

MC504/MC514 - Sistemas Operacionais

Semáforos e os Filósofos Famintos

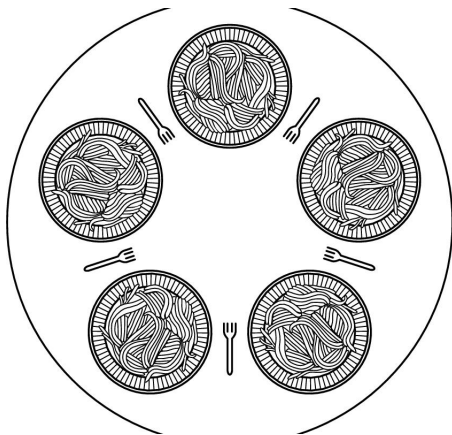
Islene Calciolari Garcia

Primeiro Semestre de 2016

Sumário

- 1 Introdução
- 2 Semáforos
- 3 Controle no garfo
- 4 Controle no filósofo

Jantar dos Filósofos

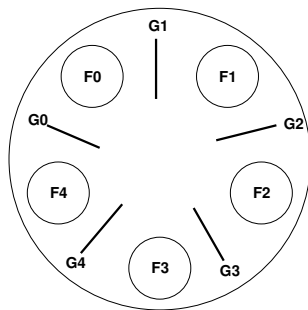


Tanenbaum: Figura 1.20

Boas soluções

- ausência de *deadlock*
- ausência de *starvation*
- alto grau de paralelismo

Representação da mesa



```
- T - T - T - T - T -  
- T - T - H - T - T -  
- T - T |H| T - T -  
- T - T |E| T - T -
```

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++;`

Implementação com Semáforos

Um semáforo por garfo

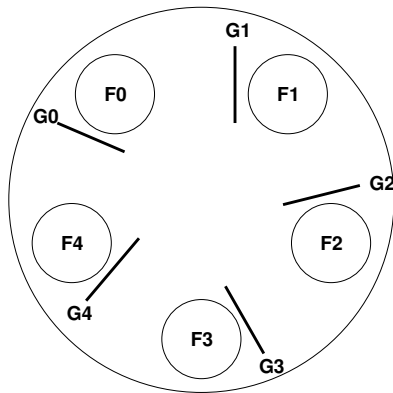
- `sem_init(&garfo, 1)`
- `wait(&garfo)`
- `signal(&garfo)`

Implementação simplista

Filósofo i:

```
while (true)
    pensa();
    wait(garfo[i]);
    wait(garfo[(i+1) % N]);
    come();
    signal(garfo[i]);
    signal(garfo[(i+1) % N]);
```

Deadlock



Veja códigos: `deadlock.c` e `deadlock-bug-exibicao.c`

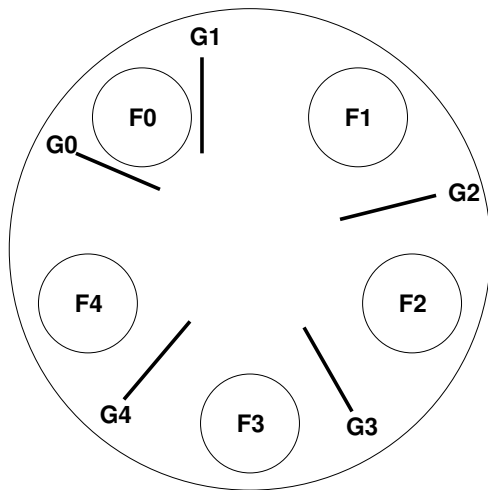
Outra tentativa...

```
semaforo lock = 1;
```

Filósofo i:

```
while (true)
    pensa();
    wait(lock);
    wait(garfo[i]);
    wait(garfo[(i+1) % N]);
    come();
    signal(garfo[(i+1) % N]);
    signal(garfo[i]);
    signal(lock);
```

Baixíssimo paralelismo



Veja código: `sem_central.c`

O que acontece se `lock == 2`?

```
semaforo lock = 2;
```

Filósofo i:

```
while (true)
    pensa();
    wait(lock);
    wait(garfo[i]);
    wait(garfo[(i+1) % N]);
    come();
    signal(garfo[(i+1) % N]);
    signal(garfo[i]);
    signal(lock);
```

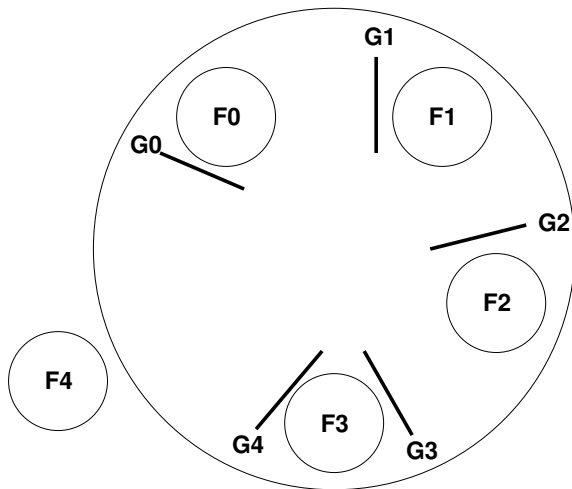
Menos lugares à mesa

```
semaforo lugar_mesa = N-1;
```

Filósofo i:

```
while (true)
    pensa();
    wait(lugar_mesa);
    wait(garfo[i]);
    wait(garfo[(i+1) % N]);
    come();
    signal(garfo[(i+1) % N]);
    signal(garfo[i]);
    signal(lugar_mesa);
```

Menos lugares à mesa

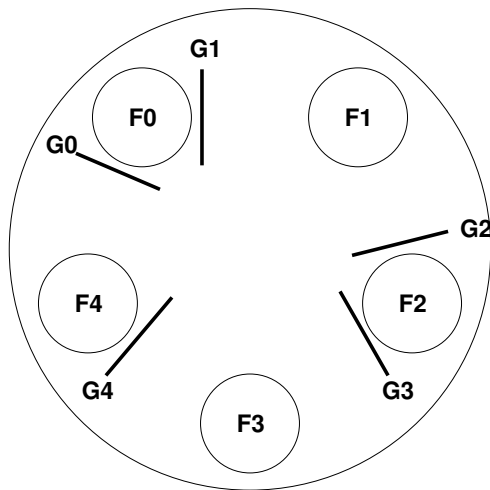


Veja código: `limite_lugares.c`

Solução assimétrica

```
while (true)
    pensa();
    if (i % 2 == 0)
        wait(garfo[i]);
        wait(garfo[(i+1) % N]);
    else
        wait(garfo[(i+1) % N]);
        wait(garfo[i]);
    come();
    signal(garfo[(i+1) % N]);
    signal(garfo[i]);
```

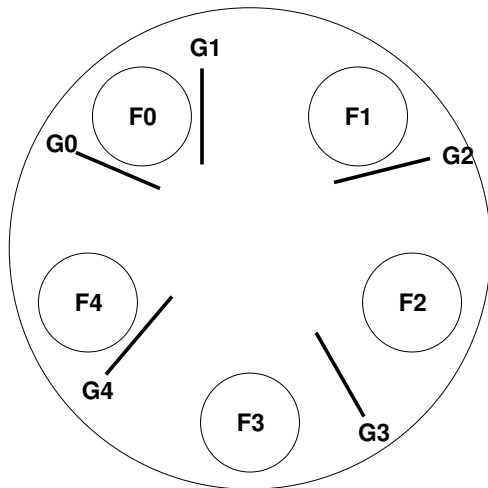

Solução assimétrica



Veja código: `assimetrica.c`

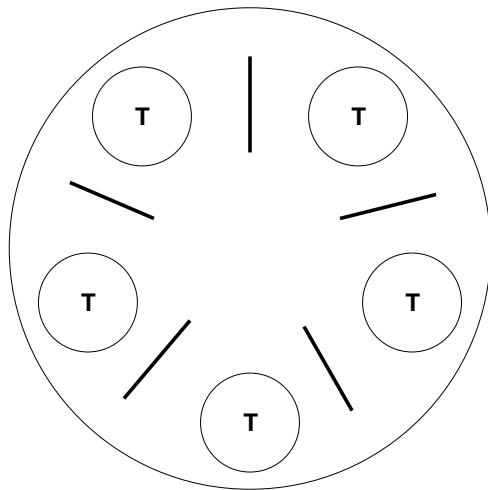
Solução assimétrica

Baixo paralelismo?!



Filósofos famintos

Um semáforo por filósofo



Solução do livro Tanenbaum

```
semaforo lock;  
semaforo filosofo[N] = {0, 0, 0, ..., 0}  
int estado[N] = {T, T, T, ..., T}
```

Filósofo i:

```
while (true)  
    pensa();  
    pega_garfos();  
    come();  
    solta_garfos();
```

Solução do livro Tanenbaum

Codificação simplificada

```
pega_garfos()
    wait(lock);
    if (estado[fil_esq] != E && estado[fil_dir] != E)
        estado[i] = E;
        signal(lock);
    else
        estado[i] = T;
        signal(lock);
        wait(filosofo[i]);
```

Solução do livro Tanenbaum

Codificação simplificada

```
testa_garfos(int i)
    if (estado[i] == H && estado[fil_esq] != E &&
        estado[fil_dir] != E)
        estado[i] = E;
        signal(filosofo[i]);
```

```
solta_garfos()
    wait(lock);
    estado[i] = T;
    testa_garfos(fil_esq);
    testa_garfos(fil_dir);
    signal(lock);
```

Solução do livro Tanenbaum

Codificação original

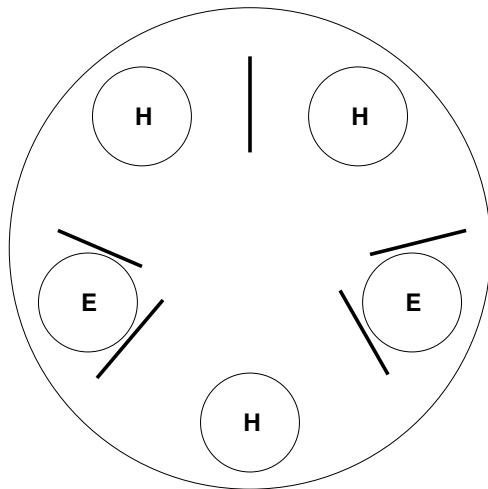
```
testa_garfos(int i)
    if (estado[i] == H && estado[fil_esq] != E &&
        estado[fil_dir] != E)
        estado[i] = E;
        signal(filosofo[i]);
```

```
pega_garfos()
    wait(lock);
    estado[i] = H;
    testa_garfos(i);
    signal(lock);
    wait(filosofo[i]);
```

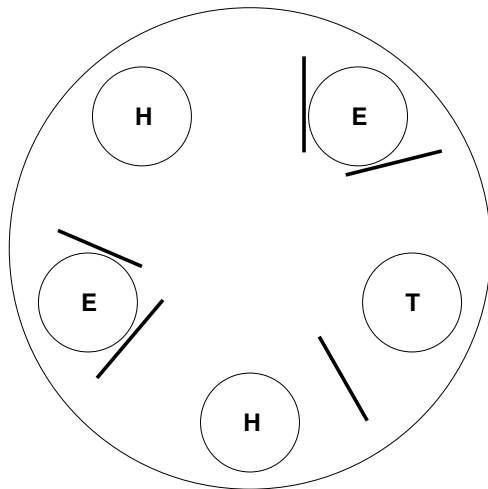
```
solta_garfos()
    wait(lock);
    estado[i] = T;
    testa_garfos(fil_esq);
    testa_garfos(fil_dir);
    signal(lock);
```

Veja código: `tanen-sem.c`

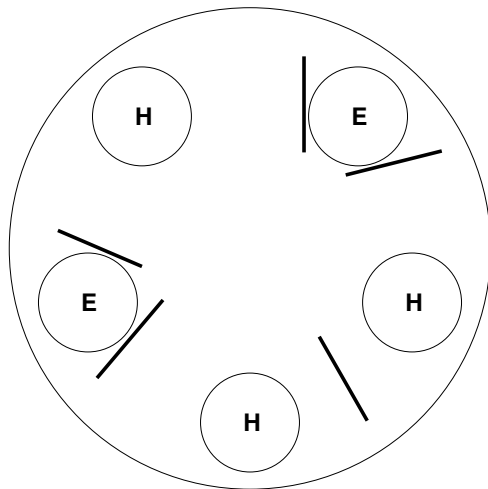
Alto paralelismo



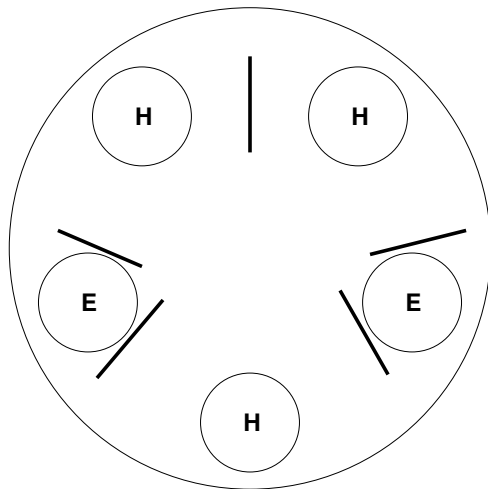
Alto paralelismo



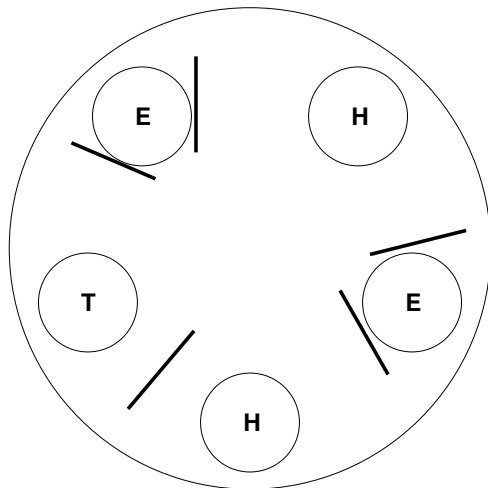
Alto paralelismo



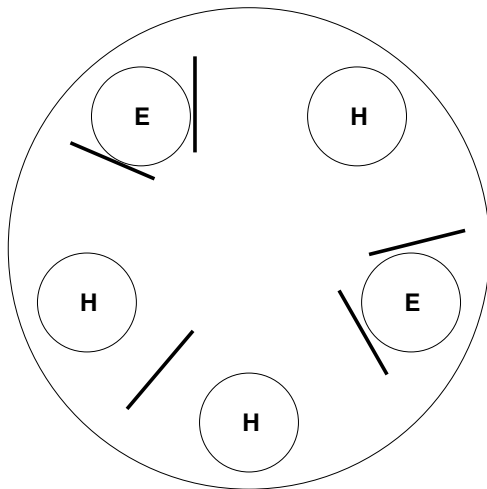
Alto paralelismo



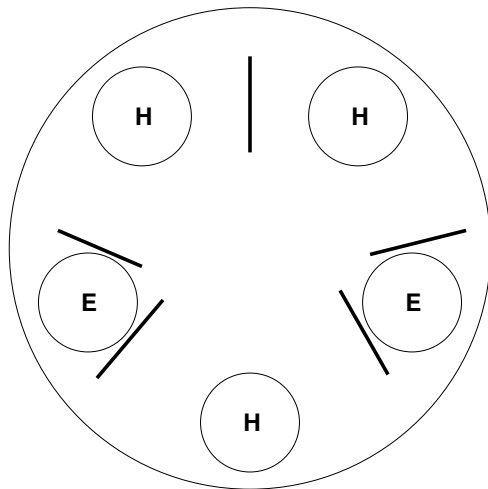
Alto paralelismo



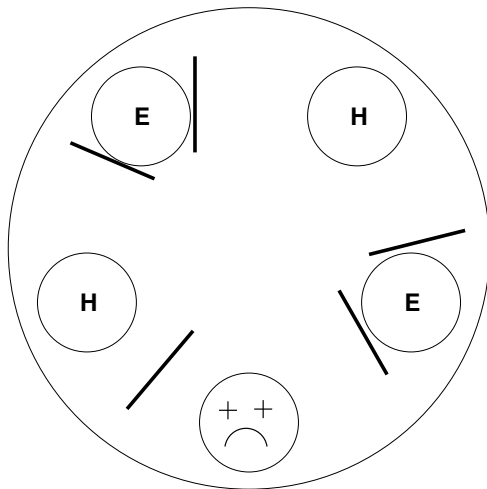
Alto paralelismo



Alto paralelismo



Starvation



Como matar os filósofos de fome?

- É preciso ajustar os tempos.
- Veja o código: `tanen-4-2.c` e `tanen-5-1.c`
- Como implementar `tanen-8-2.c`?
- É possível garantir alto paralelismo e ausência de starvation ao mesmo tempo?