Exercício 02

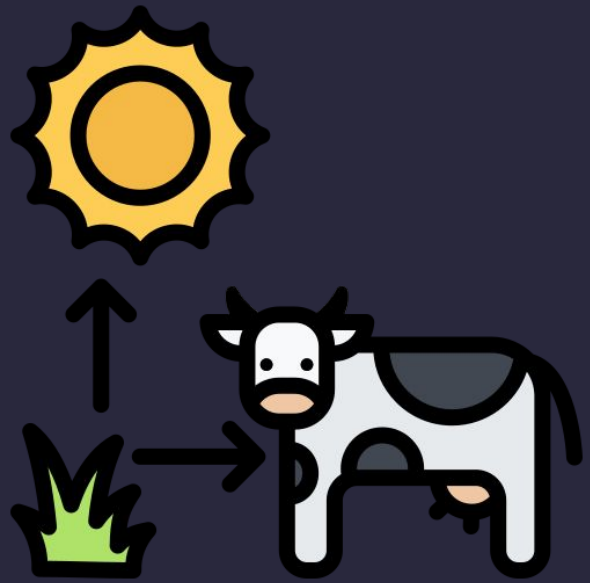
Prog. Concorrente e Distribuída (IF711)

/TABLE OF CONTENTS

/01

QUESTÃO 01

Produtor x Consumidor



/PRODUTOR

/CONSUMIDOR

/PRODUCER

```
1.  void* producer(void* args) {
2.    while (1) {
3.      int x = rand() % 100;
4.      sleep(1);
5.      sem_wait(&semEmpty);
6.      pthread_mutex_lock(&mutexBuffer);
7.      buffer[count] = x;
8.      count++;
9.      pthread_mutex_unlock(&mutexBuffer);
10.     sem_post(&semFull);
11.   }
12. }
```

Adicionamos no buffer, se não já tiver adicionado, caso já tenhamos, o produtor espera.

/CONSUMER

```
1.  void* consumer(void* args) {
2.      while (1) {
3.          int y;
4.          sem_wait(&semFull);
5.          pthread_mutex_lock(&mutexBuffer);
6.          y = buffer[0];
7.          for (int i = 0 ; i < count-1 ; i++){
8.              buffer[i]=buffer[i+1];
9.          }
10.         count--;
11.         pthread_mutex_unlock(&mutexBuffer);
12.         sem_post(&semEmpty);
13.         printf("Got %d\n", y);
14.         sleep(1);
15.     }
16. }
```

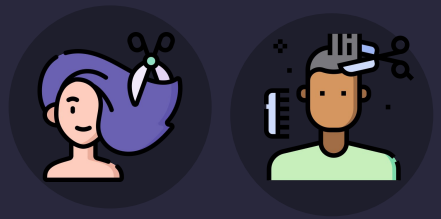
Retiramos valor do buffer, se o buffer estiver vazio, os consumidores esperam até o produtor colocar algo.

/02

QUESTÃO 02

Clientes x Barbeiro

/PRODUTOR

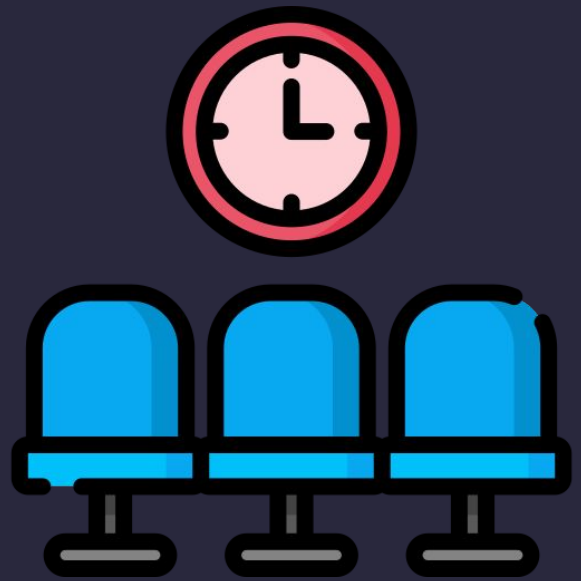


/CONSUMIDOR

/BUFFER DE EVENTOS

Recebe os eventos de que os clientes querem cortar o cabelo e armazena em um sistema FIFO

```
1.  pthread_mutex_t mutexBuffer;
2.  int buffer[15];
```



/ DesejoCortarCabelo

```
1.  void* DesejoCortarCabelo(void* args) {
2.    while (1) {
3.      int x = rand() % 100;
4.      sleep(1);
5.      sem_wait(&semEmpty);
6.      pthread_mutex_lock(&mutexBuffer);
7.      buffer[count] = x;
8.      count++;
9.      pthread_mutex_unlock(&mutexBuffer);
10.     sem_post(&semFull);
11.   }
12. }
```

Clientes que chegam na barbearia chamam essa função, eles se comportam como o "produtor". Chegada dos clientes é controlada pelo buffer, se o buffer atingir capacidade máxima, clientes não podem entrar.

/ CortarCabelo

```
1.  void* CortarCabelo(void* args) {
2.      while (1) {
3.          int y;
4.          sem_wait(&semFull);
5.          pthread_mutex_lock(&mutexBuffer);
6.          y = buffer[0];
7.          for (int i = 0 ; i < count-1 ; i++){
8.              buffer[i]=buffer[i+1];
9.          }
10.         count--;
11.         pthread_mutex_unlock(&mutexBuffer);
12.         sem_post(&semEmpty);
13.         sleep(1);
14.     }
15. }
```

Barbeiro se comporta como consumidor, vai consumindo os eventos colocados no buffer. Se não existem clientes no buffer, ele espera.

```
14 esperando
40 esperando
54 esperando
77 esperando
-> 14 cortou o cabelo.
2 esperando
77 esperando
-> 40 cortou o cabelo.
71 esperando
-> 54 cortou o cabelo.
78 esperando
86 esperando
-> 77 cortou o cabelo.
34 esperando
56 esperando
```

/OBRIGADA!

/ALGUMA PERGUNTA?

Eneri emd
Nathalia npl
Wilson jwcfj