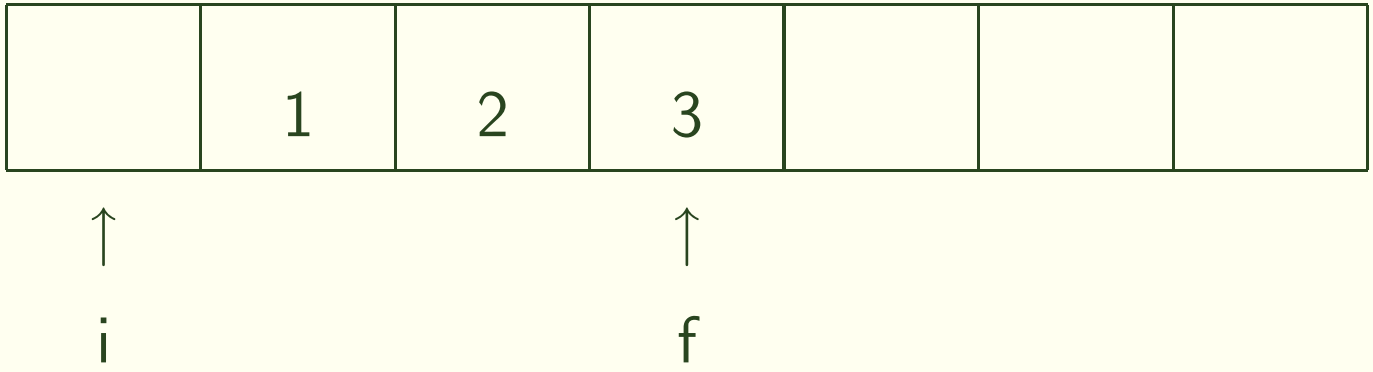
MO806/MC914
Tópicos em Sistemas Operacionais
2s2006

Produtor e Consumidor

Problema do Produtor-Consumidor

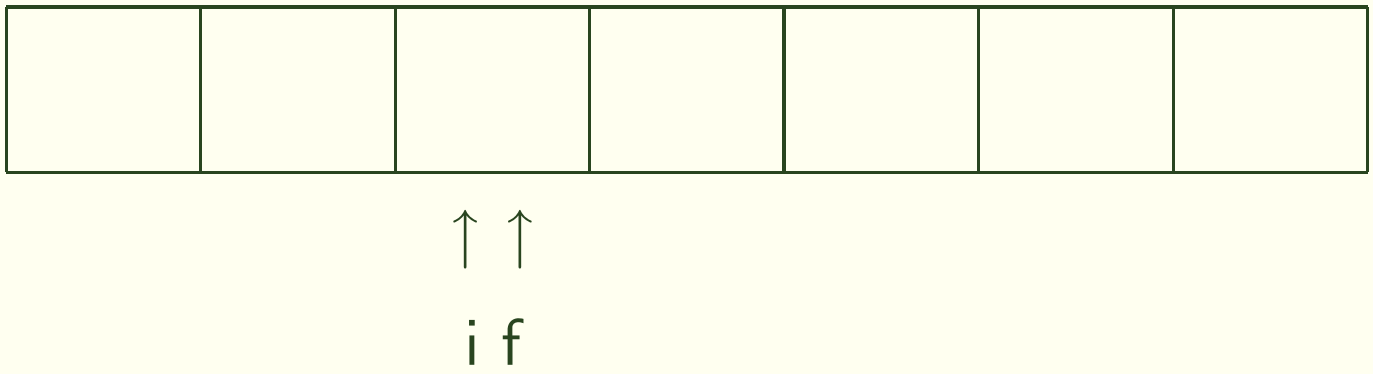
- Dois processos compartilham um $buffer$ de tamanho fixo
- O produtor insere informação no $buffer$
- O consumidor remove informação do $buffer$

Controle do buffer



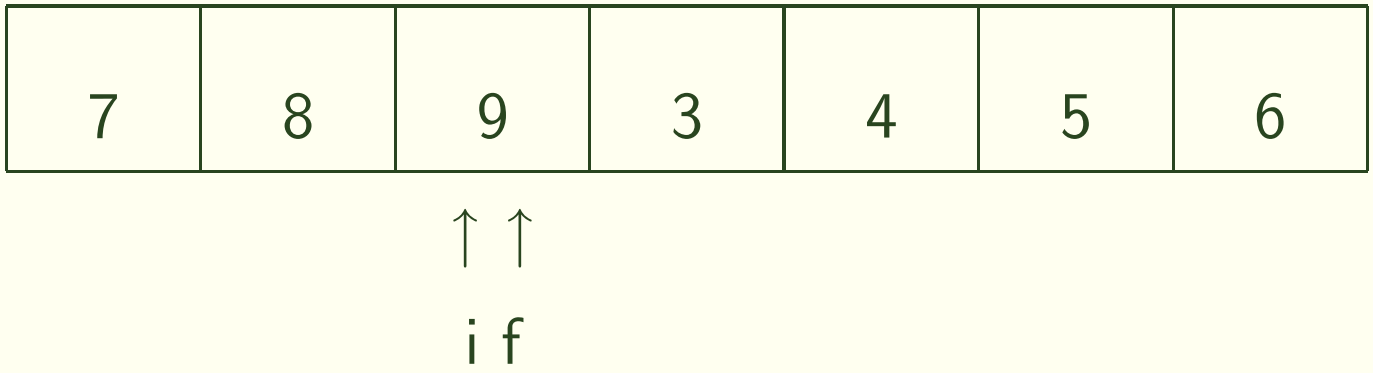
- **i:** aponta para a posição anterior ao primeiro elemento
- **f:** aponta para o último elemento
- **c:** indica o número de elementos presentes
- **N:** indica o número máximo de elementos

Buffer vazio



- **i == f**
- **c == 0**

Buffer cheio



- **i == f**
- **c == N**

Comportamento básico

```
int buffer[N];
 int c = 0;
 int i = 0, f = 0;
```

Produtor

```
  while (true)
    f = (f+1)%N;
    buffer[f]= produz();
    c++;
```

Consumidor

```
 while (true)
    i = (i+1)%N;
    consome(buffer [i]);
    c--;
```

Veja código: prod-cons-basico.c

Problemas

1. produtor insere em posição que *ainda* não foi consumida
2. consumidor remove de posição *já* foi consumida

Veja código: prod-cons-basico-bug.c

Tentativa com espera ocupada

```
int buffer[N];
 int c = 0;
 int i = 0, f = 0;
```

Produtor

```
while (true)
  while (c == N);
  f = (f+1)%N;
  buffer[f]= produz();
  c++;
```

Consumidor

```
while (true)
  while (c == 0);
  i = (i+1)%N;
  consome(buffer[i]);
  c--;
```

Veja código: prod-cons-basico-busy-wait.c

Condição de disputa

Produtor	Consumidor
`c++;`	`c--;`
`mov rp,c`	`mov rc,c`
`inc rp`	`dec rc`
`mov c,rp`	`mov c,rc`

Veja código: prod-cons-basico-race.c

Tentativa com sleep

```
int buffer[N];
 int c = 0;
 int i = 0, f = 0;
```

Produtor

```
  while (true)
    if (c == N) sleep();
    f = (f + 1);
    buffer[f]= produz();
    c++;
    if (c == 1)
      wakeup_consumidor();
```

Consumidor

```
 while (true)
   if (c == 0) sleep();
   i = (i+1);
   consome(buffer [i]);
   c--;
   if (c == N - 1)
     wakeup_produtor();
```

Lost wakeup

- Um processo pode receber a chamada wakeup enquanto ainda não estava dormindo

Semáforos

- Semáforos são *contadores especiais* para recursos compartilhados.
- Proposto por Dijkstra (1965)
- Operações básicas (atômicas):
  - decremento (down, wait ou P)
    bloqueia se o contador for nulo
  - incremento (up, signal (post) ou V)
    nunca bloqueia

Semáforos

Comportamento básico

- sem_init(s, 5)
- wait(s)

```
if (s == 0)
   bloqueia_processo();
else s--;
```

- signal(s)

```
if (s == 0 && existe processo bloqueado)
  acorda_processo();
else s++;
```

Produtor-Consumidor com Semáforos

```
semaforo cheio = 0;
semaforo vazio = N;
```

Produtor:

```
while (true)
    wait(vazio);
    f = (f+1)%N;
    buffer[f] = produz();
    signal(cheio);
```

Consumidor:

```
while (true)
    wait(cheio);
    i = (i+1)%N;
    consome(buffer[i]);
    signal(vazio);
```

Veja código: prod-cons-sem.c

Vários produtores e consumidores

```
semaforo cheio = 0, vazio = N;
semaforo lock_prod = 1, lock_cons = 1;
```

Produtor:

```
while (true)
    wait(vazio);
    wait(lock_prod);
    f = (f + 1) % N;
    buffer[f] = produz();
    signal(lock_prod);
    signal(cheio);
```

Consumidor:

```
while (true)
    wait(cheio);
    wait(lock_cons);
    i = (i + 1) % N;
    consome(buffer[i]);
    signal(lock_cons);
    signal(vazio);
```

Vários produtores e consumidores

```
semaforo cheio = 0, vazio = N;
semaforo lock_prod = 1, lock_cons = 1;
```

Produtor:

```
while (true)
    item = produz();
    wait(vazio);
    wait(lock_prod);
    f = (f + 1) % N;
    buffer[f] = item;
    signal(lock_prod);
    signal(cheio);
```

Consumidor:

```
while (true)

    wait(cheio);
    wait(lock_cons);
    i = (i + 1) % N;
    item = buffer[i];
    signal(lock_cons);
    signal(vazio);
    consome(item);
```