Ingeniería en Sistemas Computacionales

Fundamentos de Sistemas Operativos

Actividad 11

IS730547 – Santiago Cordova Berrelleza

IS727272 - Marco Ricardo Cordero Hernández

Jal., 12 de junio de 2023

1.- Considere el siguiente programa que 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 que este intento no funciona, explique en dónde falla

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

void P(int i)
{
    int j=1-i;

    while(TRUE)
    {
        flag[i]=TRUE;

        while(turno!=i)
        {
            while(flag[j])
            {
                turno=i;
            }
        }
        /* Sección crítica */

        flag[i]=falso;

        /* Resto */
    }
}
```

Análisis

P(0)	P(1)	flag[0]	flag[1]	turno
		FALSE	FALSE	0
flag[0] = True		TRUE		
	flag[1] = True		TRUE	
while(turno!=0)				0
	while(turno!=1)			0

CS				
	while(flag[0])		TRUE	
flag[0] = False		FALSE		
	turno = 1			1
flag[0] = True		TRUE		
	while(flag[0])	TRUE		
while(turno!=0)				1
	turno = 1			1
while(flag[1])			TRUE	
	while(flag[0])	TRUE		
turno = 0				0
	turno = 1			1
while(flag[1])				
	while(flag[0])			

Este intento falla debido a que ambos procesos se quedarán en un while infinito al estar revisando siempre las banderas correspondientes en un momento indeterminado; al no cambiar las banderas, la sincronización esperada no puede ser llevada a cabo. ■

2.- Considera el algoritmo de Dekker con la siguiente modificación con la intención de que funcione con 3 procesos. Mostrar que esta solución NO FUNCIONA, haciendo una tabla que muestre la secuencia de los 3 procesos concurrentes y los valores de las variables. El que no funcione correctamente no necesariamente es que dos o más entren a la sección crítica. Considera posibilidades como de que uno de los tres no inicie y los otros dos sí.

```
int flag[3]={FALSE,FALSE,FALSE};
int turno=0;

void P(int i)
{
    int j=(i+1)%3;
    int k=(i+2)%3;

    while(1)
```

```
    {
        flag[i]=true;
        while(flag[j] || flag[k])
        {
             flag[i]=false;
             while(turno!=i);
             flag[i]=true;
        }

        CS

        turno=j;
        flag[i]=false;

        RS
    }
}
```

Análisis (caso donde P1 no inicia)

P(0); j1; k2	P(1); j2; k0	P(2); j0; k1	flag[0]	flag[1]	flag[2]	turno
			FALSE	FALSE	FALSE	0
flag[0] = true			TRUE			
		flag[2] = true			TRUE	
while(flag[1] || flag[2])				FALSE	TRUE	
		while(flag[0] || flag[1])	TRUE	FALSE		
flag[0] = false			FALSE			
		flag[2] = false			FALSE	
while(turno!=0);						0
		while(turno!=2);				0
flag[0] = true			TRUE			
		while(turno!=2);				0
while(flag[1] || flag[2])				FALSE	FALSE	
		while(turno!=2);				0

CS						
		while(turno!=2);				0
turno = 1						1
		while(turno!=2);				1
flag[0] = false			FALSE			
		while(turno!=2);				
flag[0] = true			TRUE			
while(flag[1] || flag[2])						

Al no iniciar el proceso 1 en un tiempo adecuado, el proceso 2 sufrirá inanición al estar pendiente su ejecución. De estar forma, el requisito para la resolución del problema de la concurrencia para dos procesos que habla de la espera ilimitada no se estaría cumpliendo. ■

3.- Considera el algoritmo de Peterson con la siguiente modificación con la intención de que funcione con 3 procesos. Mostrar que esta solución NO FUNCIONA, haciendo una tabla que muestre la secuencia de los 3 procesos concurrentes y los valores de las variables. Considera posibilidades como de que uno de los tres no inicie y los otros dos sí.

```
int flag[3]={FALSE,FALSE,FALSE};
int turno=0;

void P(int i)
{
     int j=(i+1)%3;
     int k=(i+2)%3;

     while(1)
     {
          flag[i]=true;
           // Quiero entrar
          turno=j;
           // pero dejo a otro entrar
          while((flag[j] || flag[k])&&turno==j){};

           CS

          flag[i]=false;
           // Ya no quiero entrar
```

```
                RS
        }
}
```

Análisis

i=0, j=1, k=2	i=1, j=2, k=0	i=2, j=0, k=1				
PROCESO 0	PROCESO 1	PROCESO 2	FLAG[0]	FLAG[1]	FLAG[2]	TURNO
			False	False	False	0
flag[0]=TRUE;			True			
	flag[1]=TRUE;			True		
		flag[2]=TRUE;			True	
turno=j;						1
	turno=j;					2
		turno=j;				0
while((flag[j] || flag[k])&&turno==j){};			True	True	True	0
	while((flag[j] || flag[k])&&turno==j){};		True	True	True	0
		while((flag[j] || flag[k])&&turno==j){};	True	True	True	0
CS						
	CS					

Cuando los tres procesos realizan su ejecución intercalada, al menos dos de ellos estarían entrando a su sección crítica, por ende, este intento es incorrecto. ■

4.- ¿Qué aprendiste?

A través de la realización de estos ejercicios, se ha podido aprender acerca del análisis de intentos para la resolución del problema de la concurrencia, de forma que ahora no es posible reconocer cuando un algoritmo es correcto o incorrecto según los requerimientos establecidos para el mismo contexto. También, se ha adquirido el conocimiento de una técnica útil para el análisis visual de

lo que se intenta demostrar, haciendo este procesos mucho más sencillo e incluso eficiente para señalar cuando los algoritmos funcionarán o no.