Sistemas operativos

Primera prueba de Evaluación a Distancia (PED1)

Curso 2016-2017

Centro asociado de Cádiz

Juan Manuel Merino López

31657846-W

Solución ejercicio 1

I) **Falsa**. El bloqueo de interrupciones, siendo la solución más simple para garantizar la exclusión mutua con apoyo del hardware presenta, sin embargo, serios inconvenientes. Primero, el rendimiento del sistema se puede degradar bastante al no permitirle atender las interrupciones más prioritarias en el momento en que llegan. Además, se deja en manos del proceso la decisión de ceder el procesador y si este nunca lo devuelve el sistema se quedará colgado. Por otra parte, no sirve para el caso de sistemas multiprocesadores.

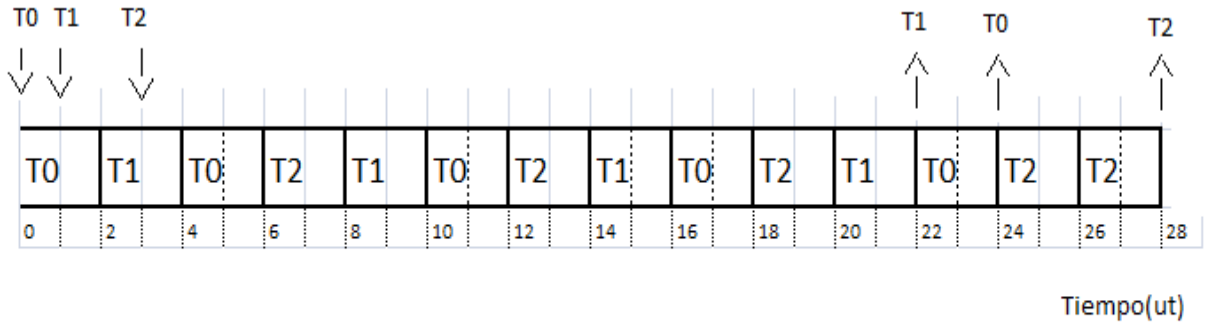
II) **Falsa**. La magnitud del tiempo de conmutación de un cambio de proceso depende principalmente de factores asociados al hardware del computador, como por ejemplo, la velocidad de lectura/escritura de memoria principal, el número de registros del procesador y la existencia de instrucciones especiales en el repertorio del procesador para salvar o cargar todos los registros, por lo tanto el tiempo de conmutación de un cambio de proceso no es independiente del hardware.

III) **Falsa**. Esta técnica consiste en mantener en memoria principal varios trabajos simultáneamente. Esto requiere que el Sistema Operativo sea capaz de gestionar la memoria principal y la memoria secundaria, y disponga de un algoritmo de planificación de trabajos. Si un trabajo que se está ejecutando actualmente en el procesador requiere la realización de una operación de E/S, mientras se completa dicha operación el procesador puede ejecutar otro trabajo. De esta forma se consigue aumentar el rendimiento del procesador. Por lo tanto, esta técnica no viene definida por el número de procesadores existente en el computador.

IV) **Falsa**. Este algoritmo puede ser expropiativo o no expropiativo. En su implementación expropiativa, si llega a la cola de procesos preparados un proceso B con una prioridad mayor que la del proceso A actual, entonces se interrumpe la ejecución de A y se planifica al proceso B. Por el contrario en la no expropiativa la planificación de B sólo se realiza cuando A se bloquea o finaliza.

Solución ejercicio 2

a) Diagrama de uso de la CPU.



Se debe tener en cuenta el turno rotatorio de los trabajos con un quantum de 2 ut. En principio se planifica T0, ya que es el primero y no hay ninguno en cola, en t=1 ut, entra la primera ráfaga de T1, cuando T0 termina su quantum, éste entra en cola y se planifica el trabajo T1, que estaba en la cola de procesos preparados. En t=3 ut, entra T2, que pasa a la cola detrás de T0. Cuando T1 acaba su quantum, se planifica T0, que esta primero en la cola, cuando éste termina su quantum, se planifica el siguiente de la cola, en t=6 ut, que es T2. En este punto, T2, T1 y T0, se van alternando cada 2 ut. En t=26 ut, sólo queda T2, no hay ningún proceso en cola, así que continua hasta que finaliza.

Además, se ha tenido en cuenta que, si un trabajo del que se está ejecutando una ráfaga, termina ésta y no ha finalizado su quantum, continua con la siguiente ráfaga de éste mismo trabajo, ya que en el enunciado no se especifica si tras cada ráfaga, cada trabajo debe bloquearse a la espera de algún evento como por ejemplo una operación de E/S. En éste caso, el uso del procesador por cada trabajo sería distinto, ya que al finalizar cada ráfaga debe entrar el planificador sin esperar a que finalice su quantum.

b) Tiempos de espera y finalización de cada proceso

Para el trabajo T0, la finalización de sus 5 ráfagas de duración 3, 2, 2, 2 y 1 ut, ocurre en 5, 11, 17, 23 y 24 ut respectivamente, están marcadas con puntos discontinuos en el diagrama.

Para el trabajo T1, la finalización de sus 4 ráfagas de duración 2, 3, 1 y 2 ut, ocurre en 4, 15, 16 y 22 ut respectivamente, marcadas en el diagrama con puntos discontinuos.

Para el trabajo T2, la finalización de sus 3 ráfagas de duración 8, 1 y 1 ut, ocurre en 26, 27 y 28 ut respectivamente, puntos marcados en el diagrama con puntos discontinuos.

Así, los tiempos de finalización (TF) son 24, 22 y 28 ut, para T0, T1 y T2 respectivamente. En la siguiente tabla se muestran además, los tiempos de retorno (TR), tiempos de ejecución (TS) y tiempos de espera (TE), que son 14, 13 y 15 ut, para T0, T1 y T2 respectivamente. Sabiendo que:

$$TR = TF - T\,LL \quad \text{y que} \quad TE = TR - TS$$

	TLL	TF	TR	TS	TE
T0	**0**	**24**	**24**	**10**	**14**
T1	**1**	**22**	**21**	**8**	**13**
T2	**3**	**28**	**25**	**10**	**15**

c) Tiempos medios de respuesta de cada trabajo.

Para **T0**, los tiempos que cada ráfaga está en espera hasta que finalizan son 5, 6, 6, 6 y 1 respectivamente, como se puede ver en el diagrama de uso del procesador, así que la media es **4,8 ut**.

Para **T1**, son 2, 7, 1 y 2 respectivamente, así el tiempo medio de respuesta es **3 ut**.

Para **T2**, son 20, 1 y 1 respectivamente, así su tiempo medio de respuesta es **7,33 ut**.

Solución ejercicio 3

Las variables globales a utilizar van a ser tres, que se corresponden con la cantidad de piezas que deben fabricar los robots R1 y R2 y las que va fabricando el robot R3. Son:

int numero T1, numero_T2, numero_T3;

Los semáforos binarios que se van a utilizar serán 4:

- R1, semáforo binario que se utiliza para garantizar la exclusión mutua en el uso de la variable global numero_T1.
- R2, semáforo binario que se utiliza para garantizar la exclusión mutua en el uso de la variable global numero_T2.
- R3, semáforo binario que se utiliza para garantizar la exclusión mutua en el uso de las variables globales numero_T1, numero_T2 y numero_R3.
- mutex, semáforo binario que sincroniza el proceso entre los robots R1, R2 y R3. Al estar inicializado a 1, cuando entra en uno de los procesos, excluye a los demás, hasta que lo libera. Y continua así hasta que se van cumpliendo las condiciones en cada proceso.

```
/* Definición de variables globales y semáforos */
# define TRUE 1
semáforo_binario R1, R2, R3, mutex;
int  numero_T1 = 0;
int  numero_T2 = 0;
int  numero_T3 = 0;

void proceso_R1( ) {            /* Proceso del robot R1 que fabrica las piezas T1 */
     while (TRUE) {
       wait_sem(mutex);
       wait_sem(R1);
      if (numero_T1 < 2)  {
            fabricar_pieza_T1( );
            numero_T1 = numero_T1 + 1;
            signal_sem(R1);
            signal_sem(mutex);
     } else {
```

```
            signal_sem(R1);
            signal_sem(mutex);
    }
  }
}
void proceso_R2( ) {                /* Proceso del robot R2 que fabrica las piezas T2 */
     while (TRUE) {
       wait_sem(mutex);
       wait_sem(R2);
      if (numero_T1 < 2)  {
         fabricar_pieza_T2( );
         numero_T2 = numero_T2 + 1;
            signal_sem(R2);
            signal_sem(mutex);
     } else {
             signal_sem(R2);
             signal_sem(mutex);
    }
  }
}
void proceso_R3( )  {            /* Proceso del robot R3 que fabrica las piezas T3, estando
    while (TRUE) {                              fabricadas dos de T1 y dos de T2   */
          wait_sem(mutex);
          wait_sem(R3);
    if ( (numero_T1=2) && (numero_T2=2))  {
          fabricar_pieza_T3( );
          numero_T1 = 0;
          numero_T2 = 0;
          numero_T3 = numero_T3 + 1;    /* va acumulando cada  pieza  T3  que va fabricando */
          signal_sem(R3);
          signal_sem(mutex);
} else {
          signal_sem(R3);
          signal_sem(mutex);
    }
  }
```

```
}
void main ( )    /* Inicialización se semáforos y ejecución concurrente */
{
     init_sem(R1, 1);
     init_sem(R2, 1);
     init_sem(R3, 1);
     init_sem(mutex, 1);
     ejecución_concurrente (proceso_R1, proceso_R2, proceso_R3);
}
```
