SISTEMAS OPERATIVOS
Solución PED1 (Noviembre 2014)

Solución Ejercicio 1

I) Aparte de para garantizar la exclusión mutua, un monitor también puede utilizarse para la sincronización de procesos. Para ello dispone de un tipo especial de dato: las *variables de condición*. Cada variable de condición tiene asociada una cola de procesos bloqueados en espera de que se cumpla dicha condición. En conclusión la afirmación es **FALSA**.

II) La magnitud del *tiempo de conmutación entre procesos* depende principalmente de factores asociados al hardware del computador como por ejemplo, la velocidad de lectura/escritura de la memoria principal, el número de registros del procesador cuyo contenido hay que salvar o cargar, la velocidad del procesador y la existencia de instrucciones especiales en el repertorio del procesador para salvar o cargar todos los registros. En conclusión la afirmación es **VERDADERA**.

III) Esta afirmación es **VERDADERA** ya que el algoritmo SJF favorece a los procesos cortos frente a los procesos largos, el tiempo de espera de los procesos cortos disminuye más de lo que se aumenta el tiempo de espera de los procesos largos. Por lo tanto, el tiempo de espera promedio se decrementa.

IV) Esta afirmación es **VERDADERA** ya que cuando un programa de usuario invoca a una llamada al sistema se produce una *trampa* que provoca la conmutación hardware de modo usuario a modo supervisor y transfiere el control al núcleo.

Solución Ejercicio 2

El *tiempo de estancia* o *tiempo de retorno* T_{ri} de un trabajo *i* es el tiempo que transcurre desde que se lanza hasta que finaliza, se calcula como la suma del tiempo de ejecución o servicio T_{si} y del tiempo de espera T_{ei} de dicho trabajo.

Por otra parte, el tiempo de estancia promedio de N trabajos es:

$$\bar{T}_r = \frac{T_{r1} + T_{r2} + \ldots + T_{rN}}{N}$$

Sustituyendo la definición de tiempo de retorno se obtiene:

$$\bar{T}_r = \frac{(T_{s1} + T_{e1}) + (T_{s2} + T_{e2}) + \ldots + (T_{sN} + T_{eN})}{N}$$

En el enunciado se proporcionan los siguientes datos:

- N = 3 trabajos
- $\bar{T}_r$ = 20 ut
- T_{s1} = x + 5 ut y T_{e1} = 5 ut
- T_{s2} = x + 13 ut y T_{e2} = 5 ut
- T_{s3} = x + 4 ut y T_{e3} = T_{s3} ut

Sustituyendo estos datos en la expresión del tiempo de retorno medio se obtiene:

$$\frac{(x + 5 + 5) + (x + 13 + 5) + (x + 4 + x + 4)}{3} = 20$$

Finalmente, despejando x de la expresión anterior y operando se obtiene: **x=6 ut**

Solución Ejercicio 3

En la Figura 1 se muestra una posible solución para este ejercicio. Esta solución utiliza las siguientes variables globales y variables de condición:

- `contadorP`. Variable global de tipo entero para llevar la cuenta del número de puestos en el plato ocupados.
- `contadorC`. Variable global de tipo entero para indicar si el columpio está ocupado.
- `puesto_plato_disponible`. Variable de condición para bloquear en su cola a los procesos hasta que exista algún puesto en el plato disponible.
- `columpio_disponible`. Variable de condición para bloquear en su cola a los procesos hasta que el columpio esté disponible.

Señalar que las acciones de `comer()` y `columpiarse()` deben invocarse (tanto en ésta como en cualquier otra solución) dentro del código del proceso `canario()` para garantizar la máxima concurrencia de procesos. Si se invocaran dentro de procedimientos del monitor, como en un monitor solo se puede ejecutar un procedimiento a la vez, se tendría la situación de que no podrían comer tres canarios y columpiarse otro simultáneamente, ya que mientras un canario come no podría comer ni columpiarse ningún otro, o mientras un canario se columpia no podrían comer los demás.

```
#define N 3 /* Número de  puestos en el plato */
monitor jaula /* Definición del monitor */
    condición puesto_plato_disponible, columpio_disponible;
    int contadorP, contadorC;

    void obtener_puesto_en_plato()  /* Procedimiento del monitor */
    {
      if (contadorP == N) wait_mon(puesto_plato_disponible);
      contadorP=contadorP+1;
    }

    void dejar_puesto_plato() /* Procedimiento del monitor */
    {
      contadorP = contadorP - 1;
      signal_mon(puesto_plato_disponible);
    }

    void obtener_columpio() /* Procedimiento del monitor */
    {
      if (contadorC == 1) wait_mon(columpio_disponible);
      contadorC=contadorC+1;
    }

    void dejar_columpio() /* Procedimiento del monitor */
    {
      contadorC = contadorC - 1;
      signal_mon(columpio_disponible);
    }

    { /* Inicialización del monitor */
      contadorP=0, contadorC=0;
    }
end monitor

void canario() /* Proceso canario */
{
    jaula.obtener_puesto_plato();
    comer();
    jaula.dejar_puesto_plato();

    jaula.obtener_columpio();
    columpiarse();
    jaula.dejar_columpio();
}

main() /* Ejecución concurrente */
{
    ejecución_concurrente(canario,...,canario);
}
```

Figura 1