SISTEMAS OPERATIVOS (Código: 71902048) **Febrero 2026**

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)* ¿En qué categorías se pueden agrupar las llamadas al sistema atendiendo a su finalidad?

b) *(1 p)* Describir brevemente cada uno de los principales estados en los que se puede encontrar un determinado proceso.

c) *(1 p)* Describir brevemente el funcionamiento de la técnica de paginación por demanda.

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

e) *(1 p)* Enumerar las principales implementaciones de una lista de bloques libres y sus principales ventajas.

2. *(2 p)* En un computador con x instancias de un recurso R_1, y instancias de un recurso R_2 y z instancias de un recurso R_3 se están ejecutando los procesos P_1, P_2, P_3 y P_4. En un cierto instante de tiempo T la matriz **N** de recursos máximos necesitados, la matriz **A** de recursos asignados y el vector de recursos disponibles $\mathbf{R_D}$ son:

$$\mathbf{N} = \begin{pmatrix} 3 & 3 & 1 \\ 2 & 2 & 1 \\ 1 & 1 & 1 \\ 2 & 1 & 2 \end{pmatrix} \qquad \mathbf{A} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \qquad \mathbf{R_D} = (1\ 0\ 1)$$

En cada matriz se ha asociado la fila i al proceso P_i (i = 1, 2 ,3 y 4) y la columna j al recurso R_j (j = 1, 2 y 3). Se pide:

a) *(0.5 p)* Calcular x, y y z.

b) *(1 p)* Determinar si el estado en el instante T es seguro.

c) *(0.5 p)* Aplicando el algoritmo del banquero determinar si el sistema puede admitir una petición del proceso P_2 de una instancia del recurso R_2.

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/

3. *(2 p)* En un computador con un único procesador el sistema operativo debe planificar para ejecución el conjunto de procesos que se muestra en la siguiente tabla:

Proceso	Tiempo de llegada (ut)	Tiempo de servicio (ut)
A	0.0	8
B	0.4	4
C	1.0	1

Supuesto que el planificador del sistema operativo implementa un *algoritmo FCFS*:

a) *(1 p)* Dibujar el diagrama de uso de la CPU.

b) *(1 p)* Determinar el tiempo de retorno y el tiempo de espera de cada proceso.

4. *(1 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.

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

Nota 2: 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.

Nota 3: En la resolución de este ejercicio se deben utilizar las operaciones para semáforos generales definidas en el libro base de la asignatura.

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Febrero 2026

Solución Ejercicio 1

a) Atendiendo a su finalidad las llamadas al sistema disponibles en un determinado sistema operativo se pueden agrupar de forma general en las siguientes categorías:

- *Control de programas en ejecución (procesos)*. En esta categoría se incluyen las llamadas al sistema para crear, ejecutar, suspender un tiempo prefijado, abortar y terminar procesos. También se incluyen las llamadas para sincronizar la ejecución de un proceso con la aparición de un evento, las llamadas para obtener y establecer los atributos de un proceso, y las llamadas para asignar y liberar memoria.
- *Administración de archivos o directorios*. Dentro de esta categoría se consideran las llamadas al sistema para crear, borrar, abrir, cerrar, leer, escribir archivos o directorios. También las llamadas para obtener y establecer los atributos de los archivos o directorios.
- *Comunicaciones*. Pertenecientes a esta categoría se encuentran las llamadas al sistema para crear o borrar una conexión de comunicación. También las llamadas para enviar o recibir mensajes o para conectarse a dispositivos remotos.
- *Gestión de dispositivos*. En esta categoría se incluyen las llamadas al sistema para solicitar, liberar, leer o escribir un dispositivo. También las llamadas para obtener y establecer los atributos del dispositivo.
- *Gestión de información del sistema*. Dentro de esta categoría se encuentran las llamadas al sistema para obtener y configurar datos del sistema, o para obtener o modificar la fecha y la hora.

b) Los principales estados en los que se puede encontrar un determinado proceso son:

- *Nuevo*. El proceso acaba de ser creado pero todavía no se encuentra preparado para ser ejecutado, puesto que aunque se le han asignado algunas estructuras de datos aún no se encuentra cargado en la memoria principal.
- *Preparado*. El proceso está listo para ser ejecutado tan pronto como el planificador del sistema operativo lo considere oportuno.
- *Ejecutándose*. El proceso está siendo ejecutado en el procesador. En un computador con un único procesador solo puede existir en un determinado instante de tiempo un único proceso en este estado.
- *Bloqueado*. El proceso tiene que esperar hasta que se produzca un determinado evento, como por ejemplo, la finalización de una operación de E/S.
- *Terminado*. El proceso ha finalizado su ejecución.

Aparte de los cinco estados descritos, otros dos estados importantes definidos en algunos sistemas operativos son: *preparado en memoria secundaria* y *bloqueado en memoria secundaria*. Básicamente se trata de procesos en el estado preparado o en el estado bloqueado que han sido intercambiados total o parcialmente desde la memoria principal a la memoria secundaria, para reducir el grado de multiprogramación o hacer sitio para nuevos procesos. Posteriormente un proceso en memoria secundaria puede ser intercambiado de vuelta a la memoria principal.

c) La técnica de *paginación por demanda*, también conocida como técnica de *demanda de página*, al igual que la técnica de paginación simple divide el espacio de la memoria principal en bloques de igual tamaño denominados marcos de página. Además divide el espacio de un proceso en bloques del mismo tamaño denominados páginas. En un determinado instante de tiempo un marco de página *j* puede estar libre o ocupado por una página *i* de un proceso X.

En la paginación por demanda, a diferencia de la paginación simple, para que un proceso se pueda ejecutar no es necesario que todas las páginas del mismo estén cargadas en memoria principal, únicamente se cargan las páginas que se van referenciando durante la ejecución del proceso. Cuando se referencia a una página que no está cargada en la memoria principal el hardware produce una excepción denominada *fallo de página*. El sistema operativo se encarga de atender los fallos de página. Básicamente el tratamiento de un fallo de página requiere que se busque un marco libre o se elija uno ocupado para ser reemplazado, y se copie en el mismo desde memoria secundaria la página que produjo el fallo de página.

El sistema operativo mantiene las siguientes estructuras de datos para poder implementar la técnica de paginación por demanda: la *tabla de marcos de página*, la *lista de marcos libres* y las *tablas de páginas*. Las dos primeras estructuras son similares a las utilizadas en la paginación simple. Sin embargo las tablas de páginas usadas en paginación por demanda contienen más información que las utilizadas en paginación simple. Como mínimo, una entrada de una tabla de páginas asociada a la página *i* de un proceso, aparte del marco de página *j* donde se aloja la página, debe contener un bit denominado *presente/ausente* o *validez/invalidez* para indicar si la página está cargada en la memoria principal.

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) La lista de bloques libres se puede implementar de las siguientes formas:

- *Mapa de bits*. Cada bloque de disco de la partición del sistema de archivos tiene asignado un bit. Si el bloque está ocupado se marca con un 0 y si está libre con un 1, o viceversa. Si la partición asociada al sistema de archivos ocupa N bloques de disco, entonces el mapa de bits ocupará N bits.

  La principal ventaja de un mapa de bits es la sencillez para encontrar el primer bloque libre o un conjunto de *k* bloques libres consecutivos. Su principal inconveniente es que para que su gestión sea eficiente es necesario mantener todo el mapa de bits en memoria principal, lo cual puede que no sea posible si el tamaño de la partición del sistema de archivos es muy grande.

– *Lista enlazada de bloques de disco*. Supóngase que en un bloque de disco se puede almacenar N_D direcciones de bloque, entonces cada bloque de la lista contendrá $N_D - 1$ direcciones de bloques libres y la dirección del siguiente bloque de la lista.

La principal ventaja de esta implementación consiste en que solo es necesario mantener simultáneamente en memoria principal un bloque de la lista enlazada. Cuando se necesitan bloques libres para asignárselos a un archivo se utilizan los apuntados por el bloque de la lista enlazada. Sólo cuando se agotan los bloques libres a los que hace referencia dicho bloque es necesario cargar en memoria principal el siguiente bloque de la lista enlazada. Por otra parte, cuando se libera espacio asignado a un archivo, las direcciones de los nuevos bloques libres son incluidas en el bloque de la lista enlazada cargado en memoria principal. Si un bloque de la lista ya no tiene capacidad para almacenar más direcciones de bloques libres se escribe en el disco y se añade un nuevo bloque al principio de la lista.

Solución Ejercicio 2

a) En este apartado en realidad nos están pidiendo que calculemos los elementos del vector de recursos existentes:

$$\mathbf{R_E} = (\ x \quad y \quad z\ )$$

Este vector se calcula como

$$\mathbf{R_E} = \mathbf{R_D} + \mathbf{R_A}$$

De la matriz **A**, sumando los elementos de una misma columna, se obtiene el vector de recursos asignados

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

Luego

$$\mathbf{R_E} = (\ 1 \quad 0 \quad 1\ ) + (\ 2 \quad 3 \quad 2\ ) = (\ 3 \quad 3 \quad 3\ )$$

En consecuencia $x = 3$, $y = 3$ y $z = 3$.

b) El número de instancias de cada recurso que todavía necesita cada proceso se obtiene restando la matriz **N** y la matriz **A**.

$$\mathbf{N} - \mathbf{A} = \begin{pmatrix} 3 & 3 & 1 \\ 2 & 2 & 1 \\ 1 & 1 & 1 \\ 2 & 1 & 2 \end{pmatrix} - \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 2 & 1 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Para saber si este estado es seguro simplemente hay que comprobar si es posible completar la ejecución de todos los procesos. Para ello en primer lugar hay que comprobar si existe alguna fila i de $\mathbf{N} - \mathbf{A}$, es decir, algún proceso P_i $i = 1, ..., 4$ que cumpla la condición

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

donde el símbolo $\overset{e}{\leq}$ indica que la comparación de cada fila se debe realizar elemento a elemento, es decir, se compara el elemento $\mathrm{N}_{ij} - \mathrm{A}_{ij}$ de $\mathbf{N}_i - \mathbf{A}_i$ con el elemento $\mathrm{R}_{\mathrm{D}j}$ del vector $\mathbf{R_D}$ siendo $j = 1, 2, 3$.

En este caso, la condición toma la forma

$$\begin{pmatrix} 2 & 2 & 1 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} \overset{e}{\leq} \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}$$

Se observa que solo la tercera fila, asociada al proceso P_3, cumple la condición. Luego al proceso P_3 se le pueden conceder todos los recursos que necesita para completarse aunque los solicite todos a la vez.

Supóngase que el proceso P_3 se ha completado, con lo que los recursos que tenía asignados quedan libres y se añaden al vector de recursos disponibles. La condición pasaría a ser:

$$\begin{pmatrix} 2 & 2 & 1 \\ 2 & 1 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \overset{e}{\leq} \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$$

Se observa que la cuarta fila, asociada al proceso P_4, cumple la condición. Luego al proceso P_4 se le pueden conceder todos los recursos que necesita para completarse aunque los solicite todos a la vez.

Supóngase que el proceso P_4 se ha completado, la condición pasaría a ser:

$$\begin{pmatrix} 2 & 2 & 1 \\ 2 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \overset{e}{\leq} \begin{pmatrix} 2 & 1 & 2 \end{pmatrix}$$

Se observa que la segunda fila, asociada al proceso P_2, cumple la condición. Luego al proceso P_2 se le pueden conceder todos los recursos que necesita para completarse aunque los solicite todos a la vez.

Supóngase que el proceso P_2 se ha completado, la condición pasaría a ser:

$$\begin{pmatrix} 2 & 2 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \overset{e}{\leq} \begin{pmatrix} 2 & 2 & 3 \end{pmatrix}$$

Se observa que la primera fila, asociada al proceso P_1, cumple la condición. Luego al proceso P_1 se le pueden conceder todos los recursos que necesita para completarse aunque los solicite todos a la vez.

En conclusión, como se ha podido completar la ejecución de los cuatro procesos, **el estado** S **es seguro**.

c) El algoritmo del banquero consiste en simular que se concede la petición y comprobar si el nuevo estado es seguro. De acuerdo con el enunciado si atiende la petición de una instancia del recurso R_2 por parte del proceso P_2, entonces se tendrían los siguientes valores para **A**, $\mathbf{R_A}$ y $\mathbf{R_D}$:

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \qquad \mathbf{R_A} = \begin{pmatrix} 2 & 4 & 2 \end{pmatrix} \qquad \mathbf{R_D} = \begin{pmatrix} 1 & -1 & 1 \end{pmatrix}$$

En el nuevo vector de recursos asignados $\mathbf{R_A}$ se observa que habría 4 instancias del recurso R_2 asignadas, pero esto es imposible, ya que solo existen 3 instancias de dicho recurso.

También se observa que en el nuevo vector de recursos disponibles $\mathbf{R_D}$ habría -1 instancia disponible del recurso R_2, lo cual no es posible, ya que el número de instancias disponibles de un determinado recurso siempre es un número positivo o cero.

En conclusión, **no es posible admitir** una petición del proceso P_2 de una instancia del recurso R_2.

Solución Ejercicio 3

a) Con el algoritmo FCFS, que es no expropiativo, los procesos son planificados para ejecución en orden de llegada, por lo tanto primero se ejecuta el proceso A, luego el B y finalmente el C. En la Figura 1 se muestra el diagrama de uso de la CPU.

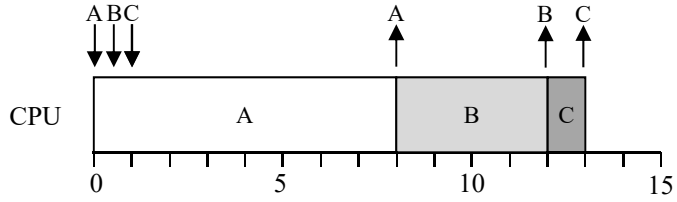

Figura 1

b) El *tiempo de retorno* se calcula como la diferencia entre el tiempo de finalización T_F y el tiempo de llegada T_{LL}:

$$T_R = T_F - T_{LL} \tag{1}$$

Mientras que el *tiempo de espera* T_E se calcula como la diferencia entre el tiempo de retorno T_R y el tiempo de servicio o ejecución T_S:

$$T_E = T_R - T_S \tag{2}$$

En la Tabla 1 se muestran los valores de T_{LL}, T_F, T_R, T_S y T_E para los procesos A, B y C.

Procesos	T_{LL} (ut)	T_F (ut)	T_R (ut)	T_S (ut)	T_E (ut)
A	0	8	8	8	0
B	0.4	12	11.6	4	7.6
C	1	13	12	1	11

Tabla 1

Solución Ejercicio 4

En la Figura 2 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 2