Threads, synchronisation and deadlock

Module 4 workshop and seminar questions/problems

OS (1DT044) and OSPP (1DT096)
February 12, 2024

Threads

1. How do threads differ from processes?

2. Why is it more "expensive" to create a new process compared to creating a new thread?

3. In short, explain the many-to-one user level thread model. Also explain what happens if one of the threads makes a blocking system call in the many-to-one user level thread model.

Need for synchronization

4. What is meant by an atomic operation? Give examples of non-atomic operations.

5. Define the following terms: Race condition, Data race, Critical section and Mutual exclusion.

Properties of lock operations

6. What is meant by a spin lock? What is meant by busy waiting?

7. In the context of mutual exclusion, what is meant by starvation?

Software based synchronization

8. What are the limitations of Petersson's solution to the mutual exclusion problem?

Hardware support for synchronization

9. Name two atomic CPU instructions that can be used to implement synchronization locks.

10. How can spin locks be constructed using the two atomic instructions from above?

Abstractions for synchronization

11. What operations can be performed on a semaphore and how do these operations work?

12. What operations can be performed on a mutex lock and how do these operations work?

13. What is the difference between a mutex lock and a semaphore?

14. When implementing semaphores and mutex locks, how can busy waiting be avoided?

Deadlock

15. Name and explain the four necessary conditions for deadlock?

16. Explain the differences between deadlock prevention and deadlock avoidance.

17. Explain how deadlock prevention can be used to prevent circular wait.

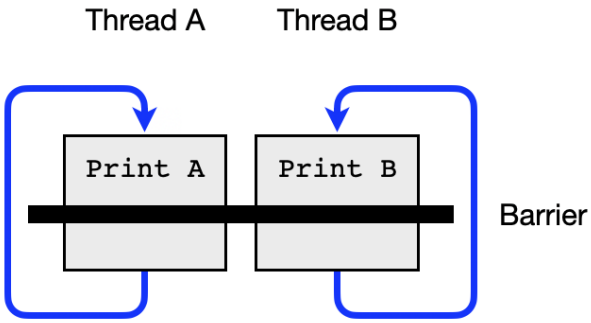
18. What conclusions regarding deadlock can be made using a resource allocation graph (RAG)?

Dining philosophers

19. Explain the Dining philosophers problem.

Barrier synchronization

20. Two threads A and B both executes a loop concurrently with each other. In the loop, Thread A prints A and thread B prints B.



In the loops there should be a **barrier** such that the first thread to reach the barrier must wait for the other to also reach the barrier. Once both threads have reached the barrier the threads are allowed to continue with the next iteration.

For each iteration the order between the threads should not be restricted. The following is an example of a valid trace of execution: $ABBABA$. The following is an example of an invalid trace of execution: $ABBBAB$.

Explain how two semaphores can be used to enforce the two threads to have a rendezvous at the barrier after each iteration.

21. Could two mutex locks be used as drop in replacements for the two semaphores when solving the barrier problem above, where you replace `wait` with `lock` and `signal` with `unlock`? Justify your answer.

Bounded buffer

22. Explain how semaphores can be used to synchronize access to a bounded buffer.

Banker's algorithm

23. Is Banker's algorithm an example of deadlock prevention or deadlock avoidance? Justify your answer.

24. Consider a system with four tasks T_0, T_1, T_2, T_3 and four resources A, B, C, D. The initial state S_0 for Banker's algorithm is defined by:

$$Allocation = \begin{bmatrix} 1 & 2 & 0 & 2 \\ 0 & 1 & 1 & 2 \\ 1 & 0 & 0 & 1 \\ 2 & 0 & 1 & 0 \end{bmatrix} \quad Max = \begin{bmatrix} 2 & 2 & 2 & 4 \\ 2 & 2 & 4 & 3 \\ 2 & 1 & 2 & 1 \\ 3 & 3 & 4 & 5 \end{bmatrix} \quad Available = \begin{bmatrix} 2 & 1 & 2 & 1 \end{bmatrix}$$

need = $\begin{bmatrix} 1 & 0 & 2 & 2 \\ 2 & 1 & 2 & 1 \\ 1 & 1 & 2 & 0 \\ 1 & 3 & 3 & 5 \end{bmatrix}$

Calculate the *Allocation* and *Need* matrices and the *Available* vector for the new state S_1 after T_1 makes a request for one more instance of the resource C and one more instance of resource D. Determine if S_1 is safe by running the safety algorithm step by step, show your calculations in the below table.

Available

Step	A	B	C	D	Choice
1	2	1	2	0	2
2	3	1	2	1	1
3	3	2	4	1	0
4	4	4	4	5	3
5	6	4	5	5	SAFE

Priority inversion

25. Use a figure to explain what is meant by priority inversion.

26. Explain how priority inheritance solves the problem with priority inversion.