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

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

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

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

b) *(1 p)* ¿Qué es una *tabla de páginas paginada*? Si se utiliza una tabla de páginas paginada ¿Cuántos accesos a memoria se requieren para convertir una dirección de memoria virtual a una dirección de memoria física?

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

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

2. *(2 p)* Enumerar y describir los *principales estados* en que se puede encontrar un determinado proceso.

3. 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	6
B	1	3
C	2	7
D	3	4

El planificador del sistema operativo implementa un *algoritmo de turno rotatorio* con un cuanto de 2 ut y un tiempo de cambio de proceso despreciable. 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.

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

4. *(2 p)* En la Figura 1 se muestra el pseudocódigo en C de un cierto programa que hace uso de un semáforo general para regular el acceso a 10 instancias de un cierto recurso compartido. Modificar adecuadamente este pseudocódigo basado en C para conseguir la misma funcionalidad haciendo uso de **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: En la resolución de este ejercicio se deben utilizar las operaciones para semáforos binarios definidas en el libro base de la asignatura.

```
#define TRUE 1
#define N 10
semáforo_general S;

void proceso()
{
    while(TRUE)
    {
      wait_sem(S);
      utilizar_recurso();
      signal_sem(S);
    }
}


void main()
{
    init_sem(S,N);
    ejecución_concurrente(proceso, proceso, proceso,...);
}
```

Figura 1

SISTEMAS OPERATIVOS (Cód. 71902048)
Solución Examen Febrero 2023

Solución Ejercicio 1

a) 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 todas las instancias de dichos recursos deben forman parte de caminos que sean ciclos.

b) El espacio lógico de un proceso puede ser bastante grande, sobre todo en sistemas que soportan memoria virtual. La tabla de páginas de un proceso se ubica en memoria principal ocupando un rango de direcciones físicas contiguas, para facilitar su localización y acceso. Si el proceso posee un espacio de direcciones lógicas grande, entonces su tabla de páginas ocupará una cantidad no despreciable de memoria física contigua.

Una forma de tratar a las tablas de páginas grandes es descomponerla también en páginas, se tiene por tanto una *tabla de páginas paginada*. De esta forma la tabla de páginas se fragmenta y puede ubicarse en memoria principal de forma no contigua. Así se pueden distinguir dos tipos de páginas:

- *Páginas ordinarias*. Contienen instrucciones y datos del espacio lógico de un proceso.
- *Páginas de tablas de páginas*. Contienen entradas de una tabla de páginas.

Para localizar en memoria principal a las páginas de una tabla de páginas se utiliza otra tabla de páginas denomina *tabla de páginas de primer nivel* o *tabla de páginas primaria*. A las entradas de la tabla de páginas original contenidas en una página se les denomina *tabla de páginas de segundo nivel* o *tabla de páginas secundaria*. Por lo tanto para cada proceso existe una única tabla de primer nivel y varias tablas de segundo nivel, tantas como páginas ocupe la tabla de páginas original del proceso.

Si la tabla de páginas de primer nivel resulta demasiado grande entonces podría plantearse descomponerla también en páginas. En ese caso se estaría añadiendo un nuevo nivel de paginación y se tendría un sistema de paginación de tres niveles. En teoría pueden implementarse todos los niveles que se deseen. Cada nivel de paginación implica un acceso a memoria principal en el proceso de traducción de una dirección de memoria lógica o virtual a una dirección de memoria física. Así, dos niveles de paginación requieren dos accesos, tres niveles tres accesos, etc. En consecuencia el retardo en la traducción de una dirección y por tanto en la ejecución de una instrucción puede ser importante. Por ello es raro utilizar más de tres niveles de paginación.

c) El método de acceso a disco 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.

d) La mayoría de los sistemas de archivos modernos, como por ejemplo NTFS de Windows, ext4 de Linux y APFS de Mac-OS, implementan la técnica conocida como *registro por diario* (*journaling*), que 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.

Solución Ejercicio 2

El tiempo de vida de un proceso, es decir, el tiempo comprendido desde su creación hasta su finalización, puede ser conceptualmente dividido en un conjunto de estados que describen el comportamiento del proceso. Aunque el número de estados y su nombre depende de cada sistema operativo, algunos de los estados más habituales en que puede encontrarse un proceso son los siguientes:

- *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.

Solución Ejercicio 3

a) En $t = 0$ ut llega el proceso A y comienza a ejecutarse durante un cuanto de 2 ut.

En $t = 1$ ut llega el proceso B y se añade a la cola de procesos preparados para ejecución:

		B	→

En $t = 2$ ut expira el cuanto del proceso A, al que le restan 4 ut de servicio, y se planifica para ejecución el proceso B. Además llega el proceso C. De acuerdo con el enunciado, 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. Por lo tanto, el proceso C se coloca en la cola antes que el proceso A:

	A	C	→

En $t = 3$ ut llega el proceso D y se añade a la cola de procesos preparados para ejecución:

D	A	C	→

En $t = 4$ ut expira el cuanto del proceso B y se planifica para ejecución el proceso C por estar primero en la cola. El proceso B, al que le resta 1 ut de servicio, se añade a la cola:

B	D	A	→

En $t = 6$ ut expira el cuanto del proceso C y se planifica para ejecución el proceso A por estar primero en la cola. El proceso C, al que le restan 5 ut de servicio, se añade a la cola:

C	B	D	→

En $t = 8$ ut expira el cuanto del proceso A y se planifica para ejecución el proceso D por estar primero en la cola. El proceso A, al que le restan 2 ut de servicio, se añade a la cola:

A	C	B	→

En $t = 10$ ut expira el cuanto del proceso D y se planifica para ejecución el proceso B por estar primero en la cola. El proceso D, al que le restan 2 ut de servicio, se añade a la cola:

D	A	C	→

En $t = 11$ ut finaliza su ejecución el proceso B y se planifica para ejecución el proceso C por estar primero en la cola. La cola queda en el siguiente estado:

	D	A	→

Procediendo de forma similar hasta la finalización de todos los procesos se obtiene el el diagrama de uso de la CPU que se muestra en la Figura 1

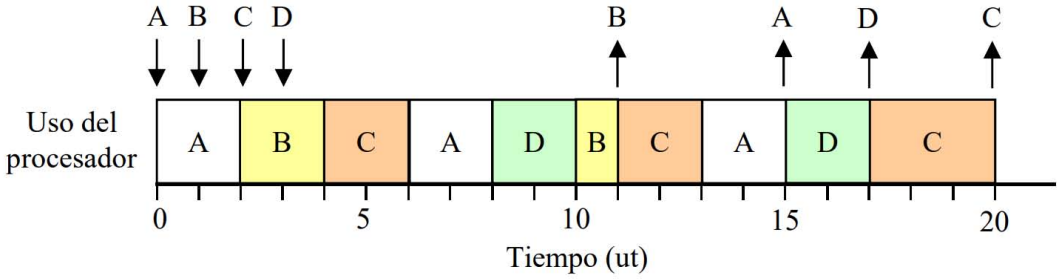

Figura 1

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

Trabajos	**T_{LL} (ut)**	**T_F (ut)**	**$T_R = T_F - T_{LL}$**	**T_S (ut)**	**$T_E = T_R - T_S$**
A	0	15	15	6	9
B	1	11	10	3	7
C	2	20	18	7	11
D	3	17	14	4	10

Tabla 1

Solución Ejercicio 4

En la Figura 2 se propone una solución que utiliza las siguientes variables globales y semáforos binarios:

- `contador`. Variable global de tipo entero para llevar la cuenta del número de instancias del recurso utilizadas. 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 a las N instancias del recurso. Se inicializa a 1.

```
/* Definición de constantes, variables y semáforos binarios */
#define TRUE 1
#define N 10
int contador=0;
semáforo_binario S1, S2;


/* Proceso */
void proceso()
{
  while(TRUE)
  {
    wait_sem(S2); /* Esperar si todas las instancias del recurso están ocupadas */
    wait_sem(S1);
    contador = contador + 1;
    if (contador < N) signal_sem(S2); /* Existen instancias del recurso libres */
    signal_sem(S1);
        utilizar_recurso();
    wait_sem(S1);
    contador = contador - 1;
    if (contador < N) signal_sem(S2); /* Queda alguna instancia del recurso libre */
    signal_sem(S1);
  }
}


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

Figura 2 – Solución del apartado a) del Ejercicio 3