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

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

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

a) *(1 p)* ¿Cuáles son los *subsistemas o componentes principales del núcleo* de un sistema operativo?

b) *(1 p)* ¿Qué se entiende por *sobrecarga del sistema*?

c) *(1 p)* ¿Qué es y en qué consiste la *paginación simple*?

d) *(1 p)* Explicar la gestión de la salida por pantalla.

e) *(1 p)* Señalar y describir brevemente tres formas diferentes de implementación de directorios.

2. *(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	7
B	2	4
C	4	2

El planificador del sistema operativo implementa un *algoritmo de turno rotatorio* con un cuanto de 2 ut. Si el cuanto del proceso en ejecución expira a la vez que la llegada de un nuevo proceso, entonces el nuevo proceso se añade a la cola de procesos preparados para ejecución antes que el proceso que termina su cuanto. Se pide:

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.

3. *(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.

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

4. *(1 p)* Una pasarela colgante de madera conecta las dos orillas de un río. La pasarela tiene una anchura que solo permite el paso de personas en un determinado sentido: izquierda o derecha. Si una persona circula en un sentido, primero pasará esta y todas los personas que estén esperando para pasar en el mismo sentido. Cuando estas hayan finalizado, comenzarán a pasar por la pasarela las personas que vayan en sentido contrario, si es que hay alguna esperando. Suponer que una persona nunca accede a la pasarela si ve que hay personas circulando en sentido contrario al que desea ir. Suponer, además, que las personas nunca se detienen cuando cruzan la pasarela. Escribir el pseudocódigo de un programa que coordine la circulación de las personas por la pasarela usando **semáforos binarios**.

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: El pseudocódigo debe tener cuatro partes: declaración de variables, código del proceso `personas_ID` asociado a las personas que desean circular en sentido izquierda-derecha, código del proceso `personas_DI` asociado a las personas que desean circular en sentido derecha-izquierda, 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 binarios definidas en el libro base de la asignatura.

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Enero 2026

Solución Ejercicio 1

a) Los *subsistemas o componentes principales 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.

b) Se denomina *sobrecarga del sistema*, o simplemente *sobrecarga* (overhead), al tiempo que el procesador se encuentra ocupado ejecutando código del sistema operativo asociado a tareas y servicios de administración que no se pueden contabilizar a ningún proceso en particular.

c) La técnica de gestión de memoria conocida como *paginación simple* consiste en dividir la memoria principal en bloques del mismo tamaño S_P denominados *marcos de página* o *páginas físicas*. El espacio de direcciones de un proceso también se divide en bloques del mismo tamaño S_P denominados *páginas* o *páginas lógicas*. Una página de un proceso se carga en un marco de página libre de memoria principal. Además las páginas de un mismo proceso no tienen porque ocupar marcos contiguos.

Cada marco de página tiene asignado un identificador numérico entero positivo j que lo identifica de forma unívoca. Al número j se le denomina *número del marco de página*. Así se habla del *marco de página j* o *marco j* de memoria principal.

Cada página del proceso tiene asignado un identificador numérico entero positivo i. Al número i se le denomina *número de página*. Así se habla de la página i del proceso X.

Un sistema operativo para implementar la paginación generalmente utiliza las siguientes estructuras de datos:

- *Tablas de páginas*. Cada proceso tiene asignada una tabla de páginas. Cada entrada *i* de la tabla de páginas de un proceso X contiene, entre otras informaciones, el marco *j* donde se encuentra almacena la página *i* del proceso X y los permisos de acceso a la página. Además puede contener información sobre la ubicación de la copia de la página en memoria secundaria, aunque a veces esta información se implementa en una tabla diferente.
- *Tabla de marcos de página*. Esta tabla tiene tantas entradas como marcos de página tiene la memoria principal. Cada entrada *j* de la tabla de marcos de página contiene, entre otras, las siguientes informaciones relativas al marco: su estado, es decir, si está libre o ocupado, punteros para crear una lista de marcos libres, y la ubicación en memoria secundaria de la copia de la página *i* contenida en el marco.
- *Lista de marcos libres*. Es consultada cuando hay que asignar espacio a los procesos que deben ser cargados en memoria principal. Cuando hay que asignar un marco se selecciona al primero de la lista. Si un marco queda libre es colocado al final de la lista.

d) La *pantalla* de un computador es un dispositivo de salida de gráficos de trama (raster graphics) también denominados gráficos de mapa de bits (bitmap graphics). La salida que se muestra en pantalla es una red de puntos denominados píxels. Cada *píxel* tiene asociado un número binario que indica el color con que se representa en la pantalla. Por ejemplo, se pueden utilizar un número de 24 bits para representar cada píxel, dedicando 8 bits para especificar cada intensidad de rojo, verde y azul. Nótese que para evitar el parpadeo de la pantalla en la misma se deben mostrar una imagen de 60 a 100 veces por segundo.

   La pantalla se suele conectar al computador a través de un *adaptador o controlador gráfico* que puede estar integrado en la propia placa base del computador, o en una tarjeta que se pincha sobre la misma. Un adaptador gráfico tiene, entre otros componentes, una *memoria de video* (Video RAM, VRAM), que es una memoria RAM en la que se almacena los pixeles de la imagen que se mostrará en la pantalla. Nótese que el tamaño en bits que ocupará una imagen en la VRAM dependerá del número de bits utilizados para representar a un píxel y del tamaño en píxels de una imagen, es decir, de la *resolución de pantalla* (1024 x 768, 1280 x 960, 1600 x 1200, etc).

   La imagen que se carga en la VRAM es generada por paquetes de rutinas de creación y manipulación de gráficos independientes del hardware, las cuales son invocadas por las aplicaciones o por el gestor de ventanas de un GUI. Estos paquetes pueden formar parte del núcleo del sistema operativo (Windows) o ser externos al mismo (UNIX y Linux). En el primer caso, dichas rutinas se ejecutarán en modo núcleo, en el segundo caso en modo usuario. Para interactuar con el adaptador gráfico las propias rutinas (Windows) o el subsistema de E/S (UNIX y Linux) invocan al driver del adaptador, que es el que conoce los detalles del hardware.

e) Los directorios se pueden implementar de diferentes formas, entre las implementaciones más comunes se encuentran las siguientes:

   - *Directorios con entradas de igual tamaño*. Todas las entradas de un directorio poseen el mismo tamaño. En cada entrada de un directorio primero se almacenan las informaciones asociadas al archivo (atributos o número de nodo-i) y a continuación el nombre del archivo.
   - *Directorios con entradas de tamaño variable*. Cada entrada del directorio tiene un determinado tamaño. En cada entrada de un directorio primero se almacena el tamaño que ocupa la entrada. A continuación se almacenan las informaciones asociadas al archivo (atributos o número de nodo-i) y finalmente se almacena el nombre del archivo que no puede superar un cierto tamaño máximo. Al nombre del archivo se le añade un carácter especial, por ejemplo un 0, para marcar el final del nombre. Además se añaden algunos caracteres de relleno para que el tamaño de una entrada sea un número entero positivo de palabras de memoria principal. De este modo se evita que dentro de una misma palabra haya información sobre dos entradas de directorios distintas, lo que complicaría la administración de los directorios.

– *Directorios con entradas de igual tamaño y uso de un montículo (heap) para almacenar los nombres de los archivos*. Cada entrada del directorio tiene el mismo tamaño, en ella se almacena un puntero al comienzo del nombre archivo dentro del montículo y las informaciones asociadas al archivo (atributos o número de nodo-i). Al final del directorio se almacena la estructura de datos de tipo montículo para almacenar de nombres de archivos. Dentro del montículo los nombres se almacenan de forma contigua incluyendo un carácter especial para marcar el final de un nombre de archivo.

Solución Ejercicio 2

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

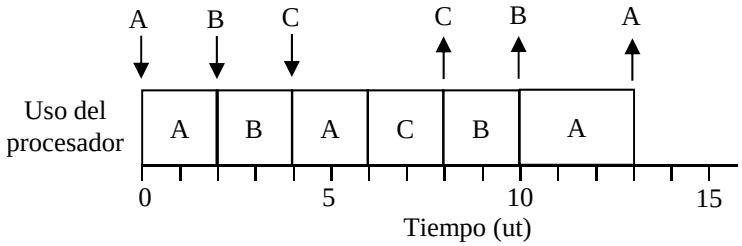

Figura 1

b) En la Tabla 1 los tiempos de llegada T_{LL}, finalización T_F, retorno T_R, servicio T_S y espera T_E de cada proceso.

Procesos	$\mathbf{T_{LL}}$ **(ut)**	$\mathbf{T_F}$ **(ut)**	$\mathbf{T_R = T_F - T_{LL}}$	$\mathbf{T_S}$ **(ut)**	$\mathbf{T_E = T_R - T_S}$
A	0	13	13	7	6
B	2	10	8	4	4
C	4	8	4	2	2

Tabla 1

Solución Ejercicio 3

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 virtual se descompone en número de página de p bits y desplazamiento dentro de la página de d bits.

- *Dirección física*:

  Del dato de la capacidad de la memoria principal C_{MP} expresada en palabras se puede determinar el tamaño n en bits de una dirección física, para ello hay que resolver la siguiente desigualdad:

  $$\min_{n} \{C_{MP} \leq 2^n\}$$

  Del enunciado se sabe que $C_{MP} = 256$ MiB $= 2^{28}$ bytes, dividiendo por la longitud de una palabra (16 bits = 2 bytes) se obtiene la capacidad expresada en palabras:

  $$C_{MP} = \frac{2^{28} \text{ bytes}}{2 \text{ (bytes/palabra)}} = 2^{27} \text{ palabras}$$

  Luego $n = 27$ bits.

  Por otra parte del tamaño de una página S_P expresado en palabras se puede determinar el tamaño d en bits del campo desplazamiento tanto de una dirección física como de una dirección virtual, para ello hay que resolver la siguiente desigualdad:

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

  Del enunciado se sabe que $S_P = 4$ KiB $= 2^{12}$ bytes, dividiendo por la longitud de una palabra se obtiene el tamaño de una página expresado en palabras:

  $$S_P = \frac{2^{12} \text{ bytes}}{2 \text{ (bytes/palabra)}} = 2^{11} \text{ palabras}$$

  Luego $d = 11$ bits.

  El tamaño f del campo número de marco de página se puede obtener, por ejemplo, de la siguiente forma:

  $$f = n - d = 27 - 11 = 16 \text{ bits}$$

- *Dirección virtual*:

  Del dato del tamaño C_A del espacio de direcciones virtuales del proceso A expresado en palabras se puede determinar el tamaño m en bits de una dirección virtual, para ello hay que resolver la siguiente desigualdad:

  $$\min_{m} \{C_A \leq 2^m\}$$

  Del enunciado se sabe que $C_A = 128$ MiB $= 2^{27}$ bytes, dividiendo por la longitud de una palabra se obtiene el tamaño del espacio virtual del proceso A expresado en palabras:

  $$C_A = \frac{2^{27} \text{ bytes}}{2 \text{ (bytes/palabra)}} = 2^{26} \text{ palabras}$$

Luego $m = 26$ bits.

El tamaño del campo desplazamiento de una dirección virtual tiene el mismo tamaño que el de una dirección física. Por su parte, el tamaño p del campo número de página se puede obtener, por ejemplo, de la siguiente forma:

$$p = m - d = 26 - 11 = 15 \text{ bits}$$

En la Figura 2 se representa el formato de una dirección física y de una dirección virtual.

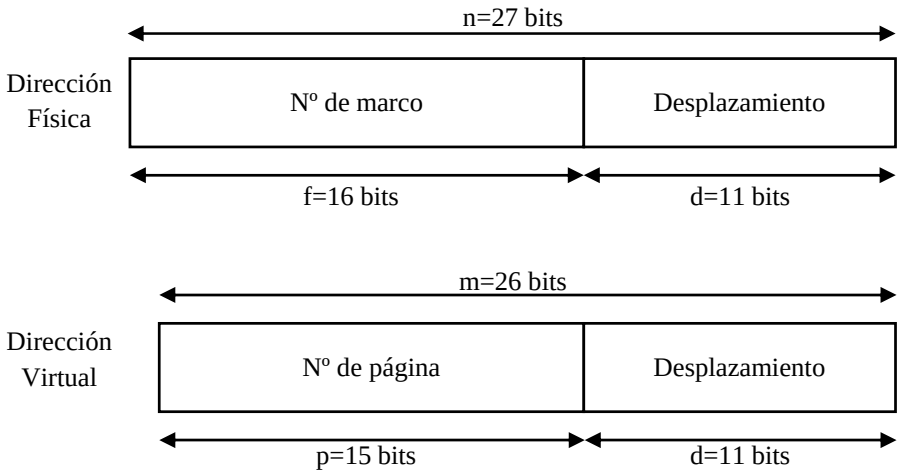

Figura 2

Solución Ejercicio 4

Nota: Con objeto de simplificar la solución de este ejercicio se ha supuesto que no se presentan problemas de inanición de procesos.

En la Figura 3 se muestra una posible solución para este ejercicio. Esta solución utiliza las siguientes variables y semáforos:

- `contID`. Variable global de tipo entero para llevar la cuenta del número de personas que quieren cruzar la pasarela en sentido izquierda-derecha.
- `contDI`. Variable global de tipo entero para llevar la cuenta del número de personas que quieren cruzar la pasarela en sentido derecha-izquierda.
- `S1`. Semáforo binario para garantizar la exclusión mutua en el uso de la variable `conID`.
- `S2`. Semáforo binario para garantizar la exclusión mutua en el uso de la variable `conDI`.
- `S3`. Semáforo binario para sincronizar la circulación de personas por la pasarela.

```
int contID=0, contDI=0; /* Definición e inicialización de las variables globales */
semáforo_binario S1, S2, S3; /* Definición semáforos binarios */

void personas_ID()
{
        wait_sem(S1);
          contID = contID + 1;
          if (contID==1) wait_sem(S3); /* Esperar si hay circulación en sentido contrario */
        signal_sem(S1);

        cruzar_pasarela();

        wait_sem(S1);
          contID = contID - 1;
          if (contID==0) signal_sem(S3); /* Desbloquear circulación en sentido contrario */
        signal_sem(S1);
}

void personas_DI()
{
        wait_sem(S2);
          contDI = contDI + 1;
          if (contDI==1) wait_sem(S3); /* Esperar si hay circulación en sentido contrario */
        signal_sem(S2);

        cruzar_pasarela();

        wait_sem(S2);
          contDI = contDI - 1;
          if (contDI==0) signal_sem(S3); /* Desbloquear circulación en sentido contrario */
        signal_sem(S2);
}

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

Figura 3 – Solución apartado a) Ejercicio 2