GSI018 – Sistemas Operacionais

Threads

Universidade Federal de Uberlândia
Faculdade de Computação
Prof. Dr. rer. nat. Daniel D. Abdala

Na Aula Anterior...

- Comunicação entre Processos
  - Pipes;
  - Memória compartilhada;
  - Sockets;
- Seção Crítica;
- Suporte em Hardware para Sincronismo;
- Semáforos;
- Mutexes;
- Barreiras;

Nesta Aula

- O Conceito de Threads;
- Exemplo de Threads;
- Threads POSIX;

O Conceito de Threads

- De maneira simplificada, um thread é uma linha de execução dentro de um processo;
- Por definição todo processo possui pelo menos uma thread, ou uma linha de execução;
- Este é o caso de sistemas operacionais que não implementam threads;
- De onde vem o nome?
  - Provavelmente porque a execução de um programa é representada graficamente por uma linha ou fio;

O Conceito de Threads

- Um modo de ver um processo é encará-lo como um meio de agrupar recursos relacionados:
- Basicamente dois componentes:
  - Recursos;
  - Execução.

Itens por Processo	Itens por Thread
Espaço de endereçamento	Contador de programa
Variáveis globais	Registradores
Arquivos abertos	Pilha
Processos filhos	Estado
Alarmes pendentes	
Sinais e manipuladores de sinais	
Informações de contabilidade	

Porque Usar Threads?

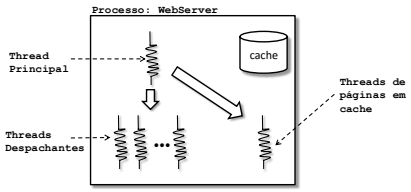
- Processos não compartilham espaço de endereçamento;
- Threads são mais rápidos – não há necessidade de invalidar páginas de memória quando se troca a thread de execução;
- Mais eficientes de criar e destruir;
- Melhor desempenho. Processo não precisa necessariamente liberar seu quanta quando bloqueia para E/S;

Exemplo

- Servidor de Páginas Web;
- Processo com thread única:
  - Recebe uma requisição
  - Atende-a até sua completude
  - Entra em espera pela próxima requisição
  - Possibilidade de perda de requisições
- Com múltiplas threads ...

7

Exemplo



8

Threads POSIX

- Até o momento abordamos apenas o modelo clássico de threads;
- Padrão IEEE 1003.1c-2001 → define o pacote Pthread;
- A maioria dos SOs baseados no UNIX implementa este padrão;
- Mais de 60 funções.

9

Principais Funções de Pthread

Chamada de Pthread	Descrição
pthread_create	Cria um novo thread
pthread_exit	Conclui a chamada de thread
pthread_join	Espera que um thread específic seja abandonado
pthread_yield	Libera a CPU ara que outro thread seja executado
pthread_attr_init	Cria e inicializa uma estrutura de atributos do thread
pthread_att_destroy	Remove uma estrutura de atributos do thread

```
#include <pthread.h>
int pthread_create(pthread_t *thread, pthread_attr_t *attr, void
*(*start_routine)(void *), void *arg);

void pthread_exit(void *retval);

int pthread_join(pthread_t th, void **thread_return);
```

10

Exemplo

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

void *thread_function(void *arg);

char message[] = "Hello World";

int main() {
    int res;
    pthread_t a_thread;
    void *thread_result;
    res = pthread_create(&a_thread, NULL, thread_function, (void *)message);
    if (res != 0) {
          perror("Thread creation failed");
          exit(EXIT_FAILURE);
    }
    printf("Waiting for thread to finish...\n");
    res = pthread_join(a_thread, &thread_result);
    if (res != 0) {
          perror("Thread join failed");
          exit(EXIT_FAILURE);
    }
```

11

Exemplo

```
    printf("Thread joined, it returned %s\n", (char *)thread_result);
    printf("Message is now %s\n", message);
    exit(EXIT_SUCCESS);
}

void *thread_function(void *arg) {
    printf("thread_function is running. Argument was %s\n", (char *)arg);
    sleep(3);
    strcpy(message, "Bye!");
    pthread_exit("Thank you for the CPU time");
}
```

12

Implementação de Threads

- Modos de Implementação:
  - No espaço do usuário;
  - No núcleo do sistema operacional;
  - Implementações Híbridas.

13

Implementação no Espaço do Usuário

- Podem ser implementadas em SOs que não suportam threads;
- As threads executam no topo de um sistema de tempo de execução (runtime), que é uma coleção de rotinas que gerenciam threads;
- Cada processo precisa manter uma tabela de threads (análoga a tabela de processos);
  - Contador de programa;
  - Registradores;
  - Ponteiro de pilha;
  - Estado.

14

Implementação no Espaço do Usuário

- Bloqueio local ocorre via chamada de uma função específica;
  - Ex: thread espera que outra termine;
- Chaveamento de threads pode ser feito em poucas instruções;
  - Pelo menos uma ordem de magnitude mais rápido que chaver processos;
- Quando uma thread decide parar de executar (`thread_yield`)
  - Salva o contexto da thread;
  - Chama automaticamente o escalonador de threads;
  - Rotinas locais, muito mais rápidas de invocar que o escalonador no núcleo;

15

Implementação no Espaço do Usuário

- Cada processo pode ter seu algoritmo de escalonamento personalizado;
- Escalam melhor, pois utilizam o espaço de endereçamento do próprio processo;
- Problemas:
  - Como implementar as chamadas do sistema com bloqueio?
  - `select`: prevê se uma syscall bloqueará
  - Reescrever parte das bibliotecas do sistema operacional;
  - Funções de wrapping: primeiro testa com `select` e se for bloquear chama o escalonador de threads e agenda a chamada bloqueante para ser invocada mais tarde pelo sistema monitor;

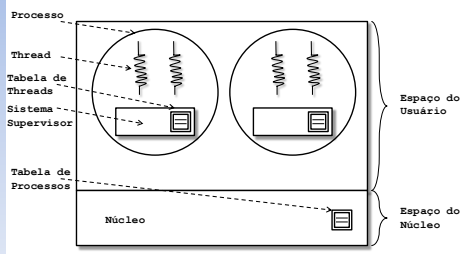
16

Implementação no Espaço do Usuário

- Problemas:
  - Threads precisam explicitamente liberar o processador para outro thread. Não há o conceito de interrupção e quanta;
  - Programadores requerem threads em programas que bloqueiam em E/S muito frequentemente;
    - Como haverá de qualquer forma um chaveamento para o núcleo o "housekeeping" das threads poderia ser feito pelo sistema operacional em modo núcleo;

17

Threads no Espaço do Usuário



18

Implementação no Núcleo

- Sistema de tempo de execução não é necessário;
- Não há tabelas de threads nos processos;
- Uma única tabela de threads eh mantida pelo SO;
- Criação, destruição, joins tudo é controlado por chamadas do sistema;
- Não há necessidade de checar chamadas ao sistema para verificar se elas bloquearão;

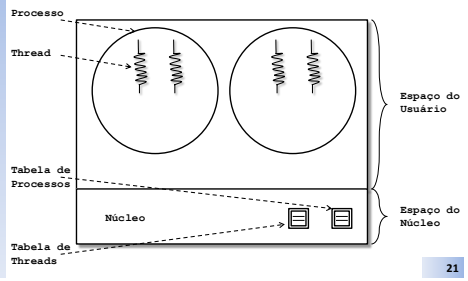
19

Implementação no Núcleo

- Em caso de falta de página, é fácil para o SO verificar se há outra thread que pode executar;

20

Threads no Núcleo



21

Implementação Híbrida

- Threads de núcleo podem ter diversas threads em espaço do usuário;
- Criar threads de usuário não bloqueantes apenas;
- Sobrecarga no programador;

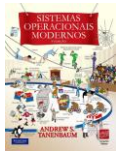
22

Threads Pop-up

- Threads são muito úteis em sistemas distribuídos;
- Geralmente um sistema cria um thread `receive` e bloqueia ela até que uma mensagem chegue;
- A chegada de uma mensagem desencadeia a criação de uma nova thread;
- Como a thread é nova é exatamente igual a todas as outros demais pop-ups ela pode ser criada rapidamente;

23

Bibliografia - Básica



- 3ª Edição
- Páginas 57-68

24

Bibliografia - Básica

- 7ª edição
- Páginas 127-150

Bibliografia - Adicional

- Capítulo 12