



3. What resources are used when a thread is created? How do they differ from those used when a process is created?

4. Describe the actions taken by the kernel to context switch

(a) Among threads.

(b) Among processes.

1. What are the differences between user-level threads and kernel-supported threads? Under what circumstances is one type “better” than the other?

2. Compare and contrast semaphores, mutexes, and monitors. What are the advantages and disadvantages of each? Are all three of these necessary for locking and unlocking data? Can any of these three be used to simulate the others?

7. Consider the procedure `put_forks` in Figure 2-20 of the textbook. Suppose that the variable `state[i]` was set to **THINKING** *after* the two calls to `test`, rather than *before*. How would this change affect the solution?

8. Solve the dining philosophers problem using monitors instead of semaphores.