

# MC504/MC514 - Sistemas Operacionais

## Barreiras

Islene Calciolari Garcia

Instituto de Computação - Unicamp

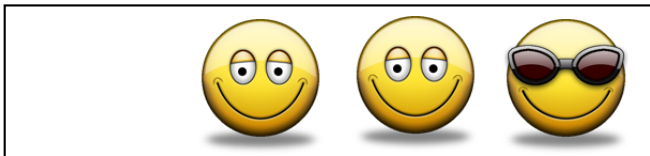
Segundo Semestre de 2013

# Sumário

- 1 Introdução
- 2 Tentativas do livro
- 3 Barreiras restritas
- 4 Futex
- 5 glibc com semáforos?

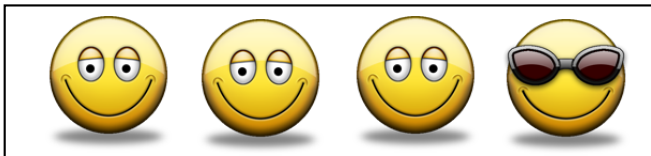
# O ponto de encontro das threads

## Aguardando N threads



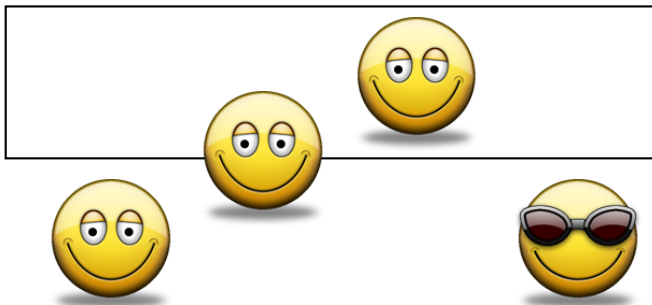
# O ponto de encontro das threads

N threads na barreira



# O ponto de encontro dos threads

Threads devem prosseguir



# O ponto de encontro das threads

Nenhuma thread pode ficar presa



# O ponto de encontro das threads

Vamos implementar?

- Grupo de N threads
- Vida monótona
  - Trabalham e
  - Sincronizam
- Vários exemplos retirados do livro The Little Book of Semaphores
- É possível ser mais eficiente?

## Primeira tentativa (3.4)

```
int c;  
sem barreira = 0;  
atomic_inc(c);  
if (c == N) sem_post(barreira);  
sem_wait(barreira);
```

- Quais são os problemas deste código?
- Quando funciona?
- Veja o arquivo `barreira1.c`



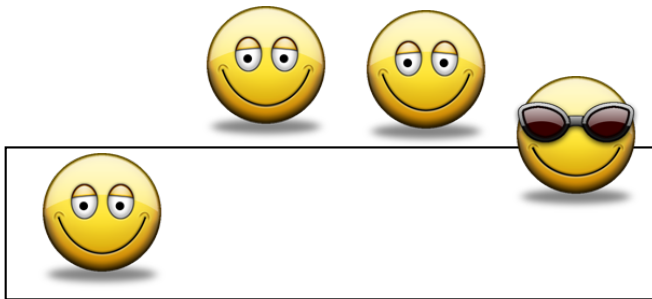
## Segunda tentativa (3.5)

```
int c;  
sem barreira = 0;  
atomic_inc(c);  
if (c == N) sem_post(barreira);  
sem_wait(barreira);  
sem_post(barreira);
```

- Veja o arquivo `barreira2.c`

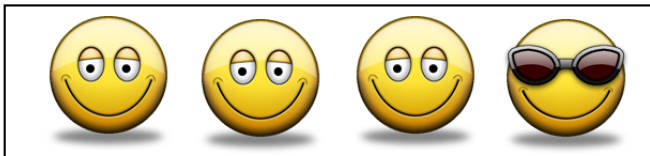
# Barreiras reutilizáveis: vários encontros

Threads devem se ressincronizar!



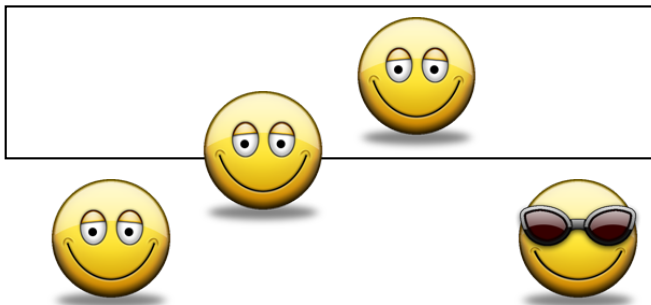
# Barreiras reutilizáveis: vários encontros

Threads devem se ressincronizar!



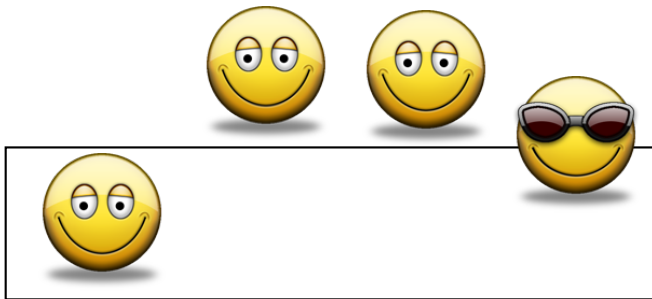
# Barreiras reutilizáveis: vários encontros

Threads devem se ressincronizar!



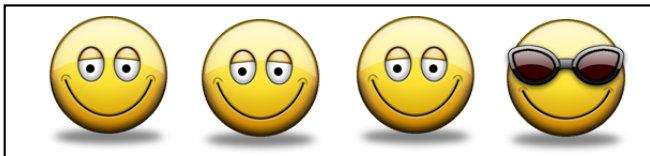
# Barreiras reutilizáveis: vários encontros

Threads devem se ressincronizar!



# Barreiras reutilizáveis: vários encontros

Threads devem se ressincronizar!



## Segunda tentativa (3.5)

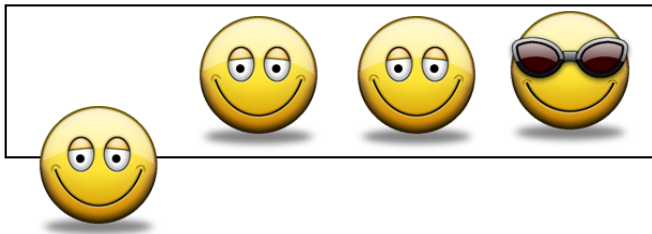
Barreiras reutilizáveis???

```
int c;
sem barreira = 0;
atomic_inc(c);
if (c == N) sem_post(barreira);
sem_wait(barreira);
sem_post(barreira);
```

- Veja o código `barreira2b.c`

# Barreiras reutilizáveis: vários encontros

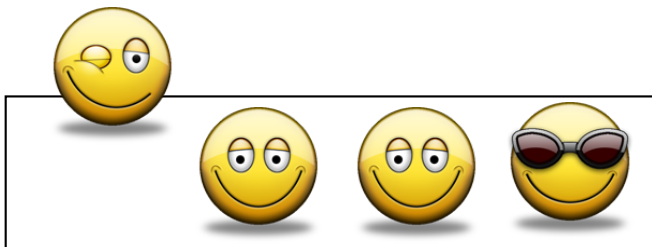
Cuidado com os espertinhos!





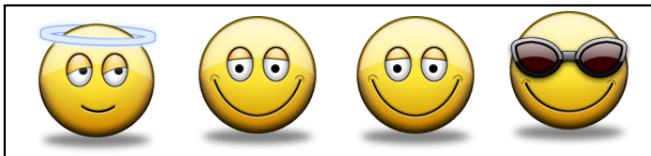
# Barreiras reutilizáveis: vários encontros

Cuidado com os espertinhos!



# Barreiras reutilizáveis: vários encontros

Cuidado com os espertinhos



## Deadlock (3.6)

```
int c; lock_t mutex;
sem barreira = 0;
lock(mutex);
c++;
if (c == N) sem_post(barreira);
sem_wait(barreira);
sem_post(barreira);
unlock(mutex);
```

- Uso claro para variáveis de condição...

## Terceira tentativa (3.7)

```
int c;
sem barreira = 0;
atomic_inc(c);
if (c == N) sem_post(barreira);
sem_wait(barreira);
sem_post(barreira);
atomic_dec(c);
if (c == 0) sem_wait(barreira);
```

- Veja o arquivo `barreira3.c`

## Quarta tentativa (3.8)

```
int c;
sem barreira = 0;
local_c = atomic_inc(c);
if (local_c == N) sem_post(barreira);
sem_wait(barreira);
sem_post(barreira);
local_c = atomic_dec(c);
if (local_c == 0) sem_wait(barreira);
```

- Veja o arquivo `barreira4.c`

# Roleta dupla

```
int c;
sem roleta_entrada = 0, roleta_saida = 1;
local_c = atomic_inc(c);
if (local_c == N)
    sem_wait(roleta_saida);
    sem_post(roleta_entrada);

sem_wait(roleta_entrada);
sem_post(roleta_entrada);
```

## Roleta dupla (continuação)

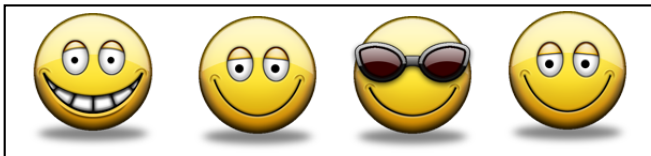
```
local_c = atomic_dec(c);  
if (local_c == 0)  
    sem_wait(roleta_entrada);  
    sem_post(roleta_saida);
```

```
sem_wait(roleta_saida);  
sem_post(roleta_saida);
```

- Veja o arquivo `barreira5.c`

# Barreiras restritas

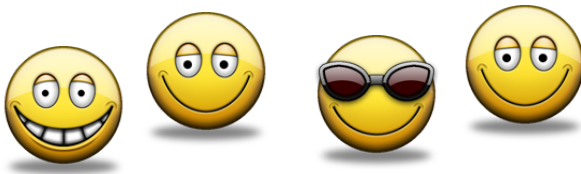
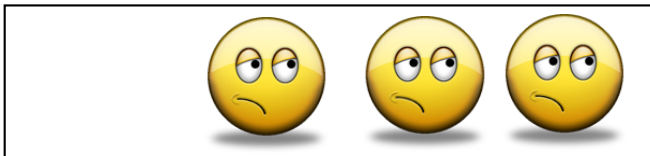
Não há lugar para todos...





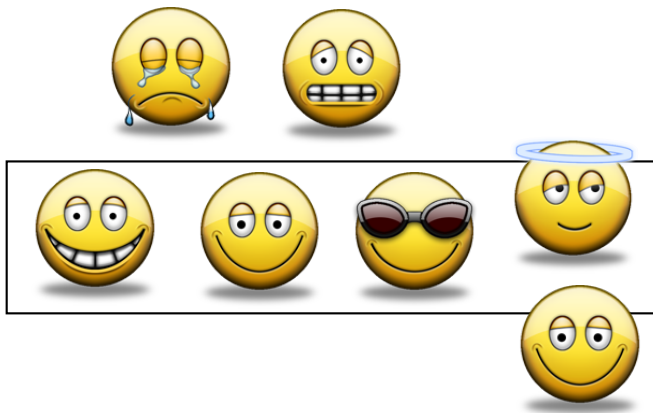
# Barreiras restritas

Um grupo deve sair para que outro possa entrar!



# Barreiras restritas

Cuidado com os espertinhos!



## Sobre desempenho...

- Duas roletas causam muitas trocas de contexto
- Na seção 3.6.6 o livro sugere um semáforo de incremento maior do que 1
- Variáveis de condição e broadcast?
- Futex?

## Tentativa com futex: roleta dupla

```
int c;
int roleta_entrada = 0, roleta_saida = 1;
local_c = atomic_inc(c);
if (local_c == N)
    roleta_saida = 0;
    roleta_entrada = 1;
    futex_wake(&roleta_entrada, N-1);
else
    futex_wait(&roleta_entrada, 0);
```

## Tentativa com futex: roleta dupla (continuação)

```
local_c = atomic_dec(c);  
if (local_c == 0)  
    roleta_entrada = 0;  
    roleta_saida = 1;  
    futex_wake(&roleta_saida, N-1);  
else  
    futex_wait(&roleta_saida, 0);
```

- Veja o arquivo `barreira_futex.c`

## Vamos analisar a implementação da glibc?

- Veja o arquivo `barrier_wait.c` e teste o comando `pthread_barrier_wait`
- Veja o código `pthread_barrier_wait.c`

## Algoritmo da glibc com semáforos

```
sem_t sem_mutex = 1, sem_barreira = 0;
```

```
sem_wait(sem_mutex);
```

```
c++;
```

```
if (c < N) {
```

```
    sem_post(sem_mutex);
```

```
    sem_wait(sem_barreira); }
```

```
local_c = atomic_dec(c);
```

```
if (local_c > 0)
```

```
    sem_post(sem_barreira);
```

```
else
```

```
    sem_post(sem_mutex);
```

- Veja o arquivo `barreira6.c`