SISTEMAS OPERATIVOS (Código: 71902048) **Febrero 2012**

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **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 los siguientes apartados:

a) *(1 p)* ¿Qué información se suele almacenar de forma general en una entrada de un directorio?

b) *(1 p)* Describir la planificación de hilos en función del tipo de hilos soportado por el sistema operativo.

2. *(2 p)* Enumerar las acciones que debe realizar el sistema operativo para crear un proceso.

3. *(2 p)* Explicar **razonadamente** qué es el *buffering* y comentar los problemas que resuelve.

4. *(2 p)* En una oficina de Correos existen 3 ventanillas de atención al cliente. Cuando un cliente entra en la oficina para realizar alguna gestión debe guardar una única cola hasta que alguna ventanilla queda libre. Explicar **razonadamente** si el pseudocódigo del programa que se muestra en la Figura 1 coordina adecuadamente la actividad de los clientes en la oficina. En caso negativo modifique el programa para que funcione correctamente.

5. *(2 p)* El sistema operativo en colaboración con el hardware gestiona la memoria principal mediante paginación por demanda. La traducción de direcciones se realiza usando una MMU con un TLB. El tiempo medio de acceso al TLB es despreciable y su tasa de aciertos es del 95 %. Determinar el tiempo medio de despacho de una referencia a memoria si se tienen: una tasa de fallos de página del 8 %, un tiempo medio de acceso a memoria de 125 ns y un tiempo medio de gestión de un fallo de página de 40 ms. Despreciar la existencia de memoria caché.

Material permitido: **Solo calculadora no programable** Tiempo: **2 horas** **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/

```
int Z=0;
semáforo_binario X, Y;

void cliente()
{
        wait_sem(X);
        Z = Z + 1;
        if (Z > 3){
          signal_sem(X);
          wait_sem(Y);
        }

        signal_sem(X);
        realizar_gestión();
        wait_sem(X);
        Z = Z - 1;
        signal_sem(Y);
        signal_sem(X);

}

main()
{
        init_sem(X,1);
        init_sem(Y,0);
        ejecución_concurrente(cliente,...,cliente);
}
```

Figura 1

Grado en Ingeniería Informática/Grado en Ingeniería en Tecnologías de la Información

SISTEMAS OPERATIVOS
Solución examen febrero 2012 (2ª semana)

Solución Ejercicio 1

a) Un *directorio* es un archivo que almacena una lista de los archivos y subdirectorios que contiene. La información que almacena cada entrada de la lista depende de cada sistema de archivos, aunque en general se suele almacenar el nombre del archivo y otras informaciones adicionales asociadas al archivo. Cuando se abre un archivo el sistema operativo cargar en memoria principal la entrada del directorio que lo referencia para poder acceder a la información que contiene.

   Algunos sistemas de archivos, como por ejemplo FAT-32 de Windows, almacenan en cada entrada de un directorio el nombre de un archivo o subdirectorio y sus atributos. Recuérdese que entre los atributos de un archivo se incluye la información necesaria que permite localizar los bloques de datos del archivo.

   Otros sistemas de archivos, como por ejemplo UFS de UNIX o ext2 de Linux, almacenan en cada entrada de un directorio el nombre de un archivo o subdirectorio y un puntero (número de nodo-i) a la estructura de datos (nodo-i) donde se almacenan los atributos del archivo.

b) La planificación de los hilos depende de los tipos de hilos soportados por el sistema operativo. Si únicamente se soportan *hilos a nivel de usuario*, el sistema operativo no es consciente de la existencia de estos hilos y por ello realiza una planificación global a nivel de proceso de acuerdo con un determinado algoritmo de planificación.

   Una vez que un proceso está siendo ejecutado en modo usuario es el hilo de planificación dentro de dicho proceso el que decide qué hilo va a ser ejecutado. Por lo tanto cada proceso puede planificar sus hilos con el algoritmo de planificación que considere más oportuno.

   Puesto que en modo usuario no se pueden tratar las interrupciones, no existen interrupciones de reloj disponibles para establecer el cuanto de un hilo. Éste se ejecuta hasta que voluntariamente decide ceder el uso del procesador o expira el cuanto del proceso. En dicho caso el sistema operativo debe seleccionar otro proceso para ser ejecutado.

   Si se soportan *hilos del núcleo*, el sistema operativo planifica estos hilos usando algún determinado algoritmo de planificación. El planificador puede tener en cuenta consideraciones relativas al rendimiento del sistema a la hora de planificar hilos del núcleo de igual prioridad.

   Por otra parte, en sistemas con una configuración del tipo menor número de hilos del núcleo que hilos de usuarios, aparte de las dos planificaciones comentadas, es necesario implementar con la biblioteca de hilos una planificación local para establecer qué hilo de usuario puede usar un hilo del núcleo.

Solución Ejercicio 2

El sistema operativo es el responsable de la creación de los procesos que se van a ejecutar. Las acciones que se deben realizar para crear un proceso dependen de cada sistema operativo, aunque de forma general se pueden distinguir las siguientes:

1. *Comprobar si el proceso puede ser creado*. El sistema operativo comprueba si existe suficiente espacio en memoria principal para crear otro proceso y si el usuario no ha excedido el número máximo de procesos que puede tener ejecutándose. Algunos sistemas operativos establecen un límite al número máximo de procesos que un usuario puede estar ejecutando para evitar exceder la capacidad de la tabla de procesos.

2. *Asignar una entrada de la tabla de procesos para el nuevo proceso*. En este instante se le asigna un identificador numérico al proceso.

3. *Reservar espacio en memoria para el proceso*. Se debe reservar espacio en memoria principal para el espacio de direcciones de memoria lógica del proceso y para la información de control del proceso.

4. *Inicializar el bloque de control del proceso*. Los campos de la entrada asociada al proceso en la tabla de procesos se inicializan con los valores adecuados.

5. *Establecer los enlaces apropiados*. El sistema operativo debe configurar los punteros adecuados para enlazar la información del proceso en las diferentes listas, tablas y colas que mantiene. Por ejemplo, el sistema operativo suele colocar al proceso creado en la cola de procesos preparados para que así puede ser planificado.

Solución Ejercicio 3

El *buffering* es una técnica implementada por el sistema operativo que consiste en almacenar temporalmente en buffers los datos que se transfieren entre un dispositivo y un proceso, o viceversa. Un *buffer* es un área de memoria principal a la que únicamente tiene acceso el sistema operativo, es decir, pertenece al espacio del núcleo. El subsistema de E/S es el encargado de asignar buffers para las operaciones de E/S. Por su parte, las rutinas de servicio de las interrupciones o los drivers, se encargan de transferir los datos entre los buffers y los dispositivos, o viceversa, dependiendo de si se trata de una operación de escritura o de lectura.

El buffering evita los problemas que plantea la transferencia directa de datos desde el dispositivo al espacio de usuario del proceso y que son los siguientes

- Obliga al sistema operativo a mantener cargada en un marco *j* de la memoria principal la página *i* mientras no se complete la operación de E/S, ya que en caso contrario se perderían los datos. Para evitar que la página sea reemplazada, el sistema operativo tiene que bloquear el marco. Las operaciones de E/S son lentas, en comparación con los accesos a memoria principal, por lo que el marco quedará bloqueado durante un tiempo importante. Cuanto mayor sea el número de marcos bloqueados, menor será el número de marcos candidatos para ser reemplazados al atender un fallo de página, lo que puede conducir a que se produzca sobrepaginación.

- Imposibilidad de poder intercambiar al proceso hasta que la operación de E/S no se haya completado.

Otra ventaja del buffering es que permite realizar durante una transferencia de datos una adaptación de velocidades entre el agente productor (proceso o dispositivo) y el agente consumidor (dispositivo o proceso). Asimismo permite lograr una adaptación entre dispositivos que tienen diferentes tamaños de unidad de transferencia de datos.

Solución Ejercicio 4

```
int Z=0;
semáforo_binario X, Y;

void cliente()
{
        wait_sem(X);
        Z = Z + 1;
        if (Z > 3){
          signal_sem(X);
          wait_sem(Y);
        }

        signal_sem(X);
        realizar_gestión();
        wait_sem(X);
        Z = Z - 1;
        signal_sem(Y);
        signal_sem(X);

}

main()
{
        init_sem(X,1);
        init_sem(Y,0);
        ejecución_concurrente(cliente,...,cliente);
}
```

Figura 1

Se observa que la solución que se propone en la Figura 1 utiliza las siguientes variables globales y semáforos binarios:

- `Z`. Variable global de tipo entero para llevar la cuenta del número de clientes en la cola o en la ventanilla.
- `X`. Semáforo binario que se utiliza para garantizar la exclusión mutua en el uso de la variable global `Z`.
- `Y`. Semáforo binario que se utiliza para sincronizar el acceso a las ventanillas de la oficina.

Analizando detenidamente la solución propuesta se observa que **no coordinada adecuadamente** la actividad de los clientes en la oficina ya que la operación `signal_sem(Y)` se realiza siempre independientemente de si existen o no clientes esperando. Nótese que si no existen clientes esperando en la cola la realización de esta operación pone el semáforo binario `Y` a 1, lo que implica que si las ventanillas están ocupadas y un cliente tiene que realizar la operación `wait_sem(Y)` para esperar en la cola del semáforo, no se bloqueará ya que el semáforo estará a 1 en vez de a 0.

En la Figura 2 se propone una solución válida.

```
int Z=0;
semáforo_binario X, Y;

void ciudadano()
{
        wait_sem(X);
        Z = Z + 1;
        if (Z > 3){
          signal_sem(X);
          wait_sem(Y);
        }
        else signal_sem(X);

        realizar_gestión();

        wait_sem(X);
        if (Z> 3) signal_sem(Y);
        Z = Z - 1;
        signal_sem(X);
}

main()
{
        init_sem(X,1);
        init_sem(Y,0);
        ejecución_concurrente(cliente,...,cliente);
}
```

Figura 2

Solución Ejercicio 5

La MMU dispone de un TLB en el cual se cargan un cierto número de entradas de la tabla de páginas del proceso en ejecución. Cuando se va a traducir una dirección virtual que referencia a una determinada página *i* se busca en el TLB la entrada *i* de la tabla de páginas. Si se encuentra se produce un acierto, en caso contrario se produce un fallo y hay que acceder a la memoria principal a leer dicha entrada de la tabla de páginas. Luego el tiempo medio de lectura t_l de la entrada de la tabla de páginas asociada a la página referenciada es:

$$t_l = h\,t_{ga} + (1-h)\,t_f$$

En la expresión anterior h es la tasa de aciertos del TLB, t_{ga} es el tiempo medio de gestión de un acierto en el TLB y t_f es el tiempo medio de gestión de un fallo en el TLB. Del enunciado se sabe que $h = 0{,}95$ y $t_{ga} = 0$. Además se puede considerar que $t_f = t_{am} = 125$ ns. Luego sustituyendo valores y operando se obtiene:

$$t_l = 0{,}95{\cdot}0 + 0{,}05{\cdot}125\,10^{-9} = 6{,}25\,10^{-9} = 6{,}25\,ns$$

Si se considera t_l, el tiempo medio de despacho de una referencia a memoria se calcula de la siguiente forma:

$$t_R = t_l + (1-p)\,t_{am} + p\,t_{gf}$$

Sustituyendo valores y operando se obtiene:

$$t_R = 6{,}25\,10^{-9} + 0{,}92{\cdot}125\,10^{-9} + 0{,}08{\cdot}40\,10^{-3} = 0{,}0032\,\text{s} = 3{,}2\,\text{ms}$$