Mutual Exclusion

- N processes execute (infinite) instruction sequences concurrently.
- Each instruction belongs to either a critical or non-critical operation.

Safety property: Mutual exclusion
- Instructions from critical sections of two or more processes must never be interleaved!
- No deadlocks: If one or multiple processes try to enter their critical sections, which tries to enter one of them, infinite waiting must be avoided.
- No starvation: Processes will eventually enter their critical sections if they are waiting for resources.

Atomic load & store operations

- Assumption 1: Memory locations of atomic load and store are atomic.
- Assumption 2: There is no atomic, non-critical, combined instruction.
- Access to the global variables is done on behalf of a local instruction.
- Other operations include read and write in the critical sections.

References for this chapter

The general mutual exclusion scenario
- A process can enter a critical section if none of its instructions belong to other critical sections.
- Safety property: Mutual exclusion
- Instructions from critical sections of two or more processes must never be interleaved!
- No deadlocks: If one or multiple processes try to enter their critical sections, which tries to enter one of them, infinite waiting must be avoided.
- No starvation: Processes will eventually enter their critical sections if they are waiting for resources.

Mutual Exclusion

Pre-conditions

- The process must have sufficient resources to enter the critical section.
- The process must not be blocked by other processes.

Post-conditions

- The process has successfully entered the critical section.
- The process has completed its tasks and released the resources.

Examples:

1. Task 1: Task 2
2. Task 3: Task 4
3. Task 5: Task 6
4. Task 7: Task 8

Mutual Exclusion: First attempt

- Task 1: Task 2
- Task 3: Task 4
- Task 5: Task 6
- Task 7: Task 8

Mutual Exclusion: Second attempt

- Task 1: Task 2
- Task 3: Task 4
- Task 5: Task 6
- Task 7: Task 8

Mutual Exclusion: Third attempt

- Task 1: Task 2
- Task 3: Task 4
- Task 5: Task 6
- Task 7: Task 8

Mutual Exclusion: Fourth attempt

- Task 1: Task 2
- Task 3: Task 4
- Task 5: Task 6
- Task 7: Task 8

Mutual Exclusion: Checking

- Task 1: Task 2
- Task 3: Task 4
- Task 5: Task 6
- Task 7: Task 8

The general mutual exclusion scenario

- If processes require resources, the resources must be allocated to the processes.
- Critical sections belonging to different processes may overlap.
- Mutual exclusion must be enforced to prevent deadlocks.

Task_Range

is mod

Critical_Section_State

is array

C1, C2: Critical_Section_State := Out_CS;

C1 := In_CS;  ------ critical_section_1;

C2 := Out_CS; C2 := In_CS;

CSS (this_Task) := Out_CS;

P2;

Making any progress?

One_Of_Two_Tasks

P1;

No interference?

P2;

Work without intervention?
Mutual Exclusion

Type

is

mod

2;

Mutual exclusion: Peterson's Algorithm

type

is

Critical_Section_State

Turn : Task_Range := Task_Range'First;

Two tasks only!

array

Critical_Section_State := (others =>  Out_CS);

critical

or

task

type

when

(this_Task  : Task_Range);

Instructions from critical sections

must never be interleaved!

end

loop

exit

when

end

end

No deadlocks

------ non_critical_section

------ critical section

exactly one.

CSS (this_Task) := In_CS;

One_Of_Two_Tasks;

No starvation!

G

Ef

be made

© 2020 Uwe R. Zimmer, The Australian National University page 220 of 758 (chapter 2: “Mutual Exclusion” up to page 253)

© 2020 Uwe R. Zimmer, The Australian National University page 221 of 758 (chapter 2: “Mutual Exclusion” up to page 253)
Types of semaphores:

- Binary semaphores: used in the earliest multitasking systems and support binary counting.
- Quantity semaphores: support counting more than 1, but are often implemented as a combination of binary semaphores.
- Signal semaphores: used for signaling.

Semaphores as supplied by operating systems and runtime environments:

- as a way to control synchronization between processes.
- as a way to implement semaphores and allow processes to communicate.
- as a way to implement semaphores and allow processes to communicate.

Beyond atomic hardware operations:

- Binary semaphores are sufficient to create all other semaphore forms.
- Quantity semaphores are used for counting more than 1.
- Signal semaphores are used for signaling.

Critical section:

- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.
- Critical section: a region of code that is mutually exclusive.


```plaintext
Semaphores

S1, S2 : Semaphore := 1;
task body Pi in
begin
  loop
    ------ non_critical_section_i;  wait (S1);
    wait (S2);
    ------ critical_section_i;
    signal (S2);  signal (S1);
  end loop
end Pi;

G Works too?

G Mutual exclusion!, No global live-lock!
G Works for any dynamic number of processes.
G Individual starvation possible!
G Deadlock possible!
```

Summary

```
• Definition of mutual exclusion
• Atomic load and atomic store operations
  • some classical errors
  • Decker's algorithm, Peterson algorithm
  • Bakery algorithm
• Realistic hardware support
  • Atomic test-and-set, Atomic exchanges, Memory cell reservations
• Semaphores
  • Basic semaphore definition
  • Operating systems style semaphores
```