Tecnológico de Costa Rica

Escuela de Computación

Maestría en Computación con énfasis en Ciencias de la Computación

MC 6004 Sistemas Operativos Avanzados

Profesor: Francisco Torres

Estudiante: Raquel Elizondo Barrios

Semana 4

27 de Febrero del 2017

I semestre 2017

Administrativo

Papers para resumir:

1. 0602
2. 0603
3. 0608

IMPORTANTE: Perfecto no va a entregar uno de los resúmenes.

Sincronización

Situación en la cual, varios programas cooperan entre ellos. Estos programas comparten o compiten por los recursos. La sincronización nace como resultado de la multiprogramación y del hecho que la protección no es suficiente.

Los programas pueden perder la CPU en cualquier momento. Cuando se le quita la CPU a un proceso, se guarda el ambiente actual que tiene el proceso. Cuando se devuelve la CPU al proceso también se carga su ambiente guardado. En este cambio de contexto, se puede sobre escribir valores en memoria y causar incoherencia en la memoria.

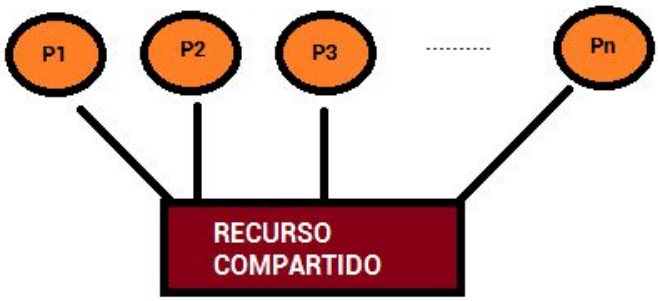
Procesos concurrentes

Dos procesos son concurrentes, si uno empieza antes que el otro termine. Los procesos concurrentes se vuelven complicados cuando compiten por el mismo recurso.

- El proceso 1 es concurrente con el proceso 2
- El proceso 3 es concurrente con el proceso 2
- El proceso 1 y 3 no son concurrentes
- El proceso 4 es concurrentes con todos los otros procesos

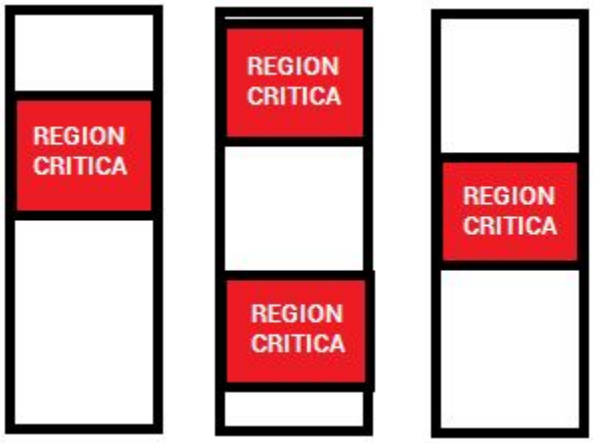
Recurso Compartido

El aquel recurso que podría ser usado por 2 o más procesos concurrentes. Los procesos podrían colaborar o competir por el curso.

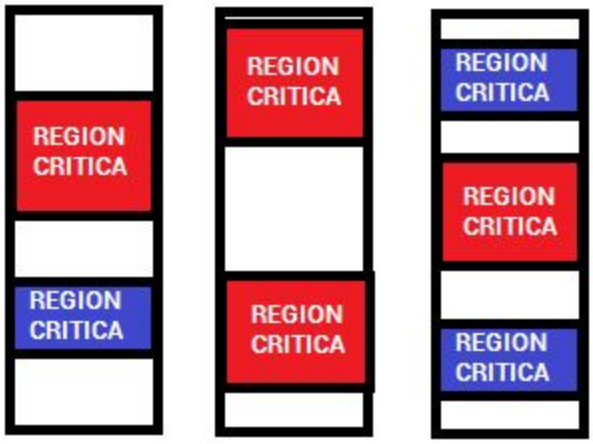


Región Crítica

Fragmento de código que debe ser ejecutado atómicamente y que hace uso de un recurso compartido de un proceso concurrente. La región crítica no es el recurso que estoy compartiendo, la región crítica es código.



Podría haber regiones críticas asociadas a diferentes recursos.



Problema de la Región Crítica

Hay un recurso compartido por 2 o más procesos. No hay que permitir que varios procesos usen un recurso al mismo tiempo porque pueden pasar cosas impredecibles. Por lo cual hay que arbitrar el recurso compartido.

Estructura de solución

Busy Waiting - *Espera activa*

"Truco" para impedir que dos o más procesos entren a la región crítica a la vez. El proceso se cicla para esperar que se den las condiciones necesarias para entrar a la región crítica. Esto ocurre en el protocolo de entrada.

Ventajas

1. No invoca al sistema operativo.

Cuando usarlo

1. Si el tiempo de espera es mejor que el cambio de contexto.
2. Poca contención por los recursos, es raro el caso que alguien quiera usar el recurso.

Desventajas

1. Mucho tiempo de espera.
2. Cuando un proceso se cicla y justo tiene la CPU, se ejecuta el mismo código muchas veces y es un código que no genera nada, preguntando por condiciones que sólo puede activar el proceso que está en la región crítica y que no tiene la CPU.

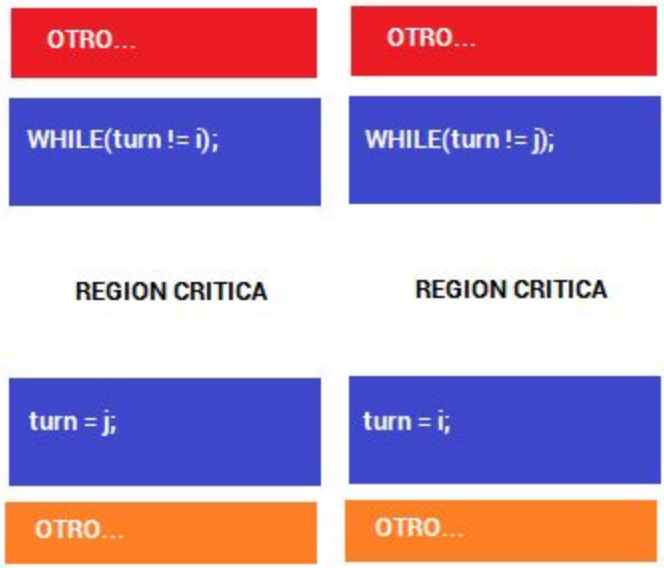
Solución al problema de la Región Crítica

RC1 - Exclusión Mutua
Solo 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
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



No cumple RC2
Características:

1. Pone un patrón de acceso

Algoritmo 1 - 2 procesos

No cumple RC3

Características:

1. Que los procesos anuncien que quieren entrar a la región crítica.

Algoritmo 3 - 2 procesos

Si sirve.

Características:

1. Está limitado a solo dos procesos.
2. Requiere colaboración del programador.

Instrucción Atómica

Durante el proceso de esta instrucción no puede haber interrupciones hasta que se termine su ejecución.

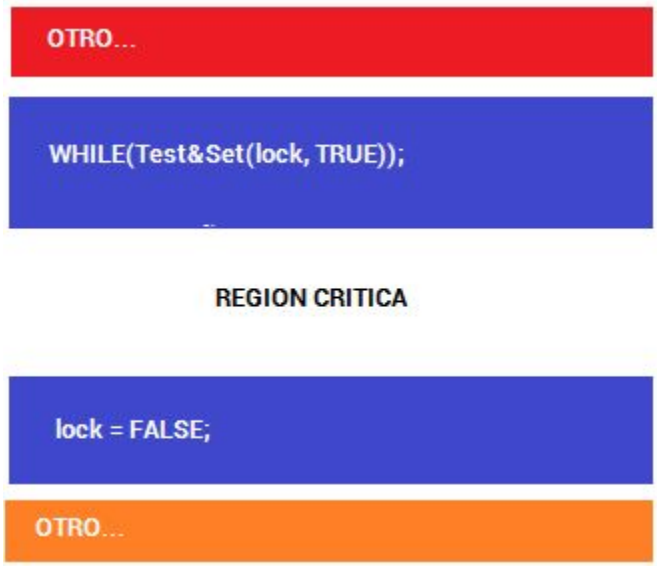
¿Como hace el Sistema Operativo para garantizar una instrucción atómica?
Apaga las interrupciones.

Test&Set(variable, valor);

Es una instrucción de lenguaje máquina atómica, que combina un read con un write. Se lee el valor actual antes de escribir el nuevo.

Test devuelve el valor de `variable` (posición en memoria)
Set asigna `valor` a `variable`

Algoritmo 4 - n procesos

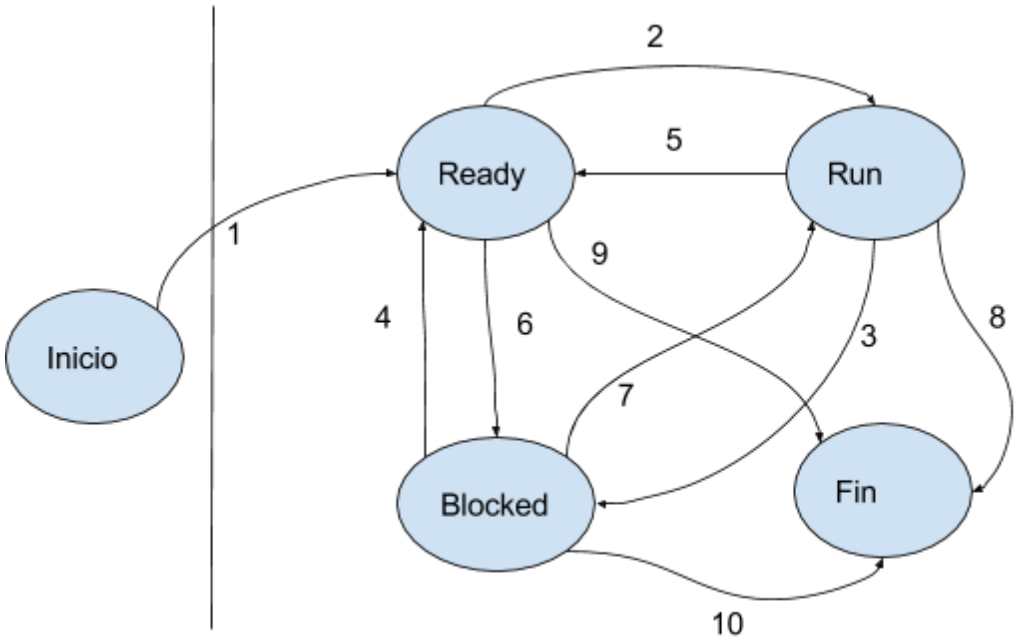


Características:

1. Corre n procesos.
2. Hay interrupción del sistema operativo. El sistema operativo no sabe nada de la región crítica.

Conceptos de Schedulling

Edsger W. Dijkstra
Científico de la computación
Holandés (1930-2002)
Turing Award 1972
Uno de los más influyentes de la disciplina.
Contribuciones en: algoritmos, sistemas operativos, compiladores, lenguajes de programación.
Primer programador oficial



- **Inicio**: el proceso aún no ha solicitado que quiere correr
- **Ready:** el proceso tiene todo lo necesario para correr excepto la CPU
- **Run:** el proceso está corriendo y tiene la CPU.
- **Blocked:** todos aquellos procesos que tienen la CPU pero no tiene todo lo necesario para correr, están esperando por algún recurso.
- **Fin:** el proceso terminó.

Descripción de los transiciones

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

IMPORTANTE: La CPU no se cuenta como un recurso

Conceptos

- **Job Scheduling/ Long Term Scheduling:** actividad en la cual se escogen los procesos que van a correr.
- **Job mix:** conjunto de procesos corriendo al mismo tiempo.
- **I/O-bound:** aquellos procesos que usan mucho los dispositivos de entrada y salida.
- **CPU-bound:** aquellos procesos que la mayor parte del tiempo están haciendo cálculos.

Semáforos

Objeto especial, administrado por el Sistema Operativo.
Operaciones limitadas y se usan con System Calls

Hay dos fundamentales:

- P(s) -> Test, lectura de variable. P de pedir
- V(s) -> Incrementar. V de deVolver

```
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 y V son instrucciones atómicas.

¿Múltiples CPUs?
Los semáforos usualmente están rodeados de Busy Waiting o Spin locks.

Algoritmo 5

Los semáforos resuelven el problema de la región crítica y son implementados por el Sistema Operativo.
Deben ser atómicos para que funcionen:

- Usan variables compartidas (semáforo y estructuras de PCBs)
- Código relativamente pequeño
- Se inhiben las instrucciones durante la ejecución

Presentación

Título: Algoritmos para sincronización estable en la memoria compartida de multiprocesos

Spin locks

Son un medio para lograr la exclusión. Los spin locks utilizan semáforos. Solo permiten que un proceso esté ejecutando o usan la memoria en un momento determinado.

1. **El bloqueo Test&Set:** si hay muchos procesos ejecutando el ciclo, puede bajarle el rendimiento; para solucionar hay dos opciones:
    a. test_and_test_and_set: pregunta por el valor pero no se queda preguntando, sino que espera a que el broadcast lo despierte.
    b. Delay: si se encuentran con la bandera ocupada entran en una delay y luego vuelve a preguntar.
2. **Bloqueo por ticket:** el proceso que llega recibe un ticket y se va a esperar que lo llamen. Al ser con ticket, los procesos se ejecutan con FIFO.
    a. Request counter: asigna el ticket
    b. Release counter: incrementa el request counter
3. **Bloqueos de cola basados en matrices:** cada proceso tiene un caché y va a preguntarse a sí mismo en su caché el valor de la bandera. Cuando un proceso termina, se ejecuta un broadcast que actualiza el caché de los procesos.
4. **Un nuevo bloqueo de colas basado en listas:**
    a. Garantiza FIFO para la adquisición del bloqueo.
    b. Itera únicamente en las variables locales de accesibilidad
    c. Requiere una pequeña cantidad de espacio para constantes por bloqueo
    d. Funciona igual en máquinas con o sin caché coherente

Barreras

Permite que n cantidad de procesos esperen a que todos los procesos terminen.

1. **Barreras centralizadas:** actualiza una pequeña cantidad de estado compartido para iniciar su llegado. Son dos contadores: cuando un proceso empieza a ejecutar aumenta el contador y a como van llegando a la barrera se va decrementando el contador.
2. **Árbol combinación de software:** sirve para recoger múltiples referencias a la misma variable compartida en una sola referencia.
3. **Diseminación o mariposa:** es una barrera redonda y cada hilo tiene su propia visión de todos los otros hilos implicados en la sincronización.
4. **Torneo:** los procesos compiten entre para llegar a la cima.