SISTEMAS OPERATIVOS (Código: 71902048) **Enero 2023**

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 http://www.uned.es/71902048/

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

a) *(1 p)* Describir el algoritmo de planificación de *turno rotatorio*.

b) *(1 p)* ¿Qué es y en qué consiste la *segmentación simple*?

c) *(1 p)* Si se conocen el tamaño de página y la capacidad de la memoria principal de un sistema con gestión de memoria virtual mediante demanda de página ¿se podrían determinar a partir de estos datos el tamaño de una *dirección virtual*?

d) *(1 p)* ¿Qué ventajas e inconvenientes tiene el método de *asignación contigua* de espacio en disco?

2. *(2 p)* Enumerar las ventajas y los inconvenientes de los *procesos multihilo*.

3. *(2 p)* Supóngase un sistema con cinco procesos (P0, P1, P2, P3 y P4) y tres recursos diferentes (R0, R1 y R2). En un instante determinado el sistema se encuentra en el siguiente estado:

	Asignados			**Máximos necesitados**		
Proceso	R0	R1	R2	R0	R1	R2
P0	1	0	0	4	4	4
P1	0	1	2	1	4	6
P2	0	0	3	3	2	5
P3	1	0	3	4	4	8
P4	1	1	0	5	1	10

Disponibles		
R0	R1	R2
3	3	2

Determinar si el estado es seguro.

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 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.

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Enero 2023

Solución Ejercicio 1

a) El *algoritmo de planificación de turno rotatorio* (*round robin*) asigna el uso ininterrumpido del procesador a un proceso durante una cantidad fija de tiempo denominada *cuanto* (quantum). Finalizado el cuanto, se genera una interrupción de reloj que produce que la ejecución del proceso actual sea interrumpida y se pase a ejecutar el primer proceso de la cola de preparados. El proceso interrumpido se coloca al final de dicha cola y tendrá que esperar turno hasta volver a utilizar el procesador. Si un proceso termina de usar el procesador antes de consumir su cuanto, entonces se planifica el primer proceso de la cola preparados sin esperar a que finalice el cuanto. Por otra parte, si finaliza un cuanto y no existen procesos en la cola de preparados, entonces el proceso que se estaba ejecutando puede seguir usando el procesador.

b) La *segmentación* es una técnica de gestión de la memoria que soporta los elementos en los que se descompone el código de un programa. Básicamente el programa es dividido por el compilador en *segmentos*. Cada segmento es una entidad lógica conocida por el programador asociada a una determinada estructura de datos o a un módulo, pero nunca a una mezcla de ambos.

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.

En la segmentación, una dirección lógica consta de dos campos (ver Figura 6.28): 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 se va a cargar un proceso en memoria principal el sistema operativo debe buscar, en la estructura de datos que mantenga para registrar el espacio ocupado y libre de la memoria principal, un hueco para cada segmento. Dentro de un hueco, a cada segmento se le asigna justo el espacio (en unidades de asignación) que necesita, el espacio sobrante pasa a formar un nuevo hueco. Con este esquema de asignación los segmentos de un proceso no tienen por qué estar ubicados en particiones contiguas.

Además, 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.

c) En la técnica de paginación una dirección lógica o virtual se descompone en dos campos: número de página y desplazamiento. Para determinar el tamaño del campo número de página se requiere conocer el número total de páginas de que consta un proceso y para determinar el tamaño del campo desplazamiento se necesita conocer el tamaño de página.

Con los datos que indican en el enunciado, capacidad de la memoria principal y tamaño de página, solo se puede calcular el tamaño del campo desplazamiento. Faltaría conocer el tamaño de un proceso para poder obtener, dividiendo por el tamaño de páginas, el número de páginas y determinar

el tamaño del campo número de página. Recuérdese que en la técnica de paginación por demanda, el tamaño de un proceso puede ser mayor que el tamaño de la memoria principal.

En conclusión, **no se pueden determinar a partir de estos datos el tamaño de una dirección virtual**.

d) El *método de asignación contigua de espacio en disco* presenta las siguientes ventajas:

- *Minimiza las operaciones de búsqueda en disco, aumentando así su rendimiento*.
- *Permite soportar archivos tanto de acceso secuencial como de acceso aleatorio*, ya que la localización de los bloques de un archivo se realiza de forma rápida y sencilla.

Los principales inconvenientes de este método son:

- *Produce fragmentación externa en el disco*.
- *El sistema operativo debe conocer el tamaño que ocupará el archivo que se va a crear*, lo cual en la mayoría de las ocasiones es un dato que no es posible predecir a priori.

Solución Ejercicio 2

Las principales ventajas de los procesos multihilos son las siguientes:

- *Aumento del rendimiento del sistema*. La creación de un hilo nuevo dentro de un proceso ya existente requiere de menos tiempo que la creación de un nuevo proceso. En algunos sistemas crear un hilo es varios órdenes de magnitud más rápido que crear un proceso. Asimismo el tiempo necesario para finalizar un hilo es menor que el tiempo necesario para finalizar un proceso. Además, un cambio de proceso requiere más tiempo que un cambio de hilo dentro de un mismo proceso.
- *Ahorro de recursos*. La creación de un hilo nuevo dentro de un proceso ya existente requiere menos recursos del sistema que la creación de un proceso nuevo.
- *Comunicación más eficiente*. La comunicación entre dos procesos independientes requiere la intervención del sistema operativo para proporcionar los mecanismos necesarios de comunicación y protección. Por el contrario, como los hilos de un proceso comparten memoria y archivos se pueden comunicar entre ellos sin la intervención del sistema operativo.
- *Mayor aprovechamiento de las arquitecturas multiprocesador*. En un proceso multihilo cada hilo podría estar ejecutándose en paralelo en cada uno de los procesadores de la máquina. Por el contrario un proceso tradicional solo puede usar un único procesador a la vez.
- *Simplificación de la estructura de las aplicaciones*. El uso de múltiples hilos generalmente ayuda a simplificar la estructura de aplicaciones software que tienen que realizar varias tareas diferentes.

El uso de un proceso de múltiples hilos para implementar una aplicación introduce diversas complicaciones o inconvenientes en la programación de la aplicación, la mayoría asociadas a la necesidad de sincronización en el acceso a las estructuras de datos que comparten los hilos. Estas complicaciones se pueden resolver diseñando con cuidado la ejecución de los múltiples hilos y utilizando mecanismos de sincronización.

Solución Ejercicio 3

Del enunciado se deduce que el estado en que se encuentra el sistema es el siguiente:

$$S = \left\{ \mathbf{N} = \begin{pmatrix} 4 & 4 & 4 \\ 1 & 4 & 6 \\ 3 & 2 & 5 \\ 4 & 4 & 8 \\ 5 & 1 & 10 \end{pmatrix} \quad \mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 0 & 0 & 3 \\ 1 & 0 & 3 \\ 1 & 1 & 0 \end{pmatrix} \quad \mathbf{R_E} = \begin{pmatrix} 6 & 5 & 10 \end{pmatrix} \quad \mathbf{R_D} = \begin{pmatrix} 3 & 3 & 2 \end{pmatrix} \right\}$$

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 $i = 1, ..., 5$ 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 $N_{ij} - A_{ij}$ de $\mathbf{N}_i - \mathbf{A}_i$ con el elemento R_{Dj} del vector $\mathbf{R_D}$ siendo $j = 1, 2, 3$. En este caso, la condición toma la forma

$$\begin{pmatrix} 3 & 4 & 4 \\ 1 & 3 & 4 \\ 3 & 2 & 2 \\ 3 & 4 & 5 \\ 4 & 0 & 10 \end{pmatrix} \overset{e}{\leq} \begin{pmatrix} 3 & 3 & 2 \end{pmatrix}$$

Se observa que solo la tercera fila, asociada al proceso P2, cumple la condición. Luego al proceso P2 se le pueden conceder todos los recursos que necesita para completarse aunque los solicite todos a la vez.

Supóngase que el proceso P2 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:

$$\begin{pmatrix} 3 & 4 & 4 \\ 1 & 3 & 4 \\ 0 & 0 & 0 \\ 3 & 4 & 5 \\ 4 & 0 & 10 \end{pmatrix} \overset{e}{\leq} \begin{pmatrix} 3 & 3 & 5 \end{pmatrix}$$

Se observa que la segunda fila, asociada al proceso P1, cumple la condición. Luego al proceso P1 se le pueden conceder todos los recursos que necesita para completarse aunque los solicite todos a la vez.

Supóngase que el proceso P1 se ha completado, la condición pasaría a ser:

$$\begin{pmatrix} 3 & 4 & 4 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 3 & 4 & 5 \\ 4 & 0 & 10 \end{pmatrix} \overset{e}{\leq} \begin{pmatrix} 3 & 4 & 7 \end{pmatrix}$$

Se observa que la primera fila, asociada al proceso P0, cumple la condición. Luego al proceso P0 se le pueden conceder todos los recursos que necesita para completarse aunque los solicite todos a la vez.

Supóngase que el proceso P0 se ha completado, la condición pasaría a ser:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 3 & 4 & 5 \\ 4 & 0 & 10 \end{pmatrix} \overset{e}{\leq} \begin{pmatrix} 4 & 4 & 7 \end{pmatrix}$$

Se observa que la cuarta fila, asociada al proceso P3, cumple la condición. Luego al proceso P3 se le pueden conceder todos los recursos que necesita para completarse aunque los solicite todos a la vez.

Supóngase que el proceso P3 se ha completado, la condición pasaría a ser:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 4 & 0 & 10 \end{pmatrix} \overset{e}{\leq} \begin{pmatrix} 5 & 4 & 10 \end{pmatrix}$$

Se observa que la quinta fila, asociada al proceso P4, cumple la condición. Luego al proceso P4 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 cinco procesos, el estado S es **seguro**.

Solución Ejercicio 4

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) signal_sem(S2); /* Hay sitio para comer 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