene en su casa una jaula llena de canarios en la que hay un plato de alpiste y un columpio. Todos los canarios quieren primero comer del plato y luego columpiarse, sin embargo sólo tres de ellos pueden comer del plato al mismo tiempo y solo uno de ellos puede columpiarse. Escribir el pseudocódigo basado en C de un programa que usando *semáforos binarios* coordine la actividad de los canarios. Dicho programa debe tener tres partes: declaración de variables y semáforos, código del proceso `canario`, y código de la función principal para inicializar los semáforos y lanzar la ejecución concurrente de los procesos.

Nota: Antes de escribir el pseudocódigo se debe explicar adecuadamente el significado de cada uno de los semáforos binarios y variables que se van a utilizar en el mismo.

5. *(2 p)* El sistema operativo en colaboración con el hardware gestiona la memoria principal mediante paginación por demanda. El tiempo medio de acceso a memoria es de 115 ns. La traducción de direcciones se realiza usando una MMU con banco de registros. El tiempo medio de acceso al banco de registros es despreciable. La atención de un fallo de página emplea en promedio 10 ms si existe disponible un marco vacío o si la página reemplazada no se modifica y 35 ms si la página reemplazada se modifica. La página que se va a reemplazar se modifica el 30 % de las veces. ¿Cuál es la tasa máxima aceptable de fallos de página para obtener un tiempo medio de despacho de una referencia a memoria menor de 230 ns? Despreciar la existencia de memoria caché.

Grado en Ingeniería Informática/Grado en Ingeniería en Tecnologías de la Información

SISTEMAS OPERATIVOS
Solución examen febrero 2012 (1ª semana)

Solución Ejercicio 1

a) Un *enlace* permite conectar un directorio con un archivo o subdirectorio ya existente en otro directorio. Un archivo o subdirectorio referenciado por un enlace se dice que está *compartido*.

Para simplificar la explicación se utilizará el término *directorio origen* para denominar al directorio en el que reside un enlace y el término *elemento destino* para denominar al archivo o subdirectorio compartido. El directorio que contiene el elemento destino se denominará como *directorio destino*.

Una posible forma de implementar un enlace es crear en el directorio origen una nueva entrada que sea una copia parcial o completa de la entrada del directorio destino asociada al archivo o subdirectorio compartido. La nueva entrada creada en el directorio origen es marcada con algún identificador especial que permita reconocer al sistema operativo que se trata de una entrada asociada a un enlace. Los enlaces con esta implementación son denominados en algunos sistemas operativos como *enlaces duros*.

Otra forma de implementar un enlace es como un tipo especial de archivo que contiene el nombre de ruta absoluta o relativa del archivo o subdirectorio compartido. A los enlaces implementados de esta forma se les denomina *enlaces simbólicos*. La entrada asociada a un enlace simbólico en el directorio origen es completamente distinta a la entrada del directorio destino asociada al archivo o subdirectorio compartido. En este caso no hace falta marcar la entrada con algún identificador especial ya que el tipo del archivo lo identifica como enlace simbólico.

b) En el *algoritmo de planificación basada en múltiples colas de prioridad* un proceso solo puede pertenecer a una determinada cola de preparados o cola de prioridad durante su tiempo de vida. En consecuencia, posee una prioridad estática. Con este algoritmo siempre se ejecuta un proceso perteneciente a la cola de preparados con mayor prioridad. Sólo cuando dicha cola está vacía se puede planificar un proceso perteneciente a una cola de preparados de menor prioridad. Un proceso a lo largo de su vida solo puede pertenecer a una determinada cola de preparados. Además para planificar cada cola se puede utilizar un algoritmo distinto: FCFS, SJF, turno rotatorio, etc.

Si se permite que un proceso pueda ir cambiando de cola de prioridad durante su existencia, es decir, que su prioridad pueda ser modificada dinámicamente, se dice que el algoritmo posee *realimentación*.

La implementación de la realimentación requiere definir un mecanismo que regule cómo se modifica dinámicamente la prioridad de los procesos durante su tiempo de existencia. O visto de otra forma, cuándo se produce la transición de un proceso de una cola de prioridad a otra.

Un posible mecanismo de regulación de la prioridad es el siguiente: cuando un proceso llega al sistema se coloca en la cola de mayor prioridad. Cuando es planificado puede usar el procesador durante un cuanto. Al finalizar el cuanto es colocado en la siguiente cola de menor prioridad. Cuando un proceso llega a la cola de menor prioridad, cada vez que consuma un cuanto volverá a ella hasta que consiga completarse. En consecuencia, el algoritmo de planificación de la cola de menor prioridad es de tipo turno rotatorio. El resto de colas se pueden planificar internamente con un algoritmo FCFS.

Solución Ejercicio 2

Se denomina *cambio de proceso* o *cambio de contexto* a la operación que realiza el sistema operativo consistente en interrumpir la ejecución de un proceso A para iniciar o continuar la ejecución de otro

proceso B. Entre las principales causas que motivan un cambio de contexto se encuentran las siguientes:

- *El proceso en ejecución pasa al estado bloqueado*. Un proceso A pasa al estado bloqueado cuando debe esperar por la aparición de algún evento. Mientras llega a producirse dicho evento se puede ejecutar otro proceso B.
- *La terminación (voluntaria o forzada) del proceso en ejecución*. Obviamente si un proceso termina se puede pasar a ejecutar otro.
- *El sistema operativo termina de atender una interrupción y existe un proceso B en estado preparado de mayor prioridad que el proceso actual A*. En dicho caso dependiendo de la política de planificación que siga el sistema operativo, se puede ejecutar dicho proceso B más prioritario o continuar con el proceso A que se estaba ejecutando.
- *El proceso A en ejecución ha excedido el tiempo máximo de ejecución ininterrumpida*. En los sistemas de tiempo compartido, a cada proceso se le asigna un tiempo máximo de ejecución ininterrumpida con objeto de poder atender las peticiones de todos los usuarios conectados al sistema. Superado dicho tiempo se produce un cambio de proceso, el proceso A en ejecución pasa al estado preparado y un proceso B en dicho estado pasa al estado ejecutándose.

Solución Ejercicio 3

De forma general, el software del núcleo de un sistema operativo necesario para la gestión de las operaciones de E/S se puede organizar en tres capas:

- *Subsistema de E/S*. Es el componente del sistema operativo que se encarga de efectuar todas aquellas tareas necesarias para la realización de las operaciones de E/S que son comunes a todos los dispositivos e independientes de los mismos. Es decir, el subsistema de E/S gestiona la parte independiente del dispositivo de todas las operaciones de E/S. Entre las tareas que se encarga de realizar el subsistema de E/S se encuentran las siguientes: asignación y liberación de dispositivos dedicados, bloqueo (si procede) de procesos que solicitan una operación de E/S, planificación de la E/S, buffering, invocación del driver del dispositivo adecuado, y gestión de los errores producidos en las operaciones de E/S.
- *Drivers de dispositivos*. Un driver de dispositivo contiene el código que permite a un sistema operativo controlar un determinado tipo de dispositivo de E/S. Un driver de dispositivo interactúa con el subsistema de E/S y con el controlador de E/S que controla el dispositivo. Un driver suministra al subsistema de E/S el conjunto de funciones que se pueden realizar sobre el dispositivo, tales como lectura o escritura. Además un driver puede invocar a ciertas rutinas o procedimientos del núcleo. Los procedimientos a los que tiene acceso un driver quedan definidos por la interfaz de drivers del subsistema de E/S. El driver de un dispositivo interactúa con el controlador de E/S, cargando en sus registros diferentes órdenes para que las efectúe sobre el dispositivo, comprobando su estado e inicializándolo si es necesario.
- *Manejadores de las interrupciones*. El manejador de una interrupción es una función del núcleo encargada de atender una determinada interrupción. Con objeto de que el rendimiento del computador sea óptimo, los manipuladores de interrupciones tienen una alta prioridad de ejecución. Además su código suele ser pequeño y rápido de ejecutar. Las acciones específicas que realiza un manejador de interrupciones dependen de cada tipo de interrupción. Si el driver del dispositivo se bloqueó en espera de que el controlador de E/S estuviera preparado para procesar otra petición de E/S, entonces una acción que debe realizar un manejador de interrupción es el desbloqueo del driver del dispositivo mediante el uso del mismo mecanismo de sincronización (semáforo, mensajes,...) que utilizó el driver para bloquearse.

Solución Ejercicio 4

La solución que se propone en la Figura 1 para este problema utiliza las siguientes variables globales y semáforos binarios:

- `CC`. Variable global de tipo entero para llevar la cuenta de canarios que están comiendo del plato de alpiste.
- `sem1`. Semáforo binario que se utiliza para garantizar la exclusión mutua en el uso de la variable global `CC`.
- `sem2`. Semáforo binario que se utiliza para sincronizar el acceso de los canarios al plato de alpiste.
- `sem3`. Semáforo binario que se utiliza para garantizar la exclusión mutua en el uso del columpio

```
/* Definición variables y semáforos */
int CC=0;
semáforo_binario sem1, sem2, sem3;

void canario() /* Proceso canario */
{
        wait_sem(sem1);
        CC=CC + 1;

        if(CC > 3)
        {
          signal_sem(sem1);
          wait_sem(sem2);  /* Esperar a que haya un puesto libre en el plato */
        }
        else signal_sem(sem1);

        comer();

        wait_sem(sem1);
        if(CC > 3) signal_sem(sem2); /* Avisa de que hay un puesto libre en el plato */
        CC=CC-1;
        signal_sem(sem1);

        wait_sem(sem3); /* Comprueba si puede usar el columpio, en caso negativo espera */
        columpiarse();
        signal_sem(sem3); /* Avisa de que ha terminado de columpiarse */

}

void main() /*Inicialización de semáforos y ejecución concurrente*/
{
        init_sem(sem1,1);
        init_sem(sem2,0);
        init_sem(sem3,1);
        ejecución_concurrente(canario, canario,...,);
}
```

Figura 1 – Solución Ejercicio 2

Solución Ejercicio 5

El tiempo medio de gestión de un fallo de página t_{gf} depende de si la página ha sido modificada, lo cual ocurre el 30 % de las veces. Si la página se modifica el tiempo medio de gestión del fallo es de 35 ms, en caso contrario emplea 10 ms. Luego t_{gf} se calcula de la siguiente forma:

$$t_{gf} = 0{,}7{\cdot}10\,10^{-3} + 0{,}3{\cdot}35\,10^{-3} = 0{,}0175\,\text{s}$$

El tiempo medio de despacho de una referencia a memoria es:

$$t_{\text{R}} = (1 - p)\,t_{am} + p\,t_{gf}$$

Se desea que t_{R} sea menor de 230 ns, es decir:

$$(1 - p)\,t_{am} + p\,t_{gf} < 230\,10^{-9}$$

Despejando p se obtiene:

$$p < \frac{230\,10^{-9} - t_{am}}{t_{gf} - t_{am}}$$

Sustituyendo valores y operando se obtiene finalmente que:

$$p < \frac{230\,10^{-9} - 115\,10^{-9}}{0{,}0175 - 115\,10^{-9}} \;\rightarrow\; p < 6{,}5715\,10^{-6}$$

SISTEMAS OPERATIVOS (Código: 71902048) **Febrero 2012**

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N2**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

1. Conteste **razonadamente** a los siguientes apartados:

a) *(1 p)* ¿Qué información se suele almacenar de forma general en una entrada de un directorio?

b) *(1 p)* Describir la planificación de hilos en función del tipo de hilos soportado por el sistema operativo.

2. *(2 p)* Enumerar las acciones que debe realizar el sistema operativo para crear un proceso.

3. *(2 p)* Explicar **razonadamente** qué es el *buffering* y comentar los problemas que resuelve.

4. *(2 p)* En una oficina de Correos existen 3 ventanillas de atención al cliente. Cuando un cliente entra en la oficina para realizar alguna gestión debe guardar una única cola hasta que alguna ventanilla queda libre. Explicar **razonadamente** si el pseudocódigo del programa que se muestra en la Figura 1 coordina adecuadamente la actividad de los clientes en la oficina. En caso negativo modifique el programa para que funcione correctamente.

5. *(2 p)* El sistema operativo en colaboración con el hardware gestiona la memoria principal mediante paginación por demanda. La traducción de direcciones se realiza usando una MMU con un TLB. El tiempo medio de acceso al TLB es despreciable y su tasa de aciertos es del 95 %. Determinar el tiempo medio de despacho de una referencia a memoria si se tienen: una tasa de fallos de página del 8 %, un tiempo medio de acceso a memoria de 125 ns y un tiempo medio de gestión de un fallo de página de 40 ms. Despreciar la existencia de memoria caché.

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N2**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

```
int Z=0;
semáforo_binario X, Y;

void cliente()
{
        wait_sem(X);
        Z = Z + 1;
        if (Z > 3){
          signal_sem(X);
          wait_sem(Y);
        }

        signal_sem(X);
        realizar_gestión();
        wait_sem(X);
        Z = Z - 1;
        signal_sem(Y);
        signal_sem(X);

}

main()
{
        init_sem(X,1);
        init_sem(Y,0);
        ejecución_concurrente(cliente,...,cliente);
}
```

Figura 1

Grado en Ingeniería Informática/Grado en Ingeniería en Tecnologías de la Información

SISTEMAS OPERATIVOS
Solución examen febrero 2012 (2ª semana)

Solución Ejercicio 1

a) Un *directorio* es un archivo que almacena una lista de los archivos y subdirectorios que contiene. La información que almacena cada entrada de la lista depende de cada sistema de archivos, aunque en general se suele almacenar el nombre del archivo y otras informaciones adicionales asociadas al archivo. Cuando se abre un archivo el sistema operativo cargar en memoria principal la entrada del directorio que lo referencia para poder acceder a la información que contiene.

   Algunos sistemas de archivos, como por ejemplo FAT-32 de Windows, almacenan en cada entrada de un directorio el nombre de un archivo o subdirectorio y sus atributos. Recuérdese que entre los atributos de un archivo se incluye la información necesaria que permite localizar los bloques de datos del archivo.

   Otros sistemas de archivos, como por ejemplo UFS de UNIX o ext2 de Linux, almacenan en cada entrada de un directorio el nombre de un archivo o subdirectorio y un puntero (número de nodo-i) a la estructura de datos (nodo-i) donde se almacenan los atributos del archivo.

b) La planificación de los hilos depende de los tipos de hilos soportados por el sistema operativo. Si únicamente se soportan *hilos a nivel de usuario*, el sistema operativo no es consciente de la existencia de estos hilos y por ello realiza una planificación global a nivel de proceso de acuerdo con un determinado algoritmo de planificación.

   Una vez que un proceso está siendo ejecutado en modo usuario es el hilo de planificación dentro de dicho proceso el que decide qué hilo va a ser ejecutado. Por lo tanto cada proceso puede planificar sus hilos con el algoritmo de planificación que considere más oportuno.

   Puesto que en modo usuario no se pueden tratar las interrupciones, no existen interrupciones de reloj disponibles para establecer el cuanto de un hilo. Éste se ejecuta hasta que voluntariamente decide ceder el uso del procesador o expira el cuanto del proceso. En dicho caso el sistema operativo debe seleccionar otro proceso para ser ejecutado.

   Si se soportan *hilos del núcleo*, el sistema operativo planifica estos hilos usando algún determinado algoritmo de planificación. El planificador puede tener en cuenta consideraciones relativas al rendimiento del sistema a la hora de planificar hilos del núcleo de igual prioridad.

   Por otra parte, en sistemas con una configuración del tipo menor número de hilos del núcleo que hilos de usuarios, aparte de las dos planificaciones comentadas, es necesario implementar con la biblioteca de hilos una planificación local para establecer qué hilo de usuario puede usar un hilo del núcleo.

Solución Ejercicio 2

El sistema operativo es el responsable de la creación de los procesos que se van a ejecutar. Las acciones que se deben realizar para crear un proceso dependen de cada sistema operativo, aunque de forma general se pueden distinguir las siguientes:

1. *Comprobar si el proceso puede ser creado*. El sistema operativo comprueba si existe suficiente espacio en memoria principal para crear otro proceso y si el usuario no ha excedido el número máximo de procesos que puede tener ejecutándose. Algunos sistemas operativos establecen un límite al número máximo de procesos que un usuario puede estar ejecutando para evitar exceder la capacidad de la tabla de procesos.

2. *Asignar una entrada de la tabla de procesos para el nuevo proceso*. En este instante se le asigna un identificador numérico al proceso.

3. *Reservar espacio en memoria para el proceso*. Se debe reservar espacio en memoria principal para el espacio de direcciones de memoria lógica del proceso y para la información de control del proceso.

4. *Inicializar el bloque de control del proceso*. Los campos de la entrada asociada al proceso en la tabla de procesos se inicializan con los valores adecuados.

5. *Establecer los enlaces apropiados*. El sistema operativo debe configurar los punteros adecuados para enlazar la información del proceso en las diferentes listas, tablas y colas que mantiene. Por ejemplo, el sistema operativo suele colocar al proceso creado en la cola de procesos preparados para que así puede ser planificado.

Solución Ejercicio 3

El *buffering* es una técnica implementada por el sistema operativo que consiste en almacenar temporalmente en buffers los datos que se transfieren entre un dispositivo y un proceso, o viceversa. Un *buffer* es un área de memoria principal a la que únicamente tiene acceso el sistema operativo, es decir, pertenece al espacio del núcleo. El subsistema de E/S es el encargado de asignar buffers para las operaciones de E/S. Por su parte, las rutinas de servicio de las interrupciones o los drivers, se encargan de transferir los datos entre los buffers y los dispositivos, o viceversa, dependiendo de si se trata de una operación de escritura o de lectura.

El buffering evita los problemas que plantea la transferencia directa de datos desde el dispositivo al espacio de usuario del proceso y que son los siguientes

- Obliga al sistema operativo a mantener cargada en un marco *j* de la memoria principal la página *i* mientras no se complete la operación de E/S, ya que en caso contrario se perderían los datos. Para evitar que la página sea reemplazada, el sistema operativo tiene que bloquear el marco. Las operaciones de E/S son lentas, en comparación con los accesos a memoria principal, por lo que el marco quedará bloqueado durante un tiempo importante. Cuanto mayor sea el número de marcos bloqueados, menor será el número de marcos candidatos para ser reemplazados al atender un fallo de página, lo que puede conducir a que se produzca sobrepaginación.

- Imposibilidad de poder intercambiar al proceso hasta que la operación de E/S no se haya completado.

Otra ventaja del buffering es que permite realizar durante una transferencia de datos una adaptación de velocidades entre el agente productor (proceso o dispositivo) y el agente consumidor (dispositivo o proceso). Asimismo permite lograr una adaptación entre dispositivos que tienen diferentes tamaños de unidad de transferencia de datos.

Solución Ejercicio 4

```
int Z=0;
semáforo_binario X, Y;

void cliente()
{
        wait_sem(X);
        Z = Z + 1;
        if (Z > 3){
          signal_sem(X);
          wait_sem(Y);
        }

        signal_sem(X);
        realizar_gestión();
        wait_sem(X);
        Z = Z - 1;
        signal_sem(Y);
        signal_sem(X);

}

main()
{
        init_sem(X,1);
        init_sem(Y,0);
        ejecución_concurrente(cliente,...,cliente);
}
```

Figura 1

Se observa que la solución que se propone en la Figura 1 utiliza las siguientes variables globales y semáforos binarios:

- `Z`. Variable global de tipo entero para llevar la cuenta del número de clientes en la cola o en la ventanilla.
- `X`. Semáforo binario que se utiliza para garantizar la exclusión mutua en el uso de la variable global `Z`.
- `Y`. Semáforo binario que se utiliza para sincronizar el acceso a las ventanillas de la oficina.

Analizando detenidamente la solución propuesta se observa que **no coordinada adecuadamente** la actividad de los clientes en la oficina ya que la operación `signal_sem(Y)` se realiza siempre independientemente de si existen o no clientes esperando. Nótese que si no existen clientes esperando en la cola la realización de esta operación pone el semáforo binario `Y` a 1, lo que implica que si las ventanillas están ocupadas y un cliente tiene que realizar la operación `wait_sem(Y)` para esperar en la cola del semáforo, no se bloqueará ya que el semáforo estará a 1 en vez de a 0.

En la Figura 2 se propone una solución válida.

```
int Z=0;
semáforo_binario X, Y;

void ciudadano()
{
        wait_sem(X);
        Z = Z + 1;
        if (Z > 3){
          signal_sem(X);
          wait_sem(Y);
        }
        else signal_sem(X);

        realizar_gestión();

        wait_sem(X);
        if (Z> 3) signal_sem(Y);
        Z = Z - 1;
        signal_sem(X);
}

main()
{
        init_sem(X,1);
        init_sem(Y,0);
        ejecución_concurrente(cliente,...,cliente);
}
```

Figura 2

Solución Ejercicio 5

La MMU dispone de un TLB en el cual se cargan un cierto número de entradas de la tabla de páginas del proceso en ejecución. Cuando se va a traducir una dirección virtual que referencia a una determinada página *i* se busca en el TLB la entrada *i* de la tabla de páginas. Si se encuentra se produce un acierto, en caso contrario se produce un fallo y hay que acceder a la memoria principal a leer dicha entrada de la tabla de páginas. Luego el tiempo medio de lectura t_l de la entrada de la tabla de páginas asociada a la página referenciada es:

$$t_l = h\,t_{ga} + (1-h)\,t_f$$

En la expresión anterior h es la tasa de aciertos del TLB, t_{ga} es el tiempo medio de gestión de un acierto en el TLB y t_f es el tiempo medio de gestión de un fallo en el TLB. Del enunciado se sabe que $h = 0{,}95$ y $t_{ga} = 0$. Además se puede considerar que $t_f = t_{am} = 125$ ns. Luego sustituyendo valores y operando se obtiene:

$$t_l = 0{,}95{\cdot}0 + 0{,}05{\cdot}125\,10^{-9} = 6{,}25\,10^{-9} = 6{,}25\,ns$$

Si se considera t_l, el tiempo medio de despacho de una referencia a memoria se calcula de la siguiente forma:

$$t_R = t_l + (1-p)\,t_{am} + p\,t_{gf}$$

Sustituyendo valores y operando se obtiene:

$$t_R = 6{,}25\,10^{-9} + 0{,}92{\cdot}125\,10^{-9} + 0{,}08{\cdot}40\,10^{-3} = 0{,}0032\,\text{s} = 3{,}2\,\text{ms}$$

SISTEMAS OPERATIVOS (Código: 71902048) **Septiembre 2012**

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **NO**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

1. Conteste **razonadamente** a los siguientes apartados:

a) *(1 p)* Señalar los criterios principales considerados en la planificación del procesador dependiendo del tipo de sistema operativo.

b) *(1 p)* ¿Qué reglas sigue el modelo de seguridad multinivel de Bell y La Padula?

2. *(2 p)* Explicar **razonadamente** qué sucede con un proceso que invoca una operación `signal_mon` según la solución de: a) Hoare. b)Hansen. c)Lamport y Redell.

3. *(2 p)* Enumerar y describir **brevemente** las diferentes áreas que se distinguen de forma general en la estructura de un sistema de archivos.

4. *(2 p)* En un computador con 3 instancias de un recurso R_1, 3 instancias de un recurso R_2 y 3 instancias de un recurso R_3 se están ejecutando los procesos P_1, P_2 y P_3. En un cierto instante la matriz **M** de recursos necesitados adicionalmente y la matriz **A** de recursos asignados son:

$$\mathbf{M} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \quad \mathbf{A} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

En cada matriz se ha asociado la fila i al proceso P_i y la columna j al recurso R_j (i, j = 1, 2 y 3). Detectar la posible existencia de interbloqueos usando el algoritmo de Coffman.

5. *(2 p)* Un cierto sistema operativo cuando comienza o continúa con la ejecución de un proceso realiza la prepaginación de su conjunto de trabajo en los marcos asignados al proceso. En un cierto instante de tiempo t_1 un proceso A, cuyo conjunto de trabajo en dicho instante está formado por los números de página 0, 1 y 3, se bloquea y pasa a ejecutarse otro proceso B. En t_2 el proceso A vuelve a ser planificado para ejecución y realiza la siguiente secuencia de referencias de página: 3, 1, 2, 1, 6, 7, 9, 5, 2, 3, 2, 9, 3, 3, 6 y 2. Se desea determinar el número de fallos de página producidos por la ejecución del proceso A a partir del instante t_2 supuesto que el sistema operativo asigna tres marcos a la ejecución del proceso A y utiliza el algoritmo de reemplazamiento FIFO.

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Septiembre 2012

Solución Ejercicio 1

a) El *planificador del procesador* fundamenta su elección del próximo proceso que pasará a ser ejecutado en el procesador en la optimización simultánea de alguno o algunos de los siguientes criterios cuantitativos: utilización del procesador, productividad, tiempo de entrega, tiempo de espera, tiempo de respuesta y plazo de finalización. La elección de los criterios a optimizar depende del tipo de sistema operativo:

   - *Sistema por lotes*. El planificador debe intentar maximizar la utilización del procesador y la productividad. Además debe intentar minimizar el tiempo de entrega.
   - *Sistema de tiempo compartido*. El planificador debe intentar minimizar el tiempo de respuesta.
   - *Sistema de tiempo real*. El planificador debe intentar maximizar el número de plazos de finalización cumplidos.

b) El modelo de seguridad multinivel de Bell y La Padula sigue las siguientes reglas:

   - *Imposibilidad de leer objetos de niveles superiores*. Un usuario solo puede leer un objeto con un nivel de seguridad igual o inferior al del usuario. Esta regla se conoce en la literatura como *propiedad de seguridad simple*.
   - *Imposibilidad de escribir objetos de niveles inferiores*. Un usuario puede escribir un objeto con un nivel de seguridad igual o superior. Esta regla se conoce en la literatura como propiedad * .

   Si se cumplen escrupulosamente estas dos reglas se puede garantizar que no existirán fugas de información desde un nivel de seguridad superior hacia un nivel inferior.

Solución Ejercicio 2

- *Solución propuesta por Hoare*. El proceso invocador A de la operación `signal_mon` se bloquea y se permite ejecutar en el monitor al proceso B desbloqueado por dicha operación. Cuando B finalice un procedimiento o se bloquee en una condición entonces el proceso A se desbloqueará.
- *Solución propuesta por Hansen*. El proceso invocador de la operación `signal_mon` sale del monitor inmediatamente. Luego esta operación aparecería como la sentencia final del procedimiento de un monitor.
- *Solución propuesta por Lamport y Redell*. El proceso A invocador de la operación `signal_mon` continua su ejecución hasta finalizar el procedimiento del monitor o bloquearse en una condición. Luego se retomará la ejecución del proceso B desbloqueado por `signal_mon`.

Solución Ejercicio 3

La estructura de un sistema de archivos depende de cada tipo de sistema de archivos en particular, pero de forma general se suelen distinguir en la misma las siguientes áreas:

- *Bloque de arranque*. Se sitúa al comienzo de la partición y puede contener el código necesario para arrancar un sistema operativo.
- *Estructura de datos con metadatos del sistema de archivos*. Contiene información administrativa y estadística del sistema de archivos, como por ejemplo: un identificador del tipo de sistema de archivos, el número de bloques del sistema de archivos, el número de bloques libres, etc. Algunos sistemas operativos, como por ejemplo UNIX, denominan a esta estructura *superbloque*.
- *Estructura de datos con información sobre los bloques libres en el sistema de archivos*. Generalmente denominada como *lista de bloques libres*. Puede implementarse como un mapa de bits o como una lista enlazada.
- *Estructura de datos para localizar los bloques asignados a los archivos*. Algunas de las más utilizadas son las siguientes:
  - *Lista de nodos índice*. Un *nodo índice* o *nodo-i* es una estructura de datos utilizada por algunos sistemas operativos, como por ejemplo UNIX, para almacenar los atributos de un archivo y la ubicación del archivo en disco.
  - *Tabla de asignación de archivos* (File Allocation Table, FAT). Esta tabla tiene una entrada por cada bloque físico existente en la partición del sistema de archivos. Así la entrada j de la tabla hace referencia al bloque físico j.
- *Área de datos*. Contiene los bloques libres y los bloques de datos asignados a los archivos y a los directorios.

Solución Ejercicio 4

Para poder aplicar el algoritmo de Coffman, en primer lugar hay que calcular el vector de recursos disponibles $\mathbf{R_D}$. Del enunciado se deduce que el vector de recursos existentes $\mathbf{R_E}$ es:

$$\mathbf{R_E} = (\ 3 \quad 3 \quad 3\ )$$

De la matriz **A** se puede deducir el vector de recursos asignados $\mathbf{R_A}$, sumando los elementos de una misma columna:

$$\mathbf{R_A} = (\ 2 \quad 3 \quad 2\ )$$

El vector $\mathbf{R_D}$ se obtiene de la siguiente forma:

$$\mathbf{R_D} = \mathbf{R_E} - \mathbf{R_A} = (\ 3 \quad 3 \quad 3\ ) - (\ 2 \quad 3 \quad 2\ ) = (\ 1 \quad 0 \quad 1\ )$$

Los pasos del algoritmo son los siguientes:

1) Se examina la matriz **A** y se comprueba si alguna de sus filas es igual a (0 0 0). Como no existe ninguna, no se marca ningún proceso como completado.
2) Se realiza la siguiente asignación:

$$\mathbf{X} = \mathbf{R_D} = (\ 1 \quad 0 \quad 1\ )$$

3) Para cada proceso P_i no marcado como completado, en este caso los tres procesos, se comprueba la siguiente condición:

$$\mathbf{M}_i \leq \mathbf{X}$$

3.1) La primera fila de **M** asociada al proceso P_1 no cumple la condición:

$$(\ 1 \quad 1 \quad 1\ ) \leq (\ 1 \quad 0 \quad 1\ )$$

3.2) La segunda fila de **M** asociada al proceso P_2 no cumple la condición:

$$(\ 1 \quad 1 \quad 0\ ) \leq (\ 1 \quad 0 \quad 1\ )$$

3.3) La tercera fila de **M** asociada al proceso P_3 no cumple la condición:

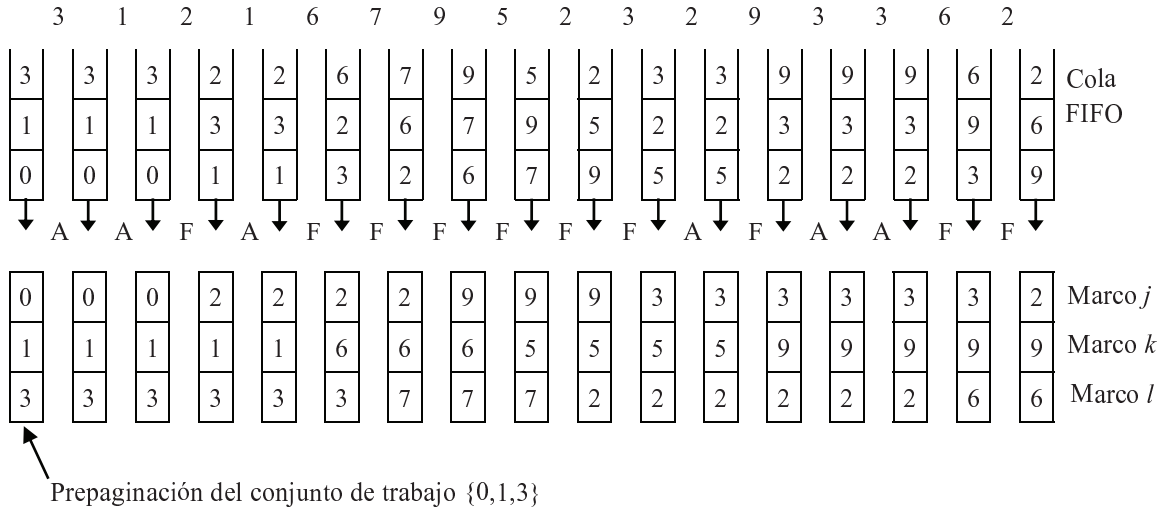
$$(\ 0 \quad 1 \quad 1\ ) \leq (\ 1 \quad 0 \quad 1\ )$$

4) Como no existe ningún proceso que cumpla la condición el algoritmo finaliza.

Se concluye que **existe interbloqueo** ya que los procesos P_1, P_2 y P_3 no han sido marcados.

Solución Ejercicio 5

En primer lugar, antes de continuar con la ejecución del proceso A, el sistema operativo carga en los tres marcos asociados al proceso, a los que denominaremos *j*, *k* y *l*, las páginas 0, 1 y 3, respectivamente, es decir, el conjunto de trabajo del proceso. Además, coloca en la cola FIFO dichos números de página. Una vez realizada la prepaginación del conjunto de trabajo, se continúa con la ejecución del proceso. Las dos primeras referencias ($i = 3$ e $i = 1$) producen sendos aciertos. La tercera referencia ($i = 2$) produce un fallo de página. La página que se sustituye es aquella cuyo número aparece al principio de la cola, en este caso $i = 0$. El número de página $i = 2$ se coloca al final de la cola y el número $i = 0$ se elimina de ella. La página $i = 2$ es cargada en el marco que contenía la página $i = 0$, es decir, en el marco *j*. Examinando de forma análoga cada referencia de la cadena se obtiene que se producen **10 fallos de página**. En la Figura se muestra el contenido de la cola FIFO y el contenido de los marcos *j*, *k* y *l* antes y después de cada referencia de la secuencia. Se indica también si dicha referencia provoca un fallo (F) o un acierto (A).



SISTEMAS OPERATIVOS (Código: 71902048) Enero 2013

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N1**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

ESTE EXAMEN CONSTA DE 5 PREGUNTAS
Preguntas 1 a 4

1. Explicar **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:

a) *(1 p)* Un sistema operativo únicamente puede aplicar la técnica de multiprogramación si el computador en el que está instalado dispone de varios procesadores.

b) *(1 p)* Para poder implementar la memoria virtual es necesario que la arquitectura del computador después de un fallo de página permita reiniciar cualquier instrucción desde la fase (búsqueda o ejecución) del ciclo de instrucción que produjo el fallo de página.

2. *(2 p)* Describir **adecuadamente** las principales configuraciones en función del número y tipo de hilos soportados por un sistema operativo.

3. *(2 p)* Enumerar y describir **brevemente** las capas de software de E/S del núcleo de un sistema operativo.

4. *(2 p)* Un cierto sistema operativo gestiona la memoria principal de un computador utilizando la técnica de particionamiento fijo. Para ello divide la memoria principal en 5 particiones: partición P0 de 10 Ki, partición P1 de 18 Ki, partición P2 de 12 Ki, partición P3 de 8 Ki y partición P4 de 16 Ki. La partición P0 está reservada para el sistema operativo, las otras cuatro particiones se utilizan para cargar procesos de usuarios. El sistema operativo asigna a cada proceso la partición más pequeña en la que quepa. Además mantiene una cola de planificación por cada partición en la que residen aquellos procesos asociados a dicha partición que tienen que esperar para ser cargados debido a que la partición está ocupada por otro proceso. En el instante de tiempo $t = 0$ ut las cuatro particiones están vacías, en $t = 1$ ut llega un proceso A de 8 Ki, en $t = 2$ ut llega un proceso B de 14 Ki, en $t = 3$ ut llega un proceso C de 18 Ki, en $t = 4$ ut llega un proceso D de 6 Ki y en $t = 5$ ut llega un proceso E de 14 Ki. Debido a su tiempo de servicio, un proceso permanece cargado en una partición un tiempo mínimo de 10 ut. Se pide:

a) *(1 p)* Realizar un dibujo **adecuadamente rotulado** que ilustre qué procesos están cargados en las diferentes particiones de memoria y cuáles residen en sus colas asociadas en el instante t= 6 ut.

b) *(1 p)* Calcular la fragmentación interna de cada partición de memoria en el instante t= 6 ut.

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N1**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

ESTE EXAMEN CONSTA DE 5 PREGUNTAS
Pregunta 5

5. *(2 p)* En una oficina de Correos existen 3 ventanillas de atención al cliente. Cuando un cliente entra en la oficina para realizar alguna gestión debe guardar una única cola hasta que alguna ventanilla queda libre. Explicar **razonadamente** si el pseudocódigo del programa que se muestra en la Figura 1 y que utiliza un monitor con la solución de B. Hansen coordina adecuadamente la actividad de los clientes en la oficina.

```
/* Definición del monitor */
monitor oficina
        condición ventanilla_disponible;
        int contador=0;
        void procedimiento1()
        {
          contador = contador + 1;
          if (contador > 3) wait_mon(ventanilla_disponible);
          realizar_gestión();
          contador = contador - 1;
          signal_mon(ventanilla_disponible);
        }

/* Proceso cliente */
void cliente()
{
        oficina.procedimiento1();
}

/* Ejecución concurrente*/
main()
{
        ejecución_concurrente(cliente,...,cliente);
}
```

Figura 1

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Febrero 2013 (1ª Semana)

Solución Ejercicio 1

a) La técnica de *multiprogramación* consiste en mantener en la memoria principal varios trabajos simultáneamente. Si el trabajo que se está ejecutando actualmente en el procesador requiere la realización de una operación de E/S, mientras se completa dicha operación el procesador puede ejecutar otro trabajo. De esta forma se consigue aumentar el rendimiento del procesador.

La multiprogramación requiere que el sistema operativo sea capaz de gestionar la memoria principal y la memoria secundaria. Además es necesario que disponga de un algoritmo de planificación de los trabajos. También es necesario que el hardware soporte mecanismos de protección de la memoria, E/S controlada por interrupciones y DMA.

De lo expuesto anteriormente se concluye que para poder aplicar la multiprogramación no es necesario que el computador disponga de varios procesadores. En consecuencia la afirmación es **FALSA**.

b) Un fallo de página puede producirse al realizar una referencia a memoria virtual en cualquier fase (búsqueda o ejecución) dentro de un ciclo de instrucción. En la fase de búsqueda el fallo de página se puede producir al traducir la dirección virtual de la próxima instrucción a ejecutar (que se encuentra contenida en el registro contador de programa). Por su parte, en la fase de ejecución se puede producir un fallo de página cuando hay que traducir la dirección virtual de algún operando o la dirección virtual donde se va a almacenar el resultado.

Para poder implementar la memoria virtual por demanda de página es necesario que la arquitectura del computador permita reiniciar cualquier instrucción después de un fallo de página. Nótese que independientemente de la fase del ciclo de instrucción en que se produzca el fallo, la instrucción debe ser reiniciada desde el principio, es decir, desde la fase de búsqueda.

De acuerdo con lo anterior se concluye que la afirmación es **FALSA** ya que solo es necesario que la arquitectura permita reiniciar las instrucciones desde la fase de búsqueda y no desde la fase de ejecución.

Solución Ejercicio 2

Dependiendo del número y tipo de hilos soportados por un sistema operativo se pueden distinguir principalmente las siguientes configuraciones:

- *Múltiples hilos de usuario sin soporte de hilos del núcleo*. En esta configuración la gestión de los hilos de usuario de un proceso se realiza mediante una biblioteca de hilos que se ejecuta en modo usuario. Cuando un hilo de un proceso realiza una llamada al sistema comienza a ejecutarse el sistema operativo. Si la realización de dicha llamada al sistema requiere el bloqueo del hilo de usuario entonces se bloquea todo el proceso completo. Además aunque el sistema soporte multiprocesamiento no es posible que varios hilos se ejecuten simultáneamente ya que solamente uno de ellos puede acceder a la vez al sistema operativo. En esta configuración el sistema operativo planifica procesos, cada proceso está constituido por uno o varios hilos de usuario.

- *Un hilo del núcleo por cada hilo de usuario*. En esta configuración el sistema operativo crea un hilo del núcleo asociado a cada hilo de usuario. En consecuencia no es necesario la utilización de una biblioteca de hilos por parte de la aplicación, basta con una interfaz de programación de aplicaciones para acceder a las utilidades de hilos del núcleo.

La principal ventaja de esta configuración es que si un hilo de usuario de un proceso se bloquea a la espera de la finalización de una llamada al sistema se puede planificar otro hilo del mismo proceso, ya que cada uno tiene asignado un hilo del núcleo distinto. El sistema operativo planifica hilos del núcleo, cada uno asociado a un hilo de usuario.

El mayor inconveniente de esta configuración radica en que cada vez que se crea un hilo de usuario es necesario crear un hilo del núcleo asociado lo que produce sobrecarga. Por este motivo los sistemas operativos limitan el número de hilos que es posible crear.

- *Menor número de hilos del núcleo que hilos de usuarios*. En esta configuración el número de hilos del núcleo que soporta el sistema operativo está limitado y es inferior al número de hilos de usuario. Para su implementación se requiere una biblioteca de hilos para gestionar los hilos de usuario. Los diferentes hilos de usuario de un proceso se asocian a un número menor o igual de hilos del núcleo. Es el programador de la aplicación el que se encarga de establecer el número de hilos del núcleo que requieren los hilos de usuario en que ha descompuesto su aplicación.

  Esta configuración, si está bien diseñada, tiene las ventajas de la configuración múltiples hilos de usuario sin soporte de hilos del núcleo y de la configuración un hilo del núcleo por cada hilo de usuario.

Solución Ejercicio 3

De forma general, el software del núcleo de un sistema operativo necesario para la gestión de las operaciones de E/S se puede organizar en tres capas:

- *Subsistema de E/S*. Es el componente del sistema operativo que se encarga de efectuar todas aquellas tareas necesarias para la realización de las operaciones de E/S que son comunes a todos los dispositivos e independientes de los mismos. Es decir, el subsistema de E/S gestiona la parte independiente del dispositivo de todas las operaciones de E/S. Entre las tareas que se encarga de realizar el subsistema de E/S se encuentran las siguientes: asignación y liberación de dispositivos dedicados, bloqueo (si procede) de procesos que solicitan una operación de E/S, planificación de la E/S, buffering, invocación del driver del dispositivo adecuado, y gestión de los errores producidos en las operaciones de E/S.

- *Drivers de dispositivos*. Un driver de dispositivo contiene el código que permite a un sistema operativo controlar un determinado tipo de dispositivo de E/S. Un driver de dispositivo interactúa con el subsistema de E/S y con el controlador de E/S que controla el dispositivo. Un driver suministra al subsistema de E/S el conjunto de funciones que se pueden realizar sobre el dispositivo, tales como lectura o escritura. Además un driver puede invocar a ciertas rutinas o procedimientos del núcleo. Los procedimientos a los que tiene acceso un driver quedan definidos por la interfaz de drivers del subsistema de E/S. El driver de un dispositivo interactúa con el controlador de E/S, cargando en sus registros diferentes órdenes para que las efectúe sobre el dispositivo, comprobando su estado e inicializándolo si es necesario.

- *Manejadores de las interrupciones*. El manejador de una interrupción es una función del núcleo encargada de atender una determinada interrupción. Con objeto de que el rendimiento del computador sea óptimo, los manipuladores de interrupciones tienen una alta prioridad de ejecución. Además su código suele ser pequeño y rápido de ejecutar. Las acciones específicas que realiza un manejador de interrupciones dependen de cada tipo de interrupción. Si el driver del dispositivo se bloqueó en espera de que el controlador de E/S estuviera preparado para procesar otra petición de E/S, entonces una acción que debe realizar un manejador de interrupción es el desbloqueo del driver del dispositivo mediante el uso del mismo mecanismo de sincronización (semáforo, mensajes,...) que utilizó el driver para bloquearse.

Solución Ejercicio 4

a) En el instante $t = 1$ ut llega el proceso A de 8 Ki. Por su tamaño le corresponde, de acuerdo con el enunciado, la partición P3. Como dicha partición está vacía se le asigna al proceso A.

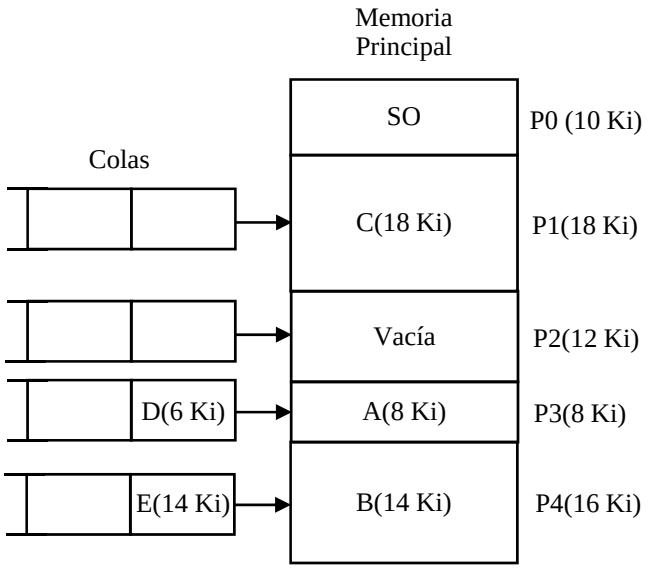
En el instante $t = 2$ ut llega el proceso B de 14 Ki. Por su tamaño le corresponde la partición P4. Como dicha partición está vacía se le asigna al proceso B.

En el instante $t = 3$ ut llega el proceso C de 18 Ki. Por su tamaño le corresponde la partición P1. Como dicha partición está vacía se le asigna al proceso C.

En el instante $t = 4$ ut llega el proceso D de 6 Ki. Por su tamaño le corresponde la partición P3. Como dicha partición está ocupada el proceso D se coloca en la cola de planificación asociada a dicha partición.

En el instante $t = 5$ ut llega el proceso E de 14 Ki. Por su tamaño le corresponde la partición P4. Como dicha partición está ocupada el proceso E se coloca en la cola de planificación asociada a dicha partición.

De acuerdo con lo anterior en el instante $t = 6$ ut el estado de las particiones de memoria y de sus colas asociadas es el que se muestra a continuación



b) El tamaño de la fragmentación interna en la partición P0 asignada al sistema operativo no se puede calcular ya que en el enunciado no se proporciona el dato del tamaño del sistema operativo.

La partición P1 tiene un tamaño de 18 Ki y está asignada al proceso C de tamaño 18 Ki. Puesto que no existe espacio libre en la partición no hay fragmentación interna.

La partición P2 no está asignada a ningún proceso en consecuencia no existe fragmentación interna.

La partición P3 tiene un tamaño de 8 Ki y está asignada al proceso A de tamaño 8 Ki. Puesto que no existe espacio libre en la partición no hay fragmentación interna.

Finalmente la partición P4 tiene un tamaño de 16 Ki y está asignada al proceso B de tamaño 14 Ki. Luego la fragmentación interna es de $16 - 14 = 2$ Ki, es decir, del 12,5 %.

Solución Ejercicio 5

Un monitor garantiza implicitamente la exclusión mutua, de tal forma que si un proceso A está ejecutando un procedimiento de un monitor ningún otro proceso B podrá ejecutar ningún otro procedimiento de dicho monitor hasta que el proceso A finalice la ejecución del procedimiento o se bloquee en una variable de condición.

Teniendo en cuenta esta propiedad de los monitores se deduce que en el monitor `oficina` propuesto la variable `contador` utilizada para llevar la cuenta del número de ventanillas ocupadas solo podrá tomar los valores 0 o 1, ya que cuando un proceso invoca al `procedimiento1` y accede al monitor se incrementa la variable `contador` y toma el valor 1, pero cuando termina su gestión se decrementa y vuelve al valor 0.

En consecuencia el pseudocódigo propuesto **no coordina adecuadamente** la actividad de los clientes en la oficina ya que solo permite usar simultáneamente una de las tres ventanillas existentes.

SISTEMAS OPERATIVOS (Código: 71902048) **Febrero 2013**

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N2**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

1. Explicar **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:

I) *(1 p)* Los *sistemas operativos distribuidos* son aquellos que se ejecutan en redes de computadores y posibilitan que un usuario en un computador de la red conozca la existencia de los otros computadores conectados, y pueda interactuar con dichas máquinas para acceder a sus contenidos y compartir sus recursos.

II) *(1 p)* Una *puerta secreta* es un fragmento de código insertado en un programa o sistema con la finalidad de recopilar información sobre la actividad de sus usuarios para enviársela a terceros.

2. *(2 p)* Enumerar y comentar **brevemente** los atributos de un archivo más frecuentemente mantenidos por un sistema operativo.

3. *(2 p)* Explicar **razonadamente** qué es un dispositivo modo bloque y un dispositivo modo carácter. Señalar algunos ejemplos de cada tipo.

4. *(2 p)* Tres procesos A, B y C se ejecutan concurrentemente en un determinado sistema. El proceso A ejecuta unas tareas (`"Tareas 1"`) y alcanza un punto de encuentro. Posteriormente realiza otras tareas (`"Tareas 2"`) y finaliza. El proceso B ejecuta unas tareas (`"Tareas 3"`) y llega al punto de encuentro. Posteriormente realiza otras tareas (`"Tareas 4"`) y finaliza. Por su parte el proceso C ejecuta unas tareas (`"Tareas 5"`) y llega al punto de encuentro. Posteriormente realiza otras tareas (`"Tareas 6"`) y finaliza. El primer proceso que llega al punto de encuentro no puede continuar su ejecución hasta que no lleguen los otros dos procesos. No se sabe qué proceso comienza a ejecutarse primero o cuál es el primero que termina. Escribir en pseudocódigo un programa que usando **semáforos binarios** coordine la actividad de los procesos A, B y C. Dicho programa debe tener cinco partes: declaración de variables y semáforos, código del proceso A, código del proceso B, código del proceso C y código para inicializar los semáforos y lanzar la ejecución concurrente de los tres procesos.

5. *(2 p)* La memoria principal de un cierto computador tiene un capacidad de 512 MiB, el sistema operativo instalado en dicho computador gestiona la memoria principal usando la técnica de paginación simple con un tamaño de página de 2 KiB. El tamaño máximo que puede tener una tabla de página es de 27 KiB y cada entrada de una tabla de páginas ocupa un tamaño de 24 bits. Calcular el tamaño máximo que puede tener el espacio de direcciones lógicas de un proceso.

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Febrero 2013 (2ª Semana)

Solución Ejercicio 1

a) La afirmación es **FALSA**, ya que la definición de *sistemas operativos distribuidos* que se da en la afirmación corresponde en realidad a la definición de los *sistemas operativos en red*. Los *sistemas operativos distribuidos* son aquellos que se ejecutan en sistemas informáticos distribuidos, los cuales se implementan mediante redes de computadores cuyos procesadores no comparten ni reloj ni memoria, y que se comunican entre sí mediante buses de alta velocidad o líneas telefónicas. En cada computador de la red se ejecuta un sistema operativo, los diferentes sistemas operativos cooperan estrechamente para dar a los usuarios la ilusión de que únicamente existe un único sistema operativo. Cuando un usuario ejecuta un programa no sabe si se está ejecutando en el procesador de su máquina o en el de otra de la red, y lo mismo se aplica al almacenamiento de archivos.

b) La afirmación es **FALSA**, ya que la definición de *puerta secreta* que se da en la afirmación corresponde en realidad a la definición de un *programa espía*. Una *puerta secreta* es un fragmento de código insertado en un programa o sistema con la finalidad de poder saltarse los procedimientos de autenticación o ganar privilegios. Se activa al introducir ciertos secuencias especiales de entrada o con una determinada secuencia de eventos.

Solución Ejercicio 2

Además del nombre de un archivo, el sistema operativo tiene que mantener otras informaciones relativas a un archivo. A los elementos componentes de dicha información se les denomina *atributos de un archivo*. La lista de atributos de un archivo varía en función de cada sistema de archivos. Entre los atributos de un archivo más frecuentemente mantenidos por un sistema de operativo se encuentran los siguientes:

- *Tipo del archivo*. Es un campo compuesto de uno o varios bits que informa al sistema operativo sobre el tipo de archivo.
- *Tamaño*. Incluye al tamaño actual en bytes, palabras o bloques. Además puede incluir información sobre el tamaño máximo permitido.
- *Localización*. Contiene la ubicación del archivo en memoria secundaria.
- *Creador y propietario*. Para identificar al usuario que creó el archivo y a su actual propietario.
- *Permisos de acceso*. Para determinar qué usuarios pueden acceder al archivo y qué operaciones (lectura, escritura, ejecución) sobre el mismo tienen permitidas.
- *Información asociada al tiempo*. Como por ejemplo la fecha y hora de creación del archivo, la fecha y hora del último acceso al archivo, y la fecha y hora de la última modificación.

Solución Ejercicio 3

De forma general los dispositivos de E/S se pueden clasificar en dos grandes categorías: *dispositivos modo bloque* y *dispositivos modo carácter*.

Un *dispositivo modo bloque* almacena información en bloques de tamaño fijo, normalmente 512 bytes o una potencia de dos múltiplo de la anterior, cada uno de los cuales tiene asignado un determinado número identificador o dirección. Ejemplos de dispositivos modo bloque son los discos y las unidades de cinta.

Un *dispositivo modo carácter* envía o recibe información como una secuencia o flujo lineal de bytes, en ellos la información no se organiza con una estructura concreta por lo que no es direccionable, y en consecuencia no permite la realización de operaciones de búsqueda. Ejemplos de dispositivos modo carácter son las impresoras, el ratón, el teclado y el módem.

Solución Ejercicio 4

La solución que se propone para modelar el punto de encuentro entre los procesos A, B y C utiliza seis semáforos binarios: `SAB`, `SBA`, `SAC`, `SCA`, `SBC` y `SCB`. Todos los semáforos son inicializados al valor 0 ya que se utilizan para sincronizar.

```
semáforos binarios SAB, SBA, SAC, SCA, SBC, SCB; /* Definición semáforos binarios*/

void proceso_A() /* Proceso A */
{
        tareas_1();
        signal_sem(SAB); /* A avisa a B */
        signal_sem(SAC); /* A avisa a C */
        wait_sem(SBA);   /* A espera a que le avise B */
        wait_sem(SCA);   /* A espera a que le avise C */
        tareas_2();
}

void proceso_B() /* Proceso B */
{
        tareas_3();
        signal_sem(SBA); /* B avisa a A */
        signal_sem(SBC); /* B avisa a C */
        wait_sem(SAB);   /* B espera a que le avise A */
        wait_sem(SCB);   /* B espera a que le avise C */
        tareas_4();
}

void proceso_C() /* Proceso C */
{
        tareas_5();
        signal_sem(SCA); /* C avisa a A */
        signal_sem(SCB); /* C avisa a B */
        wait_sem(SAC);   /* C espera a que le avise A */
        wait_sem(SBC);   /* C Espera a que le avise B */
        tareas_6();
}

main() /* Inicialización de semáforos y ejecución concurrente */
{
        init_sem(SAB,0); init_sem(SBA,0);
        init_sem(SAC,0); init_sem(SCA,0);
        init_sem(SBC,0); init_sem(SCB,0);

        ejecución_concurrente(proceso_A,proceso_B, proceso_C);
}
```

Solución Ejercicio 5

En primer lugar hay que determinar el número de entradas N_E que tendrá una tabla de páginas de tamaño máximo $C_{TP} = 27$ KiB con un tamaño de entrada $S_E = 24$ bits. Estas tres magnitudes están relacionadas a través de la siguiente expresión:

$$C_{TP} = S_E \cdot N_E$$

Despejando N_E, sustituyendo valores y operando se obtiene lo siguiente:

$$N_E = \frac{C_{TP}}{S_E} = \frac{27\,\text{KiB}}{24\,(\text{bits/entrada})} = \frac{27 \cdot 2^{10}\text{B}}{3\,(\text{B/entrada})} = 9 \cdot 2^{10}\ \text{entradas}$$

Cada entrada de una tabla de páginas de un proceso esta asociada a una página de dicho proceso, luego el número de entradas de la tabla N_E es igual al número de páginas del proceso N_P.

Finalmente multiplicando N_P por el tamaño de una página, en este caso $S_P = 2$ KiB se obtiene el tamaño máximo C_{Xmax} que puede tener el espacio de direcciones lógicas de un proceso:

$$C_{Xmax} = S_P \cdot N_P = 2(\text{KiB/página}) \cdot 9 \cdot 2^{10}\ \text{páginas} = 2 \cdot 2^{10} \cdot 9 \cdot 2^{10}\,\text{B} = 18 \cdot 2^{20}\,\text{B} = 18\,\text{MiB}$$

SISTEMAS OPERATIVOS (Código: 71902048) **Septiembre 2013**

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

1. Conteste **razonadamente** a las siguientes cuestiones:

a) *(1 p)* Enumerar la secuencia de eventos que se producen cuando un programa de usuario invoca una llamada al sistema.

b) *(1 p)* ¿Qué tipo de planificación, expropiativa o no expropiativa, es más recomendable para un sistema operativo por lotes? ¿Y para un sistema de tiempo compartido?

2. *(2 p)* Explicar **razonadamente** las tres diferencias fundamentales entre una instantánea y una copia de seguridad.

3. *(2 p)* Enumerar las ventajas y los inconvenientes de la paginación simple.

4. *(2 p)* En un cierto computador se ha monitorizado el uso del procesador durante un intervalo de observación de 180 ut. Determinar la sobrecarga (expresada en tanto por ciento) asociada a la ejecución de tareas administrativas del sistema operativo si se sabe que el procesador no ha sido utilizado durante 25 ut, y que dos procesos A y B han sido ejecutados exclusivamente en modo usuario durante un tiempo de 110 ut.

5. *(2 p)* Considérense los procesos A, B y C que comparten un recurso del que existen 12 instancias. En la siguiente tabla se muestra el número de instancias asignadas y el número máximo de instancias necesitadas por cada proceso en un cierto instante de tiempo T.

Proceso	Instancias asignadas	Instancias máximas necesitadas
A	3	4
B	5	8
C	1	5

Se pide:

a) Determinar para el instante T la matriz **N** de recursos máximos necesitados por cada proceso, la matriz **A** de recursos asignados a cada proceso, el vector $\mathbf{R}_{\mathrm{E}}$ de recursos existentes y el vector $\mathbf{R}_{\mathrm{D}}$ de recursos disponibles.

b) Haciendo uso de las matrices y vectores obtenidos en el apartado anterior determinar **razonadamente** si el estado del sistema en el instante T es seguro.

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Septiembre 2013

Solución Ejercicio 1

a) Cuando un programa de usuario invoca a una llamada al sistema se produce una trampa, que provoca la conmutación hardware de modo usuario a modo supervisor y transfiere el control al núcleo. El núcleo examina el identificador numérico de la llamada (cada llamada tiene asignado uno) y los parámetros de la llamada para poder localizar y ejecutar a la rutina del núcleo asociada a la llamada al sistema. Cuando dicha rutina finaliza, almacena el resultado en algún registro y provoca la conmutación hardware de modo supervisor a modo usuario para que el proceso de usuario que invocó la llamada continúe su ejecución.

b) Una planificación expropiativa produce una mayor sobrecarga al sistema que una no expropiativa, ya que se ejecuta con mayor frecuencia el planificador y se realizan más cambios de proceso; pero proporciona un mejor servicio a la población de procesos ya que previene que un proceso monopolice el uso del procesador. Además, la realización de una planificación expropiativa requiere que el núcleo del sistema operativo disponga de mecanismos de sincronización para evitar la posible corrupción de sus estructuras de datos.

   En un sistema operativo por lotes una buena elección es utilizar una planificación no expropiativa o una planificación expropiativa con cuantos grandes para cada proceso, ya que no existen usuarios que se encuentren esperando impacientemente en sus terminales por una respuesta rápida a sus peticiones.

   Por su parte, en un sistema de tiempo compartido o interactivo se debe utilizar planificación expropiativa con objeto de poder atender las peticiones de todos los usuarios.

Solución Ejercicio 2

Las tres diferencias fundamentales entre una instantánea y una copia de seguridad son:

- *Las instantáneas se toman de forma prácticamente inmediata y no ocupan espacio en el momento de su creación*. Tomar una instantánea solo requiere cambiar el número de versión actual. Además no se copia ningún bloque de disco hasta que sea necesario modificarlo.

- *Las instantáneas pueden ser restauradas con gran rapidez*. Para ello basta con volver a montar el sistema de archivos a partir de la versión anterior del sistema que se encuentra en el disco. Esto proporciona un mecanismo de recuperación frente a fallos muy eficiente. Por el contrario, la restauración de una copia de seguridad requiere en general mucho tiempo ya que obliga a reescribir completamente todos los archivos que se encuentren en la copia.

- *Las instantáneas residen en el mismo disco físico que los datos que respaldan*. De este modo si el disco se daña se perderá el acceso a las instantáneas y no podría restaurarse el sistema de archivos. Por este motivo las instantáneas deben complementarse copias de seguridad que almacenen los datos en un medio físico de respaldo separado que pueda utilizarse en caso de desastre.

Solución Ejercicio 3

Las ventajas de la *paginación simple* son las siguientes:

- *No produce fragmentación externa*. La memoria principal está dividida en marcos y cualquier marco de página libre está disponible para alojar la página de un proceso.
- *Produce una sobrecarga pequeña*. La gestión de las estructuras de datos asociadas a la paginación no requiere mucho tiempo de uso del procesador por parte del sistema operativo.
- *No necesita intervención humana*. La paginación es una técnica implementada y gestionada por completo por el sistema operativo. No requiere, por lo tanto, la intervención humana en ningún momento, a diferencia por ejemplo, del particionamiento fijo donde el administrador debe definir el número de particiones de la memoria principal.
- *Permite compartir entre varios procesos un código común*. Si el código de un programa es reentratante puede ser compartido por varios procesos, de tal forma que solo es necesario tener cargadas en memoria principal una copia de las páginas del código del programa independientemente del número de instancias de dicho programa que se ejecuten, cada instancia tendrá un proceso asociado. Por tanto permite ahorrar memoria.

La principal desventaja del uso de la paginación es que produce *fragmentación interna*. Puede demostrarse que en promedio la fragmentación interna es del orden de media página por proceso.

Una característica de la paginación, que en algunas ocasiones puede considerarse como una ventaja y en otras como un inconveniente, es que *la paginación es invisible al programador*. Un programador desconoce y tampoco necesita conocer a la hora de programar en cuáles y cuántas páginas se va a descomponer una determinada parte de su programa.

Solución Ejercicio 4

Se cumple la siguiente relación:

$$T = T_{no\_cpu} + T_{mu} + T_{mn}$$

Donde T es el intervalo de observación, T_{no_cpu} es el tiempo que la CPU ha estado inactiva, T_{mu} es el tiempo de ejecución en modo usuario y T_{mn} es el tiempo de ejecución en modo núcleo. Como los procesos se han ejecutado exclusivamente en modo usuario entonces el tiempo de ejecución en modo núcleo coincide con la sobrecarga. Despejando T_{mn}, sustituyendo valores y operando se obtiene que la sobrecarga es:

$$T_{mn} = T - T_{no\_cpu} - T_{mu} = 180 - 25 - 110 = 45\,\text{ut}$$

Para expresar la sobrecarga en tanto por ciento hay que dividirla entre el tiempo de uso del procesador y multiplicar por cien. Procediendo de esta forma se obtiene el siguiente resultado:

$$\frac{T_{mn}}{T - T_{no\_cpu}}\,100 = \frac{45}{180 - 25}\,100 = \frac{45}{155}\,100 = 29{,}03\,\%$$

Solución Ejercicio 5

a) La matriz **N** de recursos máximos necesitados por cada proceso y la matriz **A** de recursos asignados a cada proceso se obtienen directamente de la tabla dada en el enunciado, en este caso como solo existe un tipo de recurso se tienen matrices (vectores) de dimensión 3x1:

$$\mathbf{N} = \begin{bmatrix} 4 \\ 8 \\ 5 \end{bmatrix} \mathbf{A} = \begin{bmatrix} 3 \\ 5 \\ 1 \end{bmatrix}$$

El vector $\mathbf{R_E}$ de recursos existentes se obtiene también del enunciado, en este caso como solo existe un tipo de recurso el vector es un escalar:

$$\mathbf{R_E} = 12$$

Sumando los elementos de la única columna de la matriz A se obtiene el vector de recursos asignados $\mathbf{R_A}$, que también en este caso es un escalar:

$$\mathbf{R_A} = 9$$

Finalmente el vector de recursos disponibles $\mathbf{R_D}$ se obtiene de la siguiente forma:

$$\mathbf{R_D} = \mathbf{R_E} - \mathbf{R_A} = 12 - 9 = 3$$

b) En primer lugar hay que calcular el número de instancias del recurso que todavía necesita cada proceso. Éste se obtiene restando la matriz **N** y la matriz **A**:

$$\mathbf{N} - \mathbf{A} = \begin{bmatrix} 4 \\ 8 \\ 5 \end{bmatrix} - \begin{bmatrix} 3 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix}$$

En segundo lugar hay que comprobar si existe alguna fila i de $\mathbf{N} - \mathbf{A}$, es decir, algún proceso que cumpla la siguiente condición:

$$(\mathbf{N}_i - \mathbf{A}_i) \leq \mathbf{R_D} \quad i = 1, 2, 3$$

La primera fila asociada al proceso A si cumple la condición ya que

$$1 \leq 3$$

La segunda fila asociada al proceso B si cumple la condición ya que

$$3 \leq 3$$

La tercera fila asociada al proceso C no cumple la condición puesto que

$$4 \leq 3$$

Luego al proceso A se le pueden conceder todos los recursos que necesita para completarse aunque los solicite todos a la vez. Supóngase que el proceso A se ha completado, el estado del sistema pasaría a ser S_2:

$$S_2 = \left\{ \mathbf{N} = \begin{pmatrix} 0 \\ 8 \\ 5 \end{pmatrix} \quad \mathbf{A} = \begin{pmatrix} 0 \\ 5 \\ 1 \end{pmatrix} \quad \mathbf{R_E} = \ 12 \quad \mathbf{R_D} = \ 6 \right\}$$

Ahora la matriz $\mathbf{N} - \mathbf{A}$ es:

$$\mathbf{N} - \mathbf{A} = \begin{pmatrix} 0 \\ 3 \\ 4 \end{pmatrix}$$

Como antes hay que comprobar si existe alguna fila i de $\mathbf{N} - \mathbf{A}$, sin considerar $i = 1$, que cumpla la condición:

$$(\mathbf{N}_i - \mathbf{A}_i) \leq \mathbf{R_D}$$

La segunda fila asociada al proceso B si cumple la condición ya que

$$3 \leq 6$$

La tercera fila asociada al proceso P_3 también cumple la condición ya que

$$4 \leq 6$$

Luego al proceso B se le pueden conceder todos los recursos que necesita para completarse aunque los solicite todos a la vez. Supóngase que el proceso B se ha completado, el estado del sistema pasaría a ser S_3:

$$S_2 = \left\{ \mathbf{N} = \begin{pmatrix} 0 \\ 0 \\ 5 \end{pmatrix} \quad \mathbf{A} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \quad \mathbf{R_E} = \ 12 \quad \mathbf{R_D} = \ 11 \right\}$$

La matriz $\mathbf{N} - \mathbf{A}$ es:

$$\mathbf{N} - \mathbf{A} = \begin{pmatrix} 0 \\ 0 \\ 4 \end{pmatrix}$$

Se comprueba que la fila $i = 3$ de $\mathbf{N} - \mathbf{A}$ asociada al proceso C si cumple la condición ya que

$$4 \leq 11$$

Luego al proceso C se le pueden conceder todos los recursos que necesita para completarse aunque los solicite todos a la vez. En conclusión el estado del sistema en el instante T es un estado **seguro**, ya que a partir de él es posible completar la ejecución de los tres procesos.

SISTEMAS OPERATIVOS (Código: 71902048) **Enero 2014**

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N1**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

1. Explicar **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:

a) *(1 p)* En los sistemas operativos modernos el subsistema de E/S proporciona una interfaz uniforme para los drivers de los dispositivos.

b) *(1 p)* La estrategia de evitación de interbloqueos consiste en impedir que se produzca alguna de las cuatro condiciones necesarias para que se produzca el interbloqueo.

2. *(2 p)* Describir **adecuadamente** el algoritmo de planificación basado en múltiples colas de prioridad y realimentación.

3. *(2 p)* Enumerar los principios de diseño de sistemas operativos seguros propuestos por Saltzer y Schroeder.

4. *(2 p)* Un cierto sistema operativo gestiona la memoria principal mediante paginación simple. El tamaño de página utilizado es de 2048 bytes. La memoria física disponible para los procesos es de 8 MiB. Al sistema llegan dos procesos A y B cuya carga en la memoria principal consume 31566 bytes y 18432 bytes, respectivamente. Determinar la fragmentación interna y la fragmentación externa que provoca la carga de cada proceso.

5. *(2 p)* El baño de caballeros de un centro comercial posee una capacidad para cuatro caballeros. Cuando el servicio está completo los caballeros que desean pasar deben esperar fuera haciendo cola al lado de la puerta. Además si el operario de limpieza está limpiando el baño no puede pasar ningún caballero. Por otra parte, el operario solo pasa a limpiar el baño si éste está vacío. Escribir el pseudocódigo de un programa que usando **paso de mensajes** coordine la actividad de los caballeros y del operario de limpieza. Suponer que la comunicación es indirecta a través de buzones y que se dispone de la operación `send` sin bloqueo y de la operación `receive` con bloqueo. El pseudocódigo del programa debe tener cuatro partes: declaración de variables, código caballero, código operario limpieza y código para inicializar los buzones y lanzar la ejecución concurrente de los procesos.

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Febrero 2014 (1ª Semana)

Solución Ejercicio 1

a) El subsistema de E/S debe proporcionar una interfaz para los drivers de los dispositivos. En los sistemas operativos modernos, esta interfaz es *uniforme*, es decir, las funciones que debe aportar el driver y las funciones del núcleo que puede invocar son las mismas para todos los drivers. Si la interfaz no fuese uniforme habría que modificar el código del sistema operativo para adaptar el subsistema de E/S a cada nuevo driver. Además el subsistema de E/S tendría la necesidad de conocer qué funciones puede ejecutar cada driver, ya que todos no pueden ejecutar las mismas. El disponer de una interfaz de drivers de dispositivos uniforme permite cargar dinámicamente el driver de un nuevo dispositivo, sin necesidad de tener que recompilar y enlazar el código del sistema operativo. En consecuencia la afirmación es **VERDADERA**.

b) Esta afirmación es **FALSA** ya que la estrategia de *evitación de interbloqueos* consiste en examinar las posibles consecuencias de asignar los recursos solicitados y evitar el interbloqueo mediante la no asignación de recursos en situaciones potencialmente peligrosas.

Solución Ejercicio 2

En el *algoritmo de planificación basada en múltiples colas de prioridad y realimentación* existen varias colas de procesos preparados o *colas de prioridad*, cada una de las cuales solo pueda contener procesos con una determinada prioridad o dentro de un rango de prioridades. Para planificar cada cola se puede utilizar un algoritmo distinto: FCFS, SJF, turno rotatorio, etc. Siempre se ejecuta un proceso perteneciente a la cola de preparados con mayor prioridad. Sólo cuando dicha cola está vacía se puede planificar un proceso perteneciente a una cola de preparados de menor prioridad. La prioridad de un proceso puede ser modificada dinámicamente durante su existencia, en consecuencia un proceso puede ir cambiando de cola de prioridad (realimentación). La implementación de la realimentación requiere definir un mecanismo que regule cómo se modifica dinámicamente la prioridad de los procesos durante su tiempo de existencia. O visto de otra forma, cuándo se produce la transición de un proceso de una cola de prioridad a otra.

Solución Ejercicio 3

Los principios de diseño de sistemas operativos seguros propuestos por Saltzer y Schroeder son los siguientes:

- *El diseño del sistema deber ser público*. Confiar la seguridad del sistema en el hecho de que su diseño es secreto solo sirve como medida disuasoria temporal. Es cuestión de tiempo que el diseño se descubra.
- *El estado por defecto es el de no acceso*. Los derechos de acceso de un usuarios sobre un objeto deben concederse explícitamente.
- *Comprobación de los derechos de acceso*. Cada vez que un usuario solicita acceder a un recurso se debe comprobar que tiene los derechos de acceso adecuados.
- *Principio del mínimo privilegio*. Los procesos solo deben tener los derechos de acceso mínimo necesarios para completar su tarea. De esta forma se limitan los daños que puede ocasionar el ataque de un caballo de Troya.
- *Los mecanismos de protección debe ser simples y estar integrados en las capas más bajas del sistema*. De esta forma se facilita la verificación y el buen funcionamiento de las implementaciones.
- *Los mecanismos de protección deben ser aceptados por los usuarios*. Los mecanismos de protección deben ser fáciles de usar por los usuarios ya que en caso contrario los usarán incorrectamente, o directamente no los usarán.

Solución Ejercicio 4

La técnica de paginación simple **no produce fragmentación externa**, solo produce *fragmentación interna* debida al posible espacio libre que puede existir en la última página del espacio de direcciones lógicas de un proceso.

1) Cálculo de la fragmentación interna del proceso A:

   Para calcular la fragmentación interna en primer lugar se va a calcular el número de páginas N_P en que se descompone el espacio de direcciones lógicas del proceso A:

$$N_P = \text{ceil}\left(\frac{C_X}{S_P}\right)$$

   Donde C_X es el tamaño del espacio de direcciones lógicas de proceso A y S_P es el tamaño de una página. Sustituyendo valores y operando se obtiene que:

$$N_P = \text{ceil}\left(\frac{31566}{2048}\right) = \text{ceil}\,(15{,}413) = 16\,\text{páginas}$$

   El espacio ocupado por estas 16 páginas ($i = 0, 1, 2, ..., 15$) es $16 \cdot 2048 = 32768$ bytes. Puesto que el espacio de direcciones lógicas del proceso A tiene un tamaño de 31566 bytes, el espacio libre existente en la última página asignada al proceso (página $i = 15$) es igual a:

$$32768 - 31566 = 1202 \text{ bytes}$$

   Por lo tanto, la fragmentación interna provocada por la carga del proceso A es de **1202 bytes** o si se expresa en porcentaje el 58,69 % del tamaño de una página.

2) Cálculo de la fragmentación interna del proceso B:

$$N_P = \text{ceil}\left(\frac{18432}{2048}\right) = \text{ceil}\,(9) = 9 \text{ páginas}$$

El espacio ocupado por estas 9 páginas ($i = 0, 1, 2, ..., 8$) es $9 \cdot 2048 = 18432$ bytes. Puesto que el espacio de direcciones lógicas del proceso A tiene un tamaño de 18432 bytes, el espacio libre existente en la última página asignada al proceso (página $i = 8$) es igual a:

$$18432 - 18432 = 0 \text{ bytes}$$

Por lo tanto, la fragmentación interna provocada por la carga del proceso A es de **0 bytes**. Es decir, la carga de del proceso A no produce fragmentación interna.

Solución Ejercicio 5

La solución que se propone en la Figura 1 para este problema utiliza las siguientes variables globales y buzones:

- `contador`. Variable global de tipo entero para llevar la cuenta del número de caballeros.
- `B1`. Buzón que se utiliza para garantizar la exclusión mutua en el uso de la variable global `contador`.
- `B2`. Buzón que se utiliza para sincronizar la actividad del operario de limpieza y los caballeros.
- `B3`. Buzón que se utiliza para sincronizar la entrada al baño de los caballeros.

```
int contador=0;

void caballero()  /* Proceso caballero */
{
    mensaje m;

    receive(B1,m);
    contador=contador + 1;

    if(contador == 1)
    {
        receive(B2,m);  /* Esperar si el operario está en el baño */
        send(B1,m);
    }
    else if(contador > 4)
    {
        send(B1,m);
        receive(B3,m);  /* Esperar si el baño está completo */
    }
    else send(B1,m);

    usar_baño();

    receive(B1,m);
    if (contador == 1) send(B2,m);  /* Avisa al operario para que pase */
    else if (contador >4) send(B3,m);  /* Avisa a un caballero para que pase */
    contador = contador - 1;
    send(B1,m);
}

void operario_limpieza()  /* Proceso operario_limpieza */
{
    receive(B2,m);  /* Esperar si el baño está ocupado */
    limpiar_baño();
    send(B2,m);  /* Avisa al primer caballero para que pase */
}

void main()  /* Inicialización de buzones y ejecución concurrente */
{
    mensaje nulo;

    crear_buzón(B1);
    send(B1,nulo);

    crear_buzón(B2);
    send(B2,nulo) ;

    crear_buzón(B3);
    ejecución_concurrente(caballero,...,caballero,operario_limpieza);
}
```

Figura 1

SISTEMAS OPERATIVOS (Código: 71902048) **Febrero 2014**

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N2**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

1. Explicar **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:

I) *(1 p)* El número de entradas que puede tener una tabla de páginas invertida depende del número de páginas en que se divida el espacio de direcciones virtuales del proceso al que está asociada.

II) *(1 p)* El *planificador a largo plazo* de un sistema operativo da más prioridad a los trabajos limitados por CPU que a los trabajos limitados por E/S.

2. *(2 p)* Enumerar las acciones que de forma general suele realizar un sistema operativo para crear un proceso.

3. *(2 p)* Enumerar y describir las diferentes áreas que se distinguen de forma general en la estructura de un sistema de archivos.

4. *(2 p)* El acceso de los ciudadanos a una comisaria de policía para realizar gestiones relativas a sus DNIs o pasaportes está regulado por un agente de policía. Los ciudadanos esperan a la puerta de la comisaria en cola por orden de llegada y el agente cada 15 minutos avisa a los 10 primeros ciudadanos de la cola para que pasen dentro a realizar sus gestiones. Si no hay ciudadanos en la cola el agente no realiza ningún aviso y si hay N ciudadanos en la cola con N menor de 10 el agente solo realiza N avisos. Suponer que independientemente del número de ciudadanos que hayan pasado la vez anterior, el agente solo realiza su acción de avisar ciudadanos cada 15 minutos. Escribir el pseudocódigo de un programa que usando **semáforos generales** coordine la actividad del agente y los ciudadanos para acceder a la comisaria. El pseudocódigo del programa que se realice en cada apartado debe tener cuatro partes: declaración de variables, código del ciudadano, código del agente y código para inicializar los semáforos y lanzar la ejecución concurrente de los procesos.

5. *(2 p)* El sistema operativo en colaboración con el hardware gestiona la memoria principal mediante paginación por demanda. La traducción de direcciones se realiza usando una MMU con un TLB. El tiempo medio de acceso al TLB es despreciable y su tasa de aciertos es del 95 %. Determinar el tiempo medio de despacho de una referencia a memoria si se tienen: una tasa de fallos de página del 8 %, un tiempo medio de acceso a memoria de 90 ns y un tiempo medio de gestión de un fallo de página de 20 ms. Despreciar la existencia de memoria caché.

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Febrero 2014 (2ª Semana)

Solución Ejercicio 1

a) Una tabla de páginas invertida tiene asociada una entrada por cada marco *j* de memoria principal, en vez de una entrada por cada página *i* de un proceso. De ahí el nombre de tabla invertida. En consecuencia el número de entradas de la tabla de páginas invertida es fijo y es igual al número de marcos. No depende, por tanto, del número de procesos que haya en el sistema o del número de páginas en que se divida el espacio virtual de estos procesos. Por lo tanto la afirmación es **FALSA**.

b) El planificador a largo plazo debe intentar seleccionar trabajos de tal forma que en el sistema exista una mezcla adecuada de procesos limitados por CPU y procesos limitados por E/S. Si la mayoría de los trabajos seleccionado están limitados por CPU, entonces los dispositivos de E/S se encontrarían la mayor parte del tiempo inactivos. Por el contrario, si la mayoría de los trabajos seleccionados están limitados por E/S, sería el procesador el que se encontraría la mayor parte del tiempo inactivo. En ambos casos se estaría produciendo un desequilibrio no deseable en el uso de los recursos del sistema. Por lo tanto la afirmación es **FALSA**.

Solución Ejercicio 2

Las acciones que se deben realizar para crear un proceso dependen de cada sistema operativo, aunque de forma general se pueden distinguir las siguientes:

1. *Comprobar si el proceso puede ser creado*. El sistema operativo comprueba si existe suficiente espacio en memoria principal para crear otro proceso y si el usuario no ha excedido el número máximo de procesos que puede tener ejecutándose.
2. *Asignar una entrada de la tabla de procesos para el nuevo proceso*. En este instante se le asigna un identificador numérico al proceso.
3. *Reservar espacio en memoria para el proceso*. Se debe reservar espacio en memoria principal para el espacio de direcciones de memoria lógica del proceso y para la información de control del proceso.
4. *Inicializar el bloque de control del proceso*. Los campos de la entrada asociada al proceso en la tabla de procesos se inicializan con los valores adecuados.
5. *Establecer los enlaces apropiados*. El sistema operativo debe configurar los punteros adecuados para enlazar la información del proceso en las diferentes listas, tablas y colas que mantiene.

Solución Ejercicio 3

La estructura de un sistema de archivos depende de cada tipo de sistema de archivos en particular, pero de forma general se suelen distinguir en la misma las siguientes áreas:

- *Bloque de arranque*. Se sitúa al comienzo de la partición y puede contener el código necesario para arrancar un sistema operativo. La tabla de particiones indica la *partición activa*, es decir, aquella que contiene el código de arranque del sistema operativo que se quiera cargar al arrancar el computador.

- *Estructura de datos con metadatos del sistema de archivos*. Contiene información administrativa y estadística del sistema de archivos, como por ejemplo: un identificador del tipo de sistema de archivos, el número de bloques del sistema de archivos, el número de bloques libres, etc. Algunos sistemas operativos, como por ejemplo UNIX, denominan a esta estructura *superbloque*.

- *Estructura de datos con información sobre los bloques libres en el sistema de archivos*. Generalmente denominada como *lista de bloques libres*. Puede implementarse como un mapa de bits o como una lista enlazada. Cuando se necesita espacio en disco para crear un archivo o para aumentar el tamaño de un archivo ya existente, el sistema operativo consulta esta estructura para asignar dicho espacio.

- *Estructura de datos para localizar los bloques asignados a los archivos*. Algunas de las más utilizadas son las siguientes:

  – *Lista de nodos índice*. Un *nodo índice* o *nodo-i* es una estructura de datos utilizada por algunos sistemas operativos, como por ejemplo UNIX, para almacenar los atributos de un archivo y la ubicación del archivo en disco. Cada nodo-i tiene asociado un número entero positivo que lo identifica de manera única. Una entrada de la lista de nodos-i contiene la dirección física o número de bloque del disco donde se encuentra el nodo-i asociado a un determinado archivo.

  – *Tabla de asignación de archivos* (File Allocation Table, FAT). Esta tabla tiene una entrada por cada bloque físico existente en la partición del sistema de archivos. Así la entrada j de la tabla hace referencia al bloque físico j. Si el bloque físico j está asignado a un archivo, entonces la entrada j en la FAT contiene la dirección física del siguiente bloque del archivo.

- *Área de datos*. Contiene los bloques libres y los bloques de datos asignados a los archivos y a los directorios.

Solución Ejercicio 4

En la Figura 1 se muestra una posible solución haciendo uso de **semáforos generales**.

```
int contador=0;      /*Para almacenar el número de ciudadanos
                       en la cola de la puerta de la comisaria*/
semáforo_general S1; /*Para garantizar la exclusión mutua en el uso de la variable contador*/
semáforo_general S2; /*Para sincronizar la espera de los ciudadanos en la cola*/

void ciudadano() /*Código ciudadano*/
{
    wait_sem(S1);
    contador=contador+1;
    signal_sem(S1);

    wait_sem(S2); /*Esperar en la cola a que le avise el agente*/

    realizar_gestión();
}

void agente() /*Código agente*/
{
    int numero_avisos;

    while(TRUE)
    {
        /*Inicio bloque aviso ciudadanos*/
        wait_sem(S1);

        if (contador < 10) numero_avisos=contador;
        else numero_avisos=10;

        while(numero_avisos > 0)
        {
            signal_sem(S2); /*Avisa a un ciudadano*/
            numero_avisos=numero_avisos-1;
            contador=contador-1;
        }

        signal_sem(S1);
        /*Fin bloque aviso ciudadanos*/

        esperar_15min();
    }
}

main() /*Inicialización semáforos y ejecución concurrente*/
{
    init_sem(S1,1);
    init_sem(S2,0);
    ejecución_concurrente(ciudadanos, agente);
}
```

Figura 1

Solución Ejercicio 5

La MMU dispone de un TLB en el cual se cargan un cierto número de entradas de la tabla de páginas del proceso en ejecución. Cuando se va a traducir una dirección virtual que referencia a una determinada página *i* se busca en el TLB la entrada *i* de la tabla de páginas. Si se encuentra se produce un acierto, en caso contrario se produce un fallo y hay que acceder a la memoria principal a leer dicha entrada de la tabla de páginas. Luego el tiempo medio de lectura t_l de la entrada de la tabla de páginas asociada a la página referenciada es:

$$t_l = h\, t_{ga} + (1 - h)\, t_f$$

En la expresión anterior h es la tasa de aciertos del TLB, t_{ga} es el tiempo medio de gestión de un acierto en el TLB y t_f es el tiempo medio de gestión de un fallo en el TLB. Del enunciado se sabe que $h = 0{,}95$ y $t_{ga} = 0$. Además se puede considerar que $t_f = t_{am} = 90$ ns. Luego sustituyendo valores y operando se obtiene:

$$t_l = 0{,}95{\cdot}0 + 0{,}05{\cdot}90\, 10^{-9} = 4{,}5\, 10^{-9} = 4{,}5\, ns$$

El tiempo medio de despacho de una referencia a memoria t_R se calcula de la siguiente forma:

$$t_{\mathrm{R}} = t_l + (1 - p)\, t_{am} + p\, t_{gf}$$

donde p es la tasa de fallos de página y t_{gf} es el tiempo medio de gestión de un fallo de página. Sustituyendo valores y operando se obtiene:

$$t_{\mathrm{R}} = 4{,}5\, 10^{-9} + 0{,}92{\cdot}90\, 10^{-9} + 0{,}08{\cdot}20\, 10^{-3} = \mathbf{1{,}6000873\,ms}$$

SISTEMAS OPERATIVOS (Código: 71902048) **Septiembre 2014**

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

1. Explicar **razonadamente** si las siguientes afirmaciones son verdaderas o falsas:

a) *(1 p)* Una de las principales ventajas de implementar la matriz de acceso mediante listas de control de acceso (ACLs) es que el acceso al contenido de una ACL es muy rápido.

b) *(1 p)* El algoritmo del banquero en teoría es una técnica excelente para evitar los interbloqueos, sin embargo en la práctica realmente no se utiliza.

2. *(2 p)* Explicar **razonadamente** las características de la siguiente técnica de gestión de la memoria principal: *particionamiento fijo*.

3. *(2 p)* Enumerar y describir **brevemente** las capas de software de E/S del núcleo de un sistema operativo.

4. *(2 p)* Una persona tiene en su casa una jaula llena de canarios en la que hay un plato de alpiste y un columpio. Todos los canarios quieren primero comer del plato y luego columpiarse, sin embargo sólo tres de ellos pueden comer del plato al mismo tiempo y solo uno de ellos puede columpiarse. Escribir el pseudocódigo basado en C de un programa que usando **paso de mensajes** coordine la actividad de los canarios. Suponer que la comunicación es indirecta a través de buzones y que se dispone de la operación `send` sin bloqueo y de la operación `receive` con bloqueo. Dicho programa debe tener tres partes: declaración de variables, código del proceso canario, y código de la función principal para inicializar los buzones y lanzar la ejecución concurrente de los procesos.

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

5. Supóngase que un determinado sistema operativo asigna cinco marcos de página para la ejecución de un determinado proceso. Además utiliza el **algoritmo de reemplazamiento del reloj** y para implementarlo utiliza una lista enlazada o cola circular de cinco entradas. Cada entrada contiene el número de una página del proceso cargada en memoria. En la Figura 1 se muestra el estado de la cola circular en el instante de tiempo t_0 y un puntero a una página de la cola. Además se muestra el estado del bit referenciada r de la tabla de páginas del proceso para cada página de la cola. Supóngase que en el instante de tiempo t_0 una referencia a la página $i = 7$ produce un fallo de página. Se pide:

a) *(1 p)* Explicar el funcionamiento del algoritmo de reemplazamiento del reloj.

b) *(0.5 p)* Determinar **razonadamente** la página que sería selecciona para ser reemplazada al aplicar este algoritmo.

c) *(0.5 p)* Dibujar el estado final de la cola circular y del puntero tras aplicarse el algoritmo y realizarse el reemplazamiento.

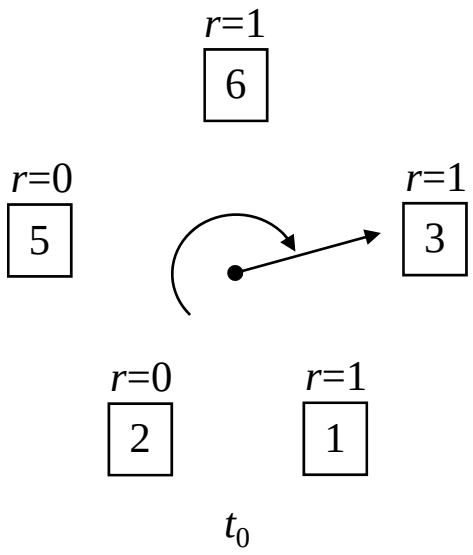


Figura 1

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Septiembre 2014

Solución Ejercicio 1

a) Una ACL se comprueba en cada acceso al objeto al que está asociada. Si los accesos son muy frecuentes o/y la lista es larga se pierde tiempo en las búsquedas en la lista. En conclusión la afirmación es **FALSA**.

b) El *algoritmo del banquero* asegura que el número de recursos asignados a todos los procesos nunca puede exceder del número de recursos del sistema. Además, nunca se puede hacer una asignación peligrosa, es decir, asignar recursos de modo que no queden suficientes para satisfacer las necesidades de todos los procesos. En teoría el algoritmo del banquero es una técnica excelente para evitar los interbloqueos. Sin embargo, en la práctica realmente no se utiliza, ya que es difícil conocer por adelantado los recursos que van a necesitar los procesos durante su ejecución. Además dicho algoritmo parte de la suposición de que el número de procesos y de recursos existentes es fijo, lo cual no tiene por qué ser cierto. En general, la población de procesos varía dinámicamente en el tiempo. Por otra parte, los recursos que inicialmente están disponibles pueden que con el tiempo no lo estén debido a posibles fallos, por ejemplo, que se quede atascado papel en una impresora o que se dañe una unidad de cinta. En conclusión esta afirmación es **VERDADERA**.

Solución Ejercicio 2

La técnica de gestión de la memoria principal conocida como *particionamiento fijo* consiste en dividir la memoria principal en un número fijo N de particiones que pueden ser de igual o de diferente tamaño. Una partición se reserva para contener el sistema operativo. Normalmente está partición se encuentra en un extremo de la memoria, o en la parte baja o en la parte alta. Las restantes $N - 1$ particiones están disponibles para cargar procesos. En cada partición solo se puede cargar un proceso.

El sistema operativo mantiene una estructura de datos, denominada *tabla de descripción de particiones*, que contiene la siguiente información por cada partición: dirección física de comienzo (también denominada dirección base), tamaño y estado (si está libre o ocupada). El sistema operativo se encarga de gestionar el estado de cada partición, ya que la dirección base y el tamaño son fijados en tiempo de arranque.

Cuando se asigna una partición a un proceso, el sistema operativo guarda el número de partición asignada en el bloque de control del proceso. Además marca como ocupada la entrada de la tabla de descripción de particiones asociada a dicha partición.

Cuando un proceso finaliza su ejecución o es intercambiado al área de intercambio en memoria secundaria su partición de memoria queda libre para ubicar a otro nuevo proceso o a un proceso intercambiado procedente del área de intercambio.

Cuando hay que asignar una partición a un proceso se busca una partición libre en la tabla de descripción de particiones.

La técnica de gestión de memoria mediante particionamiento fijo presenta como principal ventaja que es sencilla de implementar para el sistema operativo, únicamente requiere una tabla de descripción de particiones y de una o varias colas para contener a los procesos que deben ser cargados en memoria. La gestión de estas estructuras produce muy poca sobrecarga.

Los principales inconvenientes de esta técnica son que provoca fragmentación interna y que el tamaño máximo de proceso que puede ser cargado en memoria viene limitado por la partición de mayor tamaño que se defina.

Además el número máximo de procesos cargados en memoria principal o grado de multiprogramación

está limitado por el número de particiones que se definan en el momento de arrancar el sistema. Ello repercute en una disminución del rendimiento máximo del sistema, es decir, del grado de utilización del procesador y de los canales de E/S.

Solución Ejercicio 3

De forma general, el software del núcleo de un sistema operativo necesario para la gestión de las operaciones de E/S se puede organizar en tres capas:

- *Subsistema de E/S*. Es el componente del sistema operativo que se encarga de efectuar todas aquellas tareas necesarias para la realización de las operaciones de E/S que son comunes a todos los dispositivos e independientes de los mismos. Es decir, el subsistema de E/S gestiona la parte independiente del dispositivo de todas las operaciones de E/S. Entre las tareas que se encarga de realizar el subsistema de E/S se encuentran las siguientes: asignación y liberación de dispositivos dedicados, bloqueo (si procede) de procesos que solicitan una operación de E/S, planificación de la E/S, buffering, invocación del driver del dispositivo adecuado, y gestión de los errores producidos en las operaciones de E/S.

- *Drivers de dispositivos*. Un driver de dispositivo contiene el código que permite a un sistema operativo controlar un determinado tipo de dispositivo de E/S. Un driver de dispositivo interactúa con el subsistema de E/S y con el controlador de E/S que controla el dispositivo. Un driver suministra al subsistema de E/S el conjunto de funciones que se pueden realizar sobre el dispositivo, tales como lectura o escritura. Además un driver puede invocar a ciertas rutinas o procedimientos del núcleo. Los procedimientos a los que tiene acceso un driver quedan definidos por la interfaz de drivers del subsistema de E/S. El driver de un dispositivo interactúa con el controlador de E/S, cargando en sus registros diferentes órdenes para que las efectúe sobre el dispositivo, comprobando su estado e inicializándolo si es necesario.

- *Manejadores de las interrupciones*. El manejador de una interrupción es una función del núcleo encargada de atender una determinada interrupción. Con objeto de que el rendimiento del computador sea óptimo, los manipuladores de interrupciones tienen una alta prioridad de ejecución. Además su código suele ser pequeño y rápido de ejecutar. Las acciones específicas que realiza un manejador de interrupciones dependen de cada tipo de interrupción. Si el driver del dispositivo se bloqueó en espera de que el controlador de E/S estuviera preparado para procesar otra petición de E/S, entonces una acción que debe realizar un manejador de interrupción es el desbloqueo del driver del dispositivo mediante el uso del mismo mecanismo de sincronización (semáforo, mensajes,...) que utilizó el driver para bloquearse.

Solución Ejercicio 4

La solución que se propone en la Figura 1 para este problema utiliza los siguientes buzones:

- `B1`. Buzón que se utiliza para garantizar la exclusión mutua en el uso del plato. Se inicializa con tres mensajes ya que solo pueden comer del plato tres canarios simultáneamente.
- `B2`. Buzón que se utiliza para garantizar la exclusión mutua en el uso del columpio. Se inicializa con un mensaje ya que solo puede columpiarse un canario simultáneamente.

```
void canario()  /* Proceso canario */
{
    mensaje m;

    receive(B1,m);
    comer_del_plato();
    send(B1,m);

    receive(B2,m);
    columpiarse();
    send(B2,m);

}


void main()  /* Inicialización de buzones y ejecución concurrente */
{
    int h;
    mensaje nulo;

    crear_buzón(B1);
    for(h=1;h<=3;h++)
    {
       send(B1,nulo);
    }

    crear_buzón(B2);
    send(B2,nulo) ;

    ejecución_concurrente(canario,canario,canario,...);
}
```

Figura 1

Solución Ejercicio 5

a) El *algoritmo del reloj* es una implementación del *algoritmo de reemplazamiento de la segunda oportunidad* que utiliza una cola circular y un puntero, los números de páginas contenidos en la cola circular se pueden visualizar situadas sobre un círculo como si fueran las horas de un reloj y el puntero se asemejaría a una manecilla. Este algoritmo busca la página que lleva más tiempo cargada en memoria y no ha sido referenciada recientemente.

El funcionamiento del algoritmo del reloj es el siguiente: en primer lugar consulta el bit referenciada (r) de la página cuyo número i se encuentra en la entrada de la cola circular apuntada por el puntero. Si $r = 0$, entonces la página i es seleccionada para ser reemplazada por la página j. En la cola circular la entrada que contenía el número de página i es sustituido por el número de página j, el puntero pasa a apuntar a la siguiente entrada de la cola y el algoritmo finaliza. Por el contrario, si $r = 1$ entonces el algoritmo pone el bit a 0 y el puntero pasa a apuntar a la siguiente entrada (por ejemplo en sentido horario) de la cola circular. A continuación pasa a consultar el bit referenciada de la página cuyo número se encuentra en la entrada de la cola circular apuntada por el puntero. Y así sucesivamente hasta encontrar una página cuyo bit referenciada esté a cero, que será la página i elegida para ser reemplazada por la página j que ocasionó el fallo de página.

Nótese que en el caso de que todas las páginas de la cola tengan su bit referenciada a 1, el puntero daría una vuelta completa y volvería apuntar a la primera entrada que analizó al principio del algoritmo, ahora la página cuyo número se encuentra en dicha entrada tendría su bit referenciada a cero por lo que sería la página escogida.

b) Como la página $i = 3$ tiene $r = 1$, significa que hace poco que fue referenciada, su bit r se pone a 0 y el puntero pasa a apuntar a la siguiente página. Como la página $i = 1$ tiene $r = 1$, su bit r se pone a 0 y el puntero pasa a apuntar a la siguiente página. Como la página $i = 2$ tiene $r = 0$, se elige para ser reemplazada. En la posición de la cola ocupada por $i = 2$ se coloca $i = 7$ con su bit $r = 1$, ya que acaba de ser referenciada. El puntero pasa a apuntar al siguiente número de página de la cola circular.

Luego la **página i=2** es la elegida para ser reemplazada.

c) En la Figura 2 se muestra el estado de la cola circular y la posición del puntero tras realizarse el reemplazamiento de la página $i = 2$ por la página $i = 7$.

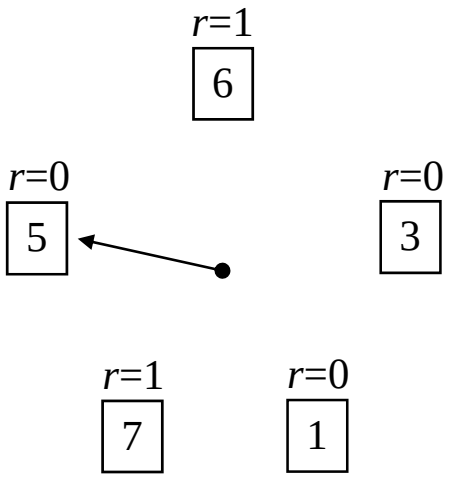


Figura 2

SISTEMAS OPERATIVOS (Código: 71902048) **Enero 2015**

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N1**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

1. Conteste **razonadamente** a las siguientes preguntas:

a) *(1 p)* ¿En qué consiste la técnica de *registro por diario (journaling)*?

b) *(1 p)* ¿En qué consiste el método LBA de acceso a un disco duro?

2. *(2 p)* Describir **adecuadamente** el *algoritmo de reemplazamiento de página mediante envejecimiento (aging)*.

3. *(2 p)* Describir el funcionamiento de las operaciones `wait_mon` y `signal_mon` de un monitor.

4. *(2 p)* Determinar cuál debe ser la duración de las ráfagas *x* de CPU del conjunto de trabajos que se muestran en la Tabla 1 sabiendo que el tiempo de espera de los trabajos T1 y T2 fue de 5 ut, en ambos casos, para el trabajo T3 su tiempo de espera coincide con su tiempo de servicio, y que el tiempo de estancia medio en el sistema fue de 20 ut.

Trabajo	Duración de las ráfagas (ut)
T1	x, 5
T2	3, x, 10
T3	4, x

Tabla 1

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N1**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

5. *(2 p)* El procesador de un computador recibe una interrupción del reloj hardware cada 10 ms. Cuando recibe la interrupción el sistema operativo salva el contexto del proceso en ejecución, ejecuta la rutina de reconocimiento de interrupciones y ésta invoca a la rutina de tratamiento de la interrupción de reloj. Cuando finaliza de atenderse la interrupción de reloj se restaura el contexto del proceso interrumpido y se continua con su ejecución. Si un proceso finaliza se ejecuta el planificador del sistema operativo, que se encarga de seleccionar el próximo proceso que será ejecutado en el procesador, y se realiza un cambio de proceso. En la Tabla 2 se muestran los tiempos promedios que tardan en ejecutarse cada una de estas tareas del sistema operativo. Determinar el instante de finalización del proceso A (expresado en μs) supuesto que en el instante de tiempo $T_1 = 100$ s llega una interrupción de reloj mientras se estaba ejecutando un proceso A al que le restaban 0,8 s de ejecución, y que durante la ejecución de dicho proceso únicamente llegan interrupciones de reloj.

Tareas del sistema operativo	**Tiempo de ejecución promedio (μs)**
Salvar el contexto de un proceso	1, 25
Restaurar el contexto de un proceso	1, 25
Cambio de proceso	6
Reconocimiento de interrupciones	2
Tratamiento de la interrupción del reloj	2
Planificador	4

Tabla 2

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Enero 2015

Solución Ejercicio 1

a) La técnica conocida como *registro por diario* (*journaling*) consiste en que el sistema operativo genera un informe con las operaciones que tiene que realizar sobre el sistema de archivos antes de realizarlas. Dicho informe se almacena en un área reservada de la partición de disco denominada *diario* (*journal*). Una vez que el informe ha sido escrito en el diario se comienzan a realizar las operaciones planificadas. Cuando dichas operaciones han sido completadas el informe se borra. Si se produce algún fallo antes de que se puedan completar todas las operaciones planificadas, cuando se reinicia el sistema operativo éste lee el diario para comprobar si existe un informe. En caso afirmativo se repiten una a una todas las operaciones para que el sistema de archivos quede de nuevo en un estado consistente.

b) El método de acceso conocido como *direccionamiento de bloques lógicos* (Logical Block Addressing, LBA) consiste en considerar un disco duro como un array de bloques lógicos de datos. Cada bloque tiene asignado un número de bloque identificativo o dirección lógica. Cada bloque lógico se hace corresponder con un sector durante el formateo físico. El bloque lógico 0 se suele hacer corresponder con el sector 0 de la primera pista del cilindro más externo. Luego se va realizando la correspondencia de forma secuencial con los restantes sectores de dicha pista. A continuación, por las restantes pistas del cilindro. Después se continúa el proceso por los restantes cilindros hasta llegar al más interno. En resumen la correspondencia consiste en numerar todos los sectores del disco de forma consecutiva comenzando por 0.

Solución Ejercicio 2

El *algoritmo de reemplazamiento mediante envejecimiento* (aging) es una aproximación al algoritmo LRU que introduce una sobrecarga mucho más pequeña ya que se puede implementar de forma bastante eficiente. Básicamente se asigna un registro de desplazamiento software de *n* bits a cada página cargada en memoria principal. El registro se inicializa a 0 cuando la página es cargada en memoria. Cada cierto tiempo T preestablecido, el sistema operativo desplaza un bit a la derecha el contenido del registro asignado a cada página cargando el bit referenciada en el bit más significativo de cada registro. Además pone a 0 el bit referenciada de cada página. La página que se selecciona para ser reemplazada es aquella cuyo registro de desplazamiento contiene el número binario más pequeño, ya que será la página menos usada recientemente. Si existen varias páginas que contienen el mismo valor en su registro, entonces la página que se selecciona para ser reemplazada se selecciona aleatoriamente entre dicho conjunto de páginas.

Solución Ejercicio 3

- `wait_mon(X)`. El proceso dentro del monitor que realiza esta operación queda suspendido en una cola de procesos asociada al cumplimiento de la condición X. En consecuencia otro proceso puede entrar en el monitor. Nótese que a diferencia de la operación `wait_sem` de los semáforos la operación `wait_mon` de los monitores siempre produce el bloqueo del proceso que la invoca.
- `signal_mon(X)`. Comprueba si la cola de procesos asociada a la condición X contiene algún proceso bloqueado. En caso afirmativo se desbloquea a un proceso. Si la cola estaba vacía esta operación no tiene ningún efecto. Nótese que a diferencia de la operación `signal_sem` de los semáforos, la operación `signal_mon` de los monitores no incrementa ningún contador, en consecuencia si no existe ningún proceso en la cola de condición la señal de aviso se pierde.

Solución Ejercicio 4

El *tiempo de estancia* o *tiempo de retorno* T_{ri} de un trabajo *i* es el tiempo que transcurre desde que se lanza hasta que finaliza, se calcula como la suma del tiempo de ejecución o servicio T_{si} y del tiempo de espera T_{ei} de dicho trabajo.

Por otra parte, el tiempo de estancia promedio de N trabajos es:

$$\bar{T}_r = \frac{T_{r1} + T_{r2} + \ldots + T_{rN}}{N}$$

Sustituyendo la definición de tiempo de retorno se obtiene:

$$\bar{T}_r = \frac{(T_{s1} + T_{e1}) + (T_{s2} + T_{e2}) + \ldots + (T_{sN} + T_{eN})}{N}$$

En el enunciado se proporcionan los siguientes datos:

- N = 3 trabajos
- $\bar{T}_r$ = 20 ut
- T_{s1} = x + 5 ut y T_{e1} = 5 ut
- T_{s2} = x + 13 ut y T_{e2} = 5 ut
- T_{s3} = x + 4 ut y T_{e3} = T_{s3} ut

Sustituyendo estos datos en la expresión del tiempo de retorno medio se obtiene:

$$\frac{(x + 5 + 5) + (x + 13 + 5) + (x + 4 + x + 4)}{3} = 20$$

Finalmente, despejando x de la expresión anterior y operando se obtiene: **x=6 ut**

Solución Ejercicio 5

Se sabe por el enunciado que cada $\Delta T = 10$ ms $= 10000\,\mu$s llega una IR. Si denotamos como T_P al tiempo disponible entre dos interrupciones de reloj para ejecutar un proceso (supuesto que no llegan otro tipo de interrupciones) se cumplirá por tanto la siguiente relación (ver Figura 1):

$$\Delta T = T_{IR} + T_P$$

Despejando T_P, sustituyendo valores y operando se obtiene:

$$T_P = \Delta T - T_{IR} = 10000 - 6{,}5 = 9993{,}5\,\mu s$$

En $T_1 = 100$ s llega una IR y al proceso A le restan por ejecutarse 0,8 s. El diagrama asociado a la ejecución del proceso A desde el instante T_1 hasta que finaliza en T_F tiene la forma general que se muestra en la Figura 2.

De la observación de este diagrama se deduce que el instante de finalización T_F del proceso A se puede calcular mediante la siguiente expresión:

$$T_F = T_1 + N\,\Delta T + T_{IR} + R = T_1 + N\,T_P + (N+1)\,T_{IR} + R$$

N es el número de periodos completos T_P que requiere el proceso A para poder completar los $T_S = 0{,}8$ s $= 800000\,\mu$s de ejecución que le restan y se calcula de la siguiente forma:

$$N = \frac{T_S}{T_P} = \frac{800000}{9993{,}5} = 80{,}052$$

Se observa que no es un número entero eso significa que necesita para completarse de 80 periodos completos T_P y una parte R de otro periodo. R es el resto de la división entera entre T_S y T_P. Se cumple por tanto la siguiente relación:

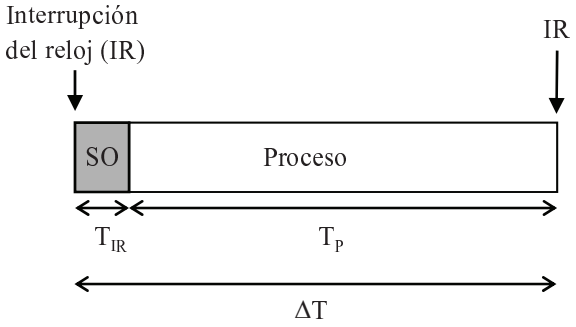


Figura 1 – Uso del procesador entre dos interrupciones de reloj

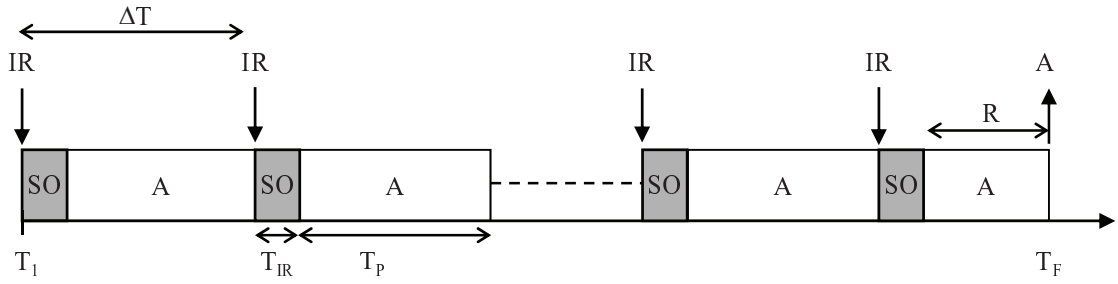


Figura 2 – Uso del procesador en el intervalo de tiempo $[T_1, T_F]$

$$T_s = N\,T_P + R$$

Luego

$$R = T_s - N\,T_P = 800000 - 80 \cdot 9993{,}5 = 520\,\mu s$$

Sustituyendo valores en la expresión del tiempo de finalización y operando se obtiene finalmente el siguiente resultado:

$$T_F = T_1 + N\,\Delta T + T_{IR} + R = 100\,10^6 + 80\,10000 + 6{,}5 + 520 = 100800526{,}5\ \mu s$$

SISTEMAS OPERATIVOS (Código: 71902048) **Febrero 2015**

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N2**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

1. Conteste **razonadamente** a las siguientes preguntas:

a) *(1 p)* Explicar la diferencia entre un *enlace duro* y un *enlace simbólico*.

b) *(1 p)* ¿Bajo que condiciones el algoritmo de planificación de peticiones de E/S a disco se implementa en el driver del disco?

2. Enumerar las ventajas y los inconvenientes de:

a) *(1 p)* Los hilos a nivel de usuario.

b) *(1 p)* Los hilos a nivel de núcleo.

3. *(2 p)* Enumerar los pasos del *algoritmo del banquero*. ¿Qué inconvenientes presenta su uso?

4. *(2 p)* El sistema operativo en colaboración con el hardware gestiona la memoria principal usando la técnica de demanda de página con un tamaño de página de 4 KiB. La memoria principal del computador tiene una capacidad de 256 MiB con un tamaño de palabra de 16 bits. La unidad direccionable es la palabra. Por otra parte el espacio de direcciones virtuales de un proceso A ocupa 128 MiB. Determinar el tamaño en bits de cada uno de los campos en que se descompone una dirección física y una dirección virtual del proceso A.
Nota: 1 KiB=1024 bytes.

5. *(2 p)* Una persona tiene en su casa una jaula llena de canarios en la que hay un plato de alpiste y un columpio. Todos los canarios quieren primero comer del plato y luego columpiarse, sin embargo sólo tres de ellos pueden comer del plato al mismo tiempo y solo uno de ellos puede columpiarse. En la Figura 1 se muestra un pseudocódigo basado en C de un programa que usando un **monitor** de nombre `jaula` coordina la actividad de los canarios. Este monitor considera la solución de Hansen en el comportamiento de la operación `signal_mon`. Explicar **razonadamente** si este pseudocódigo permite la máxima concurrencia de procesos, es decir, que al mismo tiempo tres canarios puedan estar comiendo y otro cuarto pueda estar columpiándose.

```
#define N 3 /* Número de  puestos en el plato */
monitor jaula /* Definición del monitor */
    condición puesto_plato_disponible, columpio_disponible;
    int contadorP, contadorC;

    void obtener_puesto_en_plato()  /* Procedimiento del monitor */
    {
      if (contadorP == N) wait_mon(puesto_plato_disponible);
      contadorP=contadorP+1;
      comer();
    }

    void dejar_puesto_plato() /* Procedimiento del monitor */
    {
      contadorP = contadorP - 1;
      signal_mon(puesto_plato_disponible);
    }

    void obtener_columpio() /* Procedimiento del monitor */
    {
      if (contadorC == 1) wait_mon(columpio_disponible);
      contadorC=contadorC+1;
      columpiarse();
    }

    void dejar_columpio() /* Procedimiento del monitor */
    {
      contadorC = contadorC - 1;
      signal_mon(columpio_disponible);
    }

    { /* Inicialización del monitor */
      contadorP=0, contadorC=0;
    }
end monitor

void canario() /* Proceso canario */
{
    jaula.obtener_puesto_plato();
    jaula.dejar_puesto_plato();

    jaula.obtener_columpio();
    jaula.dejar_columpio();
}

main() /* Ejecución concurrente */
{
    ejecución_concurrente(canario,...,canario);
}
```

Figura 1

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Febrero 2015

Solución Ejercicio 1

a) Un *enlace duro* es una copia total o parcial de la entrada del directorio asociada al archivo o subdirectorio que se desea compartir. Por su parte, un *enlace simbólico* es un tipo especial de archivo que contiene el nombre de ruta del archivo o subdirectorio compartido, es decir, la ubicación del archivo dentro de la estructura de directorios.

b) El algoritmo de planificación de peticiones de E/S a disco se tiene que implementar por software en el driver del disco si el controlador del disco solo puede aceptar una única petición de E/S simultáneamente.

Solución Ejercicio 2

- Ventajas de los *hilos a nivel de usuario*:
  - *Portabilidad*. Puesto que el uso de hilos de usuario no requiere la intervención del sistema operativo, una aplicación diseñada como un proceso de múltiples hilos de usuario podría ejecutarse en cualquier sistemas operativo que soportase la biblioteca de hilos que utilice dicha aplicación.
  - *Mejora del rendimiento del sistema*. Como toda la gestión de los hilos de usuario se realiza en modo usuario no es necesario realizar cambios de modo usuario a modo núcleo y viceversa, lo cual disminuye la sobrecarga del sistema.
  - *Planificación independiente*. Los hilos de usuario de un proceso puede planificarse con el algoritmo de planificación que se considere más oportuno. Dicho algoritmo puede ser distinto al algoritmo de planificación de los hilos de otro proceso y del algoritmo de planificación de procesos que utilice el sistema operativo.
- Desventajas de los *hilos a nivel de usuario*:
  - En sistemas operativos que únicamente soportan un único hilo del núcleo, cuando un hilo de usuario de un proceso entra en el estado bloqueado, entonces todo el proceso completo entra en el estado bloqueado.
  - Cuando un hilo se está ejecutando, no se puede planificar otro hilo de usuario del mismo proceso hasta que el primero no cede voluntariamente el uso del procesador.
- Ventajas de los *hilos a nivel del núcleo*:
  - Si un hilo del núcleo se bloquea se puede planificar otro hilo del mismo proceso o de otro proceso distinto.
  - En sistemas multiprocesador es posible ejecutar varios hilos del núcleo simultáneamente, cada uno de ellos en un procesador distinto.
- Desventajas de los *hilos a nivel del núcleo*:
  - Su gestión contribuye a la sobrecarga del sistema.

Solución Ejercicio 3

Los pasos de que consta el *algoritmo del banquero* son los siguientes:

1. Se parte de un estado inicial de asignación seguro S_k con $k = 0$.

$$S_k = \{\mathbf{N}, \mathbf{A}, \mathbf{R_E}, \mathbf{R_D}\} \tag{1}$$

   En la expresión anterior **N** es la matriz de recursos necesitados, **A** es la matriz de recursos asignados, $\mathbf{R_E}$ es el vector de recursos existentes y $\mathbf{R_D}$ es el vector de recursos disponibles.

2. Cuando un proceso realiza una petición de asignación de recursos, se simula que se concede la petición y se actualiza el estado del sistema. A este estado ficticio se le denota como S′:

$$S' = \{\mathbf{N}', \mathbf{A}', \mathbf{R_E}, \mathbf{R'_D}\} \tag{2}$$

3. Se comprueba si S′ es seguro.

   - En caso afirmativo se concede la petición al proceso y se actualiza el estado del sistema:

$$S_{k+1} = S' \tag{3}$$

   - En caso negativo la petición se deniega y el proceso se bloquea hasta que se le puedan conceder los recursos que solicita.

4. Volver al paso 2.

El principal inconveniente del algoritmo del banquero es que en la práctica realmente no se utiliza, ya que es difícil conocer por adelantado los recursos que van a necesitar los procesos durante su ejecución. Además dicho algoritmo parte de la suposición de que el número de procesos y de recursos existentes es fijo, lo cual no tiene por qué ser cierto. En general, la población de procesos varía dinámicamente en el tiempo. Por otra parte, los recursos que inicialmente están disponibles pueden que con el tiempo no lo estén debido a posibles fallos, por ejemplo, que se quede atascado papel en una impresora o que se dañe una unidad de cinta.

Solución Ejercicio 4

Cuando se utiliza la técnica de paginación una dirección física se descompone en los campos número de marco de página de f bits y desplazamiento dentro del marco de d bits. Por su parte una dirección lógica se descompone en los campos número de página de p bits y desplazamiento dentro de la página de d bits.

- *Dirección Física*:

  Para obtener el tamaño del campo número de marco de página se va a calcular en primer lugar el número de marcos N_{MP} en que se descompone la memoria principal si se considera un tamaño de página $S_P = 4$ KiB:

$$N_{MP} = \text{floor}\left(\frac{C_{MP}}{S_P}\right) = \text{floor}\left(\frac{256\,\text{MiB}}{4\,\text{KiB}}\right) = \text{floor}\left(\frac{2^{28}}{2^{12}}\right) = 2^{16}\ \text{marcos}$$

  Conocido N_{MP} el tamaño de este campo se obtiene resolviendo la siguiente desigualdad:

$$\min_f \left\{N_{MP} \leq 2^f\right\}$$

Luego como $N_{MP} = 2^{16}$ entonces $f = 16$ bits.

Para obtener el tamaño del campo desplazamiento primero hay que expresar el tamaño de una página en palabras:

$$S_P = \frac{4\,\text{KiB}}{2\,\text{bytes/palabra}} = \frac{2^{12}\,\text{bytes}}{2\,\text{bytes/palabra}} = 2^{11}\,\text{palabras}$$

Conocido S_P el tamaño de este campo se obtiene resolviendo la siguiente desigualdad:

$$\min_{d} \left\{ S_P \leq 2^d \right\}$$

Luego como $S_P = 2^{11}$ entonces $d = 11$ bits.

- *Dirección Lógica*:

Para obtener el tamaño del campo número de página se va a calcular en primer lugar el número de páginas N_P en que se descompone el espacio de direcciones lógicas del proceso A si se considera un tamaño de página $S_P = 4$ KiB:

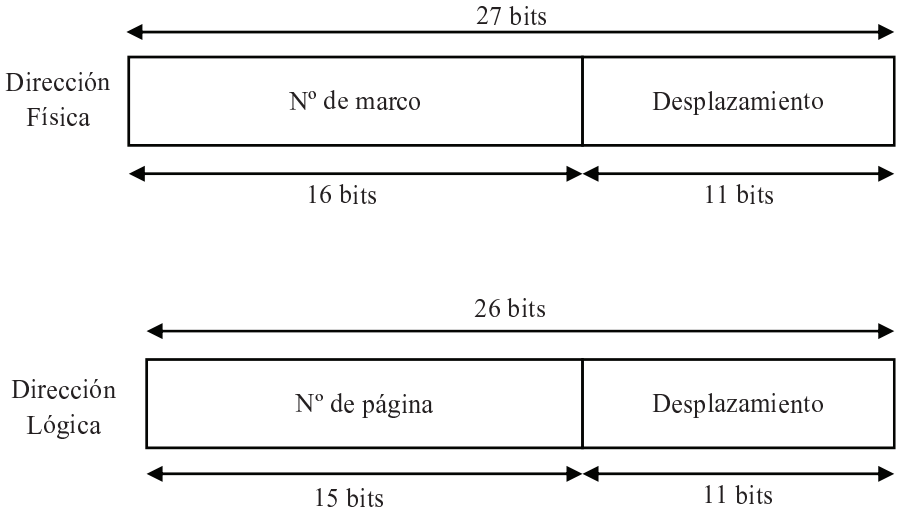
$$N_P = \text{ceil}\left(\frac{C_A}{S_P}\right) = \text{ceil}\left(\frac{128\,\text{MiB}}{4\,\text{KiB}}\right) = \text{ceil}\left(\frac{2^{17}\,\text{KiB}}{4\,\text{KiB}}\right) = \text{ceil}(2^{15}) = 2^{15}\,\text{páginas}$$

Conocido N_P el tamaño de este campo se obtiene resolviendo la siguiente desigualdad:

$$\min_{p} \{ N_P \leq 2^p \}$$

Luego $p = 15$.

El tamaño del campo desplazamiento de una dirección lógica es igual al de una dirección física, en este caso, $d = 11$ bits. En la figura se representa el formato de una dirección física y de una dirección lógica.



Solución Ejercicio 5

Se observa en el peudocódigo que las acciones de `comer()` y `columpiarse()` se invocan dentro de procedimientos del monitor. En un monitor solo se puede ejecutar un procedimiento del mismo a la vez, es decir, hasta que no termina de ejecutarse un procedimiento no puede comenzar a ejecutarse otro. Por lo tanto, se tendría la situación de que no podrían comer tres canarios y columpiarse otro simultáneamente, ya que mientras un canario come no podría comer ni columpiarse ningún otro, o mientras un canario se columpia no podrían comer los demás. En consecuencia este pseudocódigo **no permite** la máxima concurrencia de procesos.

SISTEMAS OPERATIVOS (Código: 71902048) **Septiembre 2015**

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

1. Conteste **razonadamente** a las siguientes preguntas:

a) *(1 p)* ¿Qué es la técnica de spooling y cómo se implementa?

b) *(1 p)* Explicar cuándo se produce y en qué consiste el problema de la condición de carrera.

2. *(2 p)* Enumerar y describir **brevemente** los cuatro principales factores que hay que tener en cuenta a la hora de seleccionar el tamaño de página en la técnica de gestión de la memoria principal mediante paginación por demanda.

3. *(2 p)* Enumerar y describir **brevemente** los componentes (o subsistemas) del núcleo de un sistema operativo.

4. *(2 p)* Un cierto sistema de archivos utiliza un tamaño de bloque de 16 bytes y su área de datos consta de 256 bloques. La asignación de espacio se realiza mediante el método de asignación indexada. Además en el nodo-i asociado a un archivo, entre otros datos, se almacenan las direcciones físicas de los ocho primeros bloques de datos del archivo y la dirección física de un bloque de indirección simple. Calcular el tamaño máximo en bytes que puede tener un archivo en este sistema de archivos.

5. *(2 p)* Considérense los procesos A, B, C y D cuyo tiempo de llegada, prioridad y tiempo de servicio se muestran en la Tabla 1. Supuesto que 1 es la prioridad más alta, que el tiempo de colocación en la cola de procesos preparados es despreciable y que el tiempo de cambio de contexto es de 1 ut, representar el diagrama de uso del procesador en el caso de que se utilicen los siguientes algoritmos de planificación:

a) Algoritmo de turno rotatorio con un cuanto $q = 2$ ut. Suponer que si varios procesos tienen el mismo tiempo de llegada se colocan en la cola de procesos preparados por orden de prioridad.

b) Algoritmo basado en prioridades de tipo expropiativo.

Proceso	Tiempo de llegada (ut)	Prioridad	Tiempo de servicio (ut)
A	0	1	4
B	0	2	2
C	1	3	3
D	1	4	5

Tabla 1

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Septiembre 2015

Solución Ejercicio 1

a) El *spooling* es una técnica que el sistema operativo utiliza para asignar y controlar el uso de algunos dispositivos de E/S dedicados, como es el caso de una impresora. Esta técnica se implementa con un proceso demonio y con un directorio especial, denominado *directorio de spooling* o *spool*. El proceso demonio es el único autorizado para escribir en el dispositivo. Si un proceso de usuario quiere escribir en el dispositivo, debe enviar los archivos que desea escribir al directorio de spooling. El proceso demonio irá mandando al dispositivo de salida uno a uno los archivos que encuentre en este directorio, ya sean de un proceso u otro. Usando spooling se previene que un proceso obtenga el uso de un dispositivo de salida y no lo ceda durante un tiempo excesivo, impidiendo que lo puedan usar otros procesos.

b) El problema denominado *condición de carrera* (race condition) surge cuando múltiples procesos independientes se ejecutan concurrentemente y acceden para leer o escribir en un recurso compartido. Este problema consiste en que el resultado final de la ejecución depende del orden en que se hayan planificado los procesos, es decir, en que se hayan ejecutado las instrucciones de lectura y escritura de cada proceso sobre el recurso.

Solución Ejercicio 2

Los factores que hay que tener en cuenta a la hora de seleccionar el tamaño de página son:

- *Fragmentación interna*. En promedio la fragmentación interna es de media página por proceso. En consecuencia, cuanto mayor sea el tamaño de la página mayor será la fragmentación interna y mayor será el desperdicio de espacio de la memoria principal.

- *Tamaño de una tabla de páginas*. Cada proceso tiene una tabla de páginas cargada en memoria principal que posee un número de entradas igual al número de páginas en que se descompone el espacio de direcciones virtuales del proceso. Cuanto menor sea el tamaño de una página, en más páginas se descompondrá el espacio virtual del proceso, y las tablas de páginas serán de mayor tamaño. Por lo tanto, consumirán más espacio de memoria. Además, si el tamaño de la tabla de páginas aumenta, el tiempo de cambio de proceso será mayor ya que el sistema operativo tendrá que cargar o vaciar más registros si se tiene una MMU con banco de registros o con TLB.

- *Número de fallos de página*. Si el tamaño de página es pequeño el espacio de direcciones virtual constará de más páginas por lo que la probabilidad de que se produzca un fallo de página será mayor.

- *Tiempo de uso de E/S*. El tiempo de una lectura o escritura en un disco se descompone en tiempo de búsqueda, tiempo de latencia rotacional y tiempo de transferencia. De estos tres tiempos el tiempo de transferencia es mucho más pequeño que los tiempos de búsqueda y de latencia. Por ello se tarda menos tiempo en transferir una página de tamaño grande que en transferir varias páginas de tamaño más pequeño. Por otra parte las páginas se escriben o se leen en memoria secundaria una a una. Si el tamaño de página es pequeño, el espacio de direcciones virtuales del proceso se descompondrá en un mayor número de páginas, por lo que la probabilidad de realizar operaciones de E/S será mayor. En consecuencia el tiempo empleado en E/S por el sistema operativo será mayor, tiempo de E/S que no puede ser utilizado por la ejecución de los procesos.

Solución Ejercicio 3

Los componentes o subsistemas del núcleo de un sistema operativo son:

- *Subsistema de control de procesos*. Se encarga, entre otras tareas, de crear y eliminar procesos, suspender y continuar la ejecución de los procesos, proporcionar mecanismos para la sincronización y comunicación de los procesos, y manejar los interbloqueos.
- *Subsistema de administración de la memoria principal*. Se encarga de llevar un registro de las partes de la memoria que se encuentran asignadas y a qué procesos. También se encarga de llevar un registro del espacio libre para poder asignarlo cuando se necesite, liberar memoria para asignarla a otros procesos y decidir qué procesos se van a cargar en memoria desde la memoria secundaria.
- *Subsistema de gestión de archivos*. Es el responsable de la gestión de los archivos de los diferentes sistemas de archivos existentes en la memoria secundaria. Además se encarga de fijar los tipos, los atributos, la estructura interna y los mecanismos de acceso de los archivos que soporta. También define la estructura de los directorios en los que lo usuarios y las aplicaciones organizan sus archivos, y los mecanismos de búsqueda de archivos permitidos dentro de dicha estructura de directorios. Asimismo proporciona al usuario las llamadas al sistema oportunas que le permitan operar (leer, escribir, crear, borrar, ...) con archivos y directorios. Por otra parte, este subsistema se encarga de asignar espacio a los archivos y de administrar el espacio libre de la memoria secundaria, cuyo elemento más utilizado es el disco duro interno del computador.
- *Subsistema de E/S*. Este componente oculta las particularidades del hardware de los dispositivos de E/S proporcionando una interfaz uniforme de forma que los programas de los usuarios puedan acceder a los dispositivos de E/S con un conjunto sencillo de llamadas al sistema de lectura y escritura. Para comunicarse con el hardware, el subsistema de E/S requiere un código específico para cada dispositivo denominado driver del dispositivo. Cuando un dispositivo termina una operación de E/S, el controlador de E/S avisa al driver correspondiente mediante la generación de una interrupción.

Solución Ejercicio 4

Supuesto que el área de datos tiene una capacidad suficiente, el tamaño máximo que puede tener un archivo en este sistema de archivos está limitado por el número de bloques de datos del archivos que pueden ser direccionados directa o indirectamente a través de su nodo-i asociado.

De acuerdo con el enunciado en un nodo-i asociado a un archivo se almacenan las direcciones de sus 8 primeros bloques de datos y la dirección de un bloque de indirección simple, en el cual recuérdese se almacenan direcciones de bloques de datos.

El número de direcciones N_D de bloques de datos que caben en un bloque de indirección simple se calcula de la siguiente forma:

$$N_D = \text{floor}\left(\frac{S_B}{n}\right)$$

Donde S_B es el tamaño de un bloque de datos y n es el número n de bits que ocupa la dirección física de un bloque de disco. Ésta última se obtiene resolviendo la siguiente desigualdad:

$$\min_n \{N_B \leq 2^n\}$$

Donde N_B es el número de bloques de datos existentes. Como $N_B = 256 = 2^8$ bloques entonces $n = 8$ bits.

Sustituyendo valores y operando se obtiene:

$$N_D = \text{floor}\left(\frac{16\,\text{bytes}}{8\,\text{bits}}\right) = \text{floor}\left(\frac{16\,\text{bytes}}{1\,\text{byte}}\right) = 16\,\text{direcciones}$$

Luego en un bloque de indirección simple pueden almacenar un máximo de 16 direcciones de bloques.

En consecuencia el número total de direcciones de bloques accesibles desde un nodo-i son 8 + 16 = 24 direcciones. Por lo tanto el tamaño máximo de un archivo queda limitado a **24 bloques**, es decir, a 24 (bloques)·16 (bytes/bloque) = **384 bytes**.

Solución Ejercicio 5

a) En la Figura 1 se muestra el diagrama de uso del procesador usando el algoritmo de planificación de turno rotatorio con $q = 2$ ut.

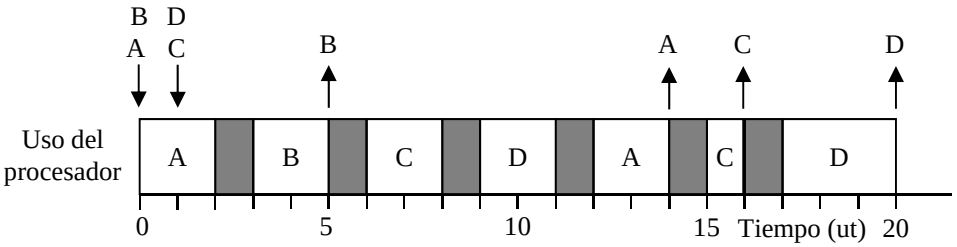


Figura 1

b) En la Figura 2 se muestra el diagrama de uso del procesador usando el algoritmo de planificación basado en prioridades de tipo expropiativo.

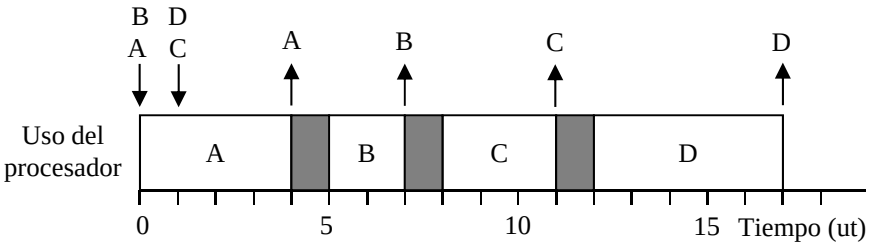


Figura 2

SISTEMAS OPERATIVOS (Código: 71902048) **Enero 2016**

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N1**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

1. Conteste **razonadamente** a las siguientes preguntas:

a) *(1 p)* ¿Qué es la *traza de ejecución* de un proceso?

b) *(1 p)* ¿Cuáles son los principales *servicios de un sistema operativo*?

2. *(2 p)* Describir **adecuadamente** cómo se realiza la traducción de direcciones lógicas a físicas y la protección en la técnica de gestión de memoria mediante *particionamiento dinámico*.

3. *(2 p)* Enumerar las acciones que de forma general realiza un driver de un dispositivo de E/S.

4. El planificador de un sistema operativo planifica los procesos usando un algoritmo basado en prioridades (la máxima prioridad es 1) de tipo no expropiativo. Se tienen que ejecutar dos procesos multihilos A y B. El proceso A de prioridad 1 consta de tres hilos: H_{A1}, H_{A2} y H_{A3}. Mientras que el proceso B de prioridad 3 consta de dos hilos: H_{B1} y H_{B2}. La prioridad y requerimientos de estos hilos son los siguientes:

- H_{A1} (prioridad 1): ráfaga de CPU de 20 ut, E/S de 60 ut y ráfaga de CPU de 20 ut.
- H_{A2} (prioridad 2): ráfaga de CPU de 60 ut.
- H_{A3} (prioridad 3): ráfaga de CPU de 20 ut, E/S de 60 ut y ráfaga de CPU de 40 ut.
- H_{B1} (prioridad 4): ráfaga de CPU de 40 ut, E/S de 60 ut y ráfaga de CPU de 20 ut.
- H_{B2} (prioridad 5): ráfaga de CPU de 40 ut, E/S de 60 ut y ráfaga de CPU de 20 ut.

Supuesto que la sobrecarga es despreciable, que el sistema operativo soporta exclusivamente hilos a nivel de usuario y que la planificación local a nivel de hilo es idéntica en los dos procesos (A y B) y está basada en prioridades de tipo no expropiativo (la máxima prioridad corresponde al valor 1); se pide:

a) *(1.5 p)* Dibujar el diagrama de uso de recursos.

a) *(0.5 p)* Determinar el número de cambios de procesos, el número de cambios de hilos y el tiempo de finalización de los procesos

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N1**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

5. *(2 p)* Un aeropuerto tiene una sola pista para el aterrizaje y despegue de aviones. Para evitar la colisión de los aparatos, la pista solo puede utilizarla un avión simultáneamente, ya sea en una operación de despegue o en una operación de aterrizaje. Hasta que no termina una operación no puede comenzar otra. Además las operaciones de aterrizaje tienen mayor prioridad que las de despegue, es decir, que una operación de despegue no puede realizarse si en el momento de solicitarse existen peticiones de operaciones de aterrizaje. En ese caso, el avión que quiere despegar quedará a la espera hasta que no quede ninguna petición de aterrizaje pendiente. Escribir el pseudocódigo de un programa que usando **paso de mensajes** coordine la actividad de los aviones en el aeropuerto. Suponer que la comunicación es indirecta a través de buzones y que se dispone de la operación `send` sin bloqueo y de la operación `receive` con bloqueo. El pseudocódigo del programa debe tener cuatro partes: declaración de variables, código del proceso `avión_aterriza`, código del proceso `avión_despega` y código para inicializar los buzones y lanzar la ejecución concurrente de los procesos.

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Enero 2016

Solución Ejercicio 1

a) *Traza de ejecución*. También denominada *hilo de control* o simplemente *hilo* o *hebra* (thread). Hace referencia a las instrucciones que ejecuta el proceso durante su tiempo de vida.

b) Los principales servicios proporcionados por un sistema operativo se pueden agrupar de forma general en las siguientes clases:

- *Ejecución de programas*.
- *Acceso a los dispositivos de E/S.*
- *Manipulación del sistema de archivos.*
- *Comunicación y sincronización entre los programas*.
- *Detección y respuesta a errores hardware y software*.
- *Protección y seguridad*. El sistema operativo debe asegurar un acceso controlado y protegido a los recursos del computador. Además debe disponer de medidas de seguridad para regular el acceso únicamente a los usuarios autorizados.
- *Contabilidad.* Un sistema operativo debe llevar registros o estadísticas del uso de los diferentes recursos del computador y monitorizar parámetros tales como la productividad o el tiempo de respuesta.

Solución Ejercicio 2

En la técnica de gestión de memoria mediante particionamiento dinámico la traducción de direcciones y la protección se implementa de la misma manera que en la técnica del particionamiento fijo, es decir, se utilizan dos registros dedicados denominados *registro base* y *registro límite*. El sistema operativo, cuando se va a ejecutar un proceso, carga en el registro base la dirección física de inicio de la partición de memoria física asociada al proceso, y carga en el registro límite el tamaño de la partición.

Cuando el procesador al ejecutar el proceso hace una referencia a una dirección lógica para buscar por ejemplo una instrucción o un operando, ésta es comparada con el valor de registro límite. Si la dirección es mayor que dicho valor, entonces se generará una excepción que al ser tratada por el sistema operativo típicamente suele producir la terminación del proceso y la notificación de dicha circunstancia al usuario propietario del proceso. Si la dirección lógica es menor que el valor del registro límite entonces la dirección lógica se suma al contenido del registro base para generar la dirección física correspondiente.

En definitiva, el registro base y el registro límite impiden el acceso de un proceso a direcciones físicas inferiores o superiores, respectivamente, a las asignadas a su espacio lógico. De esta forma se protege el espacio de direcciones lógicas del sistema operativo o el espacio de otro proceso contra el acceso involuntario o intencionado del proceso en ejecución.

Solución Ejercicio 3

De forma general un driver suele realizar las siguientes acciones:

1. Comprobar que los parámetros de la función invocada por el subsistema de E/S son correctos y que la operación de E/S solicitada se puede realizar. En caso contrario devuelve un error.

2. Traducir los parámetros de dicha función en parámetros específicos del dispositivo.

3. Comprobar si el dispositivo de E/S está ocupado atendiendo alguna petición anterior de E/S. Si el dispositivo está ocupado entonces coloca la petición en una cola. Si el dispositivo no está atendiendo ninguna petición, comprueba si se encuentra preparado, ya que quizás el driver deba activar e inicializar el dispositivo.

4. Generar un conjunto de órdenes para el controlador del dispositivo dependiendo de la petición de E/S solicitada por el subsistema de E/S. Dichas órdenes son cargadas en los registros del controlador. El driver puede comprobar que el controlador acepta una orden y que se encuentra listo para aceptar la siguiente.

5. Una vez trasmitidas todas las órdenes al controlador, el driver debe esperar a que el controlador las ejecute. Si el tiempo de espera estimado es importante, entonces el driver se bloquea usando algún mecanismo de sincronización (semáforos, paso de mensajes, etc) hasta que el controlador finalice. Cuando la operación de E/S se complete el controlador activará una interrupción. El manejador o rutina de servicio de dicha interrupción despertará al driver, si éste se bloqueó.

6. Comprobar que no se han producido errores en la operación de E/S. En dicho caso, quizás transfiera al subsistema de E/S el resultado de la operación de E/S, por ejemplo, un bloque leído en el disco, o puede que simplemente le informe de que la operación se ha completado. Si se ha producido algún error y el driver sabe cómo resolverlo realiza la acción correspondiente. Si no sabe cómo resolverlo o la solución que plantea no surte efecto, entonces informa del error al subsistema de E/S para que tome las medidas que considere oportunas.

7. Examinar la cola de peticiones de E/S pendientes, si existe alguna procede a atenderla. Si la cola está vacía el driver se bloqueará en espera de la llegada de nuevas peticiones.

Solución Ejercicio 4

a) En la Figura 1 se muestra el diagrama de uso de recursos.

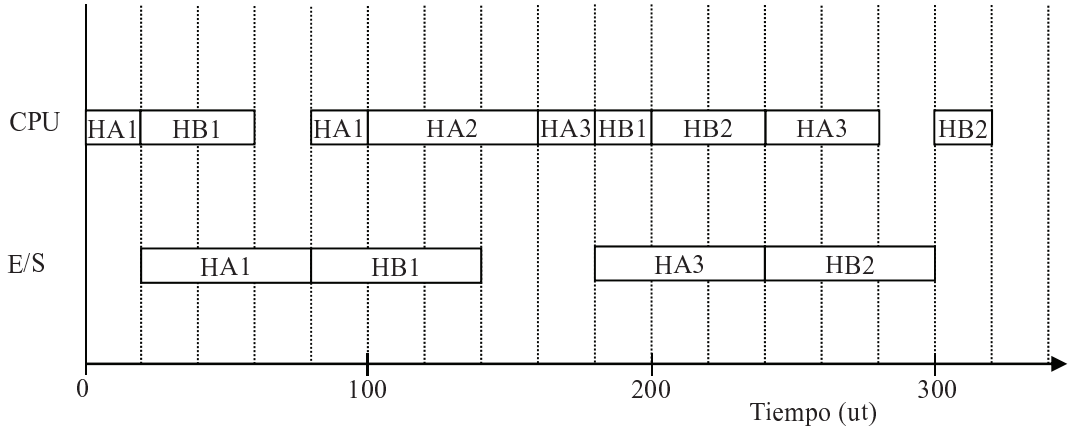


Figura 1

b) En el diagrama de uso de recursos de la Figura 1 se observa que:

- Se producen 5 cambios de proceso: en 20 ut de A a B, en el rango [60, 80] ut de B a A, en 180 ut de A a B, en 240 ut de B a A, y en el rango [280, 300] ut de A a B.
- Se producen 3 cambios de hilo: en 100 ut de HA1 a HA2, en 160 ut de HA2 a HA3 y en 200 ut de HB1 a HB2.
- El proceso A finaliza en 280 ut.
- El proceso B finaliza en 320 ut.

Solución Ejercicio 5

En las Figuras 2 y 3 se muestra una posible solución para este ejercicio. Esta solución utiliza las siguientes variables y buzones:

- `contadorPA`. Variable global de tipo entero para llevar la cuenta del número de peticiones de aterrizaje.
- `contadorEDA`. Variable global de tipo entero para llevar la cuenta del número de aviones esperando para despegar por existir aterrizajes.
- `S1`. Buzón para garantizar la exclusión mutua en el uso de la variable `contadorPA`.
- `S2`. Buzón para garantizar la exclusión mutua en el uso de la variable `contadorEDA`.
- `S3`. Buzón para garantizar la exclusión mutua en el uso de la pista de aterrizaje.
- `S4`. Buzón para avisar que no existen peticiones de aterrizaje a los aviones que desean despegar.

```
int contadorEDA=0, contadorPA=0; /* Definición variables */

void avion_aterriza()  /* Código proceso avión_aterriza */
{
  mensaje m;

  receive(S1,m);
    contadorPA=contadorPA+1;
  send(S1,m);

  receive(S3,m); /* Esperar si otro avión está usando la pista */
    aterrizar();
  send(S3,m);

  receive(S1,m);
  contadorPA=contadorPA-1;
  if (contadorPA==0)   /* Avisar a los aviones que esperan para despegar */
  {
    receive(S2,m);
    while(contadorEDA>0){
      send(S4,m);
      contadorEDA=contadorEDA-1;
    }
    send(S2,m);
    send(S1,m);
  }
  else send(S1,m);

}
```

Figura 2 – Solución ejercicio 5

```
void avion_despega()  /* Código proceso avión_despega */
{
  mensaje m;

  receive(S1,m);
  if (contadorPA>0) /* Esperar si hay peticiones de aterrizaje */
  {
    receive(S2,m);
    contadorEDA=contadorEDA+1;
    send(S2,m);
    send(S1,m);

    receive(S4,m);
   }
  else send(S1,m);

  receive(S3,m); /* Esperar si otro avión está usando la pista */
    despegar();
  send(S3,m);

}

main() /* Inicialización de semáforos y ejecución concurrente */
{
  int h;
  mensaje nulo;

  crear_buzón(S1);
  send(S1,nulo) /*Inicializar S1 con 1 mensaje*/

  crear_buzón(S2);
  send(S2,nulo) /*Inicializar S2 con 1 mensaje*/

  crear_buzón(S3);
  send(S3,nulo) /*Inicializar S3 con 1 mensaje*/

  crear_buzón(S4); /*Este buzón se deja inicialmente vacio*/


  ejecución_concurrente(avion_aterriza,avion_aterriza,...,avion_despega,avion_despega,...);
}
```

Figura 3 – Continuación solución ejercicio 5

SISTEMAS OPERATIVOS (Código: 71902048) **Febrero 2016**

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **N2**	**Aviso 1**: Todas las respuestas deben estar debidamente razonadas. **Aviso 2**: Escriba con buena letra y evite los tachones. **Aviso 3**: Solución del examen y fecha de revisión en http://www.uned.es/71902048/

1. Conteste **razonadamente** a las siguientes preguntas:

a) *(1 p)* ¿Qué es una *base de computador confiable* (Trusted Computer Base, TCB)? ¿Cuáles son las funciones del sistema operativo que debe incluir?

b) *(1 p)* ¿Cómo se pueden detectar los interbloqueos en un grafo de asignación de recursos?

2. *(2 p)* Enumerar y describir brevemente los tipos de sistemas operativos que se pueden distinguir en función de los requisitos temporales de los programas que se van a ejecutar.

3. *(2 p)* Describir el método de asignación de espacio de disco conocido como asignación indexada, ¿Cuáles son sus ventajas e inconvenientes?

4. Considérese un sistema con memoria virtual en el que se utiliza la técnica de paginación por demanda. En este sistema se han asignado a un cierto proceso X tres marcos de página para su ejecución. La cadena de referencias de página que produce la ejecución del proceso X es:

```
4 8 9 7 8 6 7 8 6 5 8 6 5 4 5 6 5 6 4 6 4
```

Determinar el número de fallos de página que se producen para los siguientes casos:

a) *(1 p)* Se utiliza el algoritmo de reemplazamiento de páginas FIFO.

b) *(1 p)* Se utiliza el algoritmo de reemplazamiento de páginas LRU.

5. *(2 p)* El acceso de los ciudadanos a una comisaria de policía para realizar gestiones relativas a sus DNIs o pasaportes está regulado por un agente de policía. Los ciudadanos esperan a la puerta de la comisaria en cola por orden de llegada y el agente cada 15 minutos avisa a los 10 primeros ciudadanos de la cola para que pasen dentro a realizar sus gestiones. Si no hay ciudadanos en la cola el agente no realiza ningún aviso y si hay N ciudadanos en la cola con N menor de 10 el agente solo realiza N avisos. Suponer que independientemente del número de ciudadanos que hayan pasado la vez anterior, el agente solo realiza su acción de avisar ciudadanos cada 15 minutos. Escribir el pseudocódigo de un programa que usando **semáforos binarios** coordine la actividad del agente y los ciudadanos para acceder a la comisaria. El pseudocódigo del programa que se realice en cada apartado debe tener cuatro partes: declaración de variables, código del ciudadano, código del agente y código para inicializar los semáforos y lanzar la ejecución concurrente de los procesos.

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Febrero 2016

Solución Ejercicio 1

a) Una *base de computador confiable* (Trusted Computer Base, TCB) es un conjunto de elementos hardware y software que permiten a un sistema informático implementar los requisitos de seguridad establecidos. Generalmente la TCB está formada por la mayoría de los elementos hardware del sistema (excepto los dispositivos de E/S) y una parte del núcleo del sistema operativo.

   Entre las funciones del sistema operativo que se deben incluir dentro de la TCB se encuentran las relativas a creación de procesos, cambio de contexto, gestión de memoria y parte de la gestión de la E/S y del sistema de archivos.

b) El grafo de asignación de recursos puede utilizarse para detectar la presencia de interbloqueos, para ello debe suponerse que ya se cumplen las condiciones de exclusión mutua y no existencia de expropiación, las cuales quedan fijadas por el sistema operativo. En el grafo de asignación de recursos pueden detectarse visualmente, o usando algún algoritmo de análisis de grafos, la condición de espera circular y la condición de retención y espera.

   Se puede demostrar que si el grafo no contiene ningún camino que sea un ciclo entonces no existe interbloqueo. Por otra parte la existencia de un ciclo no es condición suficiente para que exista interbloqueo, dependerá también del número de instancias de cada recurso implicado en el ciclo y del tipo de caminos de los que formen parte dichas instancias. Si los recursos que forman parte de un ciclo únicamente tienen una instancia entonces existe interbloqueo. Por otra parte, si en el ciclo existen recursos que tienen más de una instancia, entonces para que exista interbloqueo todos las instancias de dichos recursos deben forman parte de caminos que sean ciclos.

Solución Ejercicio 2

En función de los requisitos temporales (tiempo de ejecución o tiempo de respuesta) de los programas que se van a ejecutar, es posible distinguir los siguientes tipos de sistemas operativos:

- *Sistemas operativos por lotes* o *sistemas batch*. En estos sistemas los trabajos se procesan agrupados en lotes de trabajos con necesidades similares. Cada trabajo consta típicamente del programa a ejecutar, los datos necesarios y órdenes para el sistema operativo. El tipo de trabajos que se suelen ejecutar en estos sistemas son aquéllos que requieren un tiempo de ejecución grande, típicamente minutos u horas, y que además necesitan de poca o nula interacción con lo usuarios, como por ejemplo: análisis estadísticos, gestión de nóminas, etc.

- *Sistemas operativos de tiempo compartido* o *sistemas interactivos*. Son sistemas multiusuario con multiprogramación donde cada usuario introduce desde su terminal una orden, bien mediante el uso del teclado o del ratón, y espera por la respuesta del sistema operativo. En todo momento, gracias a la multiprogramación, un usuario cree ser el único que está interaccionando con el computador y tener a su disposición todos sus recursos. Las aplicaciones que se suelen ejecutar en estos sistemas son programas que requieren un tiempo de respuesta pequeño, típicamente menores de un segundo, ya que en caso contrario el usuario pensará que el sistema es insensible a su acciones. Ejemplo de aplicaciones interactivas son: los intérpretes de comandos, los editores y las aplicaciones con GUI.

- *Sistemas operativos de tiempo real*. Son sistemas con multiprogramación que soportan *aplicaciones de tiempo real*, que son aquellas que reciben unas entradas procedentes de unos sensores externos, a través de unas tarjetas de adquisición de datos, y deben generar unas salidas en un tiempo de respuesta preestablecido. Tales aplicaciones de tiempo real se usan en experimentos científicos, control industrial, robótica, sistemas de control de vuelo, simulaciones, telecomunicaciones, dispositivos multimedia, etc. Se pueden clasificar en dos tipos: aplicaciones de tiempo real estrictas y aplicaciones de tiempo real suaves.

- *Sistemas operativos híbridos*. Son aquellos sistemas operativos con capacidad para soportar tanto trabajos por lotes como aplicaciones interactivas o incluso aplicaciones suaves de tiempo real. Normalmente se asigna a los trabajos por lotes una prioridad de ejecución más pequeña que a las aplicaciones interactivas, y a éstas una prioridad de ejecución menor que a las aplicaciones suaves de tiempo real. Así, los trabajos por lotes se ejecutan cuando el procesador no tiene que ejecutar aplicaciones interactivas, y éstas cuando no hay que ejecutar aplicaciones suaves de tiempo real.

Solución Ejercicio 3

El *método de asignación indexada* consiste en almacenar en un nodo-i los atributos de un archivo y las direcciones físicas de los ocho o diez primeros bloques de un archivo. También se almacenan las direcciones físicas de uno o varios bloques de indirección simple, doble o triple. Un *bloque de indirección simple* es un bloque físico que almacena direcciones físicas de bloques del archivo. Por su parte un *bloque de indirección doble* es un bloque físico que almacena direcciones de bloques de indirección simple. Mientras que un *bloque de indirección triple* es un bloque físico que almacena direcciones de bloques de indirección doble. El tamaño de un nodo-i suele ser pequeño, unos pocos bytes, por lo que dentro de un bloque físico de discos se pueden almacenar múltiples nodos-i.

Cada archivo tiene asociado un nodo-i que queda identificado mediante un número entero positivo denominado *número de nodo-i*. Al principio de la partición asociada al sistema de archivos se mantiene una lista con todos los nodos-i existentes. El número de nodo-i asociado a un archivo se almacena en la entrada del directorio que contiene el archivo. Cuando se abre un archivo, su nodo-i se lee en la lista de nodos-i y se carga en memoria principal para poder conocer las direcciones físicas de los bloques del archivo y sus atributos.

El método de asignación indexada se puede usar tanto para archivos de acceso secuencial como para archivos de acceso aleatorio. No produce fragmentación externa y permite que el espacio de los bloques físicos que contienen datos del archivo pueda ser utilizado en su totalidad. Además su implementación requiere el uso de menos espacio en memoria principal que el método de asignación enlazada con FAT. Nótese que solo es necesario mantener en memoria principal los nodos-i de los archivos abiertos, no los de todos los archivos existentes en la partición.

La principal desventaja del método de asignación indexada es que el tiempo de acceso a un bloque del archivo depende de la estructura interna del nodo-i, del tamaño del archivo y de la posición del bloque al que se desea acceder. Supuesto que el nodo-i del archivo ya ha sido cargado en memoria principal, si se desea acceder a uno de los primeros bloques del archivo, como el nodo-i contiene su dirección física, simplemente hay que hacer una lectura en disco para leer dicho bloque. Pero si se trata de bloques del archivo a los que se accede a través de bloques de indirección simple, doble o triple entonces habrá que hacer una, dos o tres lecturas adicionales en disco, respectivamente. Luego en el mejor de los casos, descontando la lectura del nodo-i, hay que hacer una lectura en disco, mientras que en el peor de los casos habrá que hacer tres lecturas.

Solución Ejercicio 4

a) Se va a suponer que para implementar el algoritmo FIFO se utiliza una cola FIFO. En la Figura 1 se muestra el contenido de la cola antes y después de cada referencia de la secuencia, se indica también si dicha referencia produce un fallo (F) o un acierto (A). Se observa que se producen un total de **9 fallos de página**.

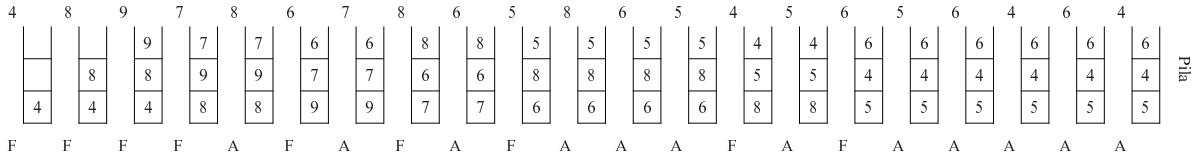


Figura 1

b) Se va a suponer que para implementar el algoritmo LRU se utiliza una lista enlazada. En la Figura 2 se muestra el contenido de la lista antes y después de cada referencia de la secuencia, se indica también si dicha referencia produce un fallo (F) o un acierto (A). Se observa que se producen un total de **7 fallos de página**.

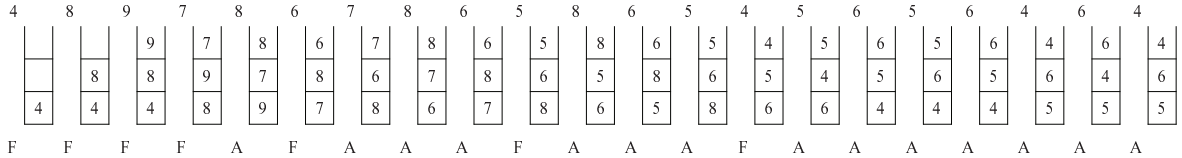


Figura 2

Solución Ejercicio 5

En la Figura 3 se muestra una posible solución haciendo uso de **semáforos binarios**.

```
int contador=0;      /*Para almacenar el número de ciudadanos
                       en la cola de la puerta de la comisaria*/
semáforo_binario S1; /*Para garantizar la exclusión mutua en el uso de la variable contador*/
semáforo_binario S2; /*Para sincronizar la espera de los ciudadanos en la cola*/

void ciudadano() /*Código ciudadano*/
{
    wait_sem(S1);
    contador=contador+1;
    signal_sem(S1);

    wait_sem(S2); /*Esperar en la cola a que le avise el agente*/

    realizar_gestión();
}

void agente() /*Código agente*/
{
    int numero_avisos;

    while(TRUE)
    {
        /*Inicio bloque aviso ciudadanos*/
        wait_sem(S1);

        if (contador < 10) numero_avisos=contador;
        else numero_avisos=10;

        while(numero_avisos > 0)
        {
            signal_sem(S2); /*Avisa a un ciudadano*/
            numero_avisos=numero_avisos-1;
            contador=contador-1;
        }

        signal_sem(S1);
        /*Fin bloque aviso ciudadanos*/

        esperar_15min();
    }
}

main() /*Inicialización semáforos y ejecución concurrente*/
{
    init_sem(S1,1);
    init_sem(S2,0);
    ejecución_concurrente(ciudadanos, agente);
}
```

Figura 3