SISTEMAS OPERATIVOS (Código: 71902048) **Febrero 2025**

Material permitido: **Solo calculadora no programable** Tiempo: **120 minutos** **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 https://www.dia.uned.es/71902048/

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

a) *(1 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.

b) *(1 p)* Enumerar las ventajas y los inconvenientes de los *hilos a nivel de usuario*.

c) *(1 p)* ¿Qué es y en que consiste la *segmentación simple*?

d) *(1 p)* Enumerar y describir brevemente las *capas de software de E/S del núcleo* de un sistema operativo.

e) *(1 p)* ¿Qué diferencia existe en una *copia de seguridad física* y una *copia de seguridad lógica*? Enumerar los diferentes tipos de copias de seguridad lógicas que se pueden realizar.

2. *(1 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 1: Antes de escribir el pseudocódigo se debe explicar adecuadamente el significado de cada una de las variables globales y semáforos binarios que se van a utilizar en el mismo.

Nota 2: En la resolución de este ejercicio se deben utilizar las operaciones para semáforos binarios definidas en el libro base de la asignatura.

3. 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 A tres marcos de página para su ejecución. La cadena de referencias de página que produce la ejecución del proceso A es:

`3 1 2 1 6 7 9 5 2 3 2 9 3 3 6 2`

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.

Material permitido: **Solo calculadora no programable** Tiempo: **120 minutos** **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 https://www.dia.uned.es/71902048/

4. *(2 p)* Si en un determinado instante de tiempo se dispone del grafo de asignación de recursos de un sistema es posible obtener la representación matricial de la asignación de recursos. Para ello hay que considerar las siguientes reglas:

1) Los recursos se representan en un grafo como cuadrados. Por lo tanto, el número q de recursos distintos viene dado por el número de cuadrados existentes en el grafo.
2) Las unidades existentes de un determinado recurso se representan en un grafo como puntos dentro del cuadrado asociado a dicho recurso. Por consiguiente, cada elemento R_{Ej} del vector $\mathbf{R_E}$ de recursos existentes se determina contabilizando el número de puntos representados dentro del cuadrado del grafo asociado al recurso j.
3) Los procesos se representan en un grafo como círculos. Luego el número de procesos p viene dado por el número de círculos existentes en el grafo.
4) Si un proceso P_i tiene asignada una unidad de un recurso R_j, entonces se representa en el grafo una flecha que va desde la unidad del recurso hacia el proceso. Por lo tanto, cada elemento A_{ij} de la matriz $\mathbf{A}$ de recursos asignados se determina contabilizando el número de flechas (unidades asignadas) que recibe el círculo-proceso P_i procedentes del cuadrado-recurso R_j.
5) El vector $\mathbf{R_A}$ de recursos asignados se puede obtener sumando las columnas de la matriz $\mathbf{A}$ obtenida en 4). También se puede obtener contabilizando el número total de flechas que sale de cada cuadrado-recurso R_j, que indican el número total de unidades asignadas de dicho recurso.
6) El vector $\mathbf{R_D}$ de recursos libres se puede obtener restando el vector $\mathbf{R_E}$ de recursos existentes obtenido en 2) del vector $\mathbf{R_A}$ de recursos asignados obtenido en 5). También se puede obtener contabilizando el número de unidades de cada cuadrado-recurso R_j de las que no parte ninguna flecha, es decir, no están asignadas.
7) Si un proceso P_i ha solicitado una unidad de un recurso R_j y se encuentra bloqueado en espera de que le sea asignada, entonces se representa en el grafo una flecha que sale del proceso hacia el recurso. Por lo tanto, cada elemento M_{ij} de la matriz $\mathbf{M}$ de recursos necesitados por cada proceso adicionalmente a los que ya posee se determina contabilizando el número de flechas (unidades solicitadas) que salen del círculo-proceso P_i hacía el cuadrado-recurso R_j.

Material permitido: **Solo calculadora no programable** Tiempo: **120 minutos** **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 https://www.dia.uned.es/71902048/

4. (**Continuación**) En un cierto instante de tiempo el grafo de asignación de recursos de un sistema es el que se muestra en la Figura 1.

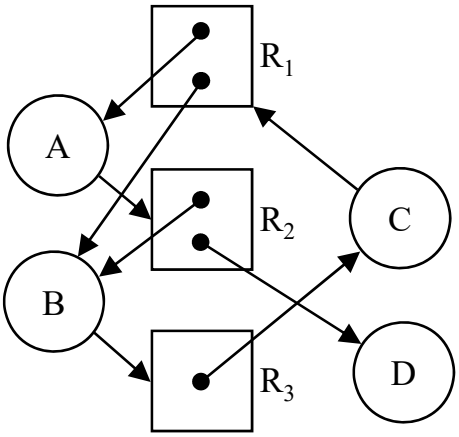

Figura 1

Se pide:

a) *(1 p)* Aplicando las 7 reglas indicadas al principio del enunciado de este ejercicio obtener la representación matricial de la asignación de recursos del sistema, es decir, $\mathbf{R_E}$, $\mathbf{A}$, $\mathbf{R_A}$, $\mathbf{R_D}$ y $\mathbf{M}$.

b) *(1 p)* Utilizando la representación matricial obtenida en el apartado anterior, determinar la posible existencia de interbloqueo mediante la aplicación del algoritmo de Coffman.

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Febrero 2025

Solución Ejercicio 1

a) Teniendo en cuenta 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 todo 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.
   - *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.

b) Los hilos de usuario tienen las siguientes ventajas:

   - *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.

   La principal desventaja de los hilos de usuario se manifiesta en aquellos sistemas operativos que únicamente soportan un único hilo del núcleo. En estos sistemas cuando un hilo de usuario de un proceso entra en el estado bloqueado, entonces todo el proceso completo entra en el estado bloqueado. Recuérdese que el sistema operativo no distingue la existencia de hilos de usuario.

Otra desventaja de los hilos de usuario es que 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. Debe tenerse en cuenta que no se pueden utilizar las interrupciones de reloj para implementar unos cuantos de ejecución (ver sección 3.6.4), ya que el tratamiento de las mismas requiere la intervención del sistema operativo.

c) La *segmentación simple* es una técnica de gestión de la memoria en la que un programa es dividido por el compilador en *segmentos*.

Cada segmento tiene asignado un nombre (código principal, subrutinas, datos, pila, etc) y una longitud. El compilador compila cada segmento comenzando por la dirección lógica 0. De esta forma cada segmento tiene su propio espacio de direcciones lógicas.

Por otra parte, cuando el sistema operativo crea un proceso asociado a un determinado programa, asigna a cada segmento un identificador numérico entero positivo *h*. Al número *h* se le denomina *número del segmento*. Así se habla del segmento *h* del proceso X.

Una dirección lógica consta de dos campos: el *número de segmento* de *s* bits y el *desplazamiento dentro del segmento* de *d* bits.

En la segmentación simple un proceso no puede ser ejecutado si todos sus segmentos no se encuentran cargados en memoria principal.

Cuando el sistema operativo tiene que cargar un proceso en la memoria principal crea una *tabla de segmentos* asociados al proceso. Esta tabla está indexada por el número de segmento, es decir, tiene una entrada por cada segmento del proceso. Cada entrada contiene la dirección física base de cada segmento y la longitud del segmento (en unidades de asignación). También contiene información relativa a los permisos de acceso al segmento (lectura, escritura, ejecución). Además puede contener información sobre ubicación de la copia del segmento en memoria secundaria. Aunque a veces esta información se implementa en una tabla diferente.

d) 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.
- *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.
- *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.

e)
- *Copia de seguridad lógica*. Únicamente contiene los directorios y archivos que el administrador o el usuario desea copiar en el medio de respaldo. Permite acceder a dichos contenidos de forma individualizada, con lo que si se ha borrado un cierto archivo del sistema de archivos se puede recuperar fácilmente de la copia de seguridad (si fue incluido en ella). Las copias de seguridad lógicas pueden ser de tres tipos:
  - *Copia completa*. Contiene todos los archivos seleccionados por el administrador o el usuario.

    - *Copia diferencial*. Contiene únicamente los archivos que han sido creados o modificados desde una última copia completa determinada.
    - *Copia incremental*. Contiene únicamente los archivos que han sido creados o modificados desde la última copia de seguridad completa o incremental.
- *Copia de seguridad física*. También conocida como *imagen de disco*. Consiste en copiar uno a uno cada uno de los bloques físicos de la partición de disco donde se ubica el sistema de archivos en el medio de respaldo elegido para contener la copia de seguridad o imagen del disco.

  Las copias de seguridad físicas son más simples y rápidas de realizar que las copias lógicas completas de todo el sistema de archivos, sin embargo son menos flexibles ya que para poder recuperar un archivo en particular es necesario restaurar o montar toda la imagen.

Solución Ejercicio 2

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

- `contador`. Variable global de tipo entero para llevar la cuenta de canarios que están comiendo del plato de alpiste. Se inicializa a 0.
- `S1`. Semáforo binario que se utiliza para garantizar la exclusión mutua en el uso de la variable global `contador`. Se inicializa a 1.
- `S2`. Semáforo binario que se utiliza para sincronizar el acceso de los canarios al plato de alpiste. Se inicializa a 1.
- `S3`. Semáforo binario que se utiliza para garantizar la exclusión mutua en el uso del columpio. Se inicializa a 1.

```
/* Definición de constantes, variables y semáforos binarios */
#define N 3
int contador=0;
semáforo_binario S1, S2, S3;


/* Proceso canario */
void canario()
{
    wait_sem(S2); /* Esperar si hay tres canarios comiendo del plato */

    wait_sem(S1);
      contador = contador + 1;
      if (contador < N) signal_sem(S2); /* Hay sitio para comer en el plato */
    signal_sem(S1);

    comer();

    wait_sem(S1);
      contador = contador - 1;
      if (contador == (N-1)) signal_sem(S2); /* Hay sitio para un canario en el plato */
    signal_sem(S1);

    wait_sem(S3); /* Esperar si el columpio está ocupado */
      columpiarse();
    signal_sem(S3); /* El columpio queda libre */
}


/* Inicialización de semáforos y ejecución concurrente */
void main()
{
    init_sem(S1,1);
    init_sem(S2,1);
    init_sem(S3,1);
    ejecución_concurrente(canario, canario,...,);
}
```

Figura 1 – Solución Ejercicio 4

Solución Ejercicio 3

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 **12 fallos de página**.

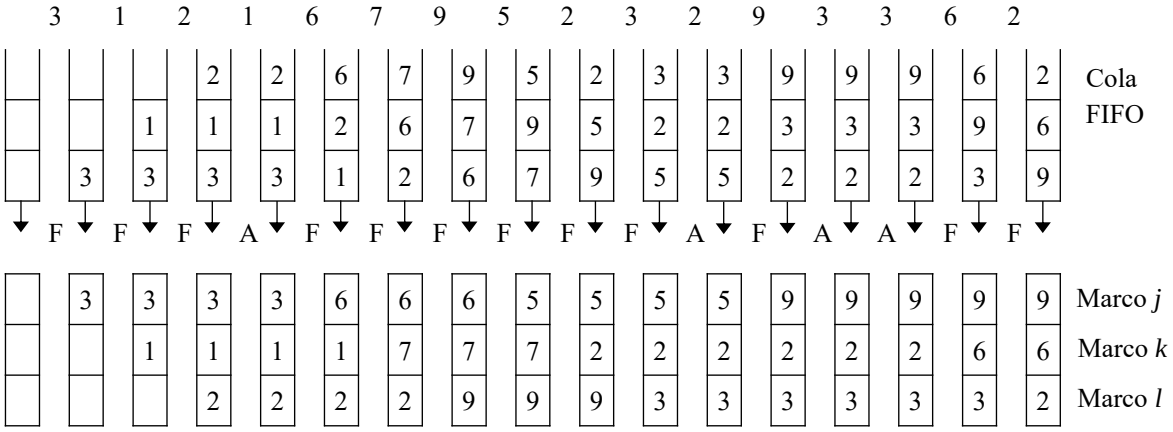

Figura 2

b) Se va a suponer que para implementar el algoritmo LRU se utiliza una lista enlazada que se gestiona como una pila. En la Figura 2 se muestra el contenido de la pila 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 **12 fallos de página**.

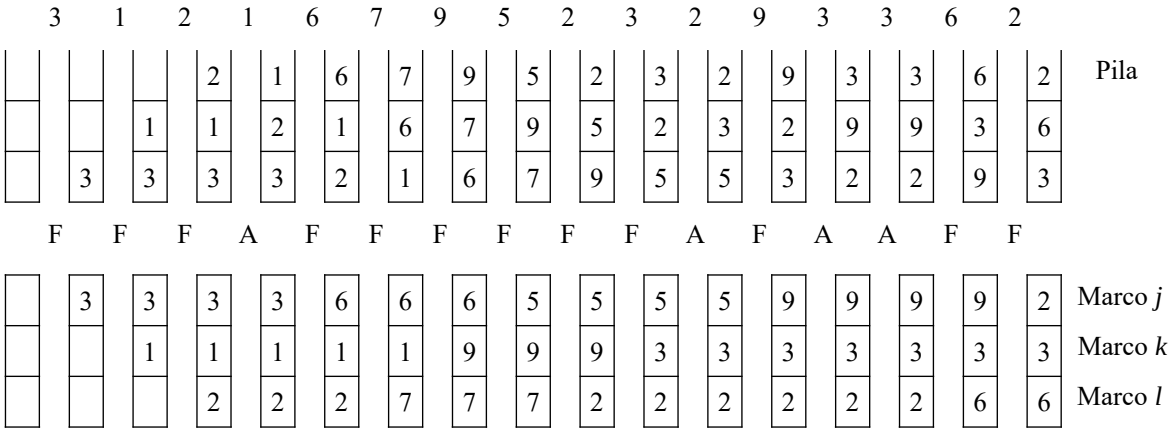

Figura 3

Solución Ejercicio 4

a) Aplicando las 7 reglas indicadas en el enunciado se obtiene la siguiente representación matricial de la asignación de recursos del sistema:

$$\mathbf{R_E} = \begin{pmatrix} 2 & 2 & 1 \end{pmatrix} \quad \mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad \mathbf{R_A} = \begin{pmatrix} 2 & 2 & 1 \end{pmatrix}$$

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

b) Se va aplicar el algoritmo de Coffman para la detección de interbloqueos:

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.
2. Se realiza la asignación

$$\mathbf{X} = \mathbf{R_D} = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$$

3. Para cada proceso no marcado, en este caso todos los procesos, se comprueba la condición

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

donde $\mathbf{M}_i$ es la fila i de la matriz **M** y el símbolo $\overset{e}{\leq}$ indica la operación de comparación entre dos vectores elemento a elemento.
Se observa

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \overset{e}{\leq} \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}$$

que solo la cuarta fila de **M**, asociada al proceso D, cumple la condición.

4. Se marca al proceso D, se supone que se ejecuta hasta su finalización y luego se liberan sus recursos. Se realiza la siguiente actualización

$$\mathbf{X} = \mathbf{X} + \mathbf{M}_4 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$$

y se comprueba si alguna fila de **M** asociada a un proceso no marcado cumple la condición

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ - & - & - \end{pmatrix} \overset{e}{\leq} \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$$

Se observa que solo la primera fila de **M**, asociada al proceso A, cumple la condición.

5. Se marca al proceso A, se supone que se ejecuta hasta su finalización y luego se liberan sus recursos. Se realiza la siguiente actualización

$$\mathbf{X} = \mathbf{X} + \mathbf{M}_1 = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}$$

y se comprueba si alguna fila de **M** asociada a un proceso no marcado cumple la condición

$$\begin{pmatrix} - & - & - \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ - & - & - \end{pmatrix} \overset{e}{\leq} \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}$$

Se observa que solo la tercera fila de **M**, asociada al proceso C, cumple la condición.

6. Se marca al proceso C, se supone que se ejecuta hasta su finalización y luego se liberan sus recursos. Se realiza la siguiente actualización

$$\mathbf{X} = \mathbf{X} + \mathbf{M}_3 = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$$

y se comprueba si alguna fila de **M** asociada a un proceso no marcado cumple la condición

$$\begin{pmatrix} - & - & - \\ 0 & 0 & 1 \\ - & - & - \\ - & - & - \end{pmatrix} \overset{e}{\leq} \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$$

Se observa que la segunda fila de **M**, asociada al proceso B, cumple la condición.

7. Se marca al proceso B. Como todos los procesos están marcados el algoritmo finaliza.

Al terminar el algoritmo todos los procesos han quedado marcados, por lo tanto, **no existe interbloqueo**.