

Q1:

What are two differences between user level threads and kernel level threads? Under what circumstances is one type better than the other?

Q2:

Describe the actions taken by a kernel to context switching between kernel level threads.

Q3:

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

Q4:

Can a multithreaded solution using multiple user level threads achieve better performance on a multiprocessor system than on a single processor system? Explain.

Q5:

Provide two programming examples in which multithreading provides better performance than a single threaded solution.