In computer science, the sleeping barber problem is a classic inter-process communication and synchronization problem between multiple operating system processes. The problem is analogous to that of keeping a barber working when there are customers, resting when there are none, and doing so in an orderly manner.

The analogy is based upon a hypothetical barber shop with one barber. The barber has one barber chair and a waiting room with a number of chairs in it. When the barber finishes cutting a customer's hair, he dismisses the customer and then goes to the waiting room to see if there are other customers waiting. If there are, he brings one of them back to the chair and cuts his hair. If there are no other customers waiting, he returns to his chair and sleeps in it.

Each customer, when he arrives, looks to see what the barber is doing. If the barber is sleeping, then the customer wakes him up and sits in the chair. If the barber is cutting hair, then the customer goes to the waiting room. If there is a free chair in the waiting room, the customer sits in it and waits his turn. If there is no free chair, then the customer leaves.

Based on a naïve analysis, the above description should ensure that the shop functions correctly, with the barber cutting the hair of anyone who arrives until there are no more customers, and then sleeping until the next customer arrives. In practice, there are a number of problems that can occur that are illustrative of general scheduling problems.

The problems are all related to the fact that the actions by both the barber and the customer (checking the waiting room, entering the shop, taking a waiting room chair, etc.) all take an unknown amount of time. For example, a customer may arrive and observe that the barber is cutting hair, so he goes to the waiting room. While he is on his way, the barber finishes the haircut he is doing and goes to check the waiting room. Since there is no one there (the customer not having arrived yet), he goes back to his chair and sleeps. The barber is now waiting for a customer and the customer is waiting for the barber. In another example, two customers may arrive at the same time when there happens to be a single seat in the waiting room. They observe that the barber is cutting hair, go to the waiting room, and both attempt to occupy the single chair.

The Sleeping Barber Problem is often attributed to Edsger Dijkstra (1965), one of the pioneers in computer science.

Many possible solutions are available. The key element of each is a mutex, which ensures that only one of the participants can change state at once. The barber must acquire this mutual exclusion before checking for customers and release it when he begins either to sleep or cut hair. A customer must acquire it before entering the shop and release it once he is sitting in either a waiting room chair or the barber chair, and also when he leaves the shop because no seats were available. This eliminates both of the problems mentioned in the previous section. A number of semaphores is also required to indicate the state of the system. For example, one might store the number of people in the waiting room.