

## Documento explicativo Laboratório 5 - Sistemas Operacionais

**Equipe:** Andreza Fernandes de Oliveira  
Thaís Nascimento  
Arina de Jesus  
Fernanda Bezerra  
Pedro de Oliveira

01. Abaixo encontra-se a execução da classe ProducerConsumerExample.



```
384341@lec34: ~/Área de Trabalho
PAR CONSUMIDOR obtém 0.
PAR CONSUMIDOR obtém 0.
ÍMPAR CONSUMIDOR obtém 0.
PRODUTOR gera 9.
PAR CONSUMIDOR obtém 9.
ÍMPAR CONSUMIDOR obtém 8.
PRODUTOR gera 5.
PAR CONSUMIDOR obtém 5.
PRODUTOR gera 0.
ÍMPAR CONSUMIDOR obtém 0.
PAR CONSUMIDOR obtém 0.
PRODUTOR gera 7.
PRODUTOR gera 0.
ÍMPAR CONSUMIDOR obtém 0.
ÍMPAR CONSUMIDOR obtém 0.
ÍMPAR CONSUMIDOR obtém 0.
PAR CONSUMIDOR obtém 0.
PRODUTOR gera 7.
PAR CONSUMIDOR obtém 7.
ÍMPAR CONSUMIDOR obtém 7.
ÍMPAR CONSUMIDOR obtém 7.
PRODUTOR gera 7.
PAR CONSUMIDOR obtém 7.
PRODUTOR gera 4.
ÍMPAR CONSUMIDOR obtém 4.
PRODUTOR gera 1.
PAR CONSUMIDOR obtém 1.
PAR CONSUMIDOR obtém 1.
ÍMPAR CONSUMIDOR obtém 1.
PRODUTOR gera 5.
PAR CONSUMIDOR obtém 5.
ÍMPAR CONSUMIDOR obtém 5.
ÍMPAR CONSUMIDOR obtém 5.
ÍMPAR CONSUMIDOR obtém 5.
PRODUTOR gera 1.
PAR CONSUMIDOR obtém 1.
ÍMPAR CONSUMIDOR obtém 1.
PRODUTOR gera 6.
PRODUTOR gera 1.
ÍMPAR CONSUMIDOR obtém 1.
PRODUTOR gera 0.
PRODUTOR gera 1.
PAR CONSUMIDOR obtém 1.
ÍMPAR CONSUMIDOR obtém 1.
PAR CONSUMIDOR obtém 1.
PRODUTOR gera 8.
ÍMPAR CONSUMIDOR obtém 8.
PAR CONSUMIDOR obtém 8.
PAR CONSUMIDOR obtém 8.
PRODUTOR gera 1.
PRODUTOR gera 0.
ÍMPAR CONSUMIDOR obtém 0.
PRODUTOR gera 5.
PAR CONSUMIDOR obtém 5.
PRODUTOR gera 5.
ÍMPAR CONSUMIDOR obtém 5.
ÍMPAR CONSUMIDOR obtém 5.
PAR CONSUMIDOR obtém 5.
PRODUTOR gera 3.
PAR CONSUMIDOR obtém 3.
PRODUTOR gera 0.
```

Ao observar essa imagem notamos alguns problemas:

- O consumidor não faz a distinção entre os números pares e ímpares;
- Os números não estão sendo removidos.

Esse problema se dá pela falta de sincronismo entre as threads consumidores e o produtor.

O que acontece é que o produtor está produzindo sem parar e os consumidores estão acessando a função take da classe Dropbox sem fazer nenhuma distinção, o que implica que as duas threads consumidoras poderão consumir aquele número mais de uma vez, até que o produtor produza um novo número.

O ideal seria: O produtor produziria e aguardaria o consumidor adequado (ímpar ou par) iria consumir aquele número para produzir um novo número. O consumidor par deve consumir o número produzido somente se for par, e o mesmo vale para o consumidor ímpar. Se ao entrar na função take o número disponível não for da paridade correspondente a thread consumidora, a thread deverá aguardar.

E por fim, sempre que produzido um novo número, as threads deverão ser acordadas.

Essa foi a lógica utilizada para a solução do problema! Fizemos uso de variáveis booleanas para controlar a disponibilidade do Produtor e Consumidor, além do uso de mecanismo de sincronização.