Informática
UFRGS

Posix threads:

- criação
- mutex
- variáveis de condição
- exemplo de barreira

Notas de aula

Cláudio Geyer.

Autores

- **Autores**
  - **C. Geyer**
- **Local**
  - **Instituto de Informática**
  - **UFRGS**
  - **disciplina: Sistemas Operacionais II**
  - **Versão**
    - V03
    - ano: 2009/2

Súmula

- **Súmula**
  - **Criação de Posix threads**
  - **Mutex**
  - **Variáveis de condição**
  - **Exemplo implementação de barreira**

Bibliografia

- **Bibliografia**
  - **Andrews, G. Foundations of Multithreaded, Parallel and Distributed Programming. Addison-Wesley, 2000**
    - Visão rápida sobre criação, semáforos, mutex e variáveis de condição

Introdução

- **Posix threads**
  - **Padrão criado na década de 1990**
  - **Biblioteca (rotinas) para programação multithreaded em C**
  - **Também chamado de Pthreads**
  - **Diversas implementações**
    - Variações especialmente com relação ao SO
  - **Muitas funções para gerência e sincronização**

Criação e gerência de threads

- **Inclusão do header padrão da biblioteca Pthreads**
  - #include <pthread.h>
- **Declaração de variáveis para descrição de atributos de threads e de threads**
  - pthread_attr_t tattr; /* atributos das therads
  - pthread_t tid; /* descritor de uma thread *?
- **Inicialização dos atributos**
  - Rotinas
    - pthread_attr_init(&tattr);
    - pthread_attr_setscope(&tattr, PTHREAD_SCOPE_SYSTEM);
  - Escalonamento global: PTHREAD_SCOPE_SYSTEM

- **Criação e gerência de threads (cont.)**
  - **Criação e disparo da thread**
    - pthread_create(&tid, &tattr, start_func, arg);
  - **Término (auto) da thread**
    - Rotina
      - pthread_exit(value);
    - *value*: valor retornado ou null
  - **Junção (join) de threads**
    - Rotina
      - pthread_join(tid, value_ptr);
    - *tid* deve ser filha da thread chamadora
    - *value_ptr:* valor retornado

- **Semáforos**
  - **Declaração**
    - sem_t semaforo;
  - **Inicialização**
    - Rotina
      - sem_init (&semaforo, SHARED, valor_inicial)
    - SHARED
      - Se zero: somente entre threads do mesmo processo
      - Se not = zero: entre threads de distintos processos
    - valor_inicial: um inteiro

Semáforos

- **Semáforos**
  - **Operação P**
    - Rotina
      - sem_wait(&semaforo);
  - **Operação V**
    - Rotina
      - sem_post(&semaforo);

- **Semáforos**
  - **Uso para exclusão mútua**
    - ...

```
sem_t   mutex;
sem_init(&mutex, 0, 1);
...
sem_wait(&mutex);
   “seção crítica”;
sem_post(&mutex);
...
```

Exemplo threads

- **Exemplo: problema produtor / consumidor**
  - **Buffer simples: 1 (uma) posição**
  - **Código:**

```
#include <pthread.h>
#include <semaphore.h>
#define SHARED 1
#include <stdio.h>

void *Producer(void *);
void *Consumer(void *);

sem_t empty, full;    /* global semaphores */
int data;   /* shared buffer */
int numIters;   /* iteration number */
```

Exemplo threads

- **Exemplo: problema produtor / consumidor**
  - **Código (cont.):**

```
/* main() -- read command line and create threads */
int main(int argc, char *argv[]) {
pthread_t pid, cid; /* thread and attributes */
pthread_attr_t attr; /* descriptors */
pthread_attr_init(&attr);
pthread_attr_setscope(&attr,
    PTHREAD_SCOPE_SYSTEM);
```

Exemplo threads

- **Exemplo: problema produtor / consumidor**
  - **Código (cont.):**

```
    sem_init(&empty, SHARED, 1); /* sem empty = 1 */
    sem_init(&full, SHARED, 0); /* sem full = 0 */

    numlters = atoi(argv[l]);
    pthread_create(&pid, &attr, Producer, NULL);
    pthread_create(&cid, &attr, Consumer , NULL);
    pthread_join(pid, NULL);
    pthread_join(cid, NULL);
  }
```

Exemplo threads

- **Exemplo: problema produtor / consumidor**
  - **Código (cont.):**

```
/* deposit 1, ..., numlters into the data buffer */
void *Producer(void *arg) {
   int produced;
   for (produced = 1; produced <= numlters; produced++) {
        sem_wait(&empty);
       data = produced;
       sem_post(&full);
   }
}
```

Exemplo threads

- **Exemplo: problema produtor / consumidor**
  - **Código (cont.):**

```
/* fetch numlters items from the buffer and sum them */
void *Consumer(void *arg) {
   int total = 0, consumed;
   for (consumed = 1; consumed <= numlters; consumed++) {
       sem_wait(&full) ;
       total = total + data;
       sem_post(&empty);
}
printf("the total is %d\n", total);
} /* program end */
```

Exemplo threads

- **Exemplo: problema produtor / consumidor**
  - **Exercícios**
    - A) quantos produtores e consumidores poderiam ser criados? Justifique.
    - B) porque o buffer (variável data) não precisa ser protegido em exclusão mútua?

Exemplos de Mutex ou Locks

- **Exemplos de Mutexs ou Locks**
  - **Mutex Posix**
    - Para threads Posix
    - Usar
      - #include <pthread.h>
    - Declaração
      - `pthread_mutex_t`
      - Tipo de variável
    - Inicialização
      - Na declaração (sintaxe C usual)
      - Pela procedure (rotina):
        `pthread_mutex_init(mutex, args)`

Exemplos de Mutex ou Locks

- **Exemplos de Mutexs ou Locks**
  - **Mutex Posix**
    - Destruição
      - `pthread_mutex_destroy(mutex)`

Exemplos de Mutex ou Locks

- **Exemplos de Mutexs ou Locks**
  - **Mutex Posix**
    - Função *lock*
      - Procedure
        - pthread_mutex_lock (mutex)
      - Procedure
        - pthread_mutex_trylock (mutex)
        - Se mutex ocupado:
        - Thread não bloqueia
        - Retorna erro

Exemplos de Mutex ou Locks

- **Exemplos de Mutexs ou Locks**
  - **Mutex Posix**
    - Função *unlock*
      - Procedure
        - pthread_mutex_unlock (mutex)
    - Existem outras procedures e opções

Exemplos de Wait/signal

- **Exemplos de wait / signal**
  - **Variáveis de condição Posix**
    - Usar
      - #include <pthread.h>
    - Associadas às variáveis *mutex*
      - Programação mais complexa
    - Declaração
      - `pthread_cond_t`
    - Inicialização
      - Na declaração
      - Pela procedure
        - `pthread_cond_init(cond)`

Exemplos de Wait/signal

- **Exemplos de wait / signal**
  - **Variáveis de condição Posix**
    - Função *wait*
      - pthread_cond_wait (condition, mutex)
      - Antes: thread deve ter o lock do mutex
      - Depois: thread libera o lock (automaticamente)
      - Thread fica bloqueada na *condition*
      - Até que um *signal* seja feito na mesma *condition*

Exemplos de Wait/signal

- **Exemplos de wait / signal**
  - **Variáveis de condição Posix**
    - Função *signal*
      - Thread deve estar com o mesmo *mutex*
      - pthread_cond_signal (condition)
        - Acorda uma thread bloqueada na *condition*
        - Automaticamente recupera o *lock* do *mutex*
      - pthread_cond_broadcast (condition)
        - Acorda todas as threads bloqueadas …

Cuidados com Posix Mutex e Condition

- **Cuidados com o uso de Mutex e Signal/Wait**
  - **No caso de Posix**
    - Sempre usar *condition* associada a *mutex*
    - Sempre liberar o *mutex* depois de acordada

Exemplo: barreira

- **Exemplo de mutex e variáveis de condição**
  - **Fonte: [ANDREWS 2000]; seção 5.5.2**
  - **Implementação de barreira com contador**
  - **Barreira**
    - Todas as threads que atingem uma barreira esperam por todas
    - Todas continuam (são liberadas) automaticamente após todas terem chegado na barreira
    - Todas: definição por um contador e um limite
      - Inicializado em zero
      - Incrementa a cada thread que atinge a barreira
      - Libera quando contador = limite

Exemplo: barreira

- **Exemplo de mutex e variáveis de condição**
  - **Código:**

```
...
pthread_mutex_t barrier;  /* lock of barrier */
pthread_cond_t go;        /* condition variable */
int numWorkers;           /* number of threads */
int numArrived = 0;       /* number that arrived */
...
```

Exemplo: barreira

- **Exemplo de mutex e variáveis de condição**
  - **Código (cont.):**

```
/* a reusable counter barrier */
void Barrier() {
   pthread_mutex_lock(&barrier);
   numArrived++;
   if   (numArrived < numWorkers)
      pthread_cond_wait(&go, &barrier);
   else {
      numArrived = 0; /* last worker awakens */
      pthread_cond_broadcast(&go);
   }
   pthread_mutex_unlock(&barrier);
...
```

- **Resumo**
  - **Posix threads**
    - Criação
    - Semáforos
    - Locks
    - Varíaveis de condição
    - Barrreira
      - Implementação com Posix lock e condição