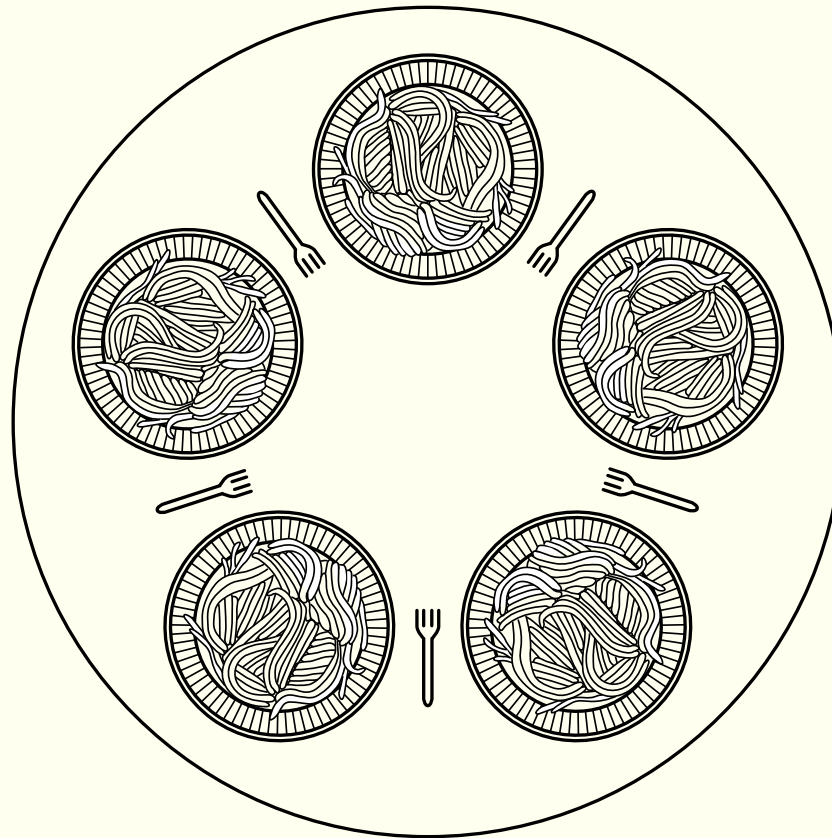


# **Processos e Threads**

## **Problema dos Filósofos Famintos**

# Jantar dos Filósofos



## Boas soluções

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

# Semáforos

## Comportamento básico

- `sem_init(s, 5)`

- `wait(s)`

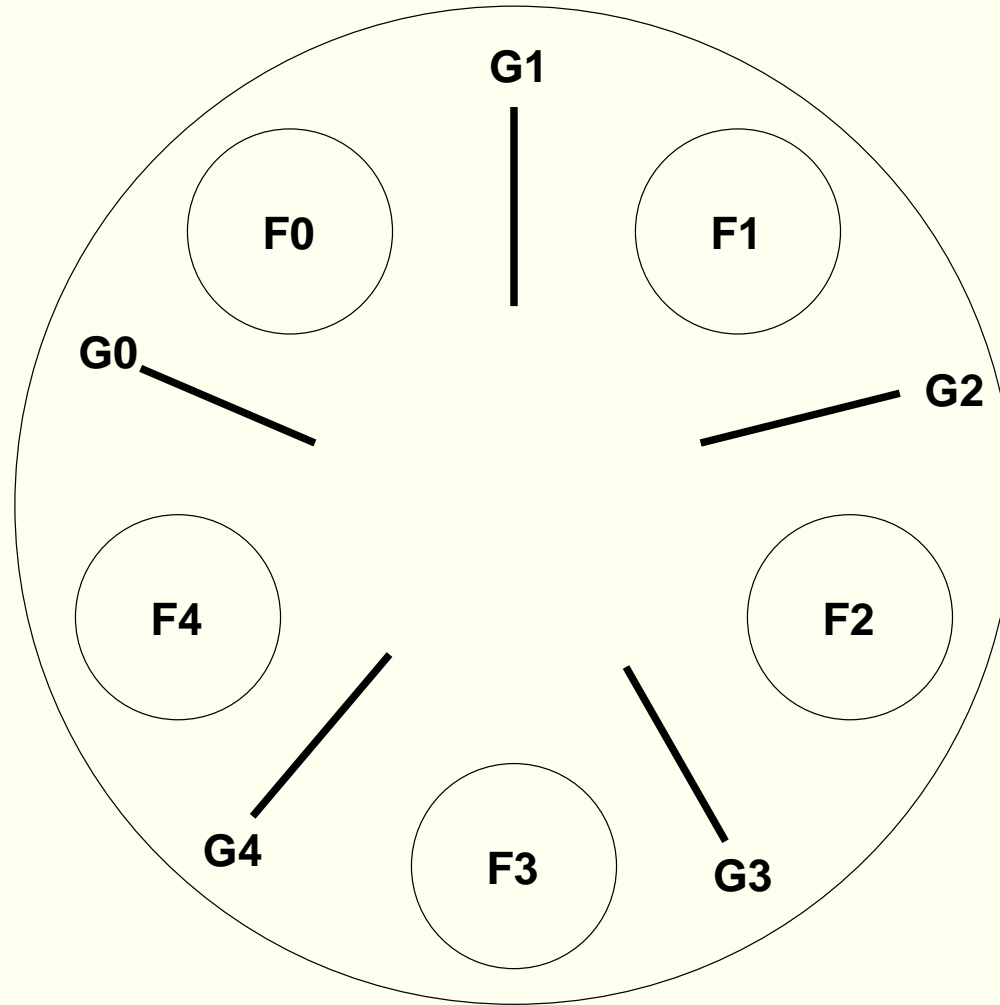
```
if (s == 0)
    bloqueia_processo();
else s--;
```

- `signal(s)`

```
if (s == 0 && existe_processo_bloqueado)
    acorda_processo();
else s++;
```

# Filósofos famintos

## Um semáforo por garfo

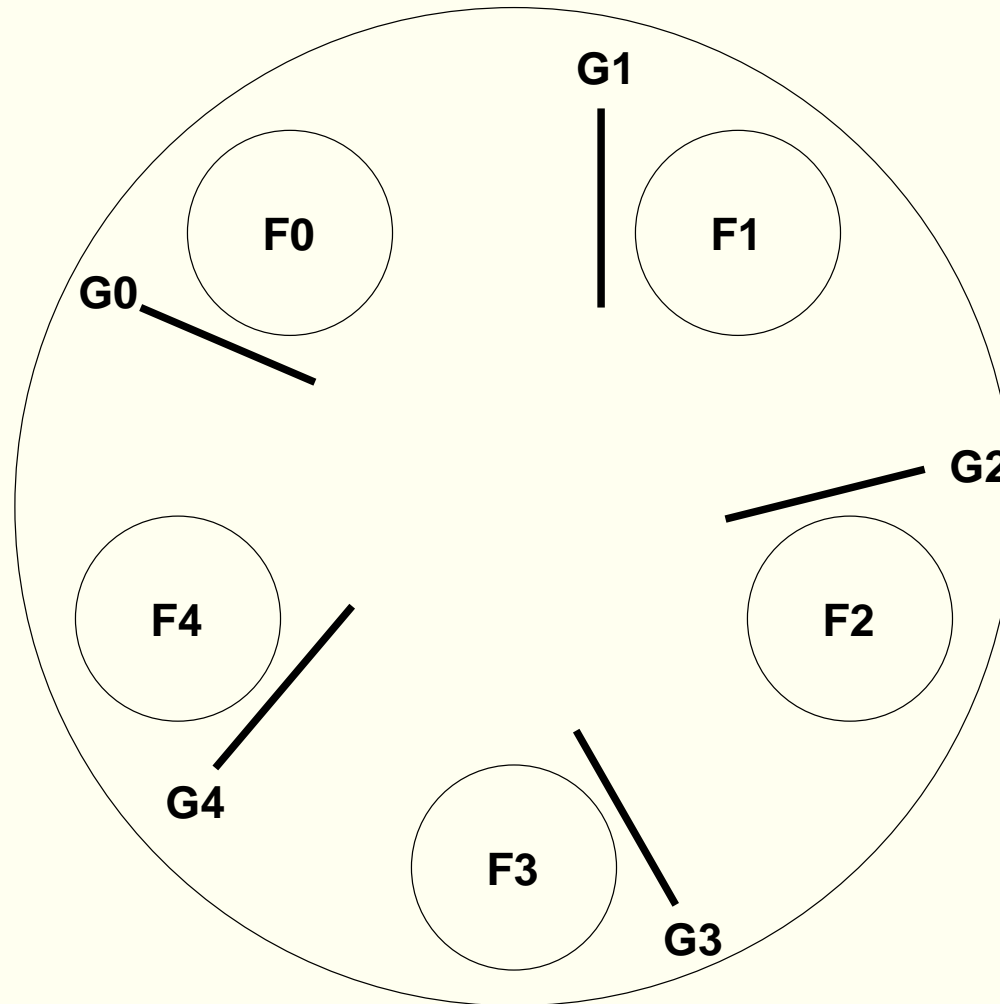


# Implementação simplista

## Filósofo i:

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

# Deadlock



Veja código: `deadlock.c`

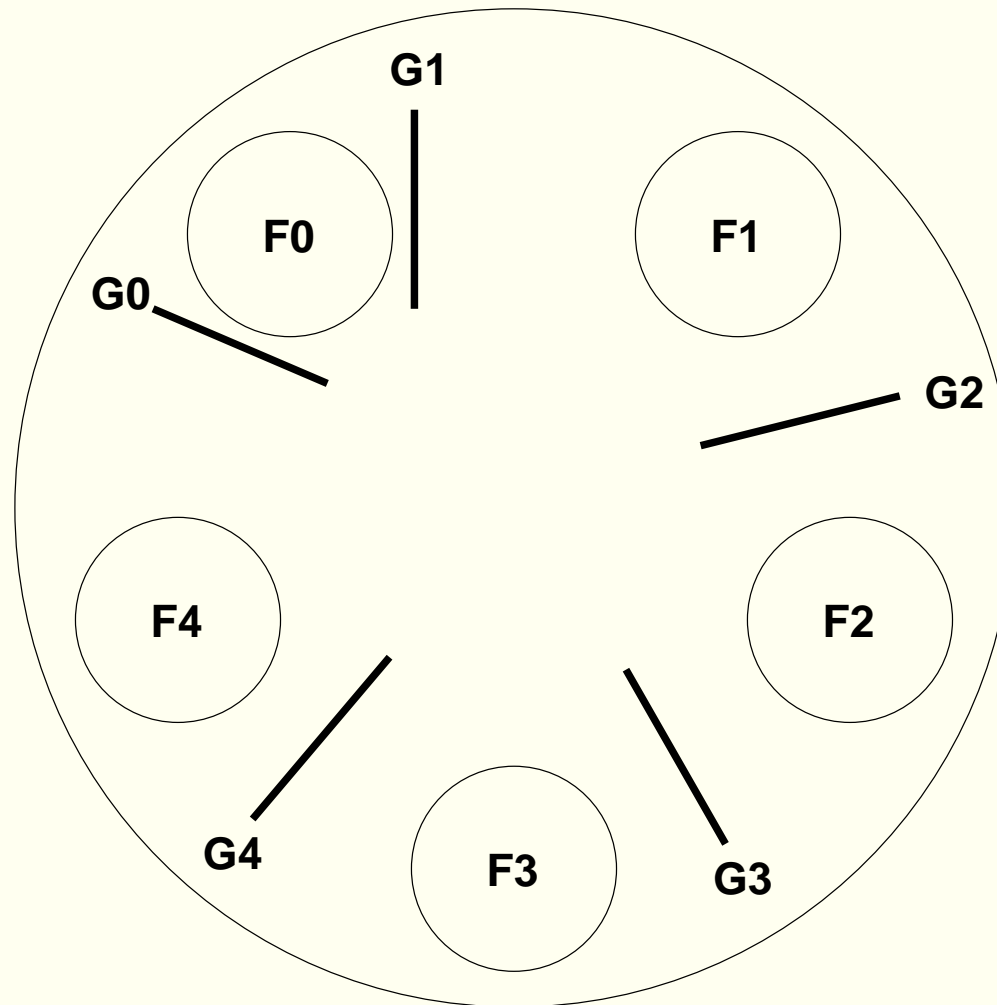
# Baixíssimo paralelismo

```
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`

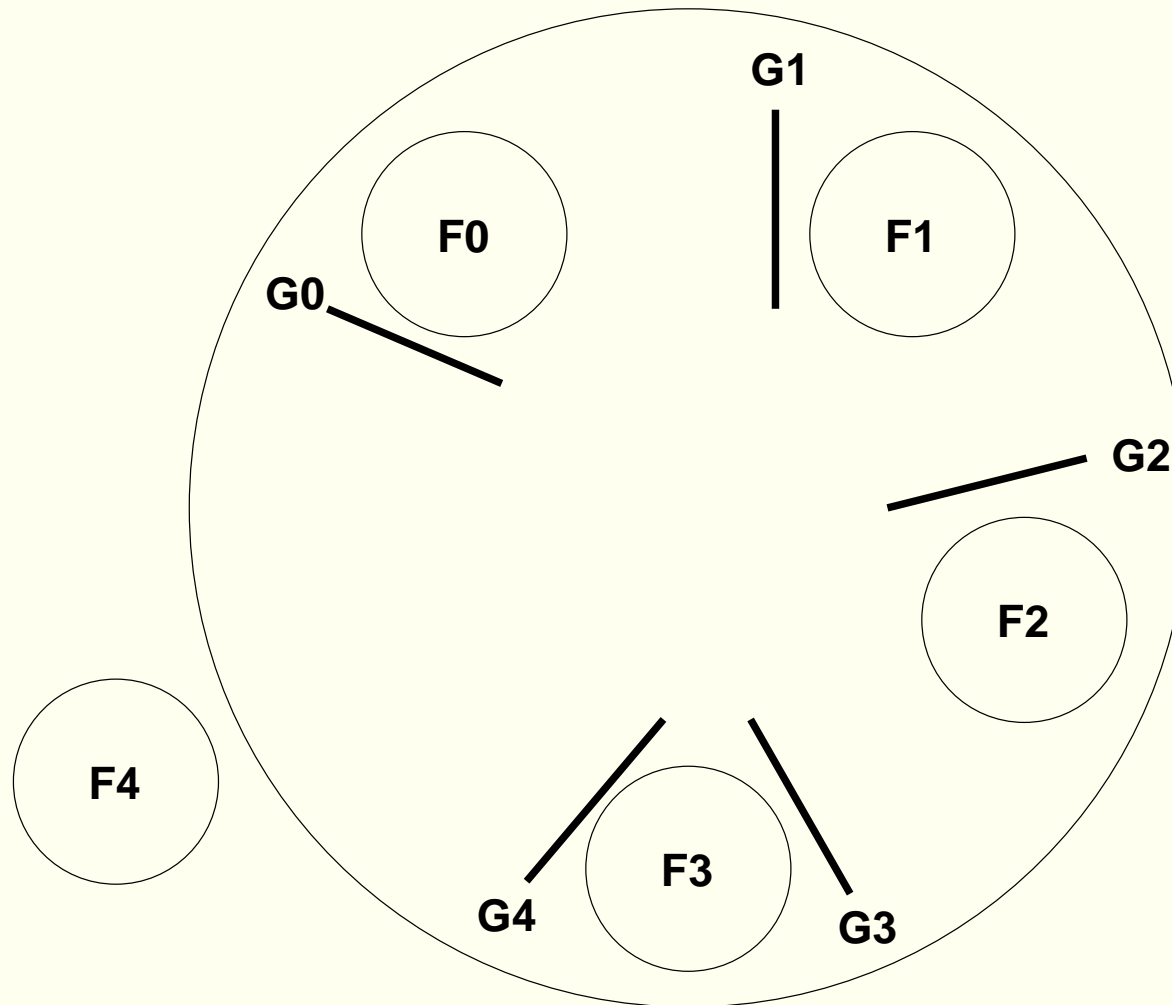
## Menos lugares à mesa

```
semaforo lugar_mesa = 4;
```

### 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

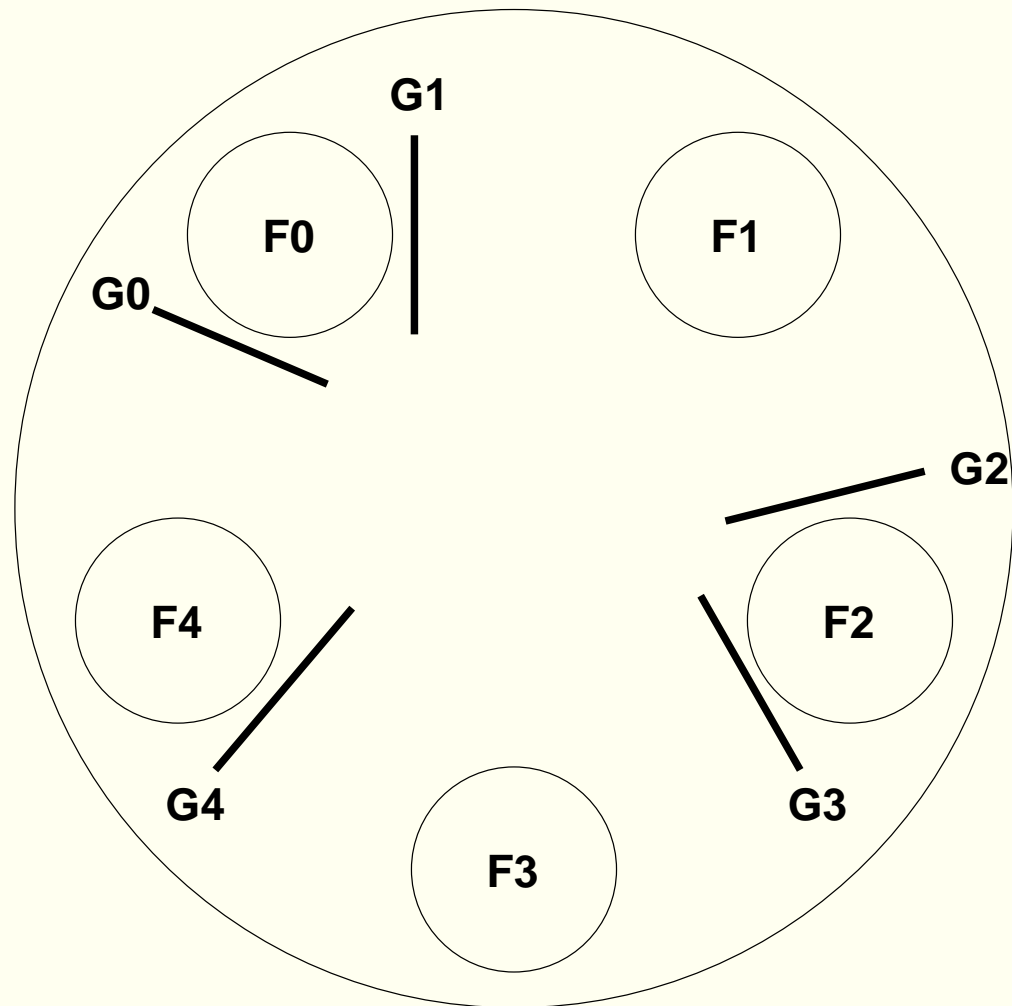


Veja código: `limite_lugares.c`

## 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]);
```

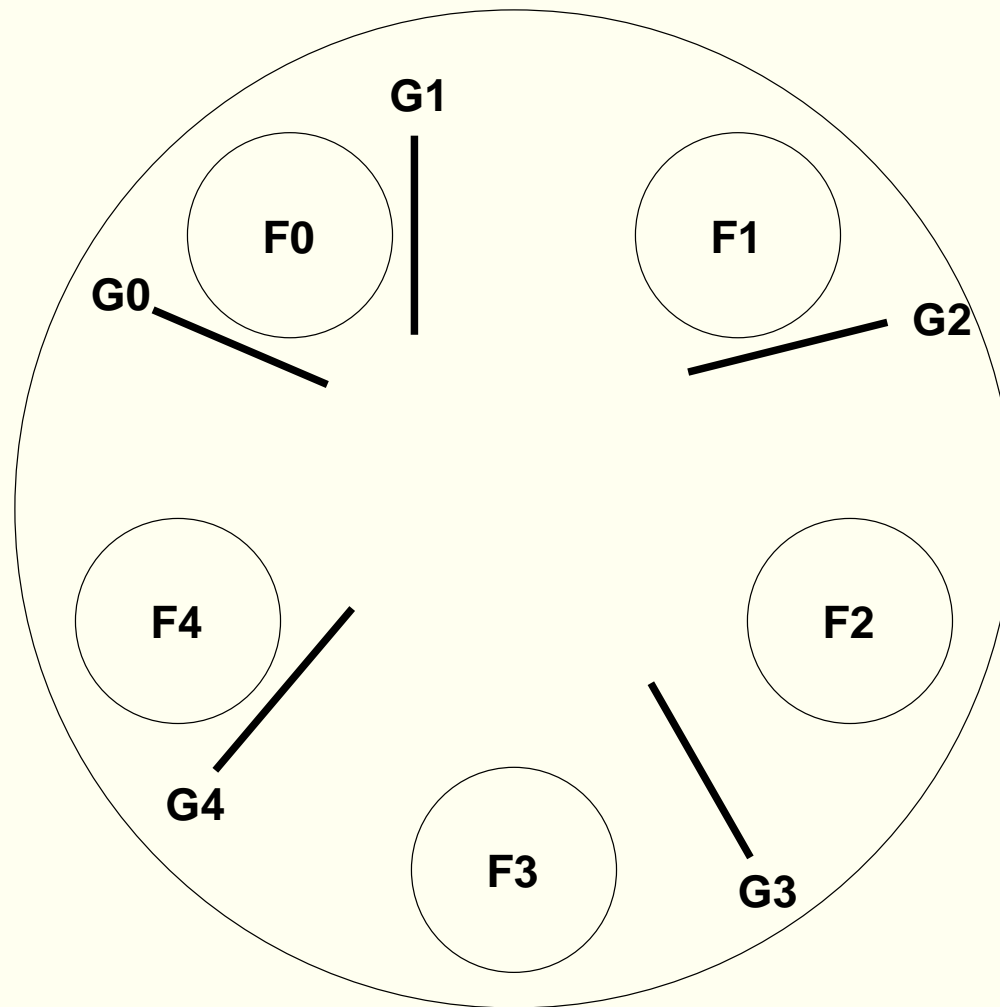
# Solução assimétrica



Veja código: `assimettrica.c`

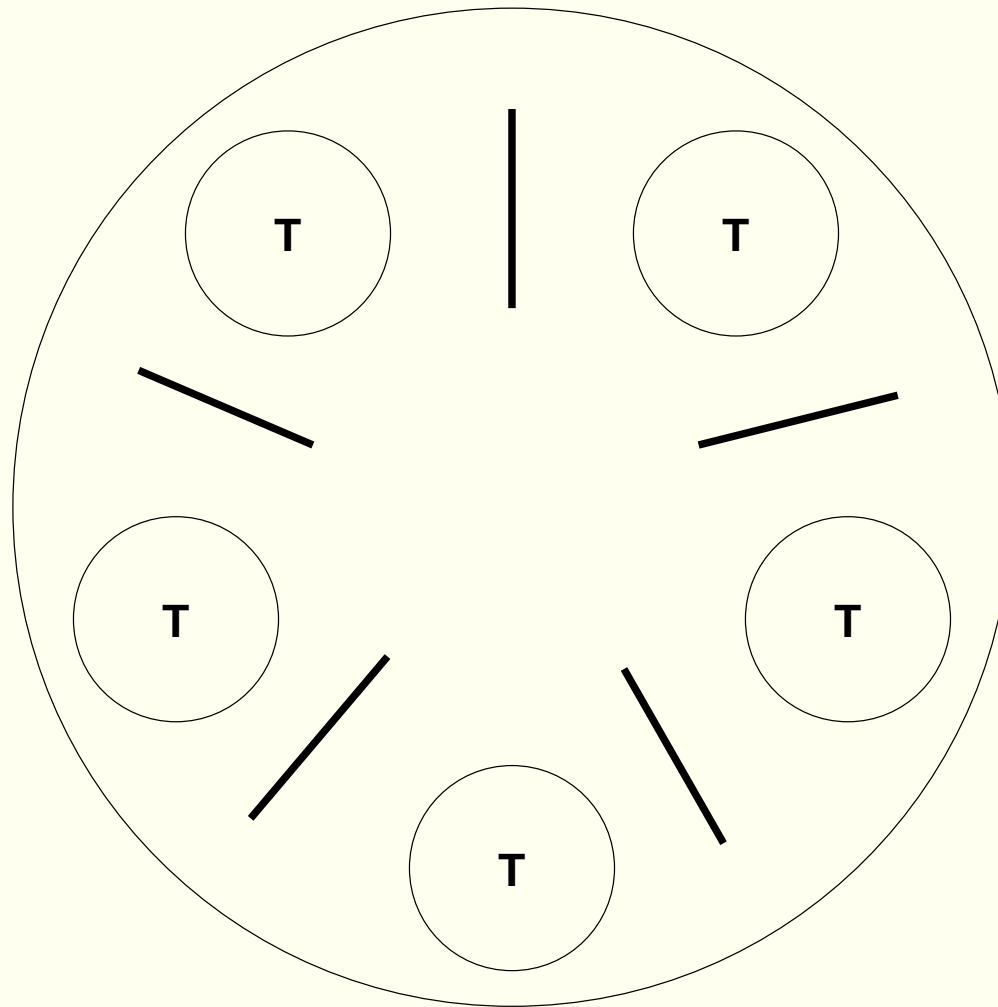
# Solução assimétrica

## 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:

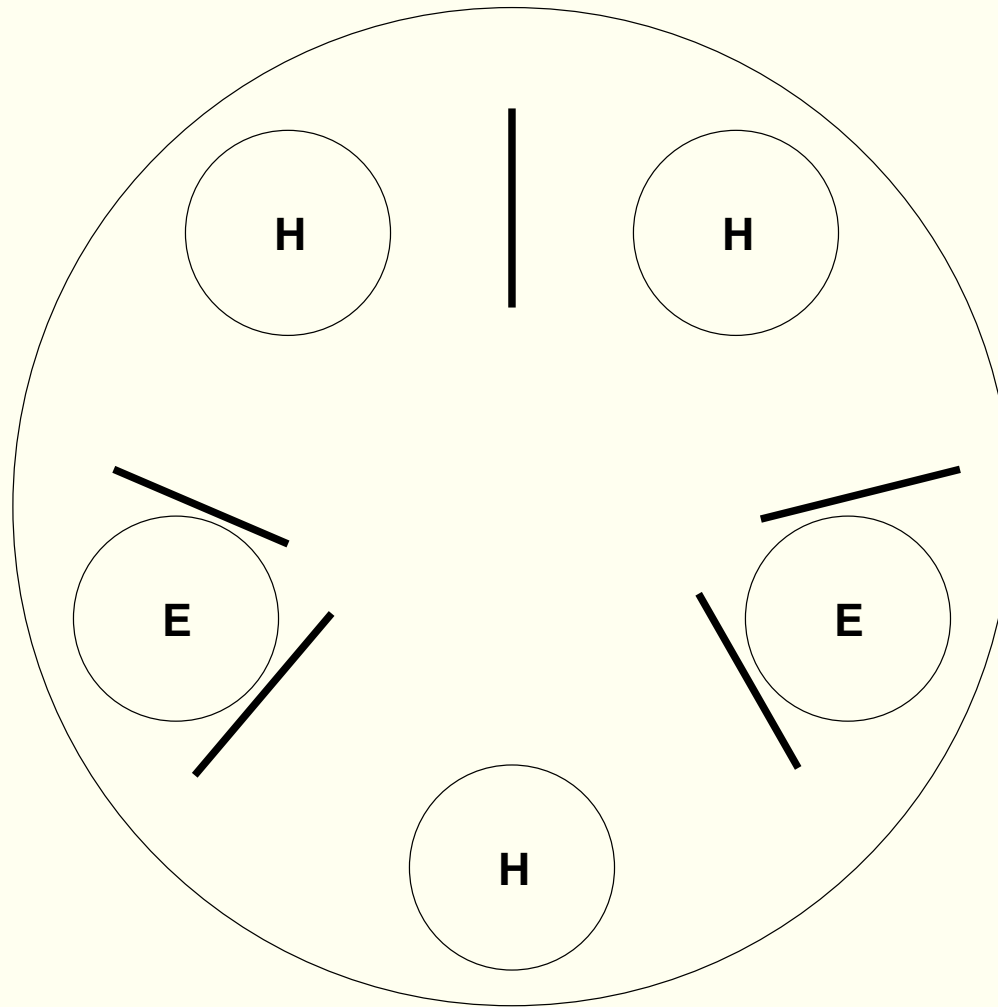
```
while (true)  
    pensa();  
    pega_garfos();  
    come();  
    solta_garfos();
```

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

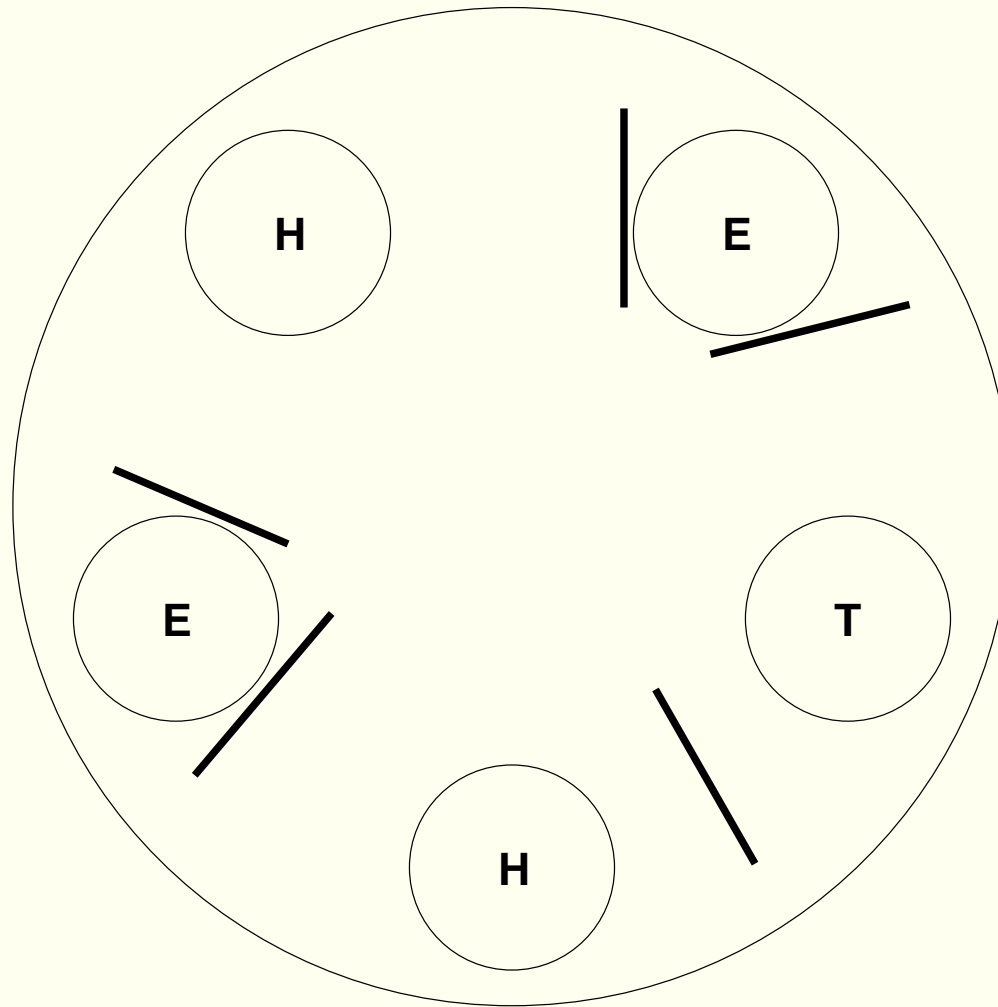
```
pega_garfos()
    wait(lock);
    estado[i] = H;
    testa_garfos(i);
    signal(lock);
    wait(filosofo[i]);
```

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

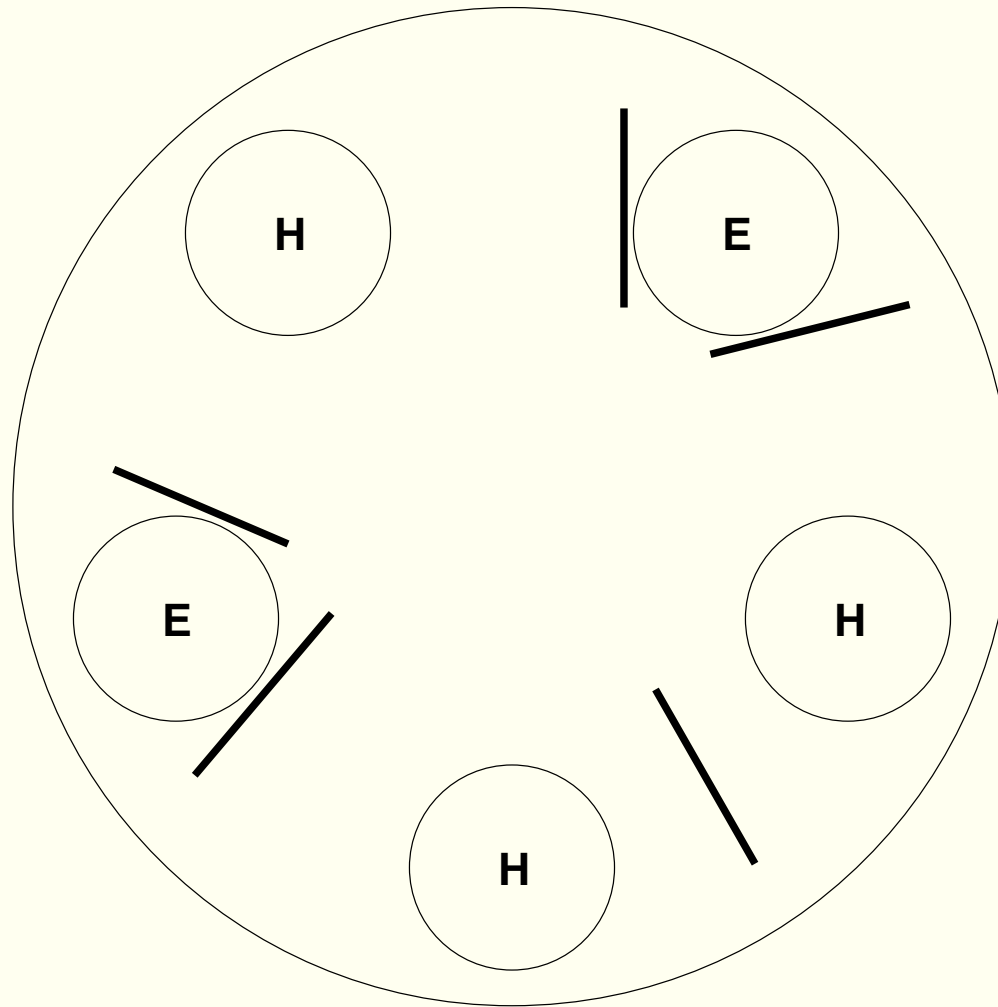
# Alto paralelismo



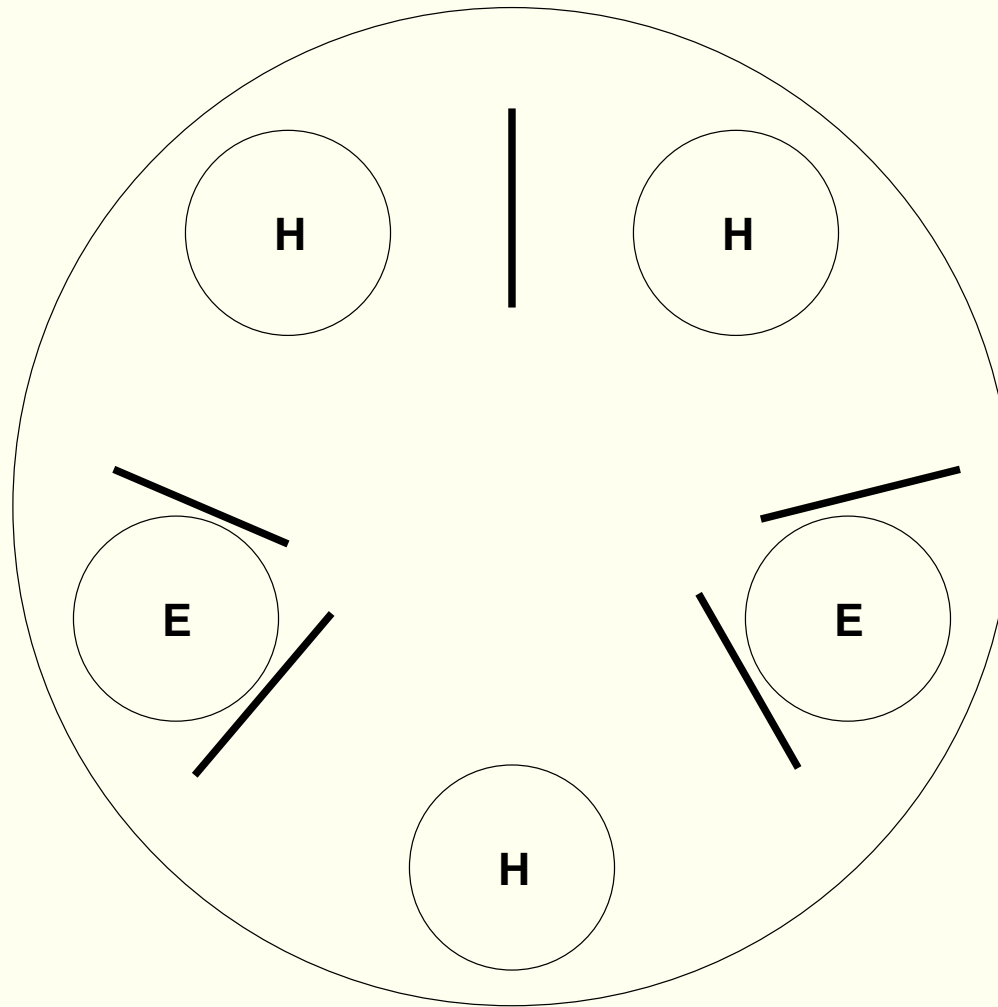
# Alto paralelismo



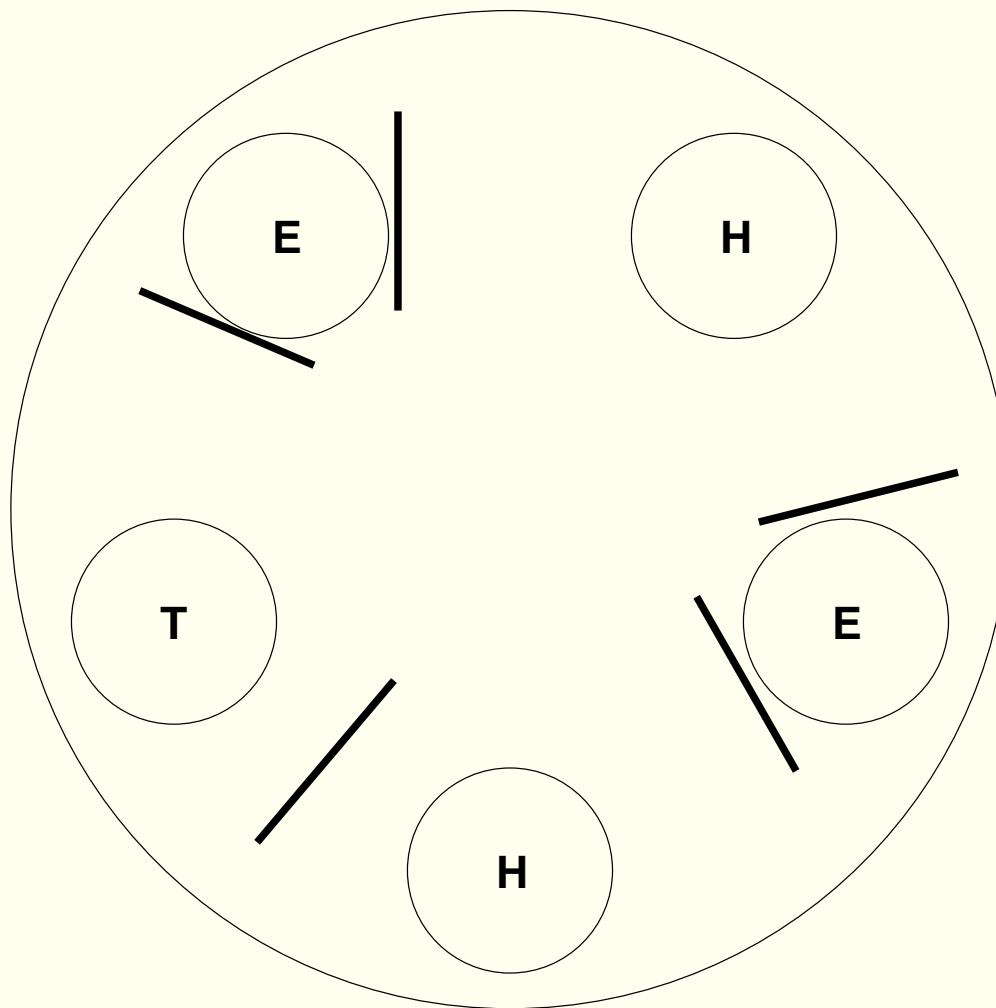
# Alto paralelismo



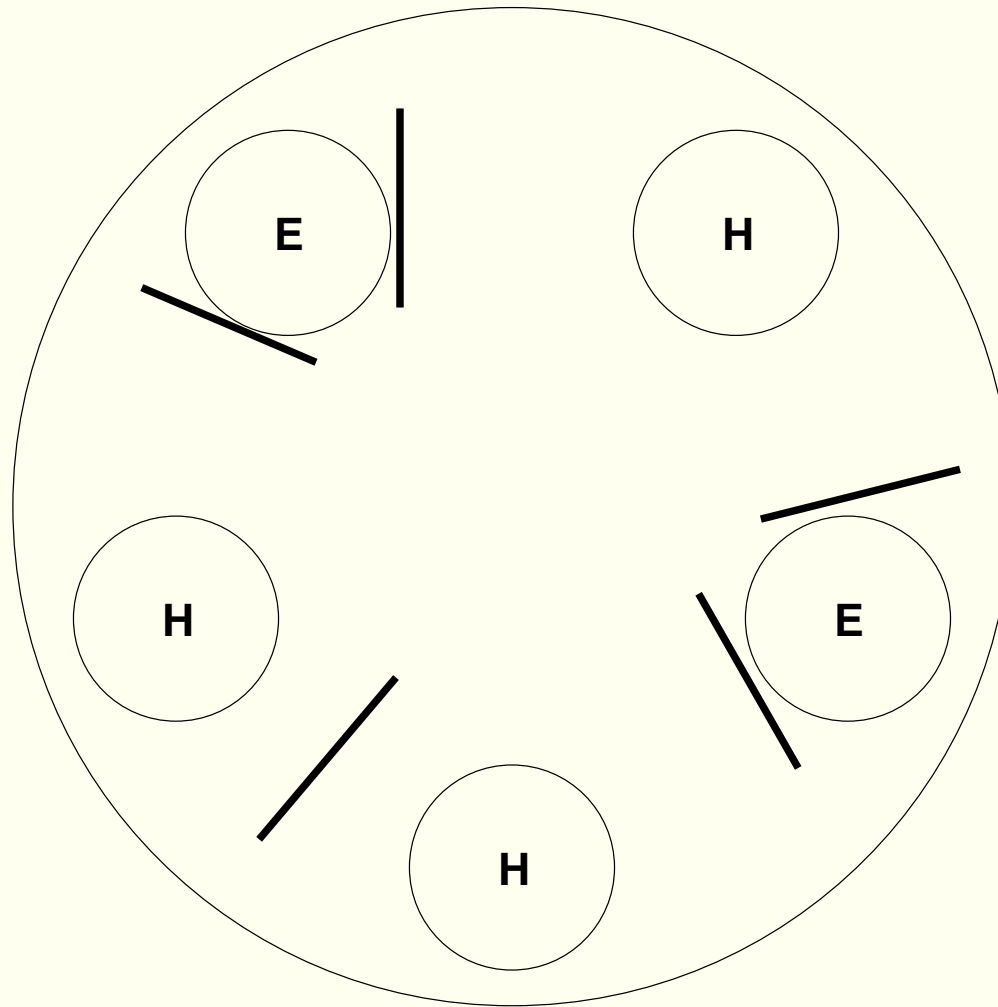
# Alto paralelismo



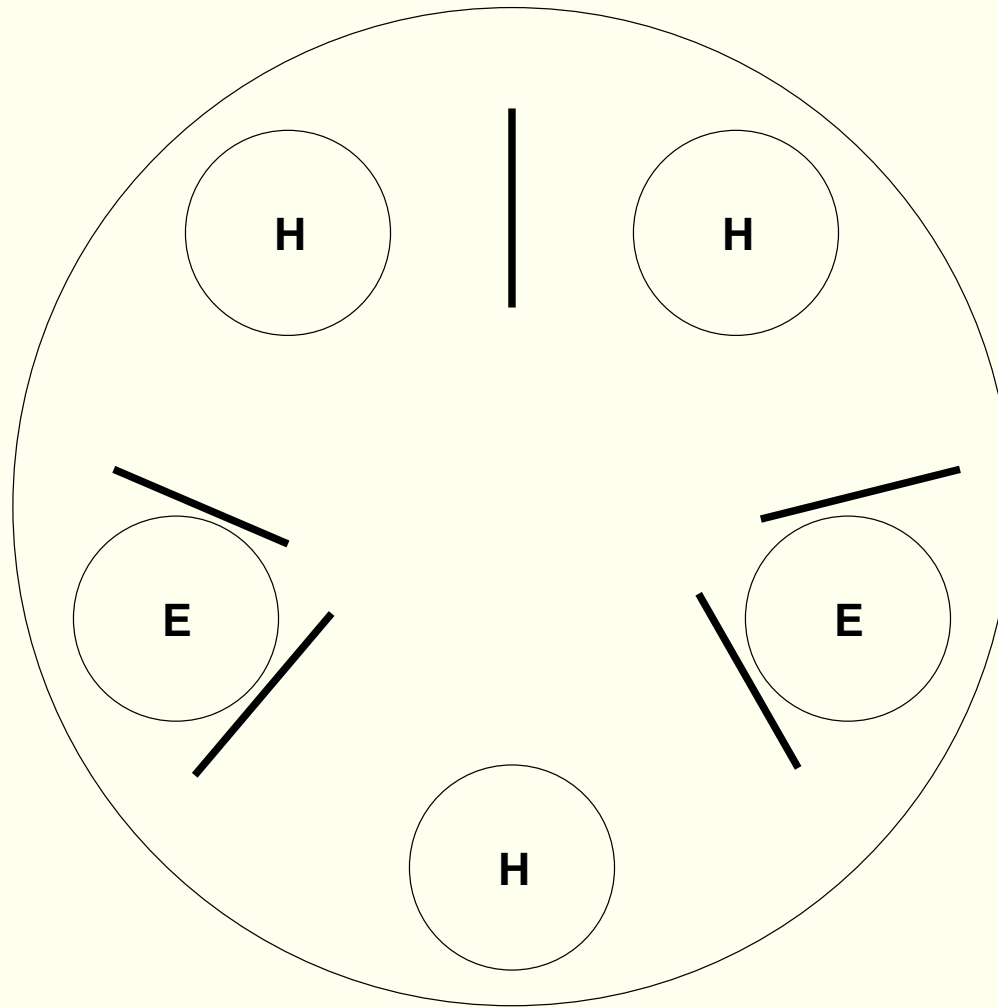
# Alto paralelismo



# Alto paralelismo



# Alto paralelismo



# Starvation

