Práctica 4
Programación Concurrente y de Tiempo Real
Universidad de Cádiz

Alejandro Serrano Fernández

October 20, 2020

1 Tercer Intento

En este intento, se inicializan las variables C1 y C2 a true para que indicar que los procesos desean entrar en la región crítica. Después comprueba si el otro proceso no quiere entrar en su sección región crítica. Si lo desea, será necesario esperar, en caso contrario, entrará en la sección crítica. Al salir de ella, el proceso establecerá su variable a false, permitiendo al otro proceso que entre en su sección crítica.

Observamos que en este intento satisface el requerimiento de la exclusión mutua.

```
Proceso 1:{
for(int i=0; i<nVueltas; i++){
        C1 = true;
        while(C2==true);
        n++;
        C1 = false;
     }
 break;
}

Proceso 2: {
for(int i=0; i<nVueltas;i++){
        C2 = true;
        while(C1==true);
        n--;
        C2 = false;
    }
}
     break;
```

Sin embargo si P0 y P1 realizan a la misma vez la primera asignación, nos encontramos en un bucle infinito, concretamente en la instrucción while de ambos procesos.

2 Cuarto Intento

Las variables C1 y C2 se incializan a true, para indicar que cualquiera de los dos procesos tienen el desea de entrar en su sección crítica. Cuando uno de los dos procesos comprueba que el otro proceso también lo desea, cede su turno , espera un tiempo, y luego reclama su derecho a entrar a la región crítica.

```
case 1:{
     for(int i=0; i<nVueltas; i++){
        C1 = true;
        while(C2==true)
        {
         C1 = false;
         C1 = true;
        }
        n++;
        C1 = false;
      }
      break;
}

case 2: {
   for(int i=0; i<nVueltas;i++){
         C2 = true;
         while(C1==true)
         {
            C2 = false;
            C2 = true;
         }
         n--;
         C2 = false;
   }
}break;
```

Aunque los procesos puedan quedarse de forma indefinida cediéndose el paso

mutuamente, no se llega a producir espera ilimitada. Este algoritmo, aunque nos devuelva el resultado esperado, no cumple con las propiedades de exactitud.

3 Algoritmo de Dekker

Para este ejercicio, he implementado el algoritmo de Dekker de tal manera que a través de dos procesos vamos modificando la variable n. Un proceso se dedica a incrementarla y otro se dedica a decrementarla, de tal manera que el resultado teórico final es 0. En la ejecución del programa, tras varias ejecuciones, verificamos que la cumple.

4 Algoritmo de Lamport

Este algoritmo es conocido como el algoritmo de la panadería. Podemos observar que para este algoritmo se cumple/verifica la exclusión mutua cuando un proceso está en su sección crítica, evitando así que otros pasen a su sección crítica e interfiera en la solución final. Cabe destacar que se garantiza que los procesos se ejecutan de forma lineal(P0,P1...). En mi caso, para comprobar que se cumple la condición de exclusión mutua, he utilizado la misma condición que los ejercicios anteriores (incrementos de 10000 y decrementos de 10000), y el resultado obtenido tras varias ejecuciones, sigue siendo el mismo, es decir, 0.

5 Algoritmo de Hyman

Cuando ejecutamos ambos procesos, podemos observar que ambos entran simultáneamente en la sección crítica, de tal manera que acaba bloqueando el programa, pues ninguno puede avanzar en su ejecución.
Este algoritmo se presentó como una solución a la exclusión mutua, un desafío para los investigadores desde los años 60, aunque solo se presentó con intereses educacionales, ya que ayuda a clarificar las cualidades de una solución correcta.