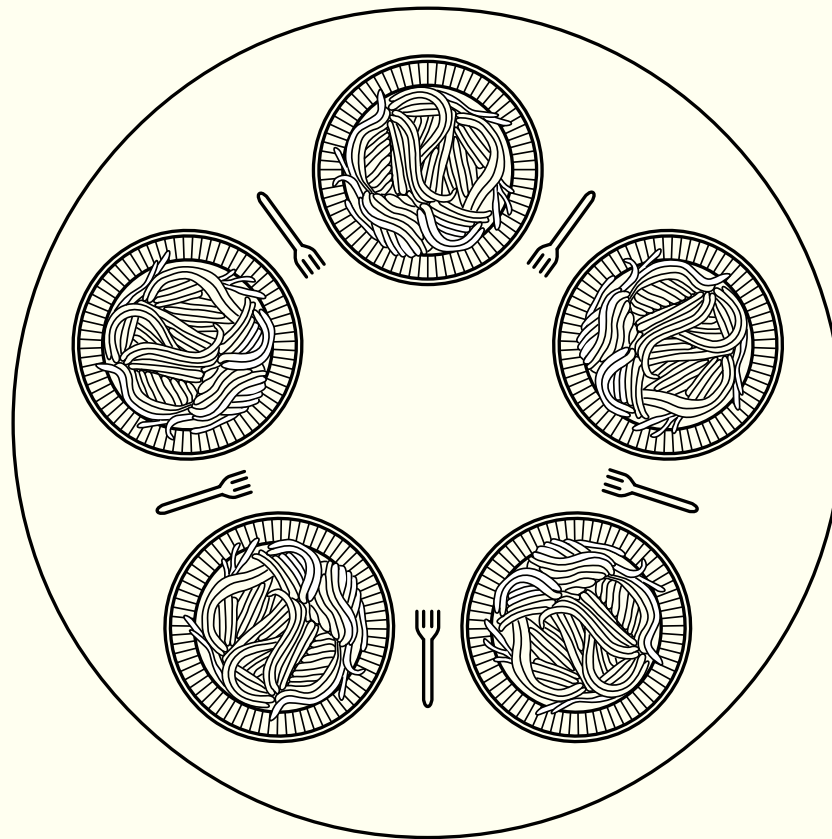


**MO806/MC914**  
**Tópicos em Sistemas Operacionais**  
2s2008

**Filósofos Famintos**

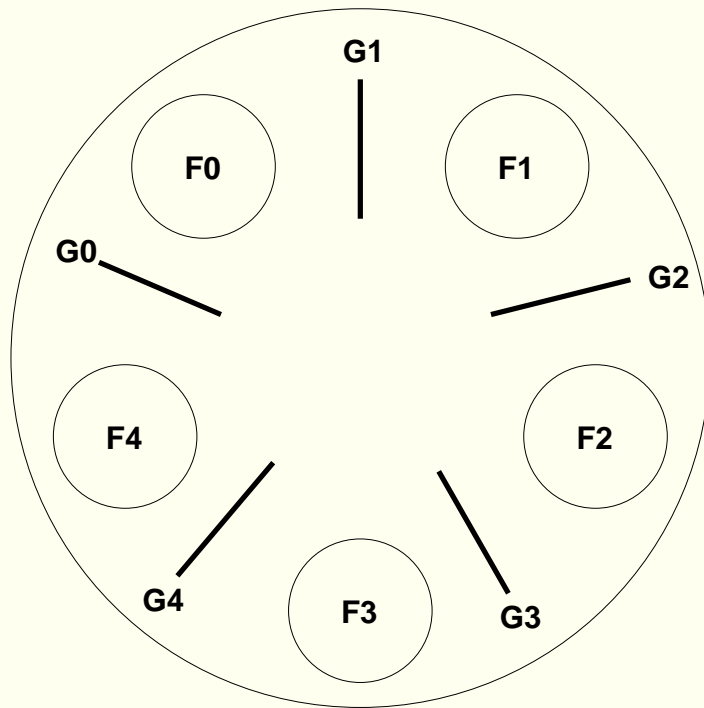
# Jantar dos Filósofos



## 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	-

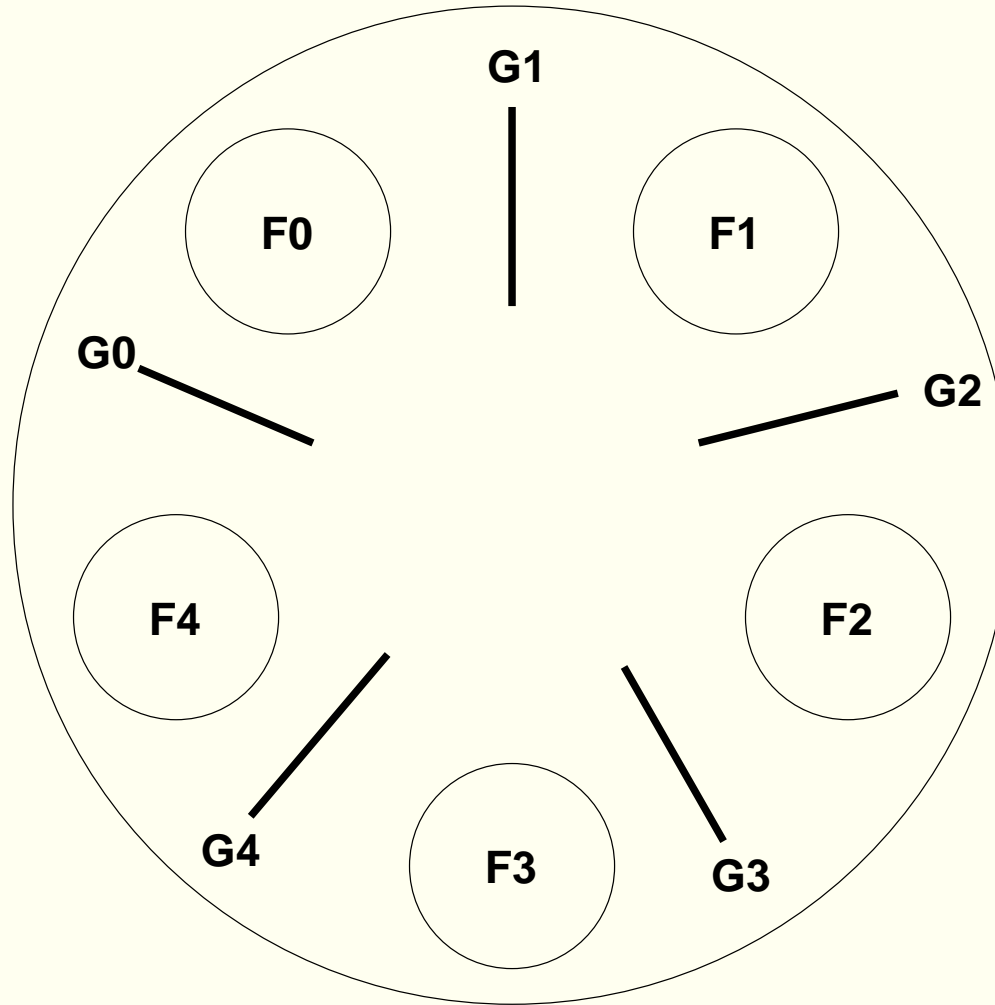
# Implementação com Semáforos

## Um semáforo por garfo

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

# Filósofos famintos

## Um semáforo por garfo

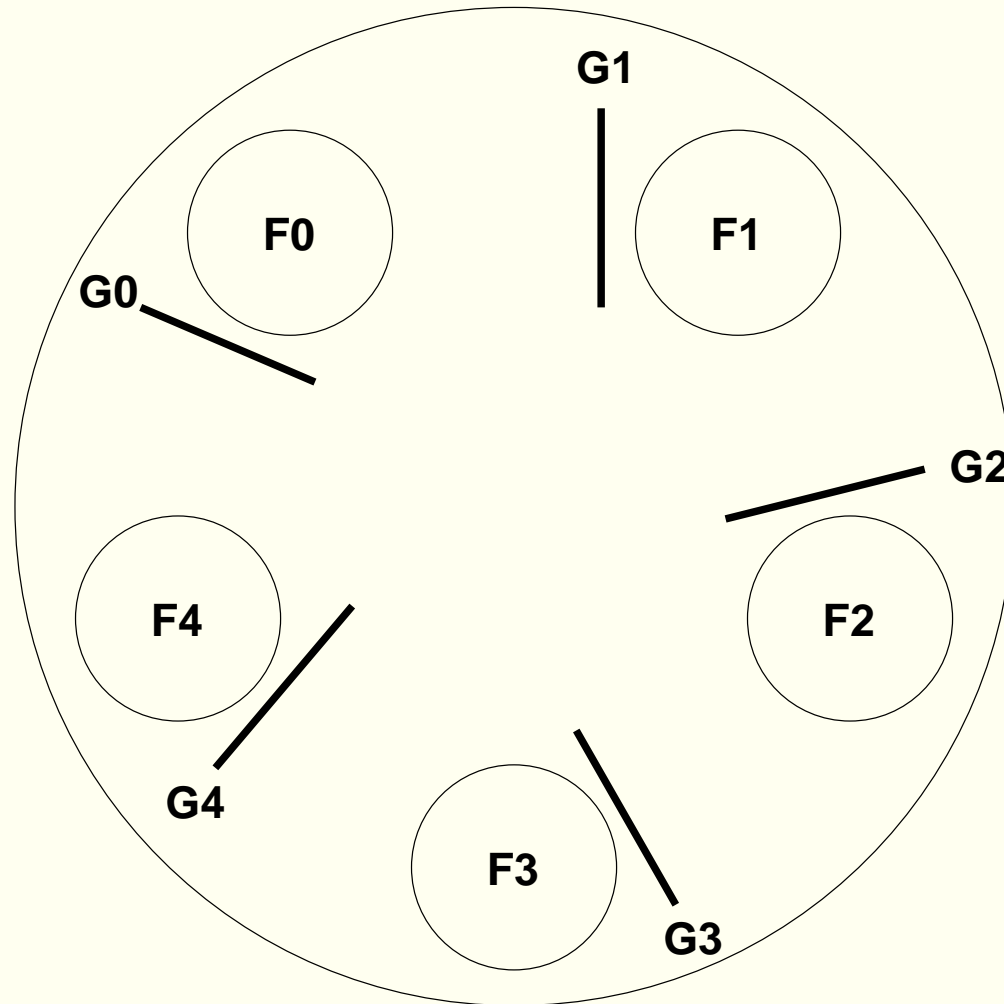


# Primeira tentativa

## 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`



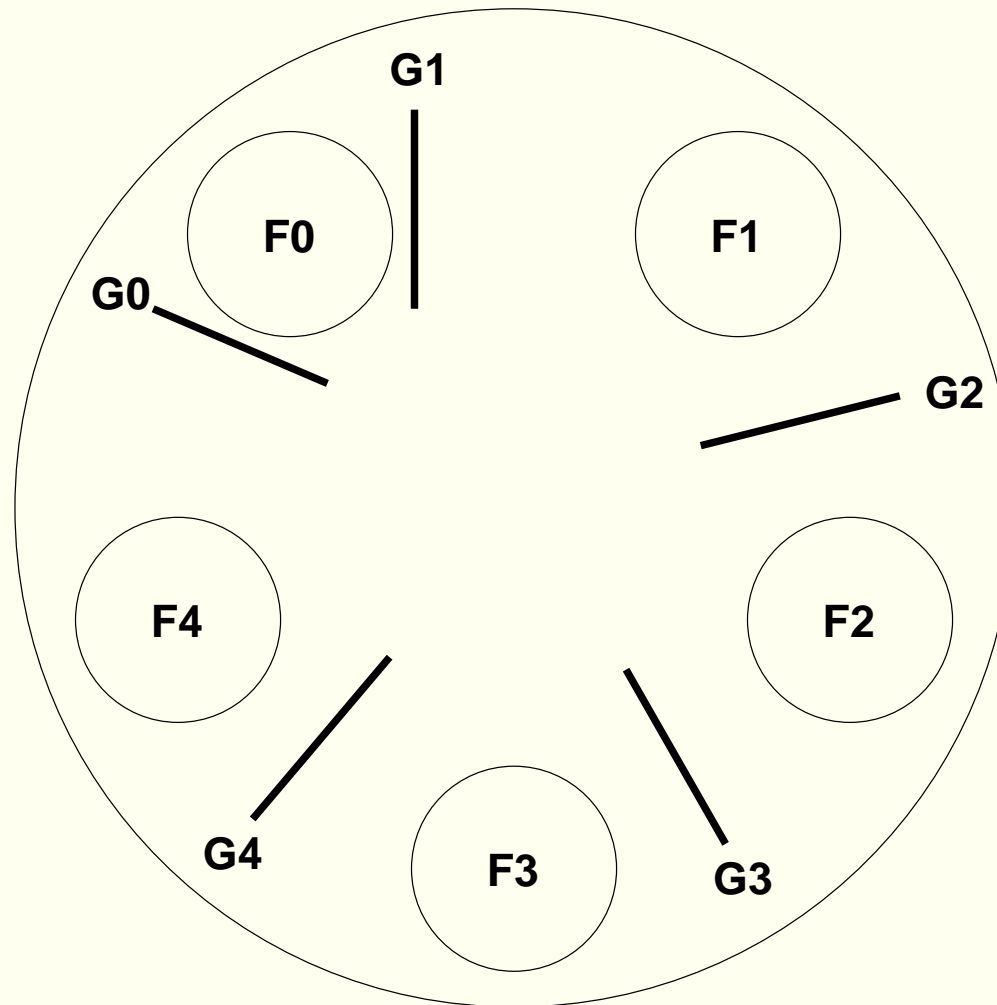
## Outra tentativa...

```
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);
```

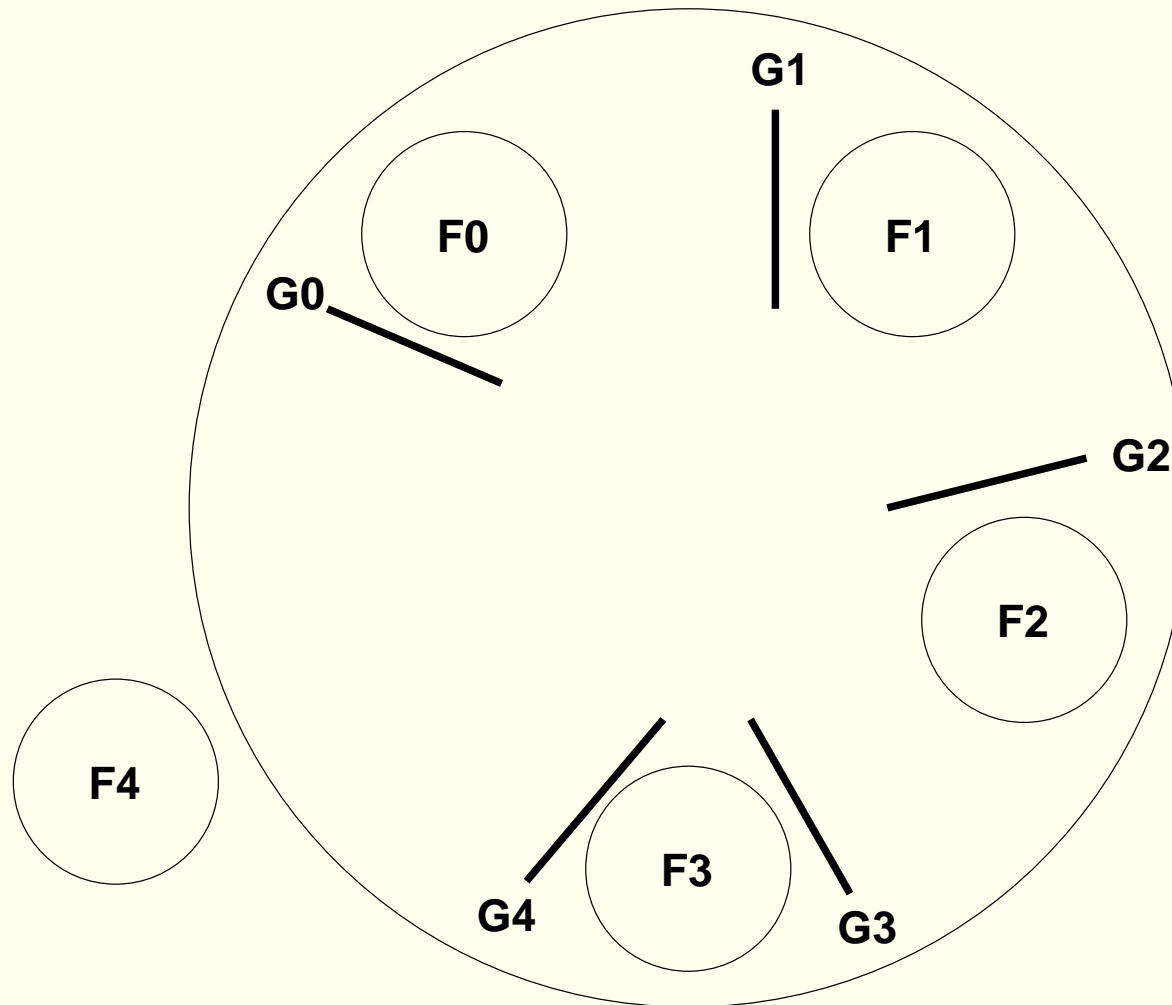
## Menos lugares à mesa

```
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

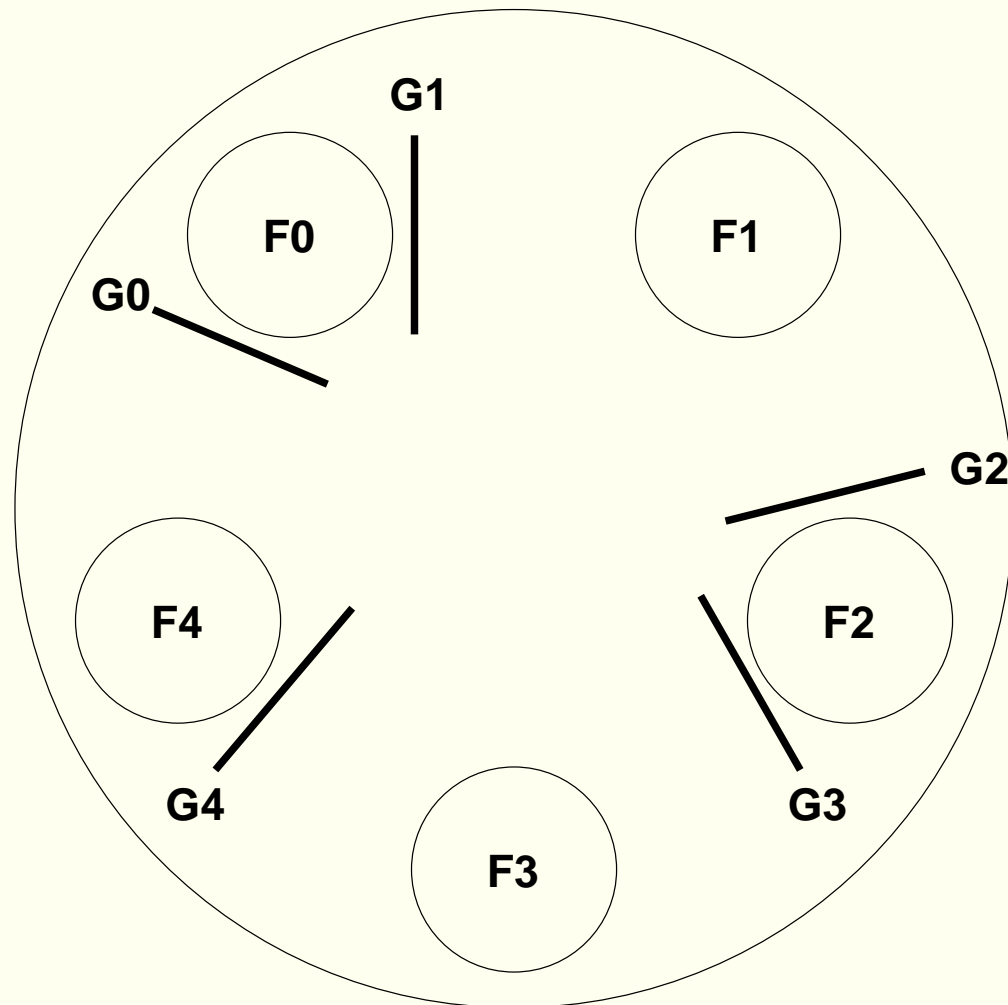


Veja código: `limite_lugares.c`

## 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]);
```

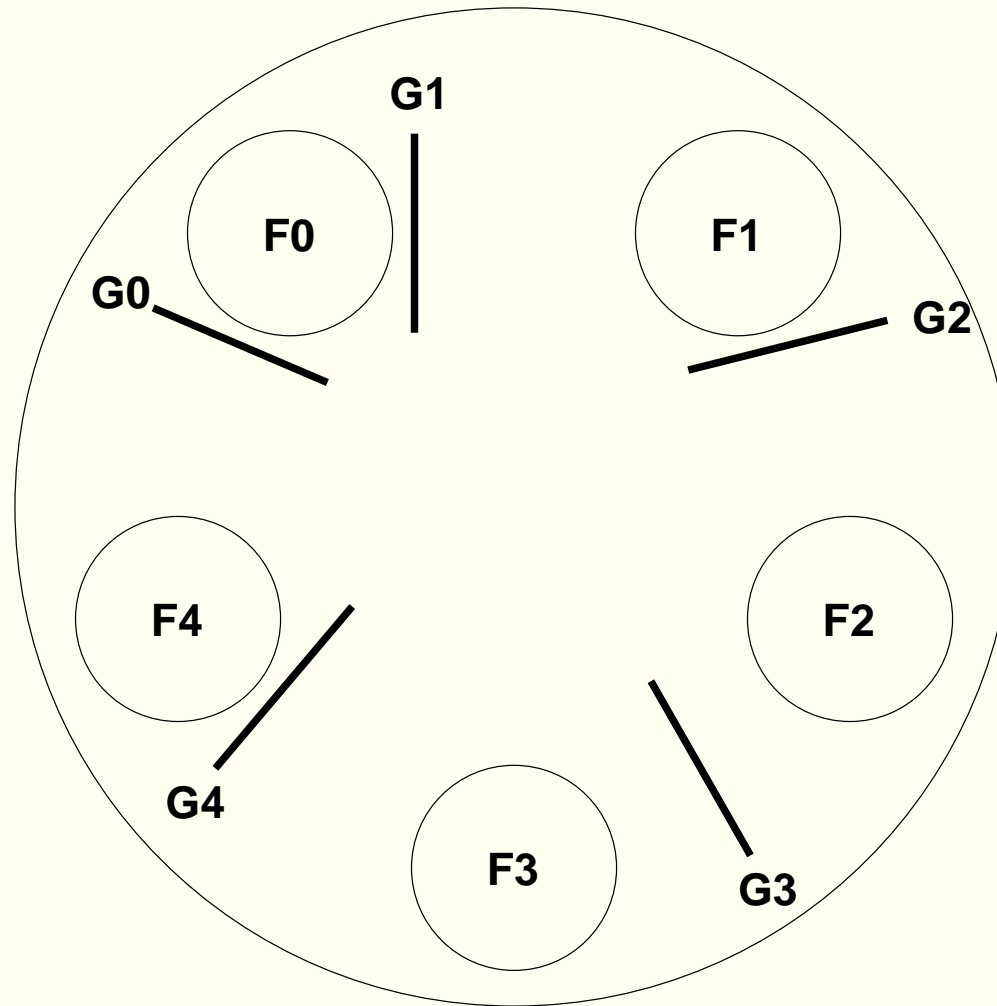
# Solução assimétrica



Veja código: `assimetrica.c`

# Solução assimétrica

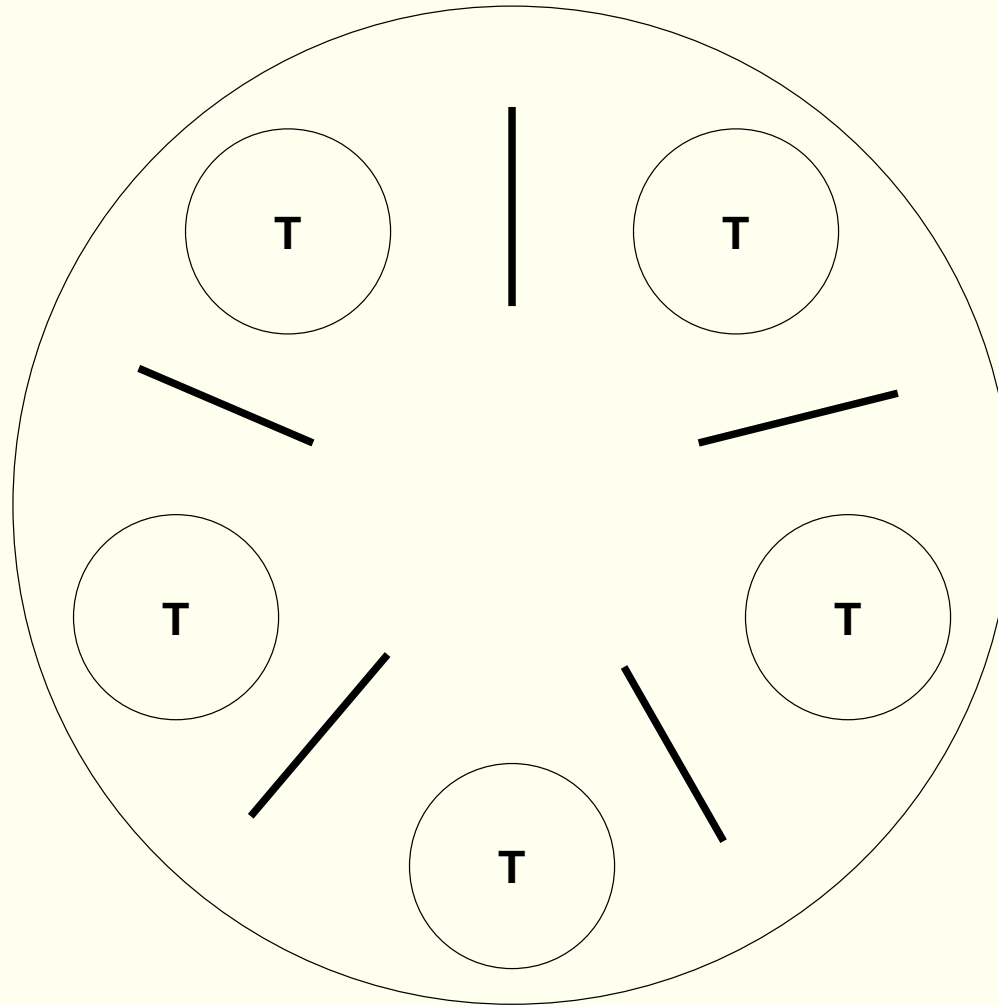
## Baixo paralelismo?!





# 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:

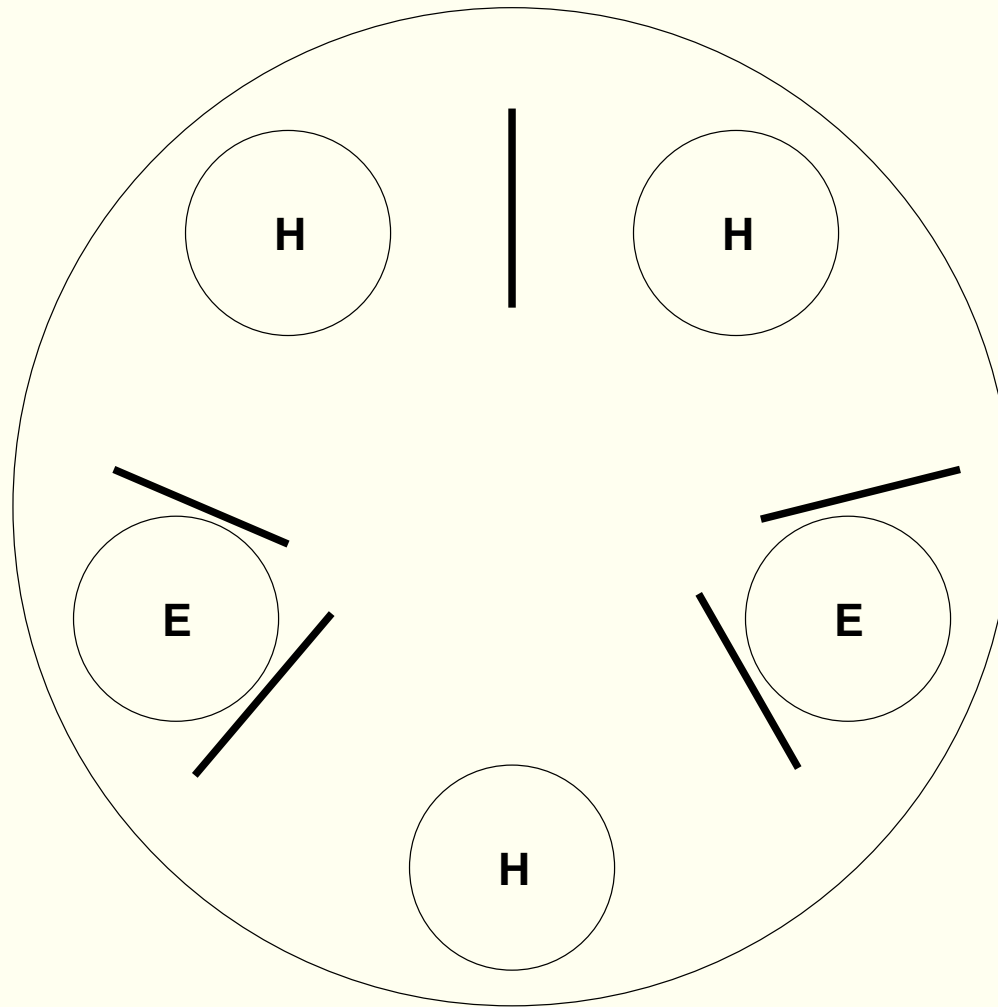
```
while (true)  
    pensa();  
    pega_garfos();  
    come();  
    solta_garfos();
```

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

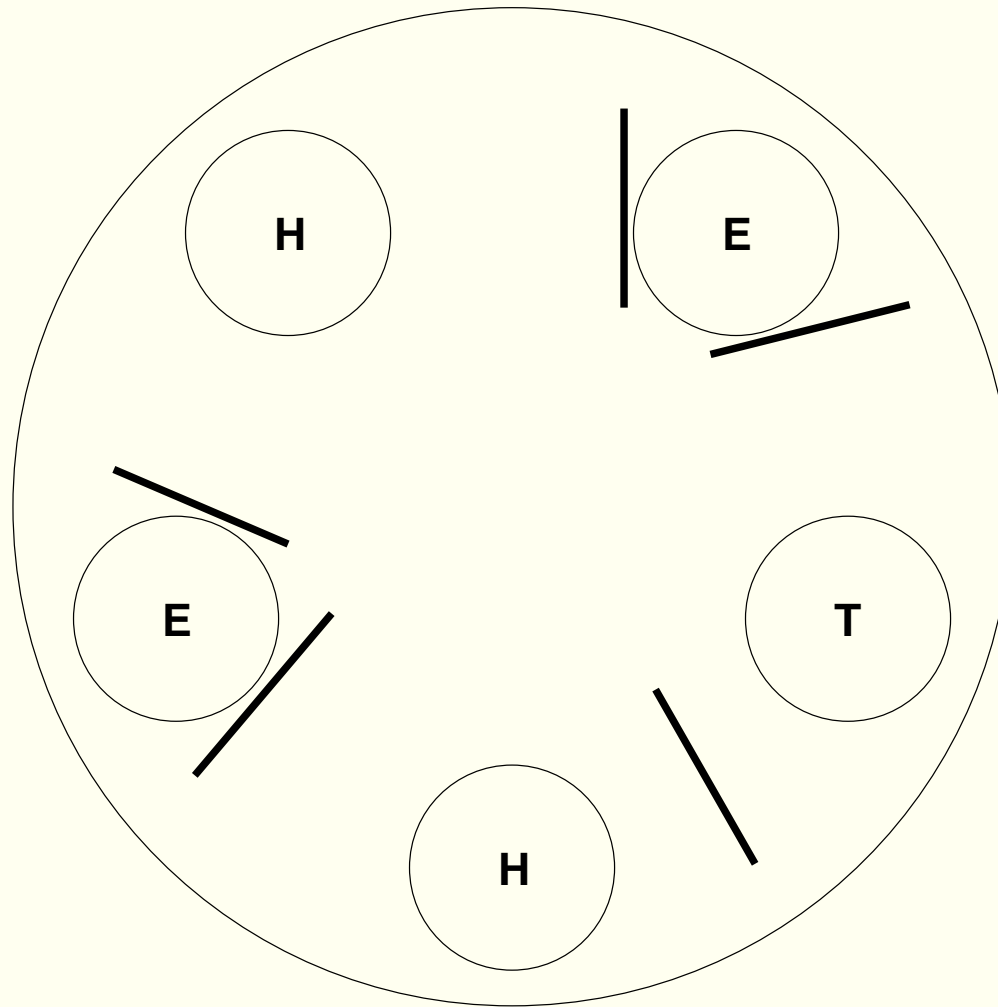
```
pega_garfos()
    wait(lock);
    estado[i] = H;
    testa_garfos(i);
    signal(lock);
    wait(filosofo[i]);
```

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

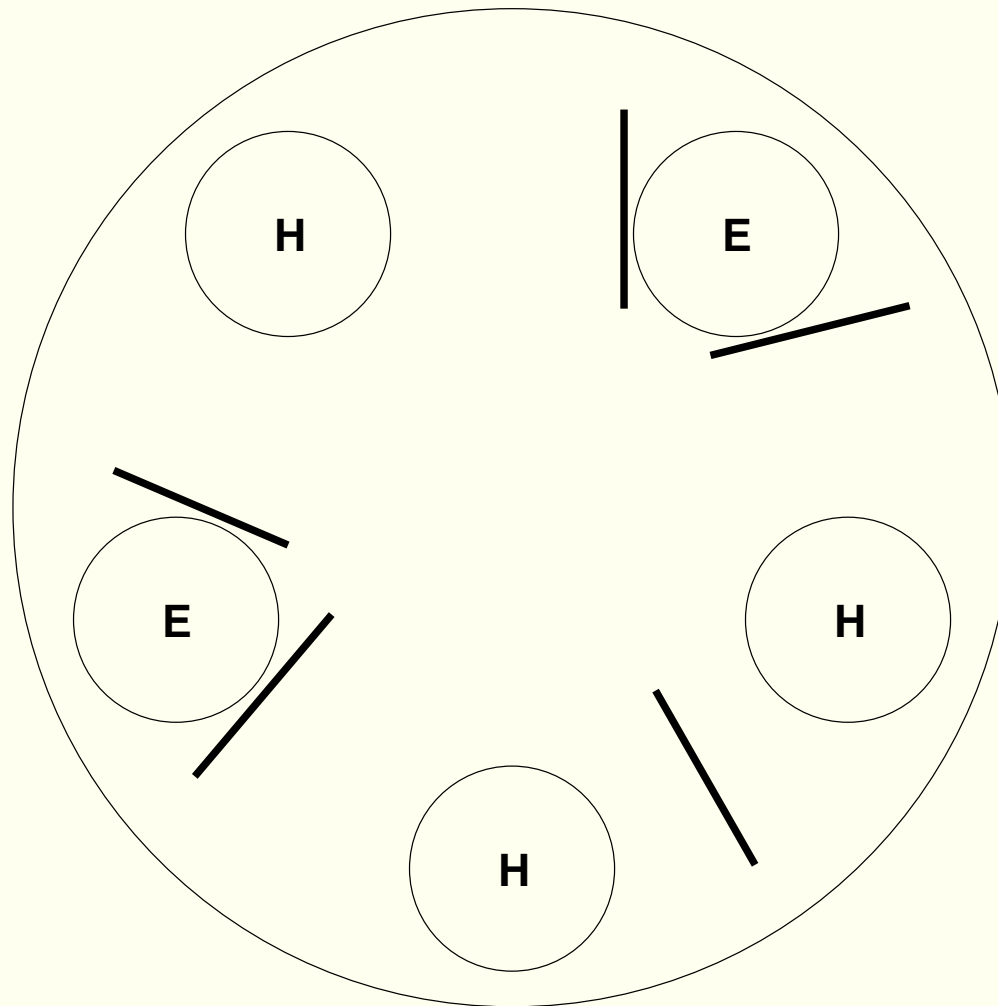
# Alto paralelismo



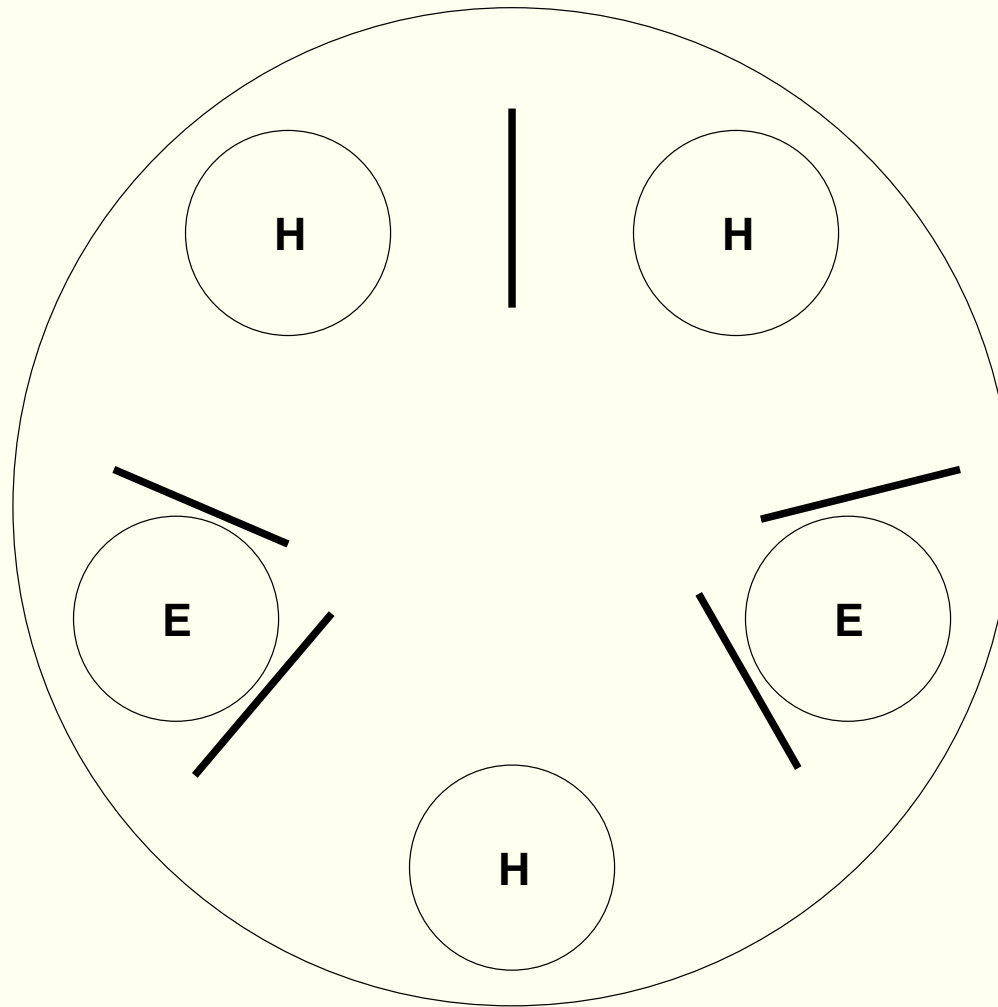
# Alto paralelismo



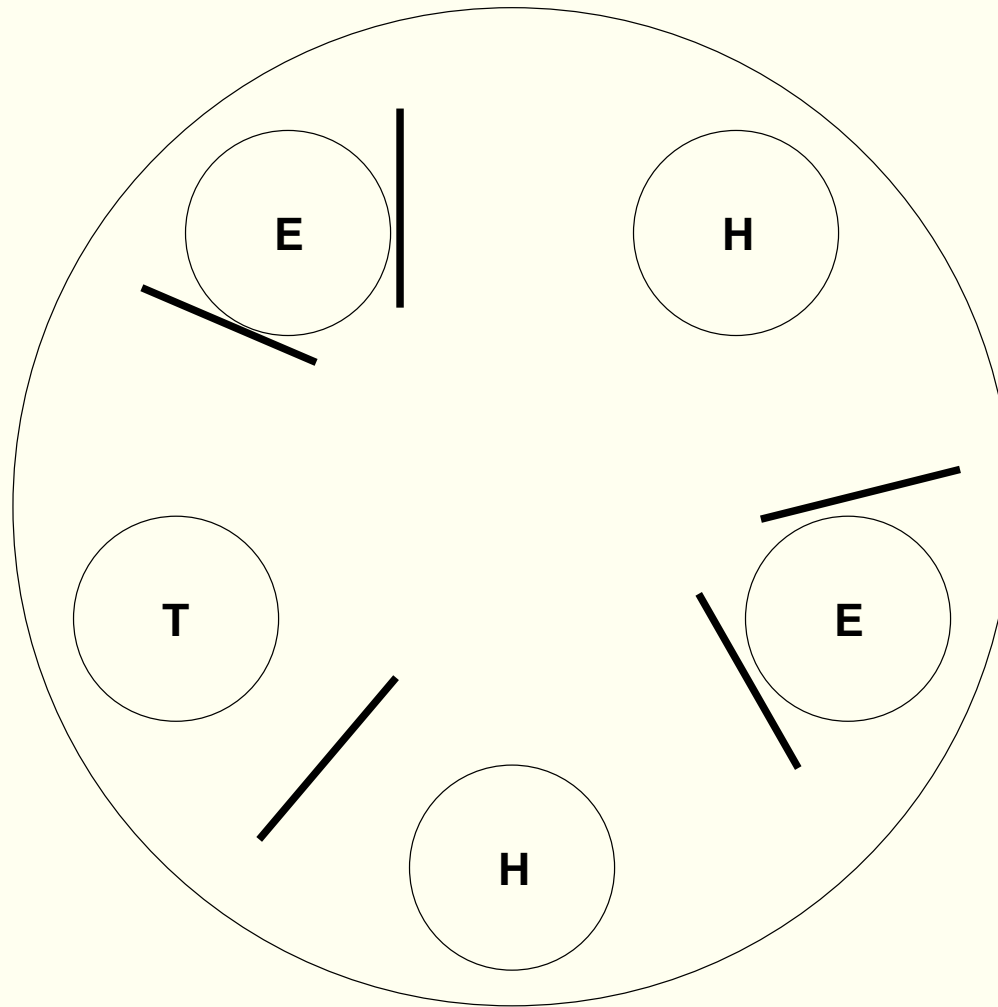
# Alto paralelismo



# Alto paralelismo

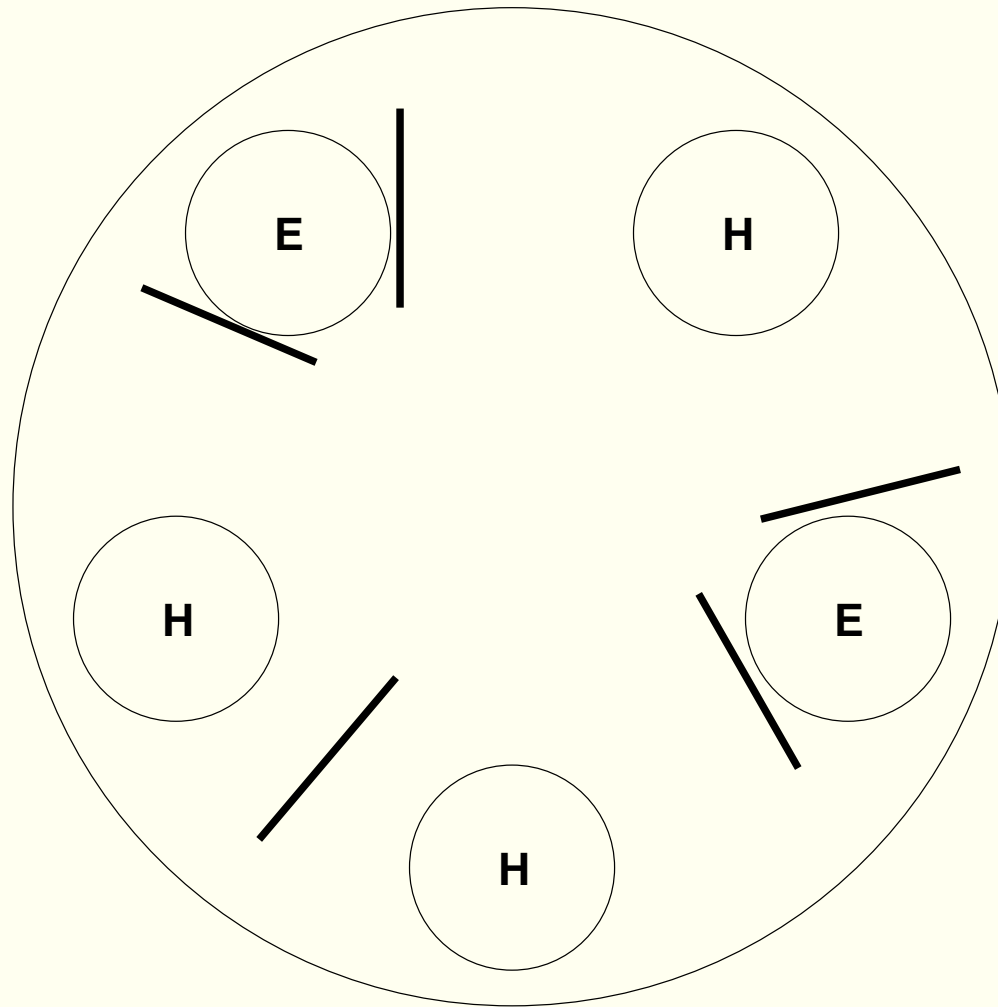


# Alto paralelismo

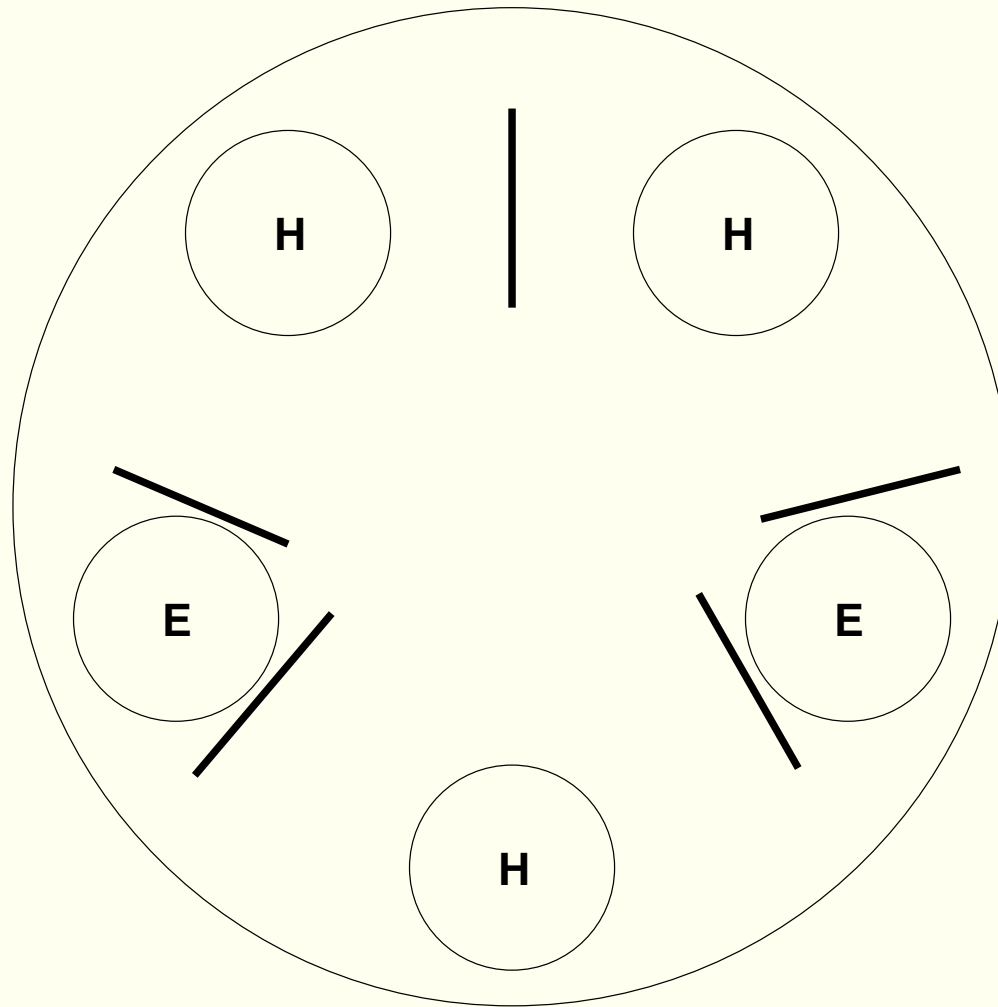




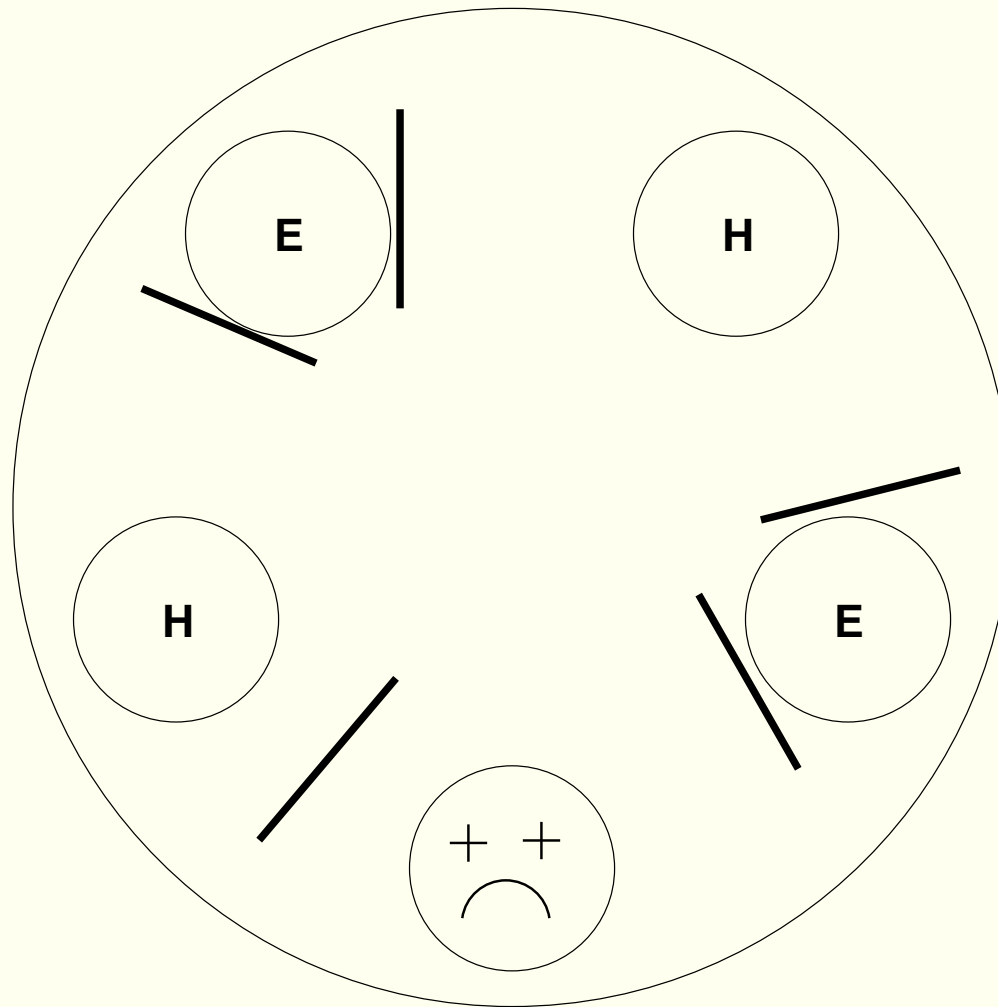
# Alto paralelismo



# 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`?