PED-1

Curso 2014-2015

Sistemas Operativos

Juan Carlos Vírseda Monforte
31336073-E C.A. Cádiz

Sistemas Operativos

1ª Pregunta

I) La primera es **FALSA,** porque los monitores no solo garantizan la exclusión mutua, sino que también proporcionan herramientas para la sincronización de procesos, disponiendo de un tipo especial de datos llamadas variables de condición.

II) Es **VERDADERA**, Ya que la magnitud del tiempo de conmutación de los cambios de procesos no es inmediata sino que consume recursos de hardware por lo que están asociados a la velocidad de lectura/escritura de la RAM, número de registros del procesador, velocidad del procesador, etc…

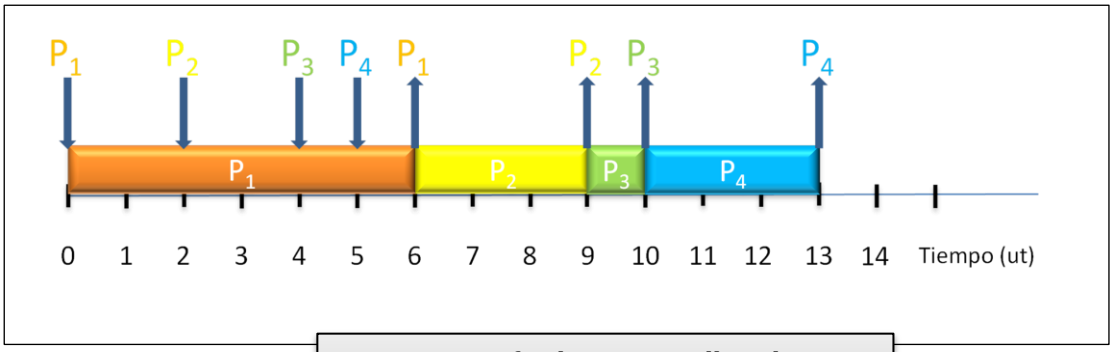
III) La tercera es **VERDADERA**. El algoritmo de planificación SJF minimiza el tiempo de espera de un conjunto dado de procesos ya que favorece (selecciona) los procesos más cortos frente a los más largos por lo que el tiempo de espera de los procesos cortos disminuye más de lo que se aumenta el tiempo de espera de los procesos largos por lo que el tiempo medio de espera se reduce.

Un ejemplo ilustra mejor esa reducción de tiempo.

Si tenemos 4 procesos con tiempos de llegada y tiempos de servicios como los de la tabla siguiente:

Proceso	Tiempo de llegada	Tiempo de Servicio
P_1	0	6
P_2	2	3
P_3	4	1
P_4	5	3

Usando un algoritmo FCFS(first-come first-served) obtenemos:



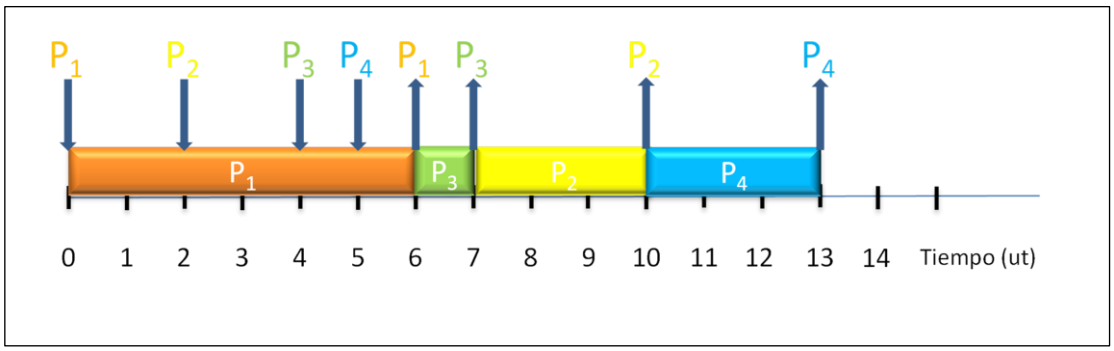
T.retorno = T.finalizacion – T.llegada

T.espera = T.retorno – T.servicio

eso	$T_{\cdot llegada}$	$T_{\cdot finalizacion}$	$T_{\cdot retorno}$	$T_{\cdot servicio}$	$T_{\cdot espera}$
P_1	0	6	6	6	0
P_2	2	9	7	3	4
P_3	4	10	6	1	5
P_4	5	13	8	3	5
Medias	2.75	9.5	**6.75**	3.25	**3.5**

Podemos observar que los tiempos medios de retorno y de espera son 6.75(ut) y 3.5(ut) respectivamente.

Usando el algoritmo SJF(shortest job first) obtenemos:



eso	T.llegada	T.finalizacion	T.retorno	T.servicio	T.espera
P_1	0	6	6	6	0
P_2	2	10	8	3	5
P_3	4	7	3	1	2
P_4	5	13	8	3	5
Medias	2.75	9.5	**6.25**	3.25	**3**

En este caso observamos que para el conjunto de procesos el tiempo medio de retorno (6,25ut) ha disminuido con respecto al del algoritmo FCFS al igual ocurre con el Tiempo de espera medio que es sensiblemente menor (3ut) en concreto un 14,29% menor.

IV) VERDADERO. Estas llamadas al sistema se producen cuando un programa usuario invoca una llamada al sistema (denominada trampa) lo que provoca la conmutación hardware de modo usuario a modo supervisor (o modo nucleo) transfiriendo el control al nucleo.

El nucleo examina los parámetros de la llamada atendiendo el servicio requerido, cuando dicha rutina termina almacena el resultado en algún registro y vuelve a provocar la conmutación hardware de modo supervisor a modo usuario para que el proceso que invocó la llamada continúe su ejecución.

2ª Pregunta

Datos del problema:

eso	$T_{\cdot retorno}$	$T_{\cdot servicio}$	$T_{\cdot espera}$
T_1		X+5	5
T_2		3+x+10	5
T_3	2y	4+x=y	Y
Medias	20		

T.espera = T.retorno – T.servicio

Luego : **T.retorno = T.espera + T.servicio** siguiendo esta formula obtenemos la siguiente tabla:

eso	$T_{\cdot retorno}$	$T_{\cdot servicio}$	$T_{\cdot espera}$
T_1	X+10	X+5	5
T_2	X+18	3+X+10	5
T_3	2X+8	4+X=Y	Y
Medias	20		

T.retorno = T.espera + T.servicio

Se nos pide que calculemos **X**. Si **T.retorno medio = $(T_{r1}+T_{r2}+T_{r3})/3$** sustituimos y obtenemos: 20 = ((X+10)+(X+18)+(2X+8))/3

Esto es igual a: 20*3= 36 + 4x ; 60-36 = 4x ; 24/4=X **Donde x = 6**

3ª Pregunta

```
#define  TRUE 1

#define  N 3 /* Capacidad del buffer del comedero */

monitor  jaula
        condición comedero_lleno, columpio_ocupado;
        int contador;
        char buffer[N];
        bool columpio_lleno;
        void comer(int canario) /* Declaracion del procedimiento comer
        del monitor*/
        {
                Contador = contador + 1;
                if (contador ==N) wait_mon(comedero_lleno);
                canario_come();
                contador = contador – 1;
                signal_mon(comedero_lleno);
         }
         void columpiar(int canario) /* Declaracion del procedimiento
         columpiar del monitor*/
         {
                if (columpio_lleno  == TRUE) wait_mon(columpio_ocupado);
                canario_columpia();
                columpio_lleno = FALSE;
                signal_mon(columpio_ocupado);
         }
         {
         /* Inicializacion del monitor */
                  contador=0;
                  columpio_lleno=FALSE;
         }
end monitor

void vidadelcanario(int canario)
{
        while (TRUE)
```

```
        {
                jaula.comer(canario);
                jaula.columpiar(canario);
        }
}
void main()
{
        ejecución_concurrente(vidadelcanario);
}
```