



CONCURRENCIA

Espera Activa

Guillermo Román Díez
`groman@fi.upm.es`

Universidad Politécnica de Madrid

Curso 2016-2017

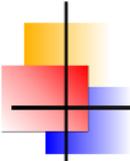
Espera Activa (busy-waiting)

“La espera activa es una técnica en la cual un proceso comprueba continuamente si una condición se cumple o si se produce un evento”

Espera Activa (busy-waiting)

“La espera activa es una técnica en la cual un proceso comprueba continuamente si una condición se cumple o si se produce un evento”





ESPERA ACTIVA (BUSY-WAITING)

- ▶ Se puede hacer la comprobación sobre el valor de una variable o sobre si se produce algún tipo de evento (p.e. pulsación de teclado, interrupción, ...)
- ▶ Se hace la comprobación *contraria a la esperada* en un bucle hasta que se cumpla la condición que estamos buscando
- ▶ Por ejemplo, podríamos usar una variable turno...

```
class Incrementador extends Thread {  
    public void run() {  
        //...  
        while(turno != INC) {}  
        // Ya se cumple turno == INC!!  
    }  
}
```

- ▶ Vamos a ver algunas posibles soluciones y los problemas que plantean



PROPUESTA 1

- ▶ Una forma de conseguir la exclusión mutua es la comprobación de una variable turno (INC o DEC) y ceder el turno al salir de la sección crítica
 - ▶ Puede hacerse con dos o más procesos

```
// Incrementador
public void run() {
    //...
    while(turno != INC) {}
    // turno == INC!!

    // SC

    turno = DEC;
    //...
}
```

```
// Decrementador
public void run() {
    //...
    while(turno != DEC) {}
    // turno == DEC!!

    // SC

    turno = INC;
    //...
}
```



PROPUESTA 1

- ▶ Una forma de conseguir la exclusión mutua es la comprobación de una variable turno (INC o DEC) y ceder el turno al salir de la sección crítica
 - ▶ Puede hacerse con dos o más procesos

```
// Incrementador
public void run() {
    //...
    while(turno != INC) {}
    // turno == INC!!

    // SC

    turno = DEC;
    //...
}
```

```
// Decrementador
public void run() {
    //...
    while(turno != DEC) {}
    // turno == DEC!!

    // SC

    turno = INC;
    //...
}
```

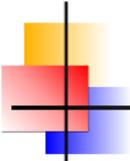
Pregunta

¿Qué problema podría tener esta propuesta?

Alternancia Estricta

“Decimos que un programa concurrente sufre de alternancia estricta cuando dos procesos intentan acceder a una sección crítica y lo hacen siempre de forma alternativa, es decir, nunca entra dos veces consecutivas el mismo proceso”

- ▶ Produce esperas innecesarias entre los procesos que no tienen el turno
- ▶ La alternancia estricta es un problema grave y provoca un buen número de bloqueos
 - ▶ El *ritmo* lo marcará el proceso más lento
 - ▶ Si los procesos tienen distinto número de accesos a la SC, al terminar el que menos accesos solicita y no entregar el turno, el resto de procesos quedarán bloqueados



PROPUESTA 2

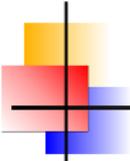
- ▶ Usar una variable para indicar la intención de acceder a la SC
- ▶ Si otro proceso quiere entrar, entonces me quedo esperando

```
// Incrementador
public void run() {
    //...
    inc_quiere = true;
    while (dec_quiere) {};

    // SC
    inc_quiere = false;
    //...
}
```

```
// Decrementador
public void run() {
    //...
    dec_quiere = true;
    while (inc_quiere) {};

    // SC
    dec_quiere = false;
    //...
}
```



PROPUESTA 2

- ▶ Usar una variable para indicar la intención de acceder a la SC
- ▶ Si otro proceso quiere entrar, entonces me quedo esperando

```
// Incrementador
public void run() {
    //...
    inc_quiere = true;
    while (dec_quiere) {};

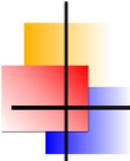
    // SC
    inc_quiere = false;
    //...
}
```

```
// Decrementador
public void run() {
    //...
    dec_quiere = true;
    while (inc_quiere) {};

    // SC
    dec_quiere = false;
    //...
}
```

Pregunta

¿Qué problema podría tener esta propuesta?



PROPUESTA 2

- ▶ Usar una variable para indicar la intención de acceder a la SC
- ▶ Si otro proceso quiere entrar, entonces me quedo esperando

```
// Incrementador
public void run() {
    //...
    inc_quiere = true;
    while (dec_quiere) {};

    // SC
    inc_quiere = false;
    //...
}
```

```
// Decrementador
public void run() {
    //...
    dec_quiere = true;
    while (inc_quiere) {};

    // SC
    dec_quiere = false;
    //...
}
```

Pregunta

¿Qué problema podría tener esta propuesta?

- ▶ Posible deadlock si ambos quieren “simultáneamente” y se quedan en el bucle de espera activa



PROPUESTA 3

- ▶ Avisamos que estamos dentro para que no entre otro proceso

```
// Incrementador
public void run() {
    //...
    while (en_sc_dec) { };
    en_sc_inc = true;

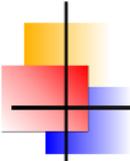
    // SC

    en_sc_inc = false;
    //...
}
```

```
// Decrementador
public void run() {
    //...
    while (en_sc_inc) { };
    en_sc_dec = true;

    // SC

    en_sc_dec = false;
    //...
}
```



PROPUESTA 3

- ▶ Avisamos que estamos dentro para que no entre otro proceso

```
// Incrementador
public void run() {
    //...
    while (en_sc_dec) { };
    en_sc_inc = true;

    // SC

    en_sc_inc = false;
    //...
}
```

```
// Decrementador
public void run() {
    //...
    while (en_sc_inc) { };
    en_sc_dec = true;

    // SC

    en_sc_dec = false;
    //...
}
```

Pregunta

¿Qué problema podría tener esta propuesta?



PROPUESTA 3

- ▶ Avisamos que estamos dentro para que no entre otro proceso

```
// Incrementador
public void run() {
    //...
    while (en_sc_dec) { };
    en_sc_inc = true;

    // SC

    en_sc_inc = false;
    //...
}
```

```
// Decrementador
public void run() {
    //...
    while (en_sc_inc) { };
    en_sc_dec = true;

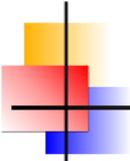
    // SC

    en_sc_dec = false;
    //...
}
```

Pregunta

¿Qué problema podría tener esta propuesta?

- ▶ No garantiza exclusión mutua. Se notifica demasiado tarde, el otro proceso puede haber entrado



PROPUESTA 4

- ▶ Incluimos un momento en el bucle de espera activa para dejar pasar a otros procesos

```
// Incrementador
public void run() {
    // ...
    inc_quiere = true;
    while (dec_quiere) {
        inc_quiere = false;
        inc_quiere = true;
    };
    // SC
    inc_quiere = false;
}
```

```
// Decrementador
public void run() {
    // ...
    dec_quiere = true;
    while (inc_quiere) {
        dec_quiere = false;
        dec_quiere = true;
    };
    // SC
    dec_quiere = false;
}
```



PROPUESTA 4

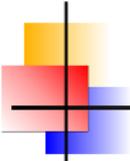
- ▶ Incluimos un momento en el bucle de espera activa para dejar pasar a otros procesos

```
// Incrementador
public void run() {
    // ...
    inc_quiere = true;
    while (dec_quiere) {
        inc_quiere = false;
        inc_quiere = true;
    };
    // SC
    inc_quiere = false;
}
```

```
// Decrementador
public void run() {
    // ...
    dec_quiere = true;
    while (inc_quiere) {
        dec_quiere = false;
        dec_quiere = true;
    };
    // SC
    dec_quiere = false;
}
```

Pregunta

¿Qué problema podría tener esta propuesta?



PROPUESTA 4

- ▶ Incluimos un momento en el bucle de espera activa para dejar pasar a otros procesos

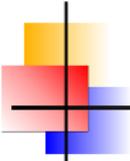
```
// Incrementador
public void run() {
    // ...
    inc_quiere = true;
    while (dec_quiere) {
        inc_quiere = false;
        inc_quiere = true;
    };
    // SC
    inc_quiere = false;
}
```

```
// Decrementador
public void run() {
    // ...
    dec_quiere = true;
    while (inc_quiere) {
        dec_quiere = false;
        dec_quiere = true;
    };
    // SC
    dec_quiere = false;
}
```

Pregunta

¿Qué problema podría tener esta propuesta?

- ▶ El tiempo de espera para entrar en la SC no está acotado (starvation)

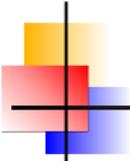


UNA SOLUCIÓN: ALGORITMO DE PETERSON

- ▶ Combina una notificación de intención de acceso a la SC con un turno

```
// Incrementador
public void run() {
    // ...
    quiere_inc = true;
    turno = DEC;
    while (quiere_dec &&
           turno == DEC) {};
    // SC
    quiere_inc = false;
}
```

```
// Decrementador
public void run() {
    // ...
    quiere_dec = true;
    turno = INC;
    while (quiere_inc &&
           turno == INC) {};
    // SC
    quiere_dec = false;
}
```



UNA SOLUCIÓN: ALGORITMO DE PETERSON

- ▶ Combina una notificación de intención de acceso a la SC con un turno

```
// Incrementador
public void run() {
    // ...
    quiere_inc = true;
    turno = DEC;
    while (quiere_dec &&
           turno == DEC) {};
    // SC
    quiere_inc = false;
}
```

```
// Decrementador
public void run() {
    // ...
    quiere_dec = true;
    turno = INC;
    while (quiere_inc &&
           turno == INC) {};
    // SC
    quiere_dec = false;
}
```

- ▶ Indicar la intención de entrar permite que el otro proceso acceda a la SC si no tenemos intención de entrar
- ▶ El turno permite resolver de forma equitativa las situaciones en las que ambos procesos quieren entrar
- ▶ Existen otras soluciones, Dekker's algorithm o Lamport's bakery algorithm