POSIX Threads

Grupo de Sistemas Distribuídos

Departamento de Informática
Escola de Engenharia
Universidade do Minho

Sistemas Operativos I
2006-2007

Conteúdo

1 Conceito
2 Propriedades
3 Caracterização
4 Aplicabilidade

Noção

Conceito

- Fluxos de execução independentes pertencentes ao mesmo processo.
- Tradicionalmente, em UNIX, os processos são criados com uma thread apenas.
- Pode ser visto como um procedimento que executa independentemente do programa principal que o define.

Conteúdo

1 Conceito
2 Propriedades
3 Caracterização
4 Aplicabilidade

Partilha

Threads do mesmo processo partilham

- instruções
- memória global (variáveis globais)
- descritores de ficheiros
- sinais (handlers e configurações)
- *current working directory*
- user e group IDs

Independência

Cada thread possui características próprias

- thread ID
- conjunto de registos (incluindo *Program Counter* e *Stack Pointer*)
- stack (variáveis locais e endereço de retorno)
- *errno*
- máscara de sinais
- prioridade

Conteúdo

1 Conceito

2 Propriedades

3 Caracterização

4 Aplicabilidade

“Processos leves!”

Threads são mais “baratas” que processos

- Sobrecarga do kernel é menor dado que este não copia memória e tabelas de descritores na criação de novas threads.
- A partilha de memória facilita a comunicação entre threads, eliminando assim a necessidade de IPC como pipes ou sockets.
- O custo da mudança de contexto entre threads é menor que entre processos.
- Curiosidade: a criação de uma thread pode ser 10 a 100 vezes mais rápida que a de um processo.

Cuidados na Utilização

Partilha de recursos requer atenção!

- Alterações, feitas por uma thread, em recursos partilhados são visíveis pelas restantes (e.g. fechar um ficheiro).
- A alteração da mesma zona de memória é possível por duas threads distintas, requerendo desta forma algum modo de sincronização.
- Dois apontadores com o mesmo valor apontam para os mesmos dados.
- Prevenir situações de corridas, que acontecem quando duas ou mais threads acedem a dados partilhados e o resultado final depende da ordem de execução das threads.

Conteúdo

1 Conceito
2 Propriedades
3 Caracterização
4 Aplicabilidade

Motivação da utilização de threads

Aumentar o desempenho

- Intercalar a execução de tarefas bloqueantes e tarefas de processamento intensivo utilizando threads.
- Rentabilizar o número de processadores utilizando threads para executar diferentes tarefas do mesmo processo.
- Algumas tarefas de um dado processo podem requerer maior prioridade no escalonamento, como tal podem ser executadas por threads com prioridade mais elevada.

Cenários de utilização

Tarefas candidatas a serem transformadas em threads

- Tarefas que bloqueiam por tempo indeterminado.
- Tarefas que utilizem massivamente o CPU.
- Tarefas que atendam eventos (e.g. executar um accept e correr uma rotina de atendimento).
- Tarefas com grau diferente de importância.
- Tarefas completamente independentes que podem perfeitamente serem executadas em paralelo.

Conteúdo

5 POSIX Threads

6 Observações

7 Exemplos

Primitivas de threads

Primitivas

- `pthread_create` (man pthread_create)
- `pthread_join` (man pthread_join)
- `pthread_exit` (man pthread_exit)
- `pthread_detach` (man pthread_detach)
- `pthread_self` (man pthread_self)

Biblioteca

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

Criação de threads

Primitiva: pthread_create

```
#include <pthread.h>

int pthread_create(
    pthread_t *thread,                  // identificador
    const pthread_attr_t *attr,         // atributo
    void *(*start_routine)(void*),      // rotina
    void *arg                           // argumento
);
```

Conteúdo

5 POSIX Threads

6 Observações

7 Exemplos

Notas práticas

- Se uma thread for “joinable” é preciso realizar o join para libertar os recursos dessa thread quando ela terminar.
- Uma thread termina quando for explicitamente invocada um pthread_exit ou implicitamente quando a função invocada pela thread retornar.
- Não retornar apontadores para variáveis automáticas, i.e. definidas dentro da função da thread.
- As funções utilizadas por threads têm o seguinte protótipo:
  `void * (*start_routine)(void *)`

Conteúdo

5 POSIX Threads

6 Observações

7 Exemplos

Exemplo 1: criar uma thread e esperar que ela termine

```
1  // Includes <stdio.h>, <stdlib.h>, <pthread.h>
2
3  void *ola( void *arg ) {
4      printf("Ola!\n");  return NULL;
5  }
6
7  int main(void) {
8      pthread_t th;  void *retv;
9
10     if (pthread_create(&th, NULL, ola, NULL)) {
11         fprintf(stderr, "pthread_create");
12         exit(EXIT_FAILURE);
13     }
14     if (pthread_join(th, &retv)) {
15         fprintf(stderr, "pthread_join");
16         exit(EXIT_FAILURE);
17     }
18     return 0;
19 }
```

Exemplo 2: imprimir o identificador da thread

```
1  // Includes <stdio.h>, <stdlib.h>, <pthread.h>
2
3  void *quemsoueu( void *arg ) {
4      printf("Thread ID: %lu\n", pthread_self());
5      return NULL;
6  }
7
8  int main(void) {
9      pthread_t th;
10     void *retv;
11
12     pthread_create(&th, NULL, quemsoueu, NULL);
13     pthread_join(th, &retv);
14
15     return 0;
16 }
```

Conteúdo

8 Exercícios

Exercício 1: contar números

Enunciado

Implemente um programa que permita contar todos números não-negativos num dado vector. A operação em questão deverá ser dividida pelo número de threads especificado na linha de comandos.

Exercício 2: contar números

Enunciado

Modifique o programa anterior de modo a que cada thread criada retorne um apontador para uma estrutura que contém o número de índices processados e o somatório dos índices que continham números positivos.

Conteúdo

9 Referências

Referências

Bibliografia

- **Foundations of Multithreaded, Parallel, and Distributed Programming**
  Autor: Gregory R. Andrews
  `http://www.cs.arizona.edu/~greg/mpdbook/`
- **Programming with POSIX Threads**
  Autor: David R. Butenhof
  `http://www.awprofessional.com/bookstore/product.asp?isbn=0201633922`