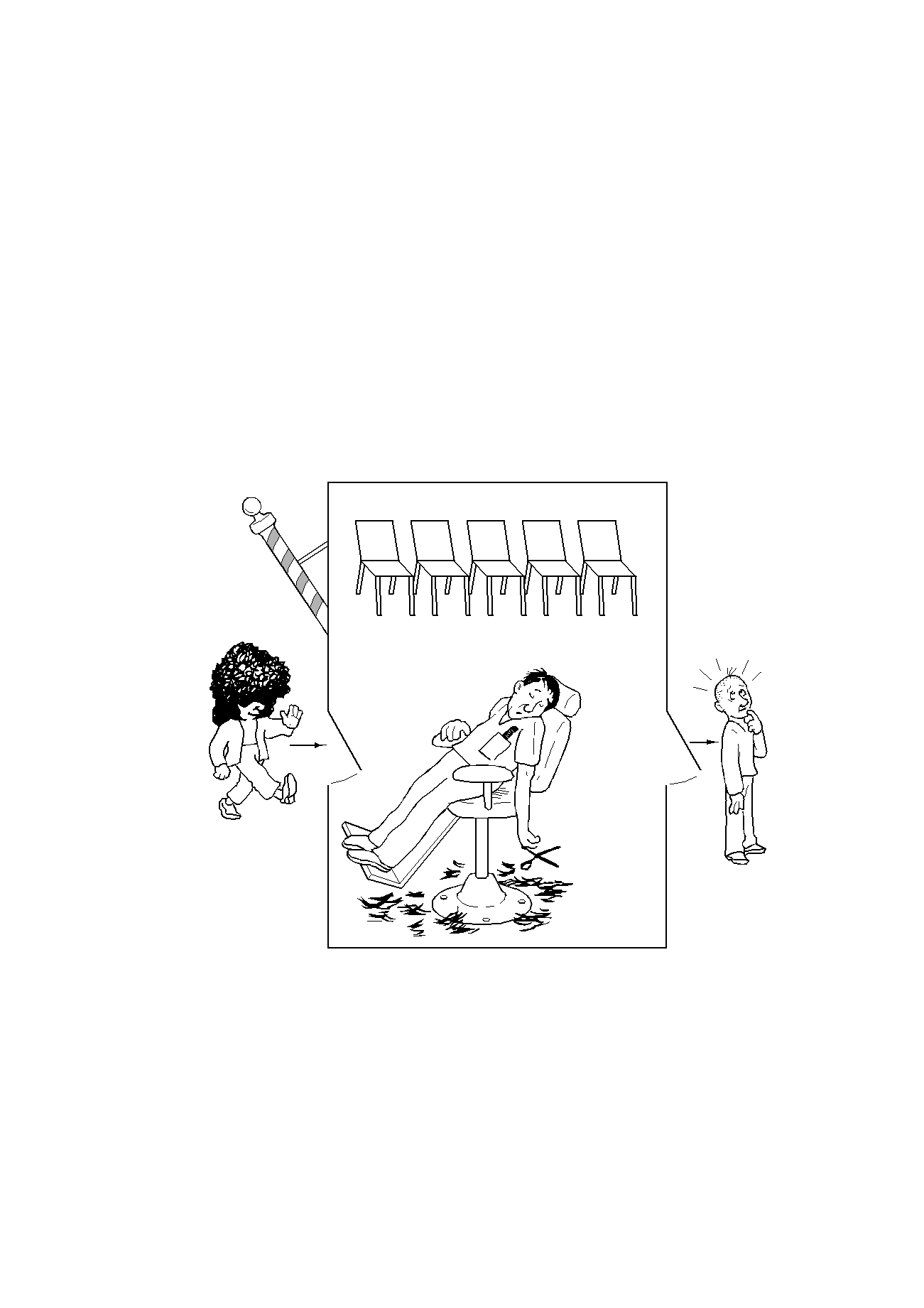
**READ ME FILE**

**ABOUT THE GIVEN PROBLEM**:The sleeping barber problem



* ***Barber:***
  + While there are people waiting for a hair cut, put one in the barber chair, and cut their hair
  + When done, move to the next customer
  + Else go to sleep, until someone comes in
* ***Customer:*** 
  + If barber is asleep wake him up for a haircut
  + If someone is getting a haircut wait for the barber to become free by sitting in a chair
  + If all chairs are all full, leave the barbershop

**Designing a solution**

* **How will we model the barber and customers?**
* **What state variables do we need?**
  + .. and which ones are shared?
  + …. and how will we protect them?
* **How will the barber sleep?**
* **How will the barber wake up?**
* **How will customers wait?**
* **What problems do we need to look out for?**

**Is this a good solution?**

**const CHAIRS = 5**

**var customers: Semaphore**

**barbers: Semaphore**

**lock: Mutex**

**numWaiting: int = 0**

***Customer Thread:***

**Lock(lock)**

**if numWaiting < CHAIRS**

**numWaiting = numWaiting+1**

**Signal(customers)**

**Unlock(lock)**

**Wait(barbers)**

**GetHaircut()**

**else *-- give up & go home***

**Unlock(lock)**

**endIf**

***Barber Thread:***

**while true**

**Wait(customers)**

**Lock(lock)**

**numWaiting = numWaiting-1**

**Signal(barbers)**

**Unlock(lock)**

**CutHair()**

**endWhile**