#10: Deadlocks and The Sleeping Barber

Feb. 16, 2023 · Wade Fagen-Ulmschneider

Sleeping Barber

A classic problem in synchronization is the "Sleeping Barber Problem" *(a barber is someone who cuts hair).*

There are two requirements for the barber:

1. If there are no customers, the barber sleeps in the chair.

2. When there is a customer, the barber wakes up and the customer sits in the chair for a haircut. If there's already a customer in the chair, the customer waits until the chair is available.

Code Overview

Barber: There is only one barber. The barber runs as a single thread in the **barber** function. The **barber** function contains a **while (1)** loop. The barber thread must only exit the **while (1)** loop when a customer with a **name == NULL** arrives.

Customer: Each customer runs as a separate thread in the **customer** function. Each customer has a unique name provided to the **customer** function. There will be many customers arriving concurrently.

Requirements

Complete the sleeping barber problem to ensure that:

● When no customers are present, the barber is waiting for a customer in a **blocked** state (**NOT** "busy waiting").

● When a customer arrives and no other customers are present, the barber is unblocked and cuts the customer's hair. The barber cuts hair by calling **cut_hair(const char *name)**, as provided in the example code. The name must be the **name** of the customer whose hair is to be cut.

● When a customer arrives and another customer is getting their haircut, the customer must wait in a **blocked** state (**NOT** busy waiting).

```
10/sleeping-barber/sleeping-barber.c

 6  struct _sleeping_barber_t {
 7
 8
 9
10  };
11  typedef struct _sleeping_barber_t sleeping_barber_t;
12
13  void init(sleeping_barber_t *sb) {
14
15
16  }
17
18  void barber(sleeping_barber_t *sb) {
19    while (1) {
20
21
22
23
24
25
26
27
28
29      cut_hair("waf");
30
31
32
33
34
35
36
37    }
38  }
39
40  void customer(sleeping_barber_t *sb, const char *name) {
41    // Each customer is a thread! :)
42
43
44
45
46
47
48
49
50
51
52
53  }
```