Ingeniería en Sistemas Computacionales

Fundamentos de Sistemas Operativos

Actividad 10

IS727272 - Marco Ricardo Cordero Hernández

IS699252 – Jorge Rodríguez Guatemala

Jal., 08 de junio de 2023

1.- Considere un programa concurrente con dos procesos P y Q, definidos a continuación. A, B, C, D y E son sentencias arbitrarias atómicas (indivisibles). Supóngase que el programa principal (no mostrado) ejecuta concurrentemente los procesos P y Q

```
P()
{
    A;
    B;
    C;
}
```

```
Q()
{
    D;
    E;
}
```

Indicar todas las posibles intercalaciones de los dos procesos anteriores (indicarlo por medio de trazas de ejecución dadas en términos de sentencias atómicas).

R: 10 posibles intercalaciones

1. ABCDE
2. ABDCE
3. DABCE
4. DABCE
5. ABDEC
6. ADBEC
7. DABEC
8. ADEBC
9. DAEBC
10. DEABC

2.- Considere la siguiente función llamada por un pthread_create donde dos hilos incrementan una variable global.

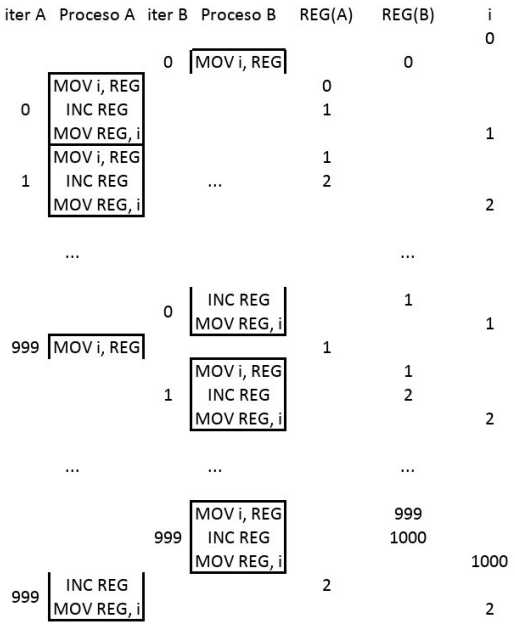
```
int cuenta=0;

void total()
{
    int i;
    for(i=0;i<1000;i++)
       cuenta++;
}
```

Determinar los valores mínimos y máximos que puede tomar la variable cuenta. Suponga que incrementar una variable en memoria requiere 3 operaciones

1. Carga la variable en un registro
2. Incrementa el registro
3. Carga el registro en la variable

R: Para el valor máximo se supone una ejecución normal/esperada o alguna combinación que resulte en el valor supuesto desde un primer análisis superficial. Este valor sería ***2000***, puesto que se tiene un loop desde 0 hasta 1999, donde se aumenta una variable contenedora en cada iteración; esta acción se repite dos veces, por ende, el resultado es el mencionado.

Para el valor mínimo existe un caso especial en el que para 2 threads, el valor de la variable siempre será ***2*** para el ejercicio actual. El razonamiento detrás de esto se explica en la siguiente tabla:



Como se puede apreciar, primero inicia cualquiera de los dos procesos y ejecuta la primera de las tres instrucciones, cargando el valor inicial del contador (0). Después, el otro proceso realiza todas sus iteraciones - 1, para que finalmente concluya si primera iteración el proceso inicial, sustituyendo todo el progreso que el segundo proceso habría realizado; esto sucede justo antes de que la última iteración del otro proceso inicie. Después de lo anterior, sucede el mismo caso pero con el segundo proceso que inició, es decir, en su última iteración ejecuta la primera de las tres instrucciones, y, como se queda con el valor aumentado del primer proceso (1), ahora este ya no tendrá el resultado de sus iteraciones anteriores. Luego, el primer proceso continua con todas sus iteraciones restantes y las finaliza, habiendo ahora un valor en el contador de 1000. Cuando lo

anterior sucede, el segundo proceso finalmente podrá continuar con las instrucciones de su última iteración, aumentado el valor que tenía de 1 a 2, y, por ende, concluyendo la ejecución con un valor de únicamente ***2*** en el contador que se esperaba que tuviera ***2000***.

3.- El código que se muestra en la Figura 1 es el intento donde dos procesos P(0) y P(1) se sincronizan a través de un algoritmo por software empleando variables en memoria compartida.

Haga un análisis de la ejecución concurrente de los dos procesos en una tabla donde se muestra la ejecución intercalada de instrucciones de ambos procesos, y muestre si esta solución funciona o no funciona, en caso de no funcionar explique el por qué.

```
boolean flag[2]={false,false};
int turno=0;

P(i)
{
  j=1-i;
  while(forever)
  {
      turno=j;
      while(flag[j] && turno!=i){};
      flag[i]=true;
      CS
      flag[i]=false;
      RS
  }
}
```

Figura 1. Código que ejecutan P(0) y P(1)

R: Mediante el siguiente análisis

Proceso (0)	Proceso (1)	flag[0]	flag[1]	turno	resultado de la expresión lógica
i = 0, j = 1	i = 1, j = 0				
		FALSE	FALSE	0	
turno = j(1)				1	
	turno = j(0)			0	
while (flag[j] && turno != i) {}					FALSO
	while (flag[j] && turno != i) {}				FALSO
flag[i] = true					
	flag[i] = true				
CS					
	CS				

Se puede concluir que esta solución al problema de la concurrencia ***no funciona***, ya que, similar a otras soluciones por software que han sido revisadas, ambos procesos acceden a su sección crítica al mismo tiempo, violando el requisito de exclusión mútua para la resolución del mismo problema.

4.- ¿Qué aprendiste?

R: Se vieron los problemas de concurrencia en gran medida y el problema y peligro que suponen para la ejecución de programas en paralelo, así como la importancia de restringir y manejar adecuaamente la sección crítica de los procesos.

Adicional a ello, se pudieron ver los análisis de ejecuciones de procesos hipotéticos y los resultados que sus ejecuciones concurrentes pudieran arrojar, así como su correctud y el porqué de sus fallos.

Finalmente, se conoció y aprendió acerca de un par de técnicas de análisis de procesos y métodos por software para resolver los problemas de concurrencia, algo que, aunque resultó desfavorable en las ejecuciones, sin duda alguna sirve para comprender los fundamentos de la operación detraś de procesos reales que posiblemente se verán más adelante.