SISTEMAS OPERATIVOS
Solución PED1 (Noviembre 2011)

Solución Ejercicio 1

I) El principal inconveniente de los hilos del núcleo radica en que su gestión contribuye a la sobrecarga del núcleo. Para resolver este problema, muchos sistemas limitan el número de hilos del núcleo que se pueden crear, además reciclan los hilos del núcleo ya existentes. En conclusión la afirmación es **FALSA**.

II) La realización del cambio de hilo a nivel de usuario dentro de un mismo proceso se realiza sin necesidad de realizar un cambio de modo y un cambio de contexto. En conclusión la afirmación es **FALSA**.

Solución Ejercicio 2

La solución que se propone en la Figura 1 para este problema utiliza las siguientes variables globales y semáforos binarios:

- `CS`. Variable global de tipo entero para llevar la cuenta de coches que desean salir.
- `CE`. Variable global de tipo entero para llevar la cuenta de coches que desean entrar.
- `sem1`. Semáforo binario que se utiliza para garantizar la exclusión mutua en el uso de la variable global `CS`.
- `sem2`. Semáforo binario que se utiliza para garantizar la exclusión mutua en el uso de la variable global `CE`.
- `sem3`. Semáforo binario que se utiliza para garantizar la exclusión mutua en el uso de la rampa por un grupo de coches que desean usarla en un cierto sentido. Un grupo puede estar formado por uno o varios coches. El primer coche de cada grupo debe realizar una operación `wait_sem(sem3)` para obtener el uso de la rampa y el último coche debe realizar una operación `signal_sem(sem3)` para liberarla.

Solución Ejercicio 3

Las características de los principales tipos de estructuras que puede presentar el núcleo de un sistema operativo son:

- *Estructura monolítica o simple*. Se caracteriza porque todos los subsistemas y las estructuras de datos del núcleo están ubicadas en un único módulo lógico, no existiendo interfaces bien definidos entre los subsistemas. El software del núcleo está escrito como un conjunto de procedimientos. Hay un procedimiento principal, un conjunto de procedimientos de servicio y un conjunto de procedimientos auxiliares. Típicamente el procedimiento principal invoca al procedimiento de servicio requerido por una llamada al sistema. A su vez el procedimiento de servicio invoca a los procedimientos auxiliares que necesita. Un procedimiento es visible por todos los demás.
- *Estructura en módulos* o *modular*. Se caracteriza por la existencia de varios módulos que pueden contener uno o varios subsistemas. Tanto los módulos como los subsistemas que contienen tienen una interfaz bien definida en términos de entradas, salidas y funciones que realizan. Además se pueden implementar como tipos de datos y objetos abstractos.

```
/* Definición variables y semáforos */
int CS=0, CE=0;
semáforo_binario sem1, sem2, sem3;

void cocheS() /* Proceso coche_que_sale */
{
        wait_sem(sem1);
        CS=CS + 1;

        if(CS == 1)
        {
          wait_sem(sem3); /* Comprobar que puede usar la rampa */
        }
        signal_sem(sem1);

        salir_del_garaje();

        wait_sem(sem1);
        CS = CS - 1;
        if (CS == 0)
        {
         signal_sem(sem3); /* Permitir el uso de la rampa a otro grupo de coches */
        }
        signal_sem(sem1);
}

void cocheE() /* Proceso coche_que_entra */
{
        wait_sem(sem2);
        CE=CE + 1;

        if(CE == 1)
        {
          wait_sem(sem3); /* Comprobar que puede usar la rampa */
        }
        signal_sem(sem2);

        entrar_al_garaje();

        wait_sem(sem2);
        CE = CE - 1;
        if (CE == 0)
        {
         signal_sem(sem3); /* Permitir el uso de la rampa a otro grupo de coches */
        }
        signal_sem(sem2);
}


void main() /*Inicialización de semáforos y ejecución concurrente*/
{
        init_sem(sem1,1);
        init_sem(sem2,1);
        init_sem(sem3,1);
        ejecución_concurrente(cocheS, cocheS,..., cocheE, cocheE,...);
}
```

Figura 1 – Solución Ejercicio 2

- *Estructura en capas* o *niveles*. Se caracteriza porque el núcleo estar organizado en una jerarquía de capas, cada una de las cuales subyace sobre la anterior. Cada capa *i* se implementa como un objeto abstracto que encapsula una serie de estructuras de datos y la implementación de las operaciones que pueden manipularlas. Dichas operaciones pueden ser invocadas por las capas de mayor nivel $i + 1$, $i + 2$,... Asimismo la capa *i* puede invocar a las operaciones de las capas de los niveles inferiores $i - 1$, $i - 2$,...

- *Estructura extensible*. Se puede considerar como un caso especial de la estructura modular. Se caracteriza por la existencia de un módulo esencial denominado *núcleo extensible* o *micronúcleo* y varios módulos accesorios denominados *extensiones del núcleo*. El micronúcleo se ejecuta en modo núcleo y se encarga de realizar únicamente los servicios absolutamente esenciales del sistema operativo. Los servicios menos esenciales del sistema operativo se implementan como extensiones del núcleo y se ejecutan en modo usuario.

Solución Ejercicio 4

a) En la Figura 2 se muestra el diagrama de uso del procesador si se utiliza el algoritmo SJF. Recuérdese que este algoritmo es no expropiativo y selecciona para ser ejecutado al proceso con tiempo de ejecución más corto.

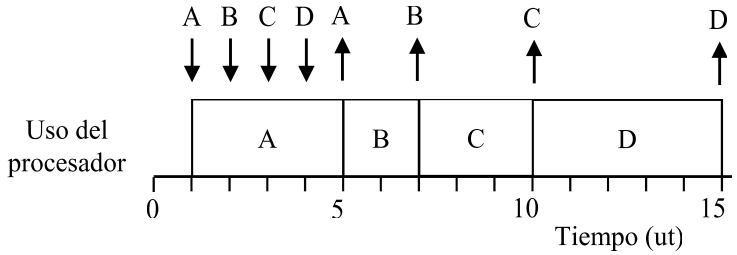

Figura 2 – Solución apartado a) Ejercicio 4

b) En la Figura 3 se muestra el diagrama de uso del procesador si se utiliza el algoritmo de turno rotatorio. Recuérdese que éste es un algoritmo expropiativo que asigna el uso del procesador a un proceso durante una cantidad fija de tiempo denominada cuanto. Finalizado el cuanto se genera una interrupción de reloj cuyo tratamiento produce que se expropie el uso del procesador al proceso en ejecución y se planifique para ejecución al primer proceso de la cola de procesos preparados para ejecución. El proceso expropiado se coloca al final de la cola de procesos preparados.

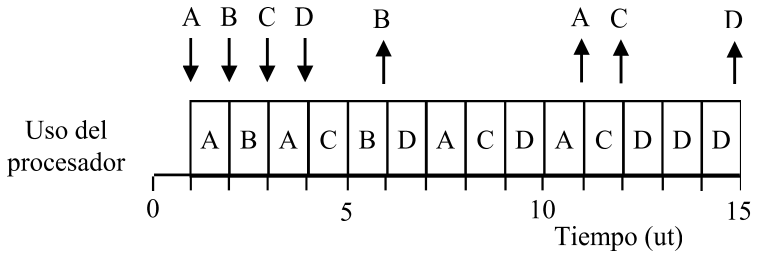

Figura 3 – Solución apartado b) Ejercicio 4

Nota: En la solución de este apartado se ha supuesto que si un proceso X entra en el estado preparado en el mismo instante en que finaliza el cuanto de un proceso Y, el proceso X se coloca en la cola de procesos preparados antes que el proceso Y. Otra solución válida sería considerar el caso contrario, es decir, que el proceso Y se coloque en la cola antes que el proceso X.

Solución Ejercicio 5

Para poder aplicar el algoritmo de Coffman, en primer lugar hay que calcular el vector de recursos disponibles $\mathbf{R_D}$. Del enunciado se deduce que el vector de recursos existentes $\mathbf{R_E}$ es:

$$\mathbf{R_E} = (\ 3 \quad 3 \quad 3\ )$$

De la matriz **A** se puede deducir el vector de recursos asignados $\mathbf{R_A}$, sumando los elementos de una misma columna:

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

El vector $\mathbf{R_D}$ se obtiene de la siguiente forma:

$$\mathbf{R_D} = \mathbf{R_E} - \mathbf{R_A} = (\ 3 \quad 3 \quad 3\ ) - (\ 2 \quad 3 \quad 2\ ) = (\ 1 \quad 0 \quad 1\ )$$

Los pasos del algoritmo son los siguientes:

1) Se examina la matriz **A** y se comprueba si alguna de sus filas es igual a (0 0 0). Como no existe ninguna, no se marca ningún proceso como completado.

2) Se realiza la siguiente asignación:

$$\mathbf{X} = \mathbf{R_D} = (\ 1 \quad 0 \quad 1\ )$$

3) Para cada proceso P_i no marcado como completado, en este caso los tres procesos, se comprueba la siguiente condición:

$$\mathbf{M}_i \leq \mathbf{X}$$

   3.1) La primera fila de **M** asociada al proceso P_1 no cumple la condición:

$$(\ 1 \quad 1 \quad 1\ ) \leq (\ 1 \quad 0 \quad 1\ )$$

   3.2) La segunda fila de **M** asociada al proceso P_2 no cumple la condición:

$$(\ 1 \quad 1 \quad 0\ ) \leq (\ 1 \quad 0 \quad 1\ )$$

   3.3) La tercera fila de **M** asociada al proceso P_3 no cumple la condición:

$$(\ 0 \quad 1 \quad 1\ ) \leq (\ 1 \quad 0 \quad 1\ )$$

4) Como no existe ningún proceso que cumpla la condición el algoritmo finaliza.

Se concluye que **existe interbloqueo** ya que los procesos P_1, P_2 y P_3 no han sido marcados.