Sistemas Operacionais Modernos

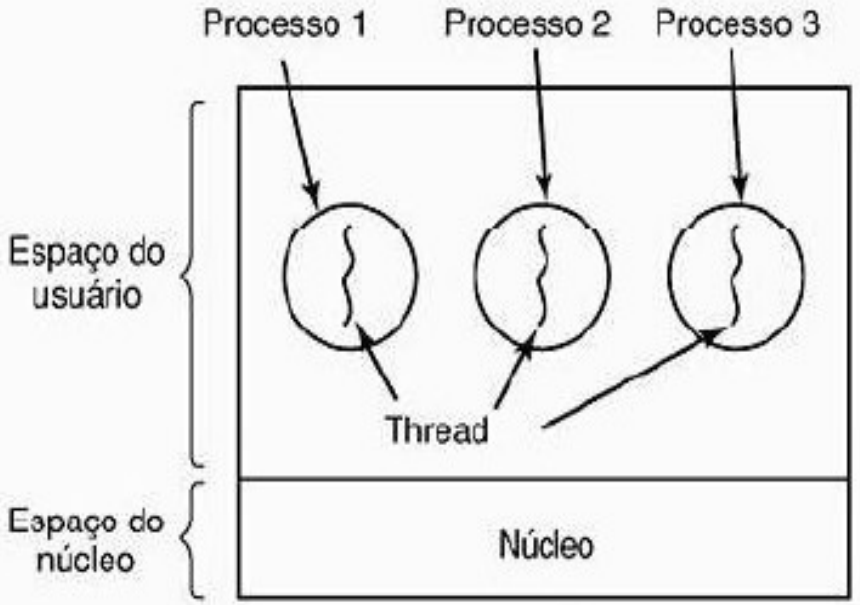
C. T. Inf. para Internet
Prof. Vinícius Alves Hax

Na aula anterior

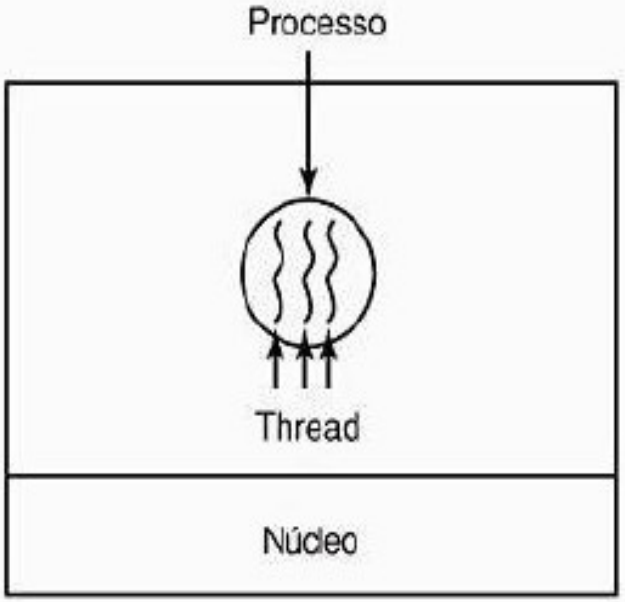
- Processos e escalonamento

Hoje

- Threads
- Comunicação entre processos


Processo 1
Processo 2
Processo 3
Espaço do usuário
Thread
Espaço do núcleo
Núcleo

(a)


Processo
Thread
Núcleo

(b)

Threads

- Threads às vezes são chamadas de processos leves
  - Compartilham informações
  - Não precisam de tanta memória
- Podem ser enxergadas pelo SO. Alguns problemas disso:
  - Como saber qual thread começar?
  - Enviar sinais para qual thread?

Threads

- Disponível na maioria das linguagens de programação
- Modelo bastante intuitivo
- Não depende de hardware especial
- As threads compartilham (a maior parte) da memória de um processo

Funções básicas de threads

- thread_create(): um processo ou thread cria outra thread;
- thread_exit(): um thread faz a sua tarefa e após isso encerra voluntariamente;
- thread_wait(): uma thread espera a execução de outra;
- thread_cancel(): uma thread finaliza outra;
- thread_yield(): a thread voluntariamente cede sua vez para outras;

Exemplo de thread_wait

```
calcula_soma_linha(numero_linha){...}


thread t[numero_linhas];
for(i=1; i<=numero_linhas; i++)
  t[i] = thread_start(calcula_soma_linha(numero_linha))
thread_wait(t); //Barreira
soma_resultados();
```

$$\begin{bmatrix} 4, & 6, & 7; \\ 6, & 10, & 11; \\ 12, & 13, & 10 \end{bmatrix}$$

Exemplo thread_cancel

```
T2 = thread_create(faz_tarefa())
while(TRUE)  {
  Texto = le_teclado();
  If (testa_ctrl_z(texto)) {
    thread_cancel(T2);
    Exit();
  }
}
```

Exemplo de thread_yield

```
tarefa_thread() {
...
while(TRUE) {
  pacote = leio_pacote_rede();
  tratamento(pacote);
  thread_yield();
  }
}
```

Comunicação interprocessos

Mesmo bem intencionados processos que compartilham informações podem gerar dados inconsistentes. Esse compartilhamento que pode gerar problemas é chamado de “condição de disputa” ou “condition race”

Exemplo

- Saldo anterior = R$100,00. Processo P1 (saque 50,00) e P2 (depósito 200,00)

  P1: saldop1 = le_saldo() // 100

  P2: saldop2 = le_saldo() // 100

  P1: saldop1 = saldop1 – 50 // 50

  P2: saldop2 = saldop2 + 200 // 300

  P1: escreve_saldo(saldop1) // 50

  P2: escreve_saldo(saldop2) // 300

Região crítica

Chamamos de “região crítica” as partes de um código que podem resultar em dados inconsistentes. Também é usado o termo “seção crítica”

Como resolver?

- Estruturas que sinalizem a exclusão mútua de acesso às regiões críticas: mutex, semáforos, monitores, etc
- O mais comum (e simples) é o mutex (*mutual exclusion*).
- Geralmente um mutex tem uma operação down() e outra up(). Down significando “Vou tentar pegar esse recurso. Se eu conseguir ele ficará indisponível”. Up significando “Já usei o recurso. Ele está disponível”

Uso de mutex

P1: down(mutex_saldo)

P1: saldop1 = le_saldo()

P2: down(mutex_saldo) // P2 ‘trava’

P1: saldop1 = saldop1 – 50

P1: escreve_saldo(saldop1)

P1: up(mutex_saldo) // Isso destrava P2 mas deixa o mutex em down – Se outro processo executar down() também irá travar

P2: executa normalmente ....

Deadlocks

Deadlocks poderia ser traduzido como “travamento mortal”

Ocorre quando, por exemplo, um processo P1 travou um recurso R1 e um processo P2 travou um recurso R2 e P1 precisa de R2 e P2 precisa de R1 para prosseguir.

Exemplo de Deadlock

- Ana e José estão fazendo TCC. Ana pegou o livro de Javascript. José pegou o livro de CSS. Um dia depois Ana descobriu que precisa do livro de CSS e José descobriu que precisa do livro de Javascript.
  - Nenhum dos dois vai devolver o seu livro e os dois ficarão “travados”

Lidando com deadlocks

- Detecção e recuperação
  - Ex: Ana ou José ligam para a biblioteca e ela propõe alguma solução
- Alocando os recursos cuidadosamente
  - Ex: Ana pega os livros de CSS e Javascript no primeiro dia
    - Porém o “sistema perde eficiência”: José fica sem livros e Ana não vai ler ambos os livros todo o tempo

Lidando com deadlocks

- Prevenção
  - Aumentando o número de recursos ou permitindo que pelo menos a leitura deles possa ser feita de forma simultânea:
    - Aumento: Comprar mais livros
    - Acesso simultâneo: Livros digitais
- Algoritmo do avestruz

Algoritmo do avestruz

Starvation

- Pode ser traduzido como “inanição” ou “morrer de fome”
- Problema relacionado ao deadlock no qual processos tentam obter os recursos que precisam e não conseguem.
  - Ex: Se um terceiro aluno estivesse precisando do livro de Javascript mas ele nunca estivesse disponível

Links das imagens

- Slide 4: Livro “Sistemas Operacionais Modernos”, Tanenbaum.
- Avestruz: https://super.abril.com.br/blog/oraculo/o-avestruz-enfia-mesmo-a-cabeca-na-areia-quando-esta-com-medo/