POSIX Threads

Mutexes

Grupo de Sistemas Distribuídos

Departamento de Informática
Escola de Engenharia
Universidade do Minho

Sistemas Operativos I
2006-2007

Conteúdo

Noção

Conceitos

- interface de exclusão mútua
- equivalente a uma fechadura que pode ser fechada (*lock*) antes de se aceder ao recurso partilhado e aberta (*unlock*) quando já não precisamos dele.

Primitivas de mutexes

Primitivas

- `pthread_mutex_init` (`man pthread_mutex_init`)
- `pthread_mutex_destroy` (`man pthread_mutex_destroy`)
- `pthread_mutex_lock` (`man pthread_mutex_lock`)
- `pthread_mutex_unlock` (`man pthread_mutex_unlock`)
- `pthread_mutex_trylock` (`man pthread_mutex_trylock`)

Biblioteca

Os programas devem ser linkados com a biblioteca pthread (**-lpthread**).

Mutex: inicialização

Métodos

- **Inicialização estática**
  (não é possível alterar os atributos do mutex)
- **Inicialização dinâmica**
  (é possível alterar os atributos do mutex)

Nota

Estado inicial do mutex: **unlocked**.

Mutex: inicialização estática

Inicialização estática

```
1  #include <pthread.h>
2
3
4  pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
5
6
7  typedef struct {
8      pthread_mutex_t mutex;
9      int valor;
10 } tipo_t;
11
12
13 tipo_t dados = { PTHREAD_MUTEX_INITIALIZER, 0 };
```

Mutex: inicialização dinâmica

Inicialização dinâmica

```
1  #include <pthread.h>
2
3  pthread_mutex_t mutex;
4
5  int main(void)
6  {
7      // ...
8      if (pthread_mutex_init(&mutex, NULL) != 0) {
9          fprintf(stderr, "pthread_mutex_init\n");
10         exit(1);
11     }
12     // ...
13     if (pthread_mutex_destroy(&mutex) != 0) {
14         fprintf(stderr, "pthread_mutex_destroy\n");
15         exit(2);
16     }
17     // ...
18 }
```

Mutex: utilização

Utilização

```
1  #include <pthread.h>
2
3  pthread_mutex_t mutex;
4
5  void * thread( void *arg )
6  {
7
8      pthread_mutex_lock(&mut);
9
10     //
11     // Seccao critica
12     //
13
14     pthread_mutex_unlock(&mut);
15
16     return NULL;
17 }
```

Exercício 1

Execução atómica vs não-atómica

O objectivo deste problema é observar os efeitos da não protecção de regiões críticas de código.

Exercício

Criar várias threads que incrementem uma variável partilhada (`x = x + 1`).

Exercício 1: primeira abordagem

Não protegendo a região crítica

```
1  long x;
2
3  void *incrementar( void *arg )
4  {
5      long i;
6
7      for (i = 0; i < 1000000; i++) {
8          x = x + 1;
9      }
10
11     return NULL;
12 }
```

Exercício 1: segunda abordagem

Protegendo a região crítica

```
1  long x;
2  pthread_mutex_t mut;
3
4  void *incrementar( void *arg )
5  {
6      long i;
7
8      for (i = 0; i < 1000000; i++) {
9          pthread_mutex_lock(&mut);
10         x = x + 1;
11         pthread_mutex_unlock(&mut);
12     }
13
14     return NULL;
15 }
```

Exercício 1: código completo (1 / 2)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <pthread.h>
4
5  #define NTHREADS 3
6
7  long x;
8  pthread_mutex_t mut;
9
10 void * incrementar(void *arg)
11 {
12     long i;
13
14     for (i = 0; i < 100000; i++) {
15         pthread_mutex_lock(&mut);
16         x = x + 1;
17         pthread_mutex_unlock(&mut);
18     }
19
20     return NULL;
21 }
```

Exercício 1: código completo (2 / 2)

```
1  int main(void)
2  {
3      pthread_t th[NTHREADS];
4      void *status[NTHREADS];
5      int i;
6      printf("Variavel global: %ld\n", x);
7
8      pthread_mutex_init(&mut, NULL);
9
10     for (i = 0; i < NTHREADS; i++) {
11         pthread_create(&th[i], NULL, incrementar, NULL);
12     }
13
14     for (i = 0; i < NTHREADS; i++) {
15         pthread_join(th[i], &status[i]);
16     }
17
18     pthread_mutex_destroy(&mut);
19
20     printf("Variavel global: %ld\n", x);
21     return 0;
22 }
```

Exercício 2

Primeira parte

Implemente um programa de gestão de uma base de dados de contas bancárias. As contas são compostas por nome do titular e saldo corrente e são identificadas por um número. As operações disponíveis deverão ser a consulta e o lançamento de valores numa conta especificada. Assuma que o número de contas é fixo (não haverá criação nem encerramento de contas). As operações disponíveis deverão ser pensadas de forma a que possam ser utilizadas num cenário multi-threaded.

Segunda parte

Modifique o programa anterior de forma a disponibilizar a operação de transferência entre contas.

Exercício 2: abordagem ao interface com a BD

Base de dados de contas: array de estruturas

```
1  typedef struct {
2      char titular[32];
3      float saldo;
4  } tConta;
5
6  float saldo(int nconta);
7
8  void depositar(int nconta, float montante);
9
10 void levantar(int nconta, float montante);
```

Conteúdo

Referências

Bibliografia

- **Programming with POSIX Threads**
  Autor: David R. Butenhof
  `http://www.awprofessional.com/bookstore/product.asp?isbn=0201633922`
- **Advanced Programming in the UNIX Environment, segunda edição**
  Autor: W. Richard Stevens e Stephen A. Rago
  Capítulo: 11 - Threads
  `http://www.apuebook.com/`