Processamento Paralelo e Distribuído

Marcelo Trindade Rebonatto

Memória Compartilhada

Roteiro

- Multiprogramação e multiprogramação leve
- Processos
- Threads
- Threads X Processos
- Modelos de Threads
- Pthreads (POSIX) básico

Multiprogramação

Conceitos

- Diversos processos na memória ao mesmo tempo
  - Windows: vários programas abertos
  - Linux: múltiplos processos (ps –fax)
- Pseudoparalelismo
  - Disputa pelo uso da CPU
  - Time slice (fatia de tempo)
- Escalonamento: *scheduling*
- Hiearquia

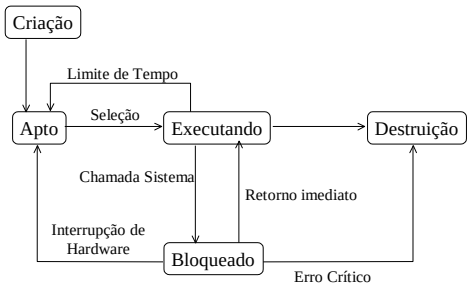
Multiprogramação

Processos

- Processo: programa em execução
  - 2 execuções do pine = 2 processos
  - Identificação pelo id (número)
- Proteção de memória
  - Área de memória exclusiva do processo
- Recursos exclusivos do processo
- Um único fluxo de execução

Multiprogramação

Estados de um processo



Multiprogramação Leve

Threads

- Processos leves
  - Recursos alocados por processo são compartilhados
    - ☆MEMÓRIA
  - Manipulação menos onerosa ao S.O.
- Múltiplos fluxos de execução
- Threads são geralmente criadas por processos

Threads

Conceitos

- Threads podem COMPARTILHAR memória
  - Não toda a memória
  - Definição de áreas de dados compartilhados
- Comunicação
  - Variáveis compartilhadas
  - Atribuição e recuperação de valores
  - **Idêntica a programação seqüencial**

Threads

Conceitos

- Escalonamento: depende do modelo
  - Pode concorrer igualmente com processos
  - Pode concorrer com o tempo para execução dentro do processo
- Cuidados com a sincronização
  - Instruções com variáveis compartilhadas consideradas críticas
  - Valores inconsistentes

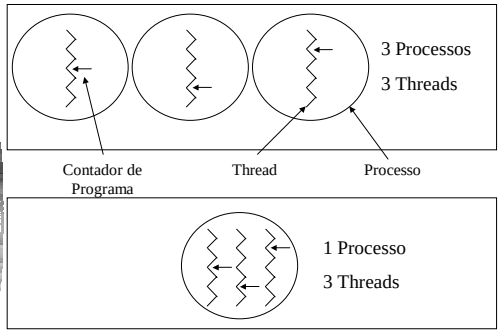
Multiprogramação Leve

Threads X Processos

- Processos
  - Uma linha de controle
  - Um contador de programa
- Threads: Light Weight Processes (LWP)
  - Múltiplas linhas de controle dentro de um processo
  - Processos leves
  - Multithreading

Threads X Processos

Multiprogramação Leve



Modelos & Implementações

Threads

- Bibliotecas
  - Integradas ou não ao S.O.
- Mecanismo de escalonamento
  - Modelo 1:1
  - Modelo N:1
  - Modelo M:N

Modelo 1:1

Threads

- "Threads sistema" (kernel)
- Direitos iguais a processos no escalonamento
- Manipuladas individualmente pelo S.O.
- Quando uma bloqueia (p.ex I/O)
  - Demais podem permanecer executando
- Modelo ideal para arquiteturas multiprocessadas

Modelo N:1

- "Threads aplicação" (usuário)
- Escalonamento dentro do processo
- Quando uma bloqueia (p.ex I/O)
  - Demais também bloqueiam
- Arquiteturas multiprocessadas não são exploradas na totalidade
- Menos onerosas (maior número de threads)

Modelo M:N

- Dentro de um processo
  - N threads sistema
  - M threads usuário
- Relação: geralmente M > N
- Benefícios
  - Manipulação leve: usuário (M)
  - Muitos fluxos de execução: sistema (N)

Manipulação de dados compartilhados

```
int x=100; /* x e uma variavel compartilhada */
```

/* thread A */	/* thread B */

/* le dado compartilhado */
a = x;
/* realiza operacao matem. */
a = a + 10;
/*altera dado compartilhado*/
x = a;
	/* le dado compartilhado */
	b = x;
	/* realiza operacao matem. */
	b = b - 10;
	/*altera dado compartilhado*/
	x = b;

- Valor final de "x"?

Manipulação de dados compartilhados

- Execução com valores inconsistentes

Thread	Instrução	X	A	B
A	a = x;	100	100	-
A	a = a + 10;	100	110	-
B	b = x;	100	110	100
A	x = a;	110	110	100
B	b = b – 10;	110	110	90
B	x = b;	90	110	90

- Valor final de “x” ?

Manipulação de dados compartilhados

- Soluções: controle para seções criticas
  - Semáforos
    - Registro composto por um inteiro e um apontador
    - Maior controle e processamento
  - Exclusão mútua: semáforo binário
    - Mutex: *Mutual Exclusion*
    - Uma estrutura de dados
    - Controle eficiente

Dados compartilhados com mutex

```
int x=100;    /* x e uma variavel compartilhada */
mutex tranca; /* mutex associado a x */
init ( &tranca ); /* inicializa o mutex */
```

```
      /* thread A */
lock ( &tranca );
/* le dado compartilhado */
a = x;
/* realiza operacao matem. */
a = a + 10;
/*altera dado compartilhado*/
x = a;
unlock ( &tranca );
```

```
      /* thread B */
lock ( &tranca );
/* le dado compartilhado */
b = x;
/* realiza operacao matem. */
b = b - 10;
/*altera dado compartilhado*/
x = b;
unlock ( &tranca );
```

- Garantido o valor final de “x”.

pthreads

Conceitos

- Biblioteca de threads que segue o padrão POSIX (*Portable Operating System Interface*)
- Normalmente distribuída com linux e solaris
- Utilizada para escrita de programas em C/C++
- Permite criação de threads
  - Sistema 1:1 (padrão)
  - Usuário N:1

pthreads

Utilização & compilação

- Dentro de programas, incluir a `pthread.h`

```
#include <pthread.h>
```

- Compilar linkeditando a biblioteca "`pthread`"

```
$ gcc thr_ex.c –o thr_ex -lpthread
```


pthreads

Criação de threads

- Instruções: definidas numa uma função

```
void *func (void *args);
```

- Criação: `pthread_create`

```
int pthread_create(pthread_t *thid,
      const pthread_attrib_t *attrib,
      void *(*funcao) (void *),
      void *args);
```

Criação de threads

pthreads

- `thid`: identificador da thread criada
- `attrib`: parâmetros de criação
  - NULL: parâmetros padrão
- `funcao`: endereço da função a ser criada
- `args`: parâmetros a serem enviados para a thread

Identificação de threads

pthreads

- Dentro da thread

```
pthread_t pthread_self( void );
```

- Fora da thread

```
pthread_t thid;
```

- `pthread_t`: tipo thread
  - casting para int:
    - `printf("%d",(int) pthread_self());`
    - `printf("%d", (int) thid);`

Destruição e sincronização

pthreads

- Tempo de vida
  - Igual ao necessário para execução
- Sincronização: disponibiliza dados

```
pthread_t pthread_join(
    pthread_t thid, void **ret);
```

- `thid`: identifica a thread
- `ret`: armazena o valor de retorno

pthreads - exemplo

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void *func(void *str) {
   printf("%s\n", (char *) str);
   printf("Sou a thread %d !\n", pthread_self());
}

int main() {
   pthread_t thid;
   char *str = "!!!!! Oi mundo !!!!!";
```


pthreads - exemplo

```
   ...
   ...

   if (pthread_create(&thid, NULL, func, str)!=0 ){
      printf("Ocorreu um erro\n");
      return(0);
   }

   printf("Foi criada a thread %d\n", thid);
   pthread_join(thid, NULL);
   printf("A thread %d ja terminou.\n", thid);

   return(1);
}
```


pthreads

Variáveis compartilhadas

- Qualquer variável global
  - Pode ser compartilhada
- Acesso:
  - Acessada e modificada por qualquer thread (processo)
- Seções críticas: mutex
- pthreads não disponibiliza semáforos
  - Mutex
  - Contador global (compartilhado)

pthreads – variáveis compartilhadas

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

int x; /* Variavel compartilhada pelas threads */

void *soma(void *str) {
   int a;

   printf("Somando ...\n");
   a = x;
   a = a + 10;
   sleep(1); /* faz a thread 'dormir' 1 segundo */
             /* e perder a CPU                  */
   x = a;
   printf("Valor de x em Somando %d\n", x);
}
```


pthreads – variáveis compartilhadas

```
void *diminui(void *str) {
   int b;

   printf("Diminuindo ...\n");
   b = x;
   b = b - 10;
   x = b;
   printf("Valor de x em Somando %d\n", x);
}

int main() {
   pthread_t t_soma;
   pthread_t t_dimi;

   x = 100;
   printf("Valor inicial de x %d\n", x);
```


pthreads – variáveis compartilhadas

```
  if (pthread_create(&t_soma, NULL, soma, NULL) != 0 ) {
      printf("Erro na cricao da thread soma\n");
      return(0);
  }

  if (pthread_create(&t_dimi, NULL, diminui, NULL)!= 0) {
      printf("Erro na criacao da thread diminui\n");
      return(0);
  }

  printf("Criadas threads %d e %d\n",t_soma, t_dimi);

  pthread_join(t_soma, NULL);
  pthread_join(t_dimi, NULL);

  printf("Valor Final de x %d\n", x);
  return(1);
}
```

Mutex

- Definição: `pthread_mutex_t tranca;`
- Inicialização

```
pthread_mutex_init(
   pthread_mutex_t *fecha,
   pthread_mutexattr_t *attrib)
```

- `fecha:` variável tipo mutex para controle
- `attrib:` utilizado para o valor inicial
  - Padrão: NULL (aberto)

pthreads

Mutex: manipulação

- lock:

```
int pthread_mutex_lock(
       pthread_mutex_t *fecha);
```

- unlock:

```
int pthread_mutex_unlock(
       pthread_mutex_t *fecha);
```

- `fecha:` variável tipo mutex de controle

pthreads

Dados compartilhados com mutex

```
int x=100; /* variavel global e compartilhada */
pthread_mutex_t trc; /* mutex associado a x   */
pthread_mutex_init(&trc,NULL); /* inicializa mutex */
```

/* thread A */	/* thread B */
pthread_mutex_lock(&trc);	pthread_mutex_lock(&trc);
/* le dado compartilhado */	/* le dado compartilhado */
a = x;	b = x;
/* realiza operacao matem. */	/* realiza operacao matem. */
a = a + 10;	b = b - 10;
/*altera dado compartilhado*/	/*altera dado compartilhado*/
x = a;	x = b;
pthread_mutex_unlock(&trc);	pthread_mutex_unlock(&trc);

pthreads