Tecnológico de Costa Rica

Escuela de Computación Maestría en Computación con énfasis en Ciencias de la Computación

Curso: Sistemas Operativos Avanzados

Profesor: Francisco Torres

Estudiante: Oscar Rodríguez Arroyo

Apuntes: 27 de Febrero, 2017

Administrativo

Quiz #2

1. Defina los siguientes conceptos
    a. Trap vs interrupción
    b. Protección de memoria
    c. Modo privilegiado
    d. Garantías de multiprogramación.
2. Compare el concepto de safety en SPIN y exokernel
3. Mencione los principales argumentos de Liedtke respecto al uso de microkernels

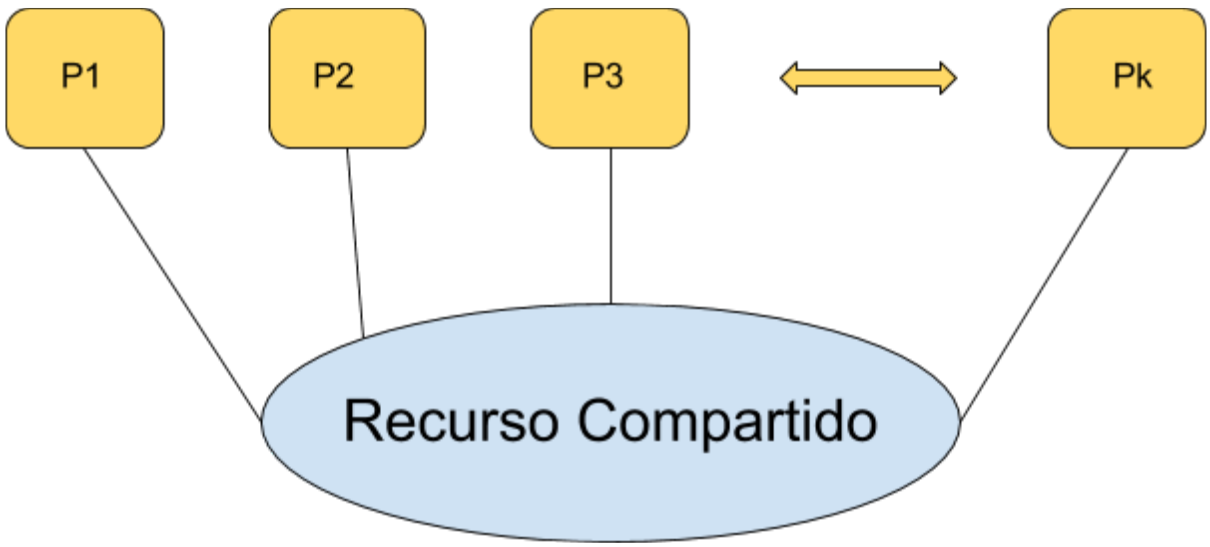
Papers a leer

- 602, 603, 608
- *Perfecto* no va a presentar un resumen

Sincronización

Es la coordinacion y cooperacion de un conjunto de procesos para asegurar la comparación de recursos de cómputo.

***Recurso*: Puede ser una variable, estructura de datos, dispositivo, etc

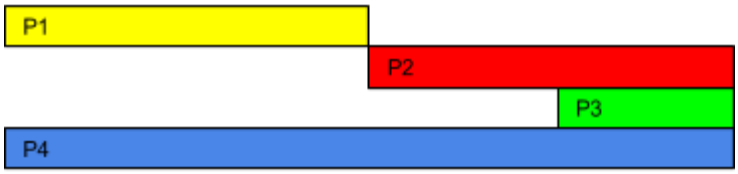


Problema

Sin una sincronización adecuada entre procesos, la actualización de variables compartidas puede inducir a errores de tiempo relacionados con la concurrencia que son con frecuencia difíciles de depurar.

Procesos Concurrentes

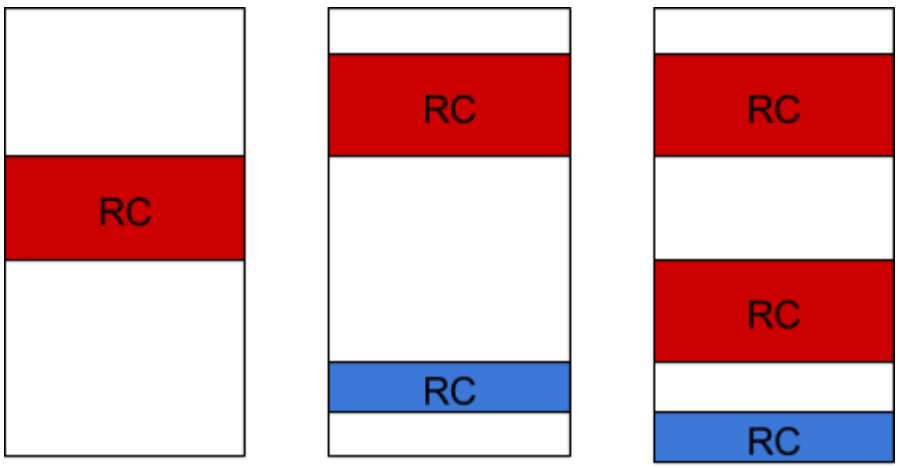
- Es un concepto de tiempo.
- La concurrencia puede ser igual que la infidelidad (según Torres)
- Dos procesos son concurrentes si uno empieza antes de que otro termine.



- El P1 no es concurrente con P2
- El P3 es concurrente con P2
- El P1 y P3 no son concurrentes
- El P4 es concurrente con P1, P2 y P3

Región Crítica

Es la parte del código donde un proceso concurrente usa un recurso compartido. Distintos procesos pueden tener su recurso compartido en distintos lugares.



Podría haber regiones críticas asociadas a diferentes recursos.

Problema de la región crítica

- Hay un recurso *compartido* por 2 o más procesos.
- **NO** se debe permitir que varios lo usen al mismo tiempo.
- Arbitrar el uso del recurso compartido.

Estructura de la solución

Busy Waiting - Espera Activa

- El proceso no puede ejecutar la región crítica si ya hay otro proceso en la misma región crítica
- **IDEA:** hacer lo que ejecute otro código
- Usualmente un *loop* del que sale sólo si puede usar la región crítica
- Busy Waiting = espera ocupada

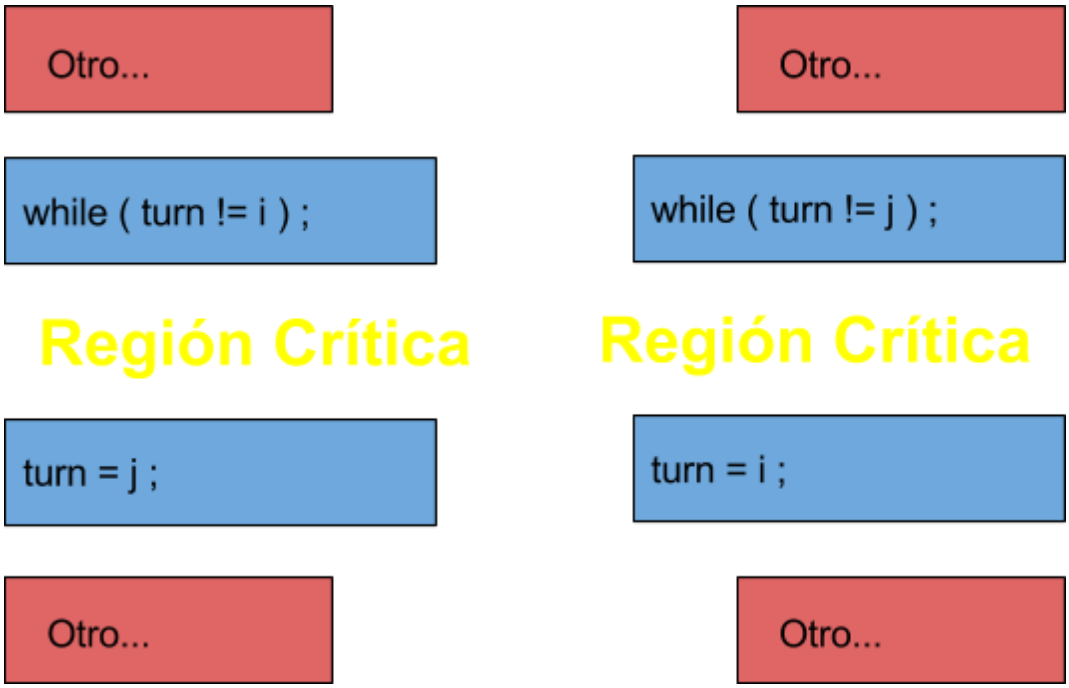
Ventajas	**Desventajas**
- No hay necesidad de invocar al sistema operativo	- Pregunta por condiciones que no se dan - Puede haber mucho tiempo de espera

Solución al problema de la región crítica

Debe cumplir lo siguiente:

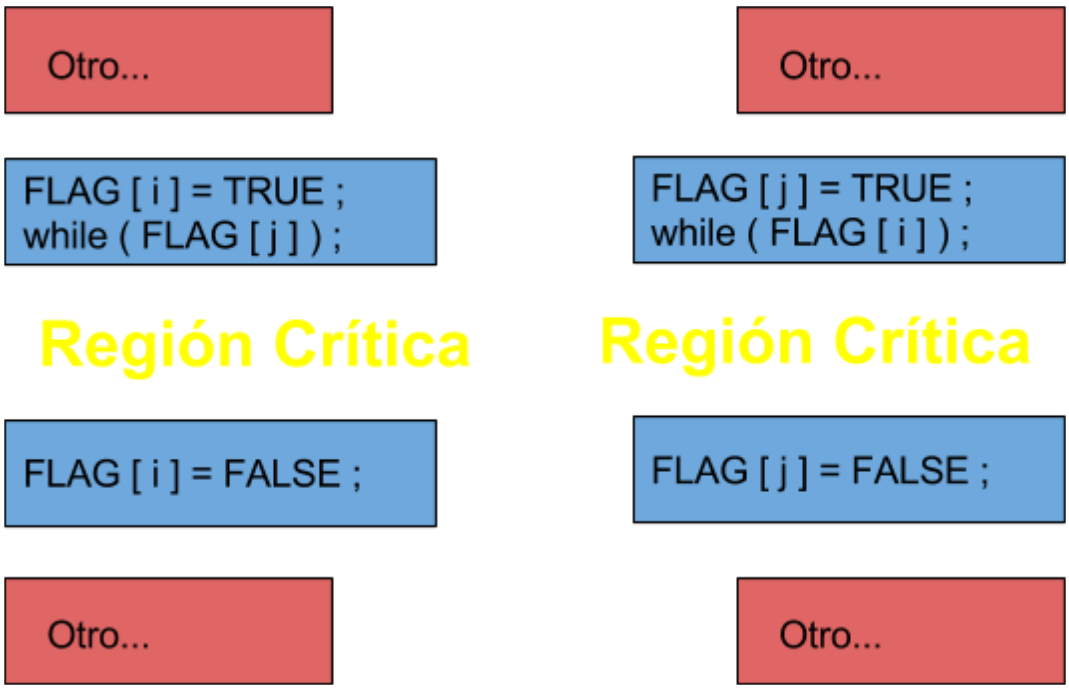
- **RC1 - Exclusión mutua**
  - Sólo un proceso puede estar en la región crítica en un momento dado
- **RC2 - Progreso**
  - Si algún proceso no le permite a otro entrar en la región crítica, es porque está interesado en la región crítica.
- **RC3 - Espera acotada (Acceso en tiempo finito)**
  - Una vez que un proceso manifiesta interés en la región crítica, debe esperar un tiempo que puede ser grande, pero finito.

Algoritmo 1 [2 procesos]



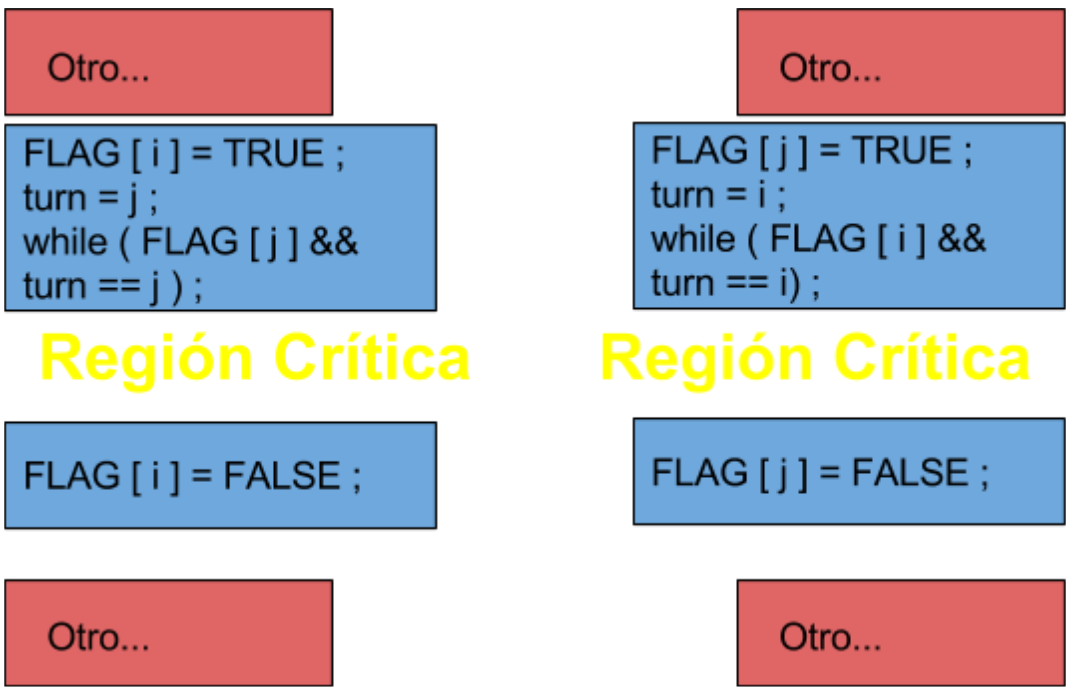
*Conclusión: la solución **no funciona** dado que impone un patrón de acceso y no cumple RC2.*

Algoritmo 2 [2 procesos]



*Conclusión: En este algoritmo los procesos anuncian que quieren entrar a la región crítica pero **no funciona** debido a que la espera no es acotada*

Algoritmo 3 [2 procesos]

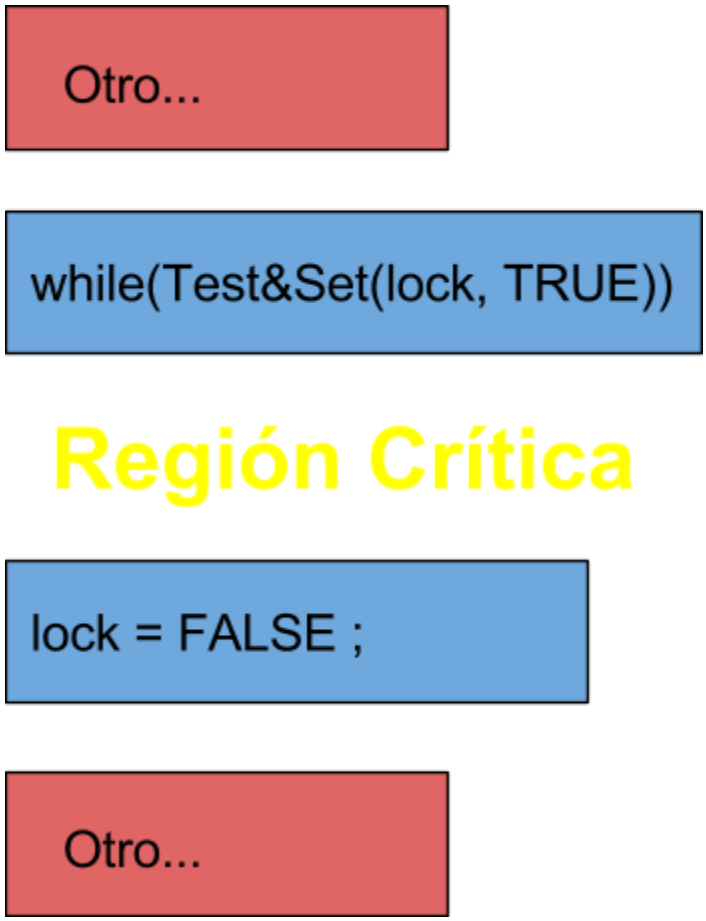


*Conclusión: En esta solución no hay problema, turn es i o es j, nos salva el memory inter-lock (sucede a nivel de hardware). **FUNCIONA***

Instrucciones de Hardware

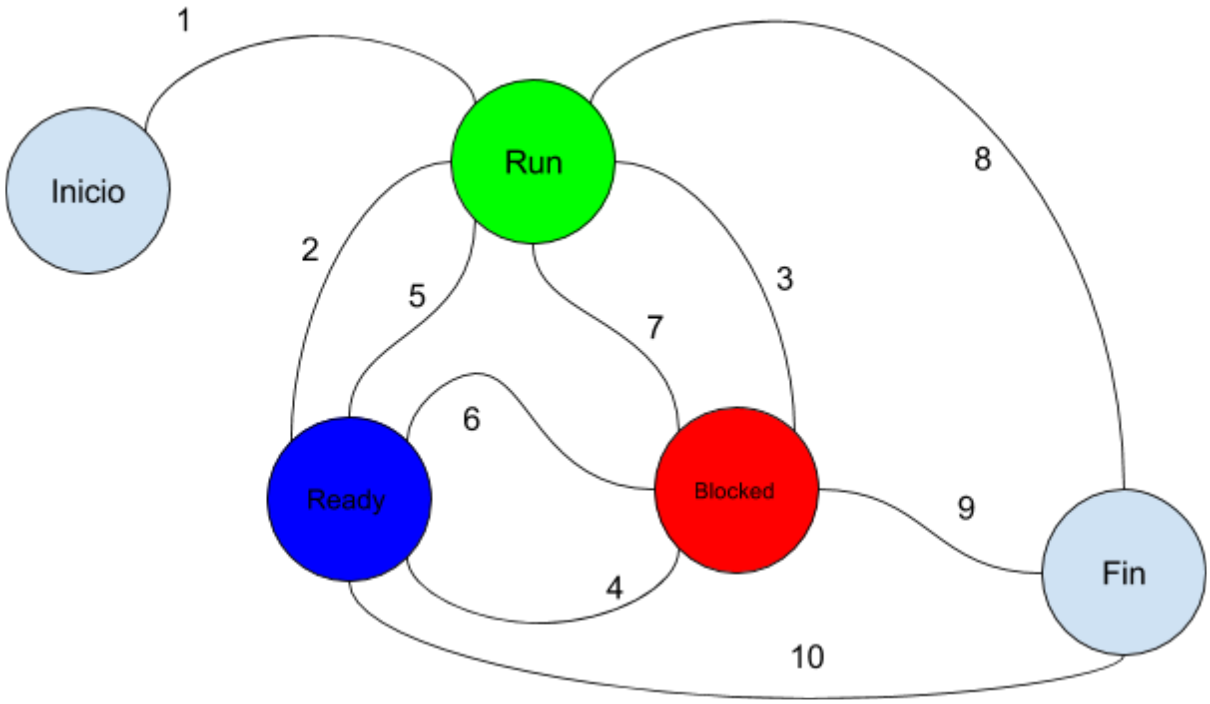
- El Hardware puede ayudar a implementar soluciones de *Busy Waiting*
- Son instrucciones atómicas que hacen “varias” cosas
- **Test & Set**
    - Test&Set(*variable*, *valor*);
    - **Test**: devuelve el valor de *variable* (posición de memoria)
    - **Set**: asigna *valor* a *variable*
    - Read + Write
    - En arquitectura Intel se llama

Algoritmo 4 [N procesos]



Ventajas	**Desventajas**
- Puede manejar N procesos - La atomicidad es la clave	- NO HAY

Conceptos de Scheduling



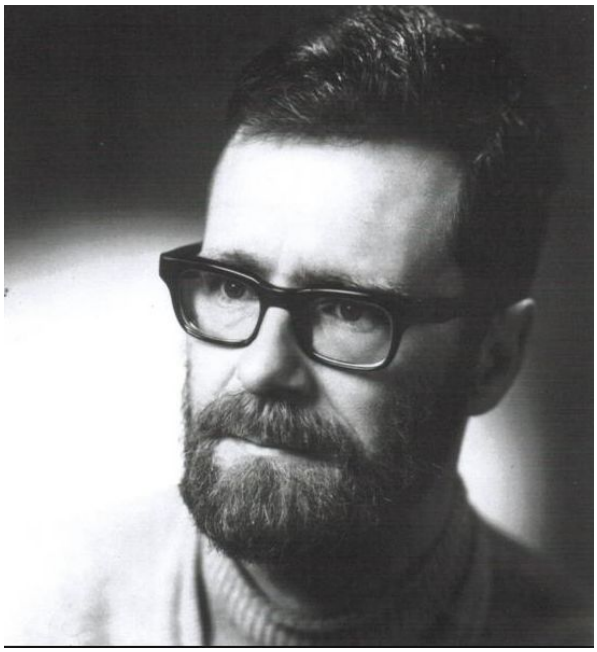
1. Inicio a Ready: un proceso que no está ejecutándose pide correr.
2. Ready a Run: escoger un proceso de ready y darle el CPU.
3. Run a Blocked: un proceso está corriendo y hace petición de un recurso que no está disponible o un recurso muy lento, esto dispara un System Call y el Sistema Operativo suspende al proceso.
4. Blocked a Ready: se activa por una interrupción de un dispositivo externo.
5. Run a Ready: le quitan el CPU al proceso, por causa de timeslice o otro criterio del Sistema Operativo.
6. Ready a Blocked: el proceso estaba listo y le quitaron algún recurso.
7. Blocked a Run: pueden haber procesos con altísima prioridad, que cuando se le devuelven los recursos se corren inmediatamente.
8. Run a Fin: el proceso terminó felizmente.
9. Blocked a Fin: el sistema operativo mató el proceso.
10. Ready a Fin: el sistema operativo mató el proceso.

Conceptos

- Job Scheduling/ Long Term Scheduling: Escoge el proceso a ejecutar
  + I/O-bound: aquellos procesos que están ligados a los dispositivos de entrada y salida.
  + CPU-bound: procesos que se dedica a hacer cálculos

- Job mix: conjunto de procesos corriendo al mismo tiempo. Debe ser variado.

Edsger Dijkstra



- Científico de computación. Holandés (1930 - 2002)
- Turing Award 1972
- Uno de los más influyentes en la disciplina
- Contribuciones en: algoritmos, sistemas operativos, compiladores, lenguajes de programación
- Primer programador oficial
- Usualmente recordado por el **algoritmo de Dijkstra**, también llamado **algoritmo de caminos mínimos.**

Semáforos de Dijkstra

- Son objetos manejados por el sistema operativo
- Tienen operaciones limitadas
- Se usan con System Calls
- Tiene 2 operaciones fundamentales
    - **P(S)**: *proberen.* Ocurre la lectura de la variable
    - **V(S)**: *vergohen.* Incrementar.

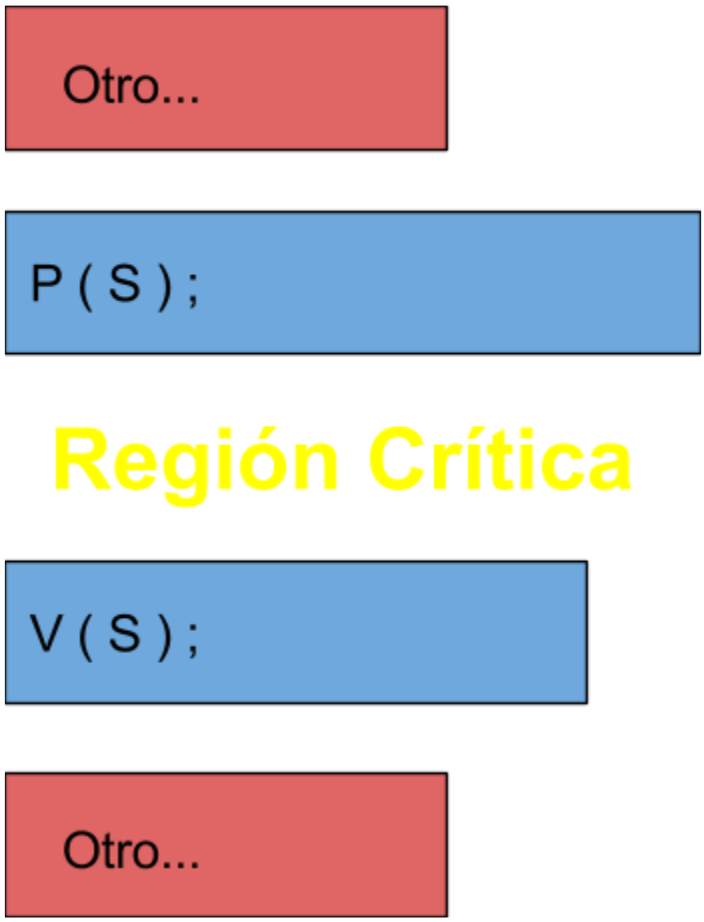
```
P(S){
        S = S - 1;
        if(S < 0){
                *****SUSPENDER PROCESO******
                Enviar a BLOCKED, en la cola asociada a S
        }
}

V(S){
        S = S + 1;
        if(S <= 0){
                *****DESBLOQUEAR PROCESO******
                Selecciona algún proceso de la cola asociada a S.
```

```
                Pasar a la cola de READY
        }
}
```

P(S) y V(S) tienen que ser instrucciones atómicas.

Algoritmo 5 [Proceso i]



- Bajo este modelo se resuelve el problema de la región crítica
- Son implementados por el sistema operativo
- Deben ser atómicos para que funcionen
    - Usan variables compartidas (semáforos y estructuras de PCBs)
    - Código relativamente pequeño y efectivo
    - Se inhiben las interrupciones durante su ejecución
- ¿En múltiples CPUs?Se deben rodear al P y al V de *busy waiting*