Computação Concorrente (DCC/UFRJ)

Módulo 3 - Semana 2: Implementação de problemas clássicos de concorrência usando semáforos

Prof. Silvana Rossetto

Dezembro 2020

Sincronização coletiva (barreira) com semáforos

- Projetar uma função **void barreira(int nthreads)** para implementar sincronização coletiva
- O parâmetro `nthreads` informa o número total de threads que devem participar da barreira
- Todas as threads deverão chamar essa função no ponto do código onde a sincronização por barreira é requerida
- Usar **semáforos** para sincronização por condição e por exclusão mútua

Essa solução está correta?

```
sem_t mutex; //exclusao mutua (iniciado com 1)
sem_t cond; //condicional (iniciado com 0)
int chegaram=0; //variavel de estado global
void barreira(int numThreads) {
    sem_wait(&mutex);
    chegaram++;
    if (chegaram < numThreads) {
      sem_post(&mutex);
      sem_wait(&cond);
    } else {
      for(int i=1; i<numThreads; i++)
         { sem_post(&cond); }
      chegaram = 0;
      sem_post(&mutex);
    }
}
```

Uma solução correta

Sincronização coletiva com semáforos

```
void barreira(int numThreads) {
    sem_wait(&mutex);
    chegaram++;
    if (chegaram < numThreads) {
      sem_post(&mutex);
      sem_wait(&cond);
      chegaram--;
      if (chegaram==0) sem_post(&mutex);
      else sem_post(&cond);
    } else {
      chegaram--;
      sem_post(&cond);
    }
}
```

Problema leitor/escritor

Condições básicas do problema:

- Os leitores podem ler simultaneamente uma região de dados compartilhada
- Apenas um escritor pode escrever a cada instante em uma região de dados compartilhada
- Se um escritor está escrevendo, nenhum leitor pode ler a mesma região de dados compartilhada

Leitores

```
int leitores=0; //leitores lendo
sem_t mutex, escrita; //inicializados com 1
void leitor(){
   sem_wait(&mutex);
   leitores++;
   if(leitores==1) //primeiro leitor
      { sem_wait(&escrita); }
   sem_post(&mutex);
   /* faz a leitura */
   sem_wait(&mutex);
   leitores--;
   if(leitores==0) //ultimo leitor
      { sem_post(&escrita); }
   sem_post(&mutex);
}
```

Escritores

```
void escritor(){
    sem_wait(&escrita);
    /* faz a escrita */
    sem_post(&escrita);
}
```

Referências bibliográficas

1. *Computer Organization and Design – the hardware/software interface*, Patterson e Hennessy, ed. 4, 2009