Computação Concorrente (DCC/UFRJ)

Aula 6: Sincronização condicional com variáveis de condição

Prof. Silvana Rossetto

24 de novembro de 2015

Problema

O sistema de controle de uma máquina de processamento de chocololate deve funcionar em dois modos:

1. modo normal: quando a temperatura externa está abaixo de 30^o
2. modo especial: quando a temperatura externa está acima de 30^o

Para cada modo, o fluxo de operação da máquina varia significantemente para garantir a qualidade do produto. **De que forma a computação concorrente poderia ser usada para ajudar a resolver esse problema?**

Sincronização por condição

Visa garantir que **uma thread fique bloqueada enquanto uma determinada condição lógica da aplicação não for satisfeita**

Variáveis de condição

Definição

Variáveis especiais que permitem que as threads esperem (bloqueando-se) até que sejam sinalizadas (avisadas) por outra thread que a condição lógica foi atendida

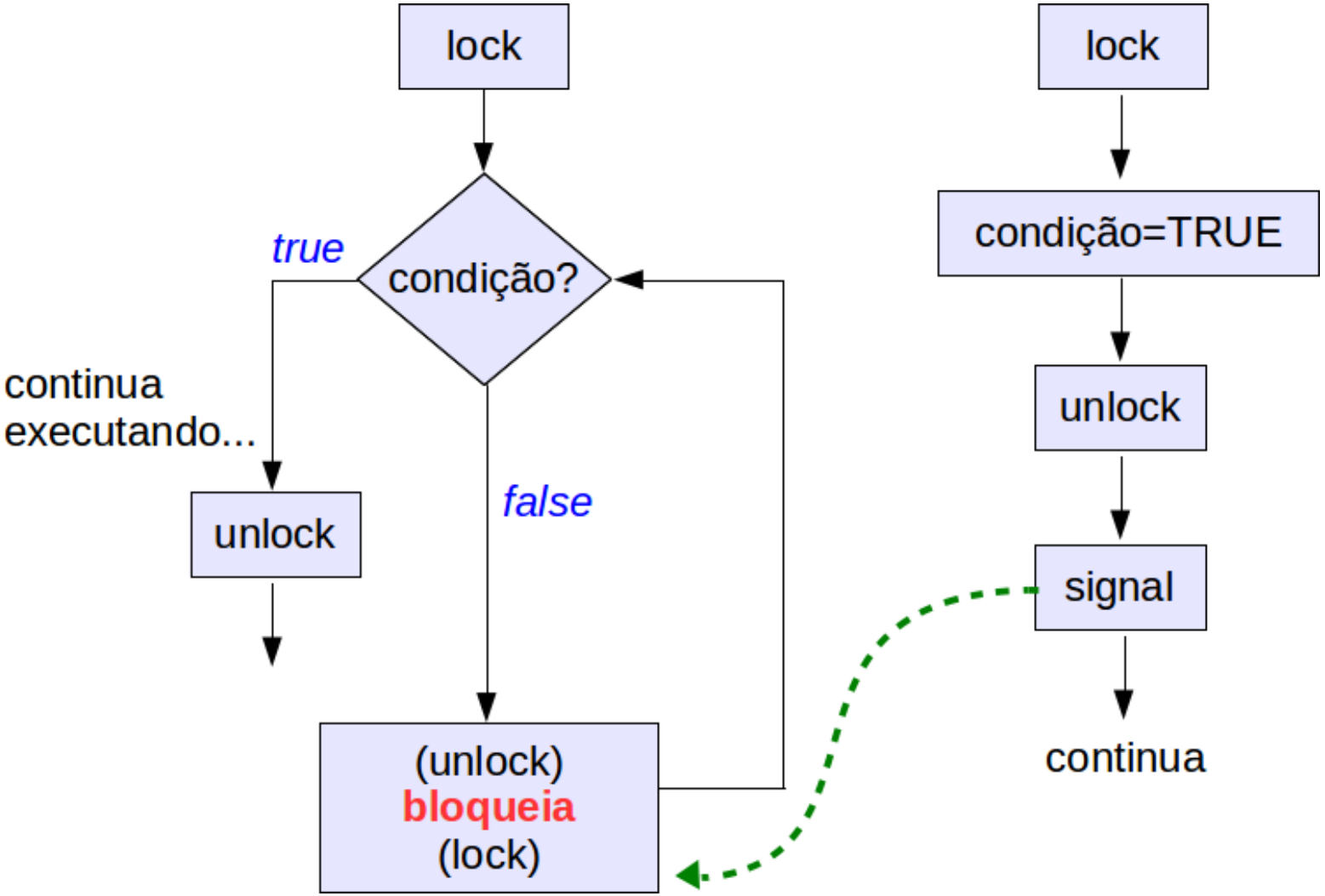
Operações básicas

- **WAIT(condvar)**: bloqueia a thread na fila da variável de condição
- **SIGNAL(condvar)**: desbloqueia uma thread na fila da variável de condição
- **BROADCAST(condvar)**: desbloqueia todas as threads na fila da variável de condição

Como funcionam as variáveis de condição

- Uma **variável de condição** é sempre usada em conjunto com uma **variável de lock**
- A thread usa o **bloco de lock para checar a condição lógica da aplicação e decidir** por WAIT ou SIGNAL
- O lock é implicitamente liberado quando a thread é bloqueada e é implicitamente devolvido quando a thread retoma a execução do ponto de bloqueio

Como funcionam as variáveis de condição



Locks X Variáveis de condição

- Locks são usados para implementar **sincronização por exclusão mútua** (controle do acesso ao dado)
- Variáveis de condição são usadas para implementar a **sincronização por condição** (lógica de execução baseado no valor do dado)

Variáveis de condição em PThreads

A biblioteca PThreads define um tipo especial chamado **pthread_cond_t** com as seguintes rotinas:

- **pthread_cond_wait (condvar, mutex)**: bloqueia a thread na condição (*condvar*) (deve ser chamada com *mutex* locado para a thread e depois de finalizado deve desalocar *mutex*)
- **pthread_cond_signal (condvar)**: desbloqueia uma thread esperando pela condição (*condvar*) (deve ser chamada com *mutex* locado e deve desalocá-lo em seguida)
- **pthread_cond_broadcast (condvar)**: usado no lugar de SIGNAL quando todas as threads na fila da condição podem ser desbloqueadas

Variáveis de condição em PThreads

- **pthread_cond_init (condvar, attr)**: inicializa a variável
- **pthread_cond_destroy (condvar)**: libera a variável

Exemplo em PThreads

Inicialização

```
pthread_mutex_t count_mutex;
pthread_cond_t count_cond;
...
pthread_mutex_init(&count_mutex, NULL);
pthread_cond_init (&count_cond, NULL);
```

Exemplo em PThreads

Thread A

```
void *A (void *t) {
  pthread_mutex_lock(&count_mutex);
  while (count != COUNT_LIMIT) {
    pthread_cond_wait(&count_cond, &count_mutex);
  }
  pthread_mutex_unlock(&count_mutex);
  ...
  pthread_exit(NULL);
}
```

Exemplo em PThreads

Thread B

```
void *B (void *t) {
  while(proximo!=0) {
    pthread_mutex_lock(&count_mutex);
    count++;
    if (count == COUNT_LIMIT) {
       pthread_cond_signal(&count_cond);
    }
    pthread_mutex_unlock(&count_mutex);
    ...
  }
  pthread_exit(NULL);
}
```

Exercício

Projetar e implementar um programa concorrente com 4 threads onde a ordem de execução das threads é controlada no programa:

- a thread 1 imprime a frase "tudo bem?"
- a thread 2 imprime a frase "hola!"
- a thread 3 imprime a frase "até mais tarde."
- a thread 4 imprime a frase "tchau!"

As threads 1 e 2 devem executar antes das threads 3 e 4 sempre (a ordem de execução entre as threads 1 e 2 não importa, assim como a ordem de execução entre as threads 3 e 4).

Primeiro trabalho

descrição do primeiro trabalho...

Referências bibliográficas

- *Computer Systems - A Programmer's Perspective* (Cap. 12)
- *Programming Language Pragmatics*, M.L.Scott, Morgan-Kaufmann, ed. 2, 2006
- *Modern Multithreading*, Carver e Tai, Wiley, 2006
- *An Introduction to Parallel Programming*, Peter Pacheco, Morgan Kaufmann, 2011