|  |  |
| --- | --- |
| CS 341 | #41 The Critical Section Problem |

~~ Welcome to the **Critical Section Problem** game show! ~~

Example Critical Section

if( data[i] > data[j] {   
 temp=data[i]; data[i] = data[j]; data[j] = temp   
}

**The Critical Section Problem**

while(running) {

1. Wait to enter the critical section if another thread is in the CS.

2. Critical Section Code; Only one thread/process in here at a time!

3. Leave critical section. Allow another waiting thread to enter.

4. // do other stuff most of the time

}

Today's prizes:

Mutual exclusion

Bounded wait

Progress

Candidate #1. Use a single, boolean "flag"

boolean flag

|  |  |
| --- | --- |
| *Thread A*  wait while the flag is up  raise the flag!  *Critical Section* code here  lower the flag!  ... | *Thread B*  wait while the flag is up  raise the flag!  *Critical Section* code here  lower the flag! ... |

// Then each thread does other work but will repeat this again sometime in the future. Problems?

Candidate #2. Give each thread its own a flag.

boolean flagA, flagB

|  |  |
| --- | --- |
| wait while B's flag is up  raise A flag  *Critical Section* code here  lower A flag | wait while A's flag is up  raise B flag  *Critical Section* code here  lower B flag |

Problems?

Candidate #3. Change the sequence order

|  |  |
| --- | --- |
| raise A flag  wait until B flag is down   *Critical Section* code here  lower A flag | raise B flag  wait until A flag is down  *Critical Section* code here lower B flag |

Problems?

Candidate #4. Try a single turn-based shared variable.  
 turn=1

|  |  |
| --- | --- |
| while( turn == 2) { }  *Critical Section* code here  turn = *2* | while( turn == 1) { }  *Critical Section* code here turn = 1 |

Problems?

Dekker’s N=2 solution (1962) to the Critical Section Problem.

raise my flag

while(your flag is raised) :

if it's your turn to win :

lower my flag

wait while your turn

raise my flag

// Do Critical Section stuff here

set your turn to win

lower my flag

Peterson's N=2 solution to the Critical Section Problem? (1981!)

raise my flag

turn = your\_id

wait while your flag is raised and turn is your\_id

// Do Critical Section stuff

lower my flag

Code Examples-

Example code for \_\_\_\_?

void lock\_init(){

flag[0] = flag[1] = 0;

turn = 0;

}

//Call before critical section

void lock(int self){

flag[self] = 1;

turn = 1 - self;

while(flag[1-self]==1 && turn==1-self);

}

// Call after critical section

void unlock(int self){

flag[self] = 0;

}

Also

https://android.googlesource.com/kernel/tegra.git/+/android-tegra-3.10/arch/arm/mach-tegra/sleep.S#58

spinlock implementation with no atomic test-and-set and no coherence

\* using Peterson's algorithm on strongly-ordered registers

\* used to synchronize a cpu waking up from wfi with entering lp2 on idle

mov r12, #1

str r12, [r2] @ flag[cpu] = 1

dsb

str r12, [r1] @ !turn = cpu

1: dsb

ldr r12, [r3]

cmp r12, #1 @ flag[!cpu] == 1?

ldreq r12, [r1]

cmpeq r12, r0 @ !turn == cpu?

beq 1b @ while !turn == cpu && flag[!cpu] == 1

mov pc, lr @ locked

Challenges with code implementations?