Aula 7

Problemas Clássicos de Sincronização

1.1 O Jantar dos Filósofos

1.2 Leitores e Escritores

1.3 O Barbeiro Dorminhoco

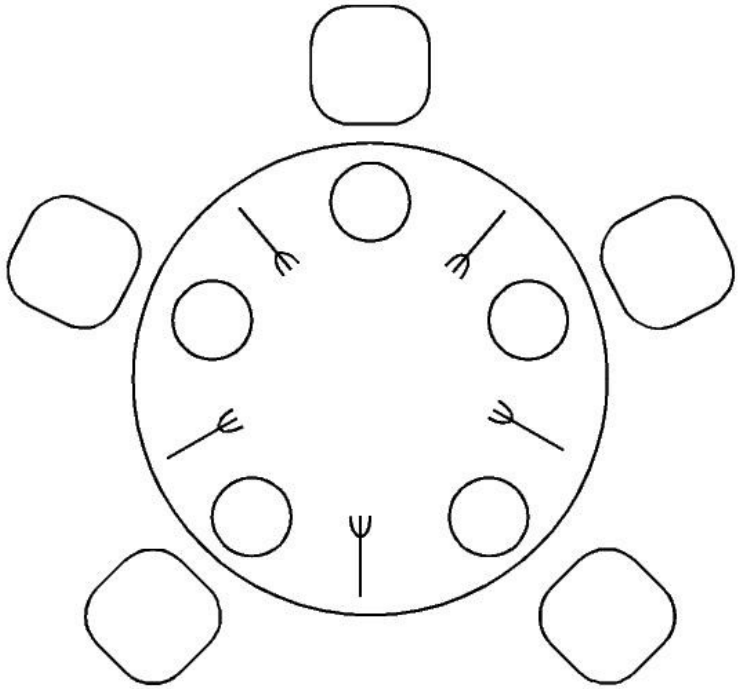
Problemas Clássicos de Sincronização

Porque estudá-los:

- Arquétipos:
  - Representação **sintética** de panoramas **reais** muito complexos
- Permitem avaliar qualidades de métodos e primitivas de sincronização

1.1 O Jantar dos Filósofos (1)

- 5 filósofos sentam-se a uma mesa circular
- Passam a vida pensando ou dormindo
- Para comer precisam de 2 garfos
- Cada garfo é compartilhado por 2 filósofos



1.1 O Jantar dos Filósofos (2)

Como resolver o problema?

Como sincronizar os filósofos, de
de tal forma que:

- Um filósofo “lento” não morra de fome (sem inanição)
- Não fiquem todos parados esperando um ao outro (sem *deadlock*)

1.1 O Jantar dos Filósofos (3)

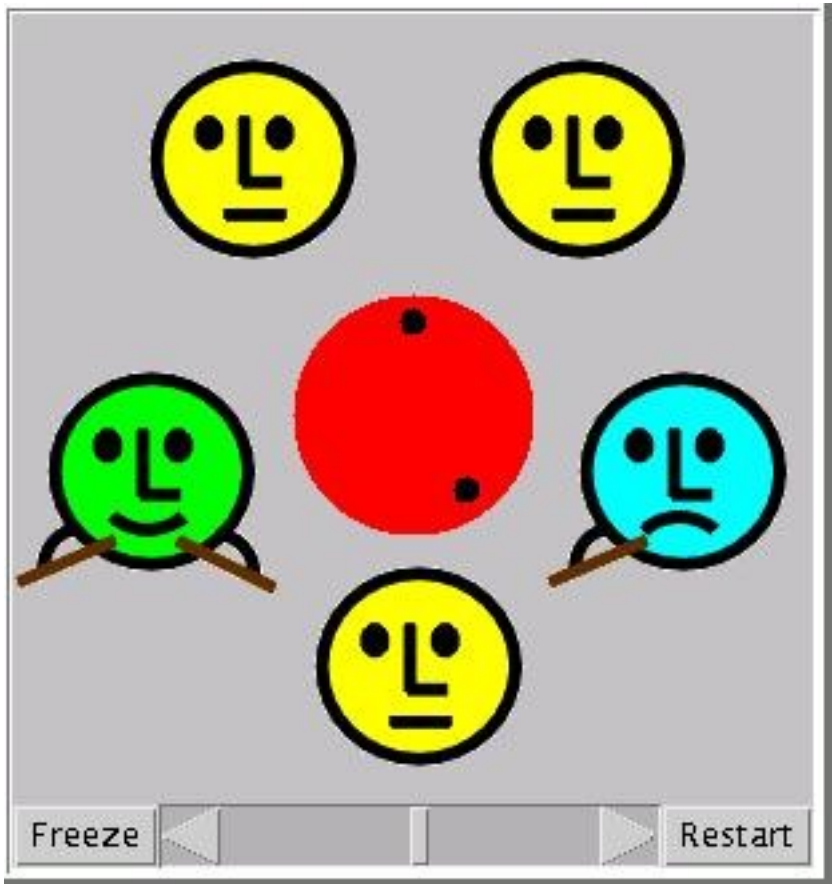
1a solução

semaphore garfo[5];

```
filosofo(i) {
  while (true) {
    think();
    wait(garfo[i]);
    wait(garfo[(i+1) % 5];
    eat();
    signal(garfo[i]);
    signal(garfo[(i+1) % 5]);
  }
}
```

1.1 O Jantar dos Filósofos (4)

Vamos ver se funciona!



Pensando

Esperando um garfo

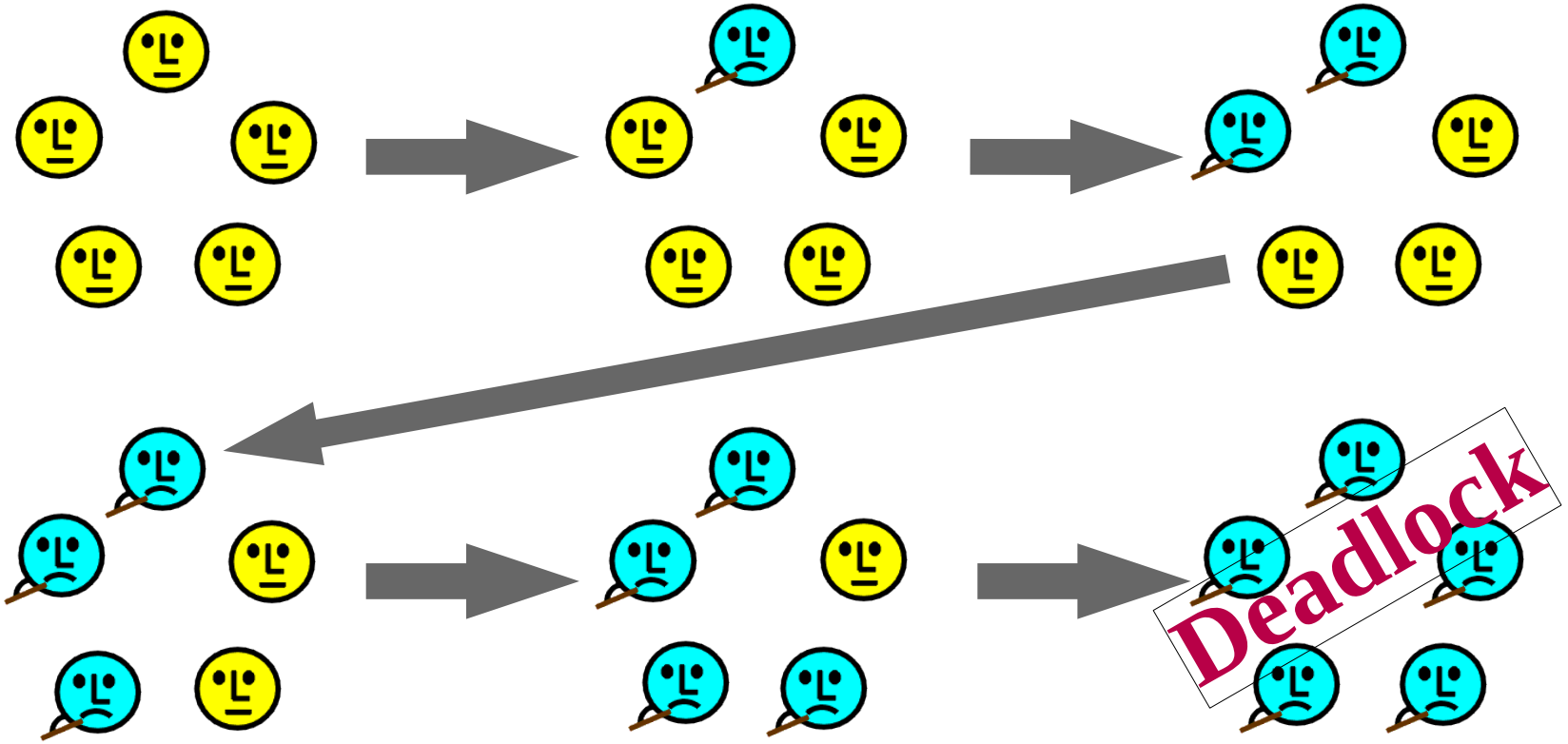
Comendo

Hit&Run!

1.1 O Jantar dos Filósofos (5)

Porque não funciona?

- Permite alocação e espera circulares



1.1 O Jantar dos Filósofos (6)
2a solução - Funciona?

semaphore garfo[5];

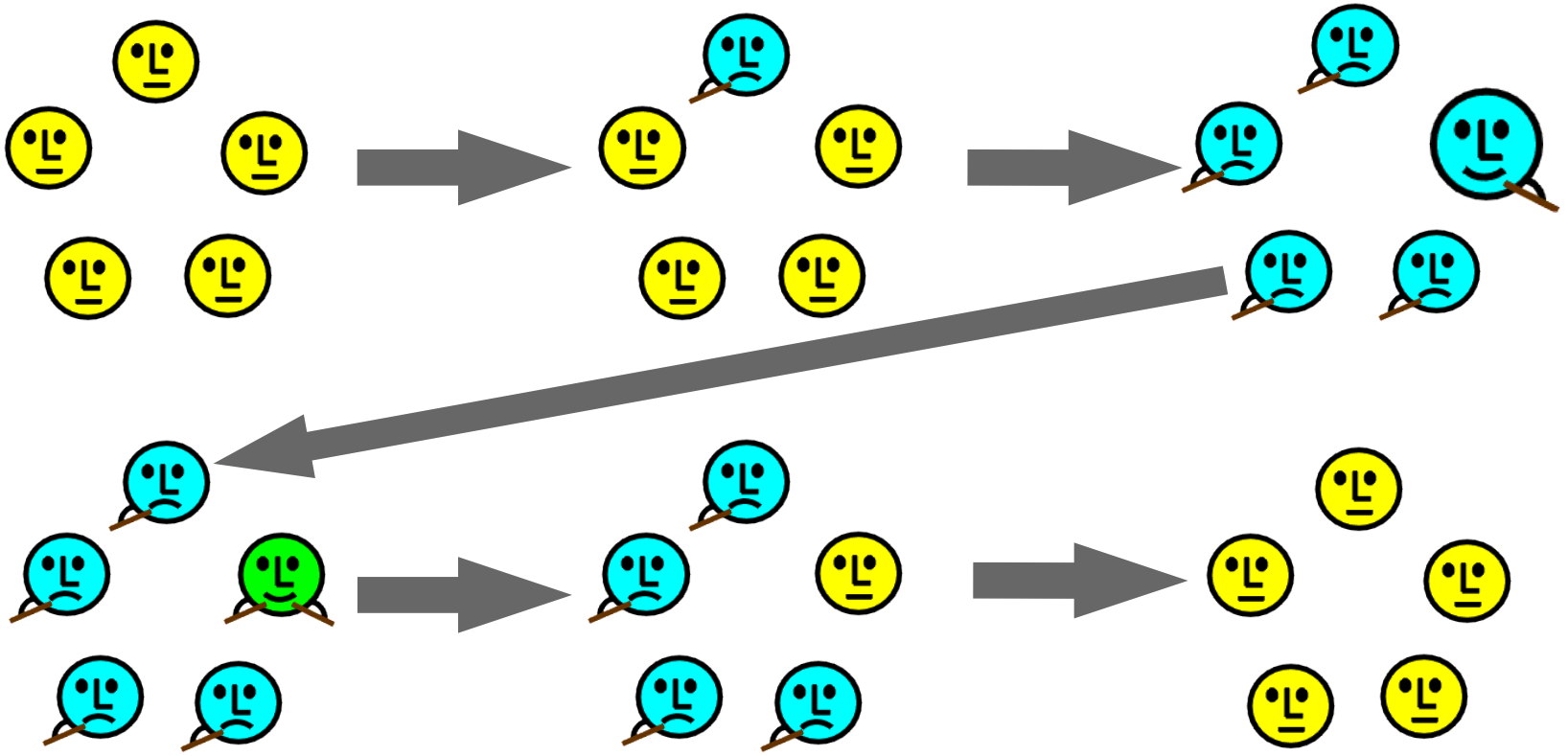
```
filosofo(i) {
  while (true) {
    think();
    if (i != 5) {
      wait(garfo[i);
      wait(garfo[(i+1) % 5];
      eat();
      signal(garfo[i]);
      signal(garfo[(i+1) % 5]);
     }
     else {
      wait(garfo[(i+1) % 5];
      wait(garfo[i);
      eat();
      signal(garfo[(i+1) % 5]);
      signal(garfo[i]);
     }
  }
}
```

Kick me!

1.1 O Jantar dos Filósofos (5)

Porque funciona agora?

- Impede alocação e espera circulares



1.1 O Jantar dos Filósofos (6)

- Ocorre inanição?
- Quando um filósofo decide comer, ele irá comer, mais cedo ou mais tarde?

1.1 O Jantar dos Filósofos (7)

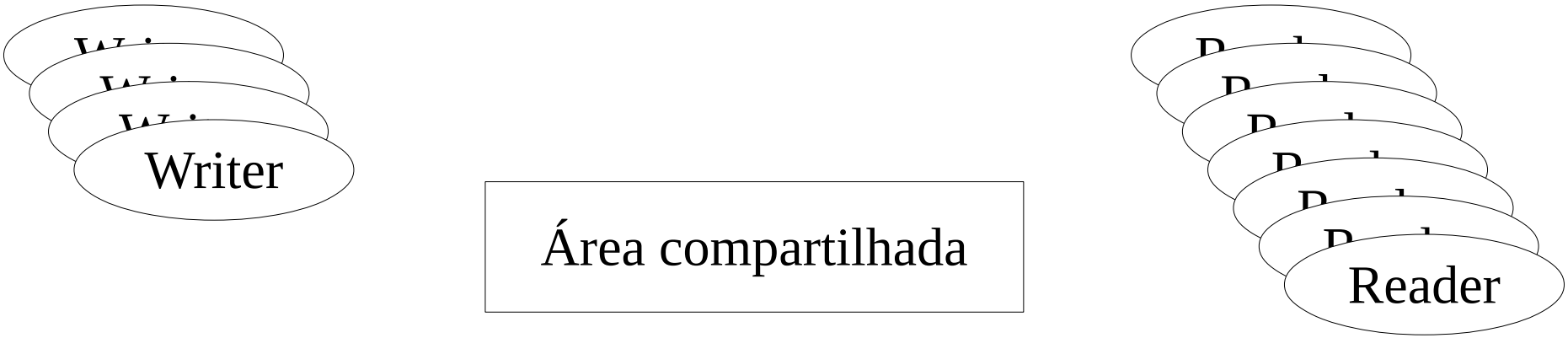
Outros cenários e soluções:

- Acesso exclusivo ao conjunto de garfos:
  - Controle central dos recursos
- Drinking philosophers:
  - Para comer => Todos os garfos
  - Bara beber => Apenas algumas das garrafas

1.2 Leitores e Escritores (1)

O problema:

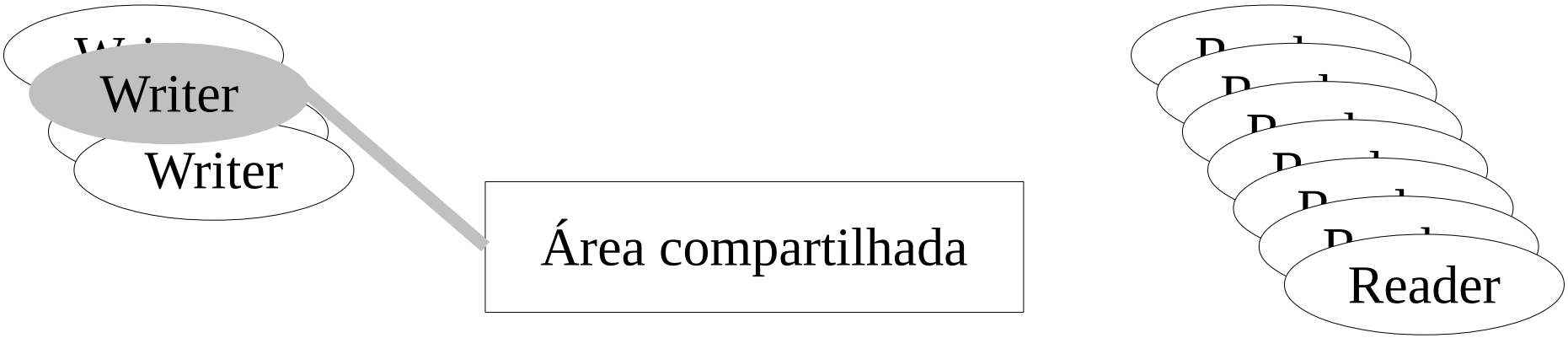
- 1 área compartilhada
- 2 Grupos de processos cooperantes:
  - Escritores lêem/escrevem no buffer: exclusão mútua
  - Leitores lêem o que foi escrito: exclusão parcial



1.2 Leitores e Escritores (2)

O problema:

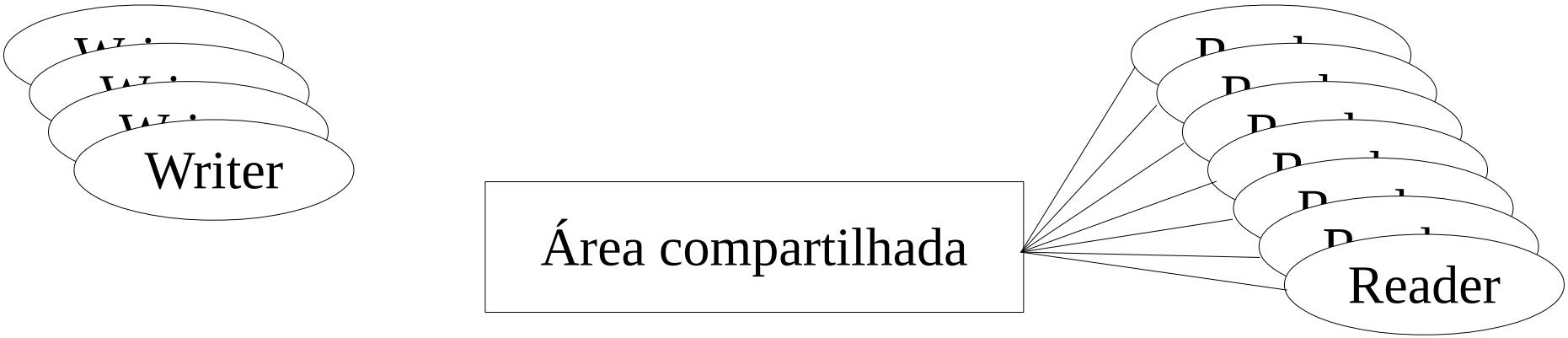
- 1 área compartilhada
- 2 Grupos de processos cooperantes:
  - Escritores lêem/escrevem no buffer: exclusão mútua
  - Leitores lêem o que foi escrito: exclusão parcial



1.2 Leitores e Escritores (3)

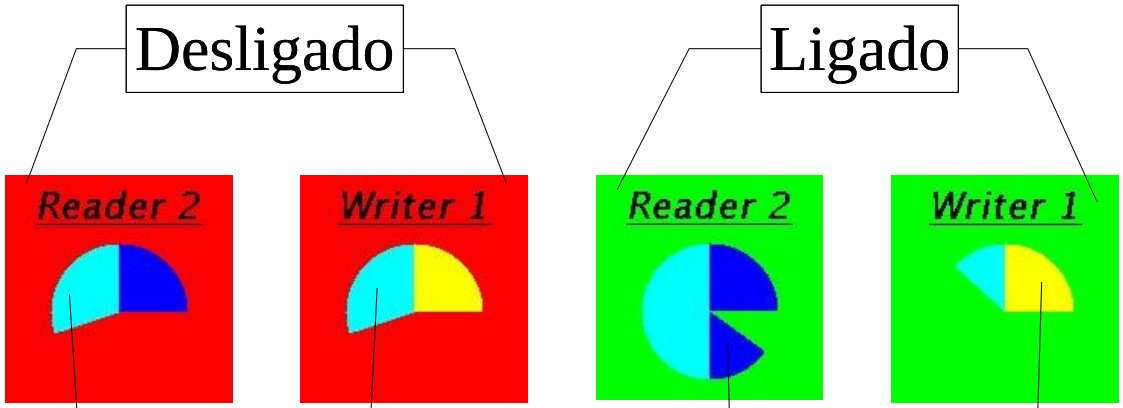
O problema:

- 1 área compartilhada
- 2 Grupos de processos cooperantes:
  - Escritores lêem/escrevem no buffer: exclusão mútua
  - Leitores lêem o que foi escrito: exclusão parcial



1.2 Leitores e Escritores (4)

Vamos ver como funciona!



Hit & Run!

1.2 Leitores e Escritores (4)

```
semaphore escrevendo = 1;// Livre p/ escrever
semaphore lendo = 1;     // Livre p/ ler
int leitores = 0;        // Quantos lendo
```

```
leitor() {
  while (true);
    wait(lendo);
    leitores++;
    if (leitores == 1)
      wait(escrevendo);
    signal(lendo);
    le();
    wait(lendo);
    leitores--;
    if (!leitores)
      signal(escrevendo);
    }
  }
}
```

```
escritor() {
  while (true);
    wait(escrevendo);
    escreve();
    signal(escrevendo);
  }
}
```

```
Problema?
   ... Sim!
   Leitores podem
   sufocar escritores!
```

Hit & Run!

1.3 O Barbeiro Dorminhoco (1)

A barbearia tem:

- Um barbeiro
- Uma cadeira de barbeiro
- Algumas cadeiras para os fregueses esperarem

Movimento fraco => barbeiro senta e dorme

Cliente chega no salão vazio => Acorda barbeiro

Cliente chega e barbeiro ocupado =>

- Assenta numa cadeira
- Vai embora se não há cadeira livre

1.3 O Barbeiro Dorminhoco (2)

```
#define CHAIRS n
semaphore
customers=0,
          barbers=0,
          mutex=1;
int waiting= 0;
```

```
Barber() {
  while (true) {
    wait(customers);
    wait(mutex);
    waiting--;
    signal(barbers);
    signal(mutex);
    cut_hair()
  }
}
```

```
Customer() {
  while (true) {
    wait(mutex);
    if (waiting < CHAIRS) {
      waiting++;
      signal(customers);
      signal(mutex);
      wait(barbers);
      get_haircut();
    }
    else
      signal(mutex);
    }
  }
}
```

Kick and cut!

Problemas Clássicos de Sincronização

Conclusão:

3 problemas clássicos:

- Jantar dos filósofos
- Leitores e Escritores
- Barbeiro dorminhoco

Modelos gerais para problemas reais de sincronização de recursos e processos