Thread Synchronization II

Søren Hansen <sha@ase.au.dk>
V0.9

Introduction

Using protection inadvertently presents some challenges, namely *when to use*, *how to use* and what might go run if used without care. These, rather abstract, notation are the topics for this lecture.

Contents

Theme(s)	Topic(s)	Reference
Buffer case	Producer/Consumer impl. using semaphores	[3]
Deadlock	Definition Dining philosophers problem (chandra/misra only superficially)	[4] [1] [5] [1]
Generel	Thread Models (another kind of thread model) Thread Safety Rules for Multithreaded Programming	[2, chap. 4] [2, chap. 5] [2, chap. 6]

Material

Slides

[1] S. Hansen, *Thread synchronization ii*, Slides - see course repos.

Local repository

[2] P. C. Chapin, *Pthread tutorial*, Tutorial, See https://redmine.iha.dk/courses/projects/i3isu/repository, 2008.

Online

[3] E. al. (). Producer–consumer problem. Wikipedia Article, [Online]. Available: `https://en.wikipedia.org/wiki/Producer%2dconsumer_problem`.

[4] ——, (). Deadlock, [Online]. Available: `https://en.wikipedia.org/wiki/Deadlock`.

[5] ——, (). Dining philosophers problem, [Online]. Available: `https://en.wikipedia.org/wiki/Dining_philosophers_problem`.

Fundamental questions to consider while reading

Thread Synchronization II

- Pitfalls
  - Initialising semaphores - how and with what number
  - What to lock and when to lock
  - Locking / unlocking - missing one spells what?

Thread Synchronization II

- Deadlock
  - Which conditions need to be present for a deadlock to occur
  - What is the problem behind the *Dining philosophers problem*
  - Which rule can you do something about, and what could you do to solve this particular problem.
- Priority inversion
  - What actually makes this problem
  - What are the two different strategies to solve the problem
  - In what way do they solve the problem and what is the consequence of which when doing so.