Processos e threads

Guilherme Henrique de Souza Nakahata

Universidade Estadual do Paraná - Unespar

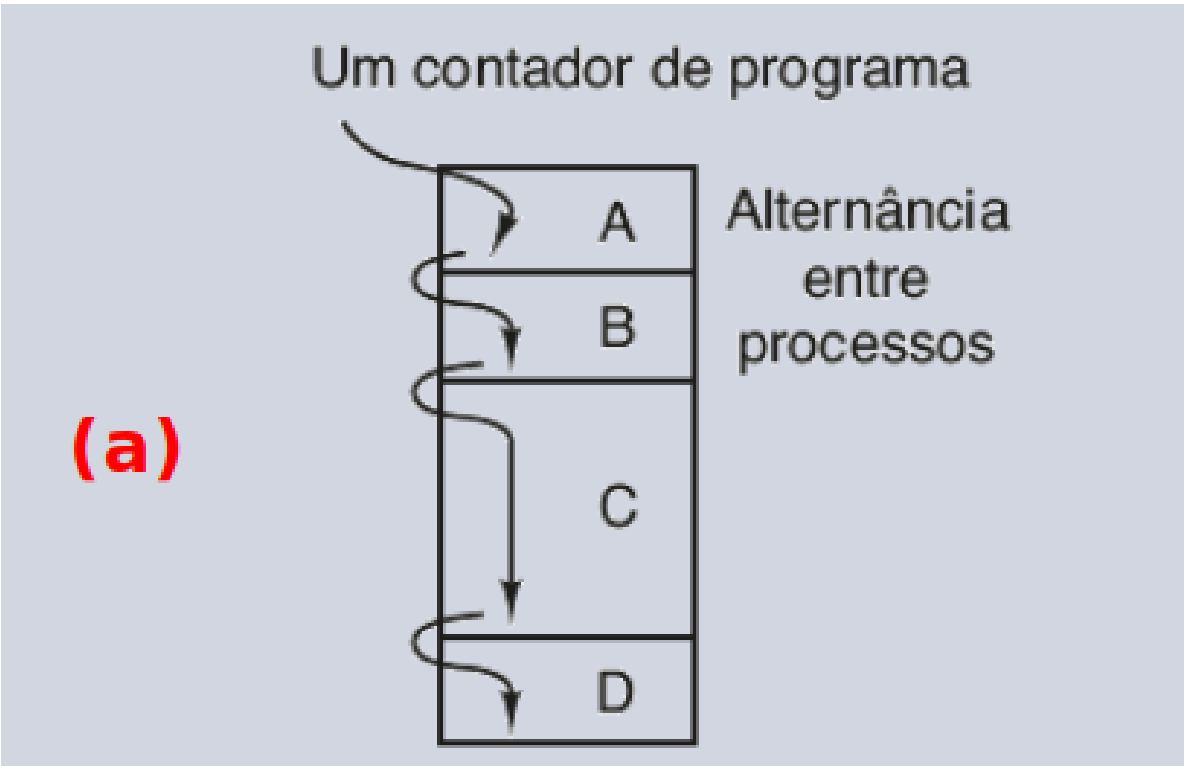
31 de Março de 2023

Processos

- O conceito mais central em qualquer sistema operacional é o **processo**: uma abstração de um programa em execução.
- Um processo é apenas uma instância de um programa em execução, incluindo os valores atuais do contador do programa, registradores e variáveis.
- Processos podem ser criados e terminados dinamicamente.
- Cada processo tem seu próprio espaço de endereçamento.

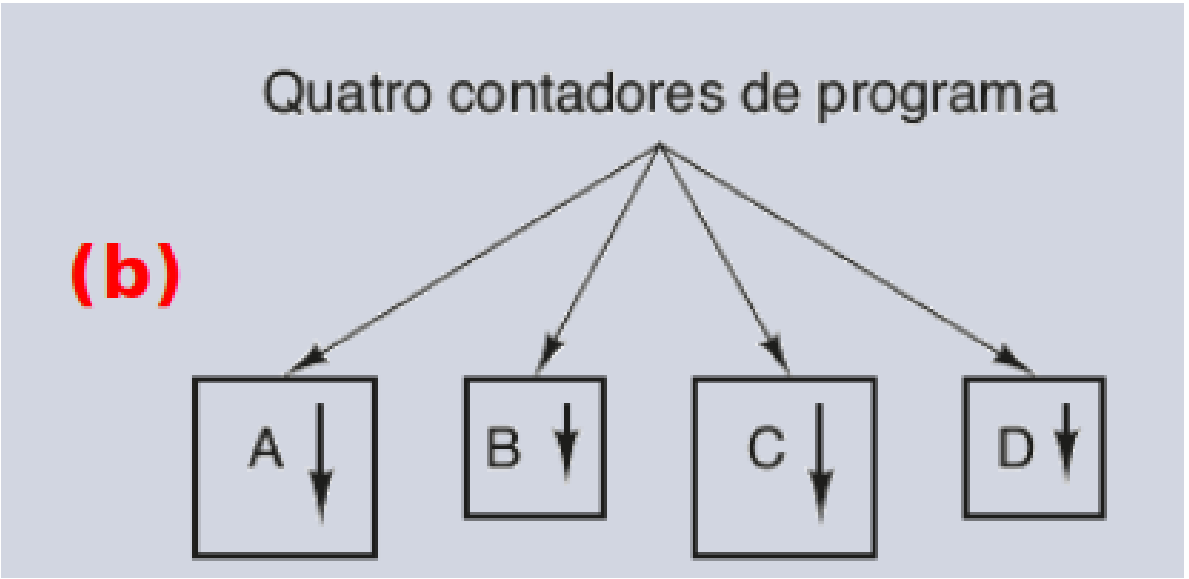
O modelo de processo

- Na figura (a) vemos um computador multiprogramando quatro programas na memória.



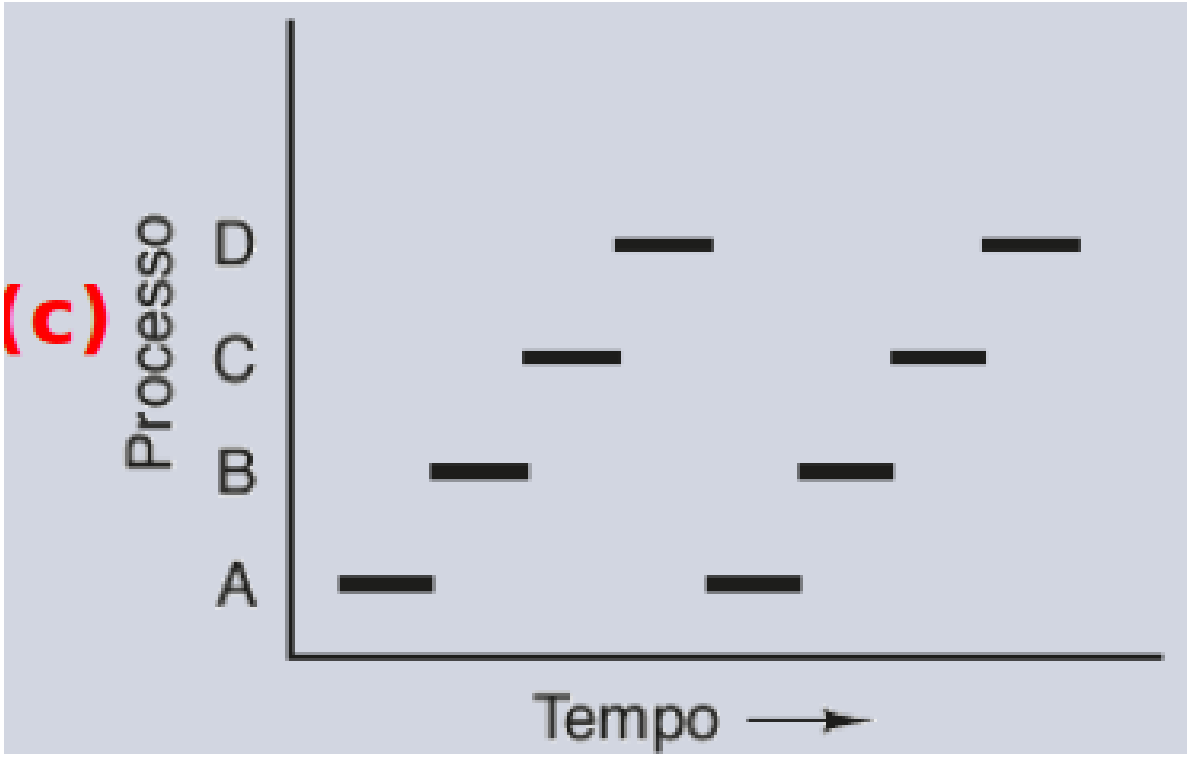
O modelo de processo

- Na figura (b) vemos quatro processos, cada um com seu próprio fluxo de controle e sendo executado independente dos outros



O modelo de processo

- Na figura (c) vemos que, analisados durante um intervalo longo o suficiente, todos os processos tiveram progresso, mas a qualquer dado instante apenas um está sendo de fato executado.



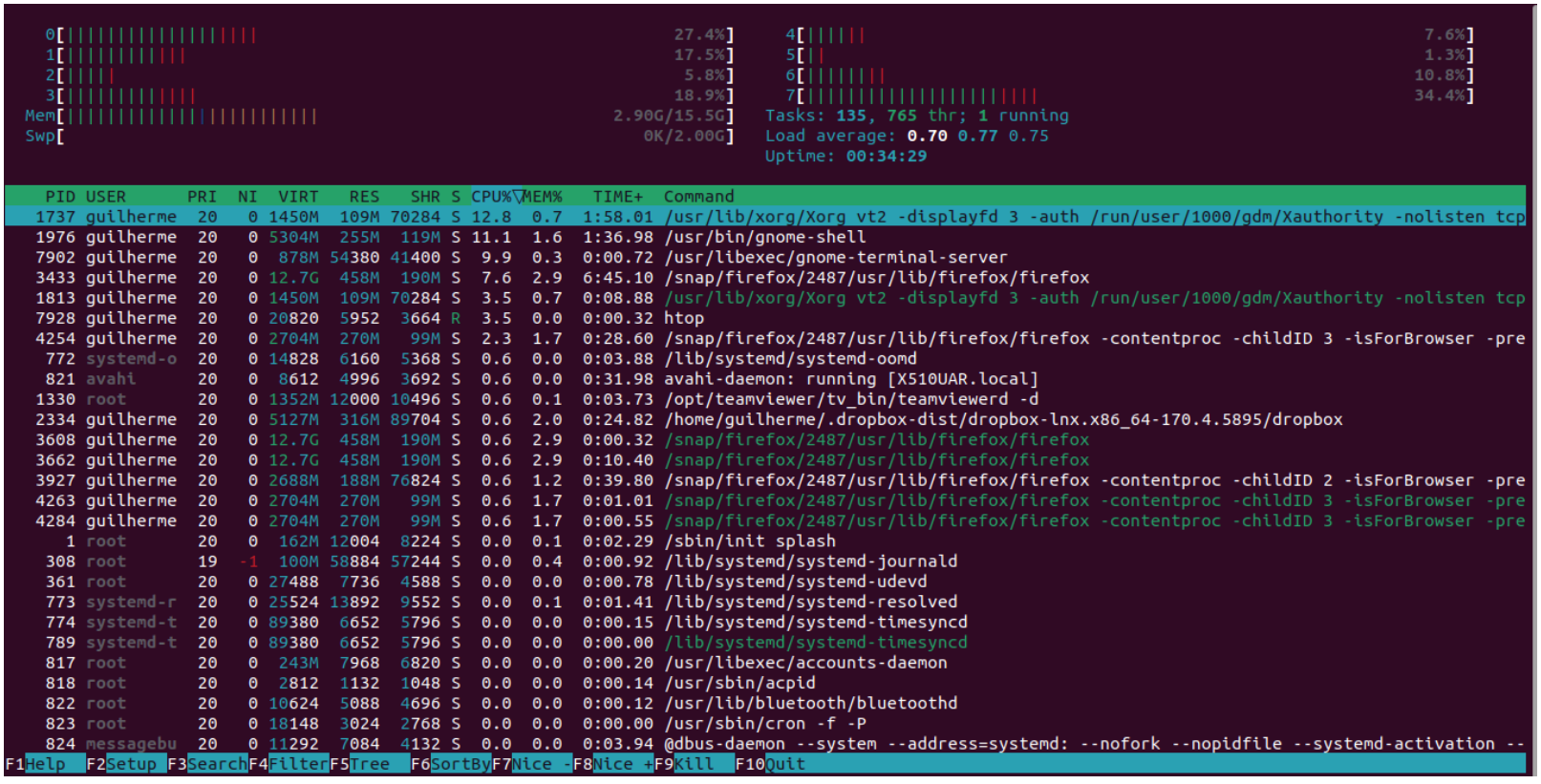
Criação de processos

- Sistemas operacionais precisam de alguma maneira criar processos.
- Quatro eventos principais fazem com que os processos sejam criados:
  1. Inicialização do sistema.
  2. Execução de uma chamada de sistema de criação de processo por um processo em execução.
  3. Solicitação de um usuário para criar um novo processo.
  4. Início de uma tarefa em lote.

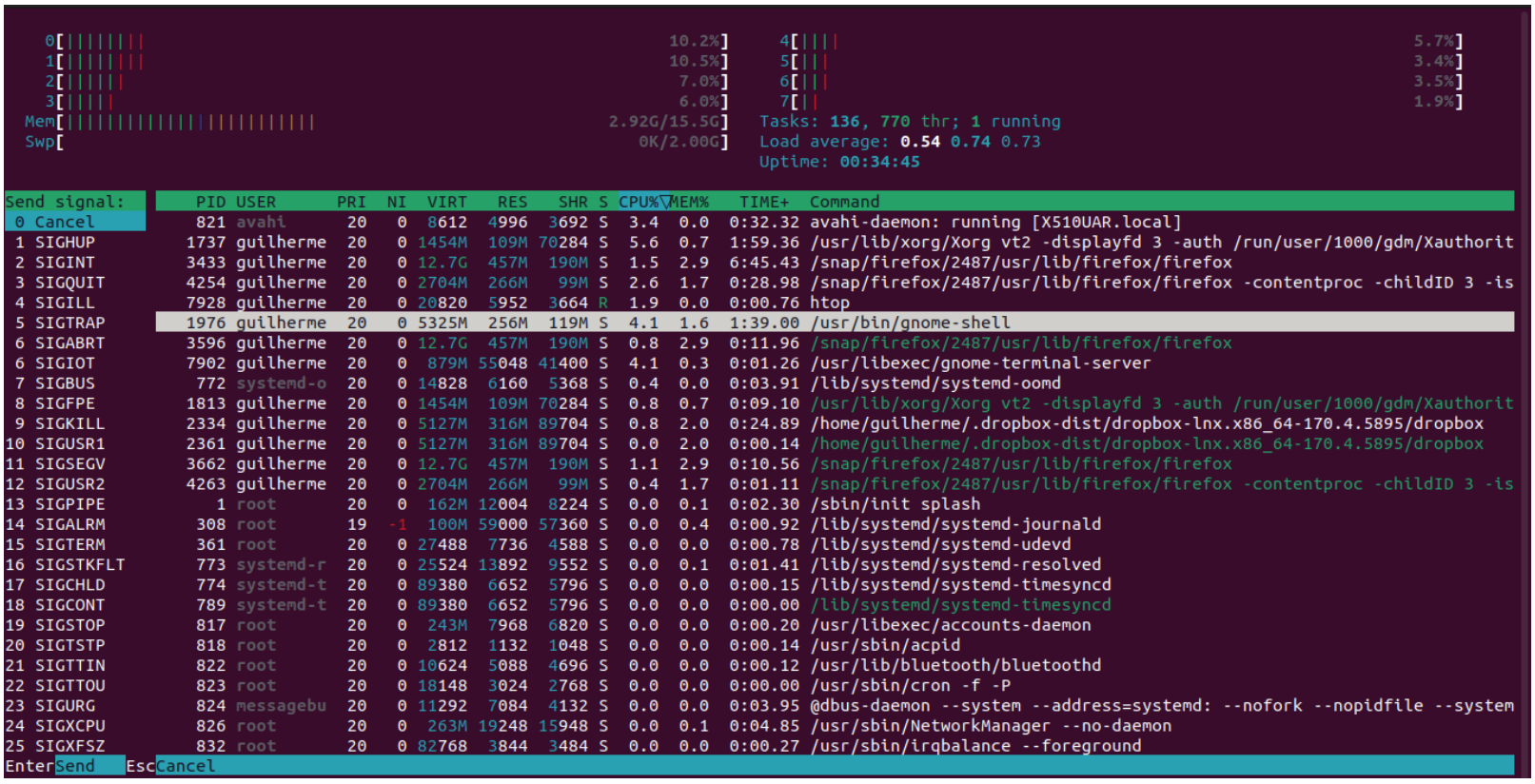
Término de processos

- Após um processo ter sido criado, ele começa a ser executado e realiza qualquer que seja o seu trabalho;
- Cedo ou tarde, o novo processo terminará, normalmente devido a uma das condições a seguir:
  1. Saída normal (voluntária).
  2. Erro fatal (involuntário).
  3. Saída por erro (voluntária).
  4. Morto por outro processo (involuntário).

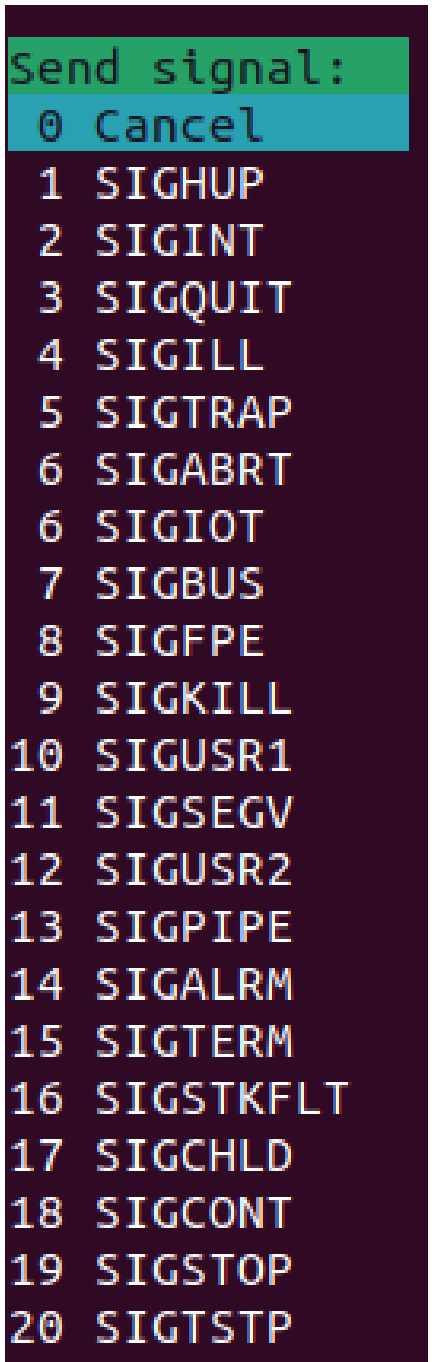
Término de processos



Término de processos



Término de processos



Hierarquias de processos

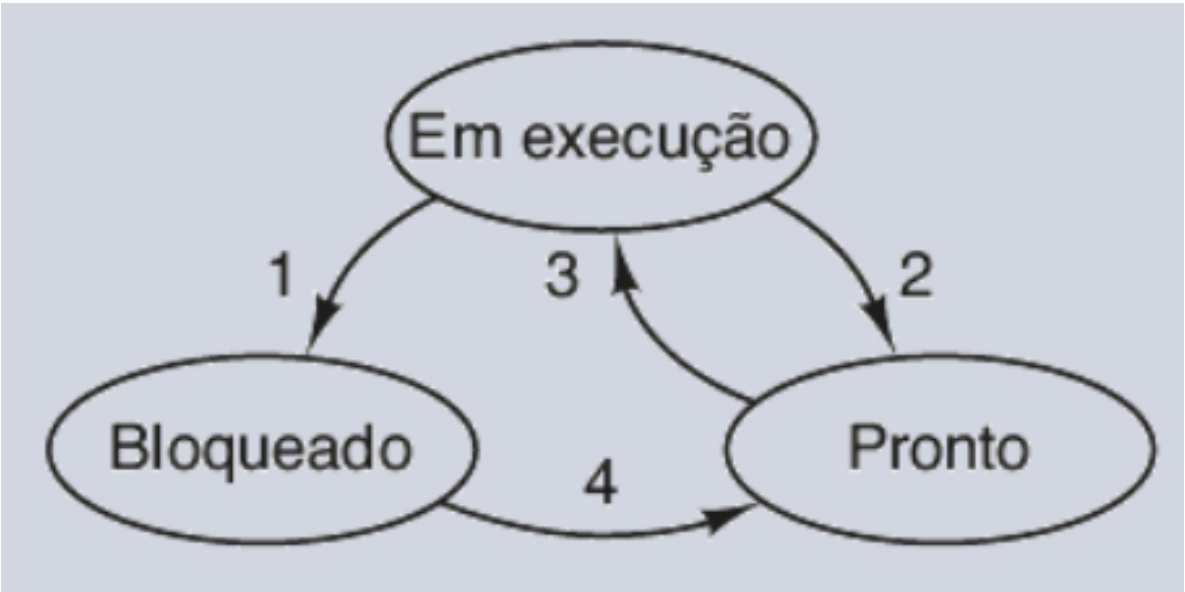
- Quando um processo cria outro, o processo pai e o processo filho continuam a ser associados de certas maneiras.
- O processo filho pode em si criar mais processos, formando uma hierarquia de processos.

Estados de processos

- Embora cada processo seja uma entidade independente, com seu próprio contador de programa e estado interno, processos muitas vezes precisam interagir entre si.
- Um processo pode gerar alguma saída que outro processo usa como entrada.

Estados de processos

- Três estados de um processo:
  - 1. Em execução.
  - 2. Pronto.
  - 3. Bloqueado.



Estados de processos

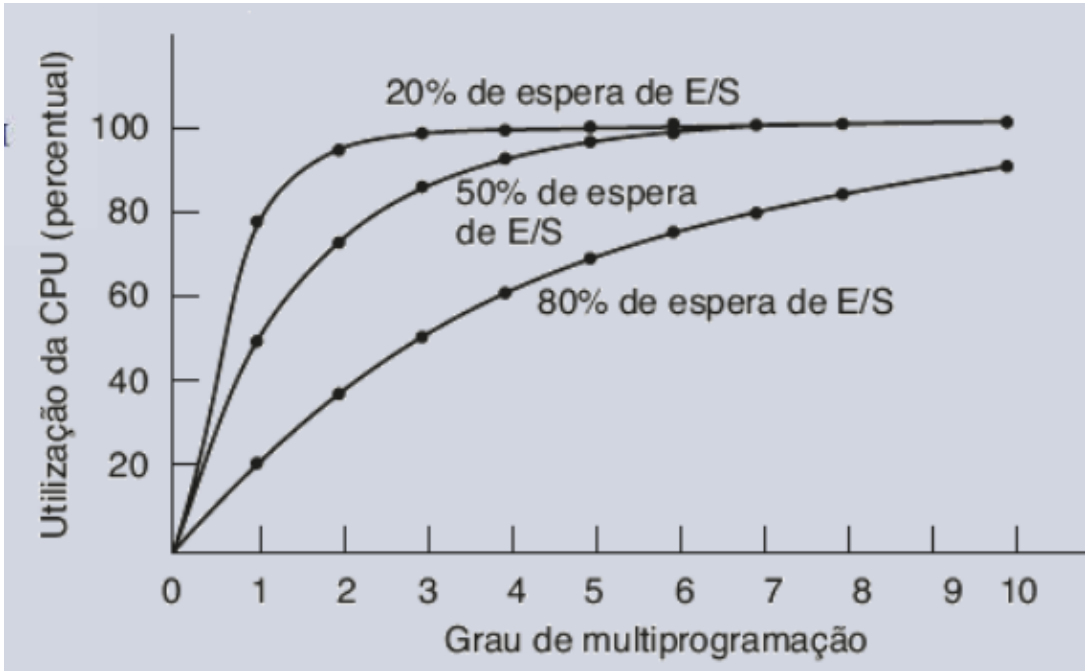
- O nível mais baixo de um sistema operacional estruturado em processos controla interrupções e escalonamento.
- Acima desse nível estão processos sequenciais.
- Todo o tratamento de interrupções e detalhes sobre o início e parada de processos estão ocultos no escalonador.

Implementação de processos

- Para implementar o modelo de processos, o sistema operacional mantém uma tabela chamada de **tabela de processos**, com uma entrada para cada um deles.
- Um processo pode ser interrompido milhares de vezes durante sua execução.
- Sempre retornando para o estado em que se encontrava antes de ser interrompido.

Modelando a multiprogramação

- Quando a multiprogramação é usada.
- A utilização da CPU pode ser aperfeiçoada.



Exemplo

Threads

- Para algumas aplicações é útil ter múltiplos threads de controle dentro de um único processo.
- Esses threads são escalonados independentemente e cada um tem sua própria pilha.
- Todas as threads em um processo compartilham de um espaço de endereçamento comum.
- Threads podem ser implementados no espaço do usuário ou no núcleo.

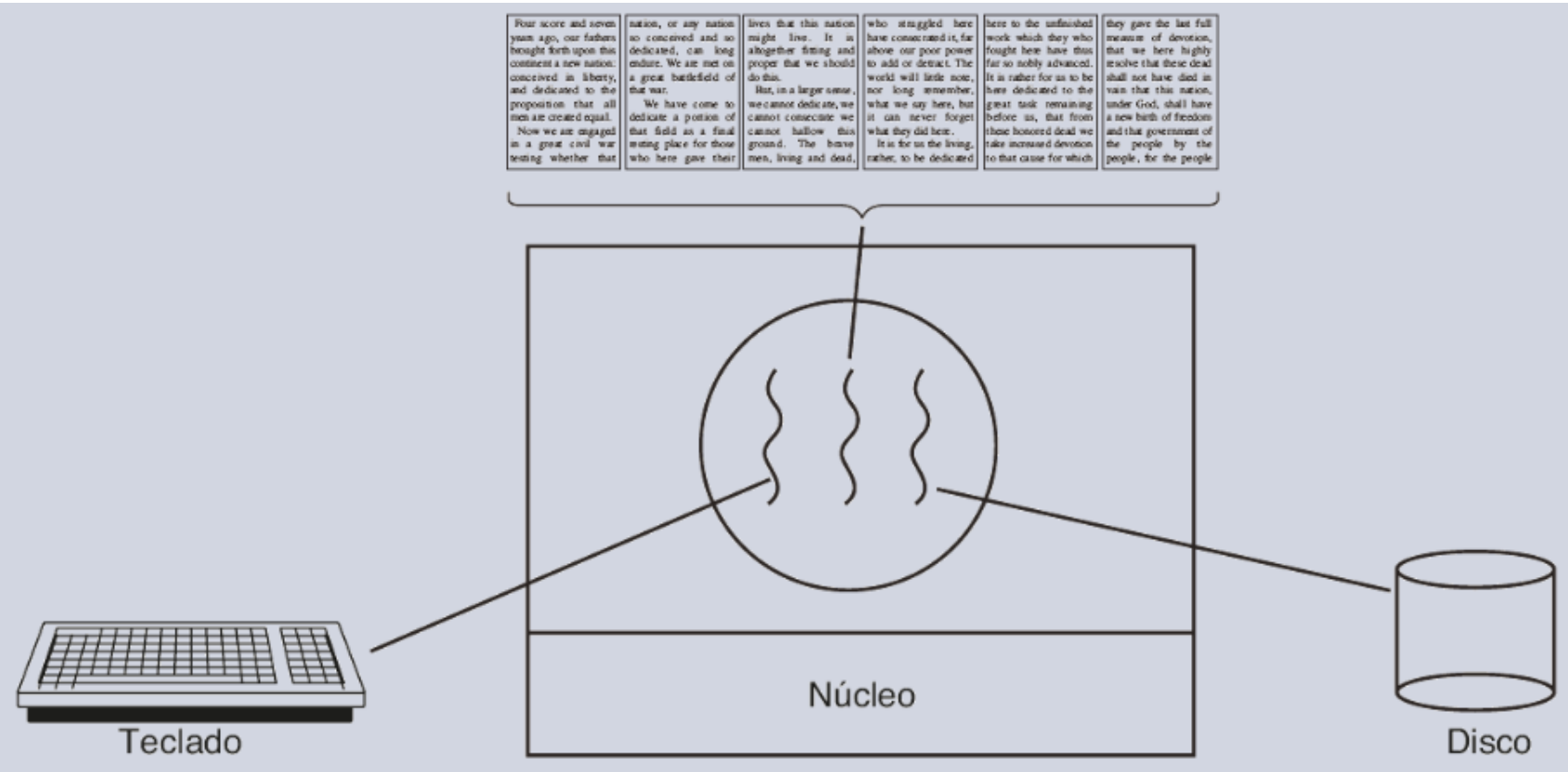
Threads

```
void iniciaThreadsGVNS(FuncaoObjetivo funcaoObjetivo, bool debug, Solucao solucaoInicial, Parametro parametro, float porcentagem, int epocas,int
qtdCamadas){

        vector<thread> threads;

        // for(int i = 0; i < thread::hardware_concurrency(); i++){
        for(int i = 0; i < 10; i++){
                threads.push_back(thread(iniciaGVNS, funcaoObjetivo,debug,solucaoInicial,parametro, porcentagem, epocas, i,qtdCamadas));
        }

        for(int i = 0; i < threads.size(); i++){
                threads[i].join();
        }
}
```

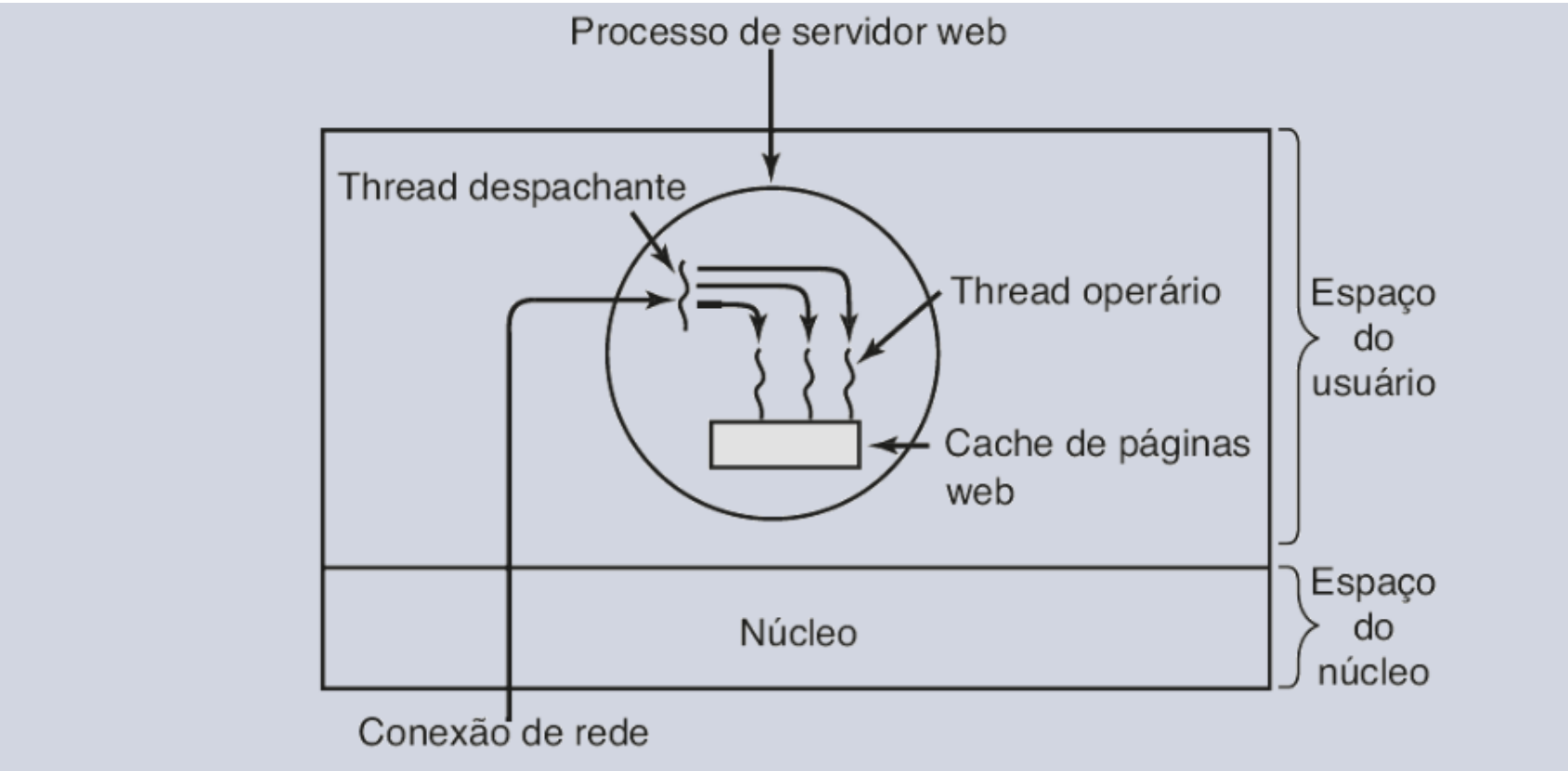
Modelos para construir um servidor

Modelo	Características
Threads	Paralelismo, chamadas de sistema bloqueantes
Processo monothread	Não paralelismo, chamadas de sistema bloqueantes
Máquina de estados finitos	Paralelismo, chamadas não bloqueantes, interrupções

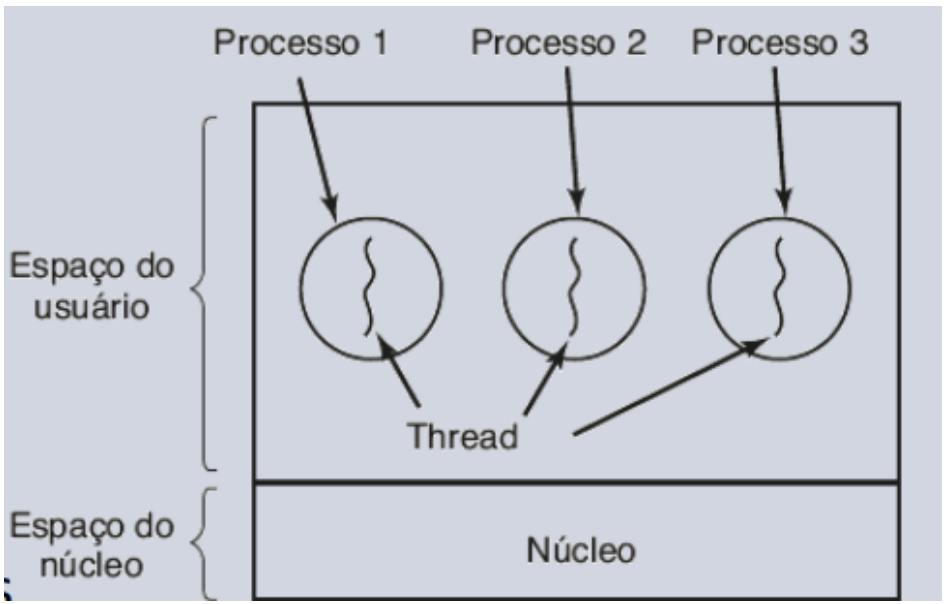
Um processador de texto com três threads

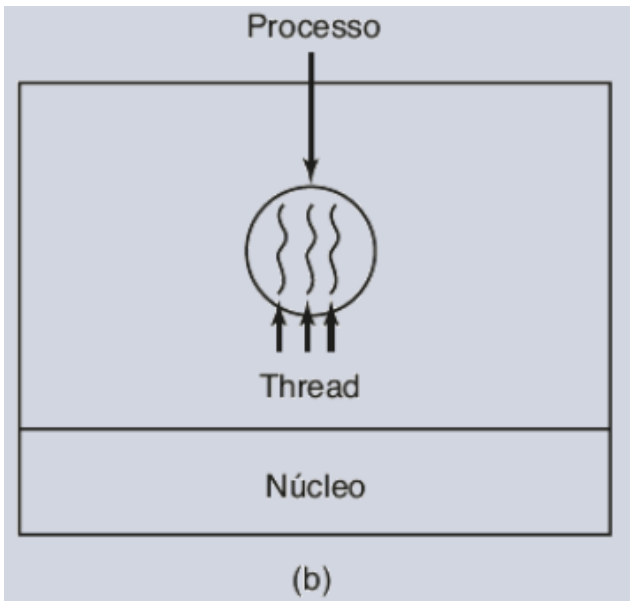


Um servidor web multithread

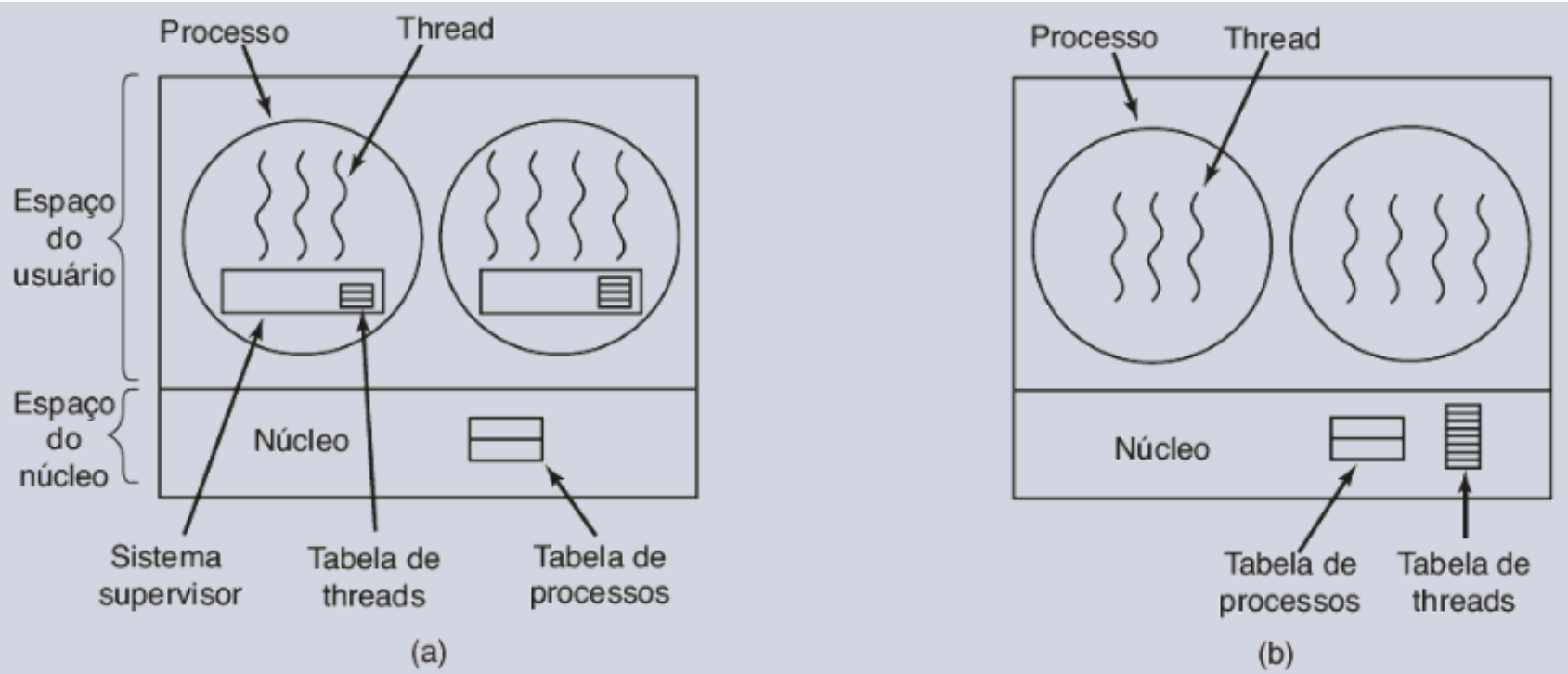


O modelo de thread clássico

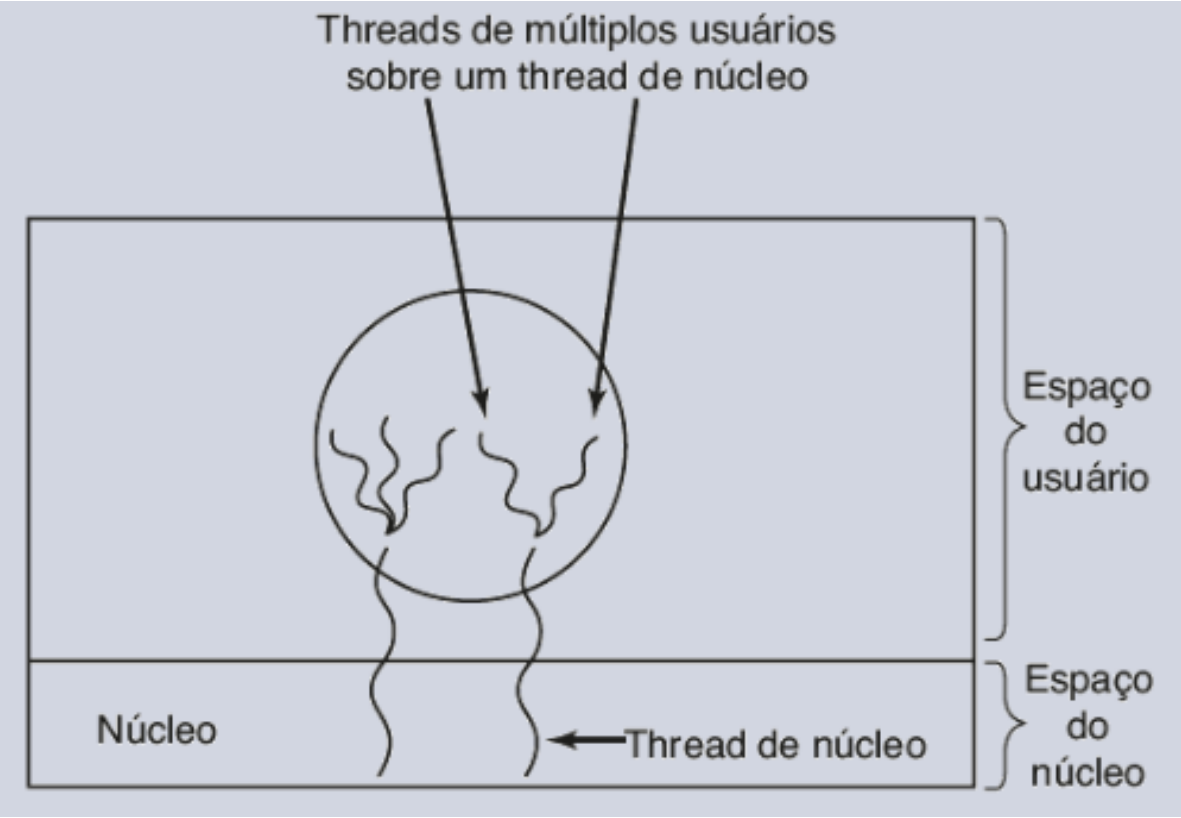




Implementando threads no espaço e no núcleo

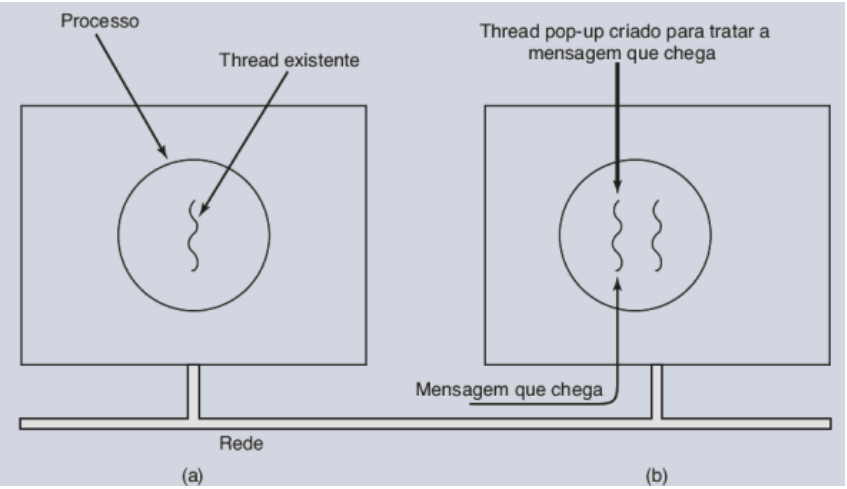


Implementações híbridas



Threads pop-up

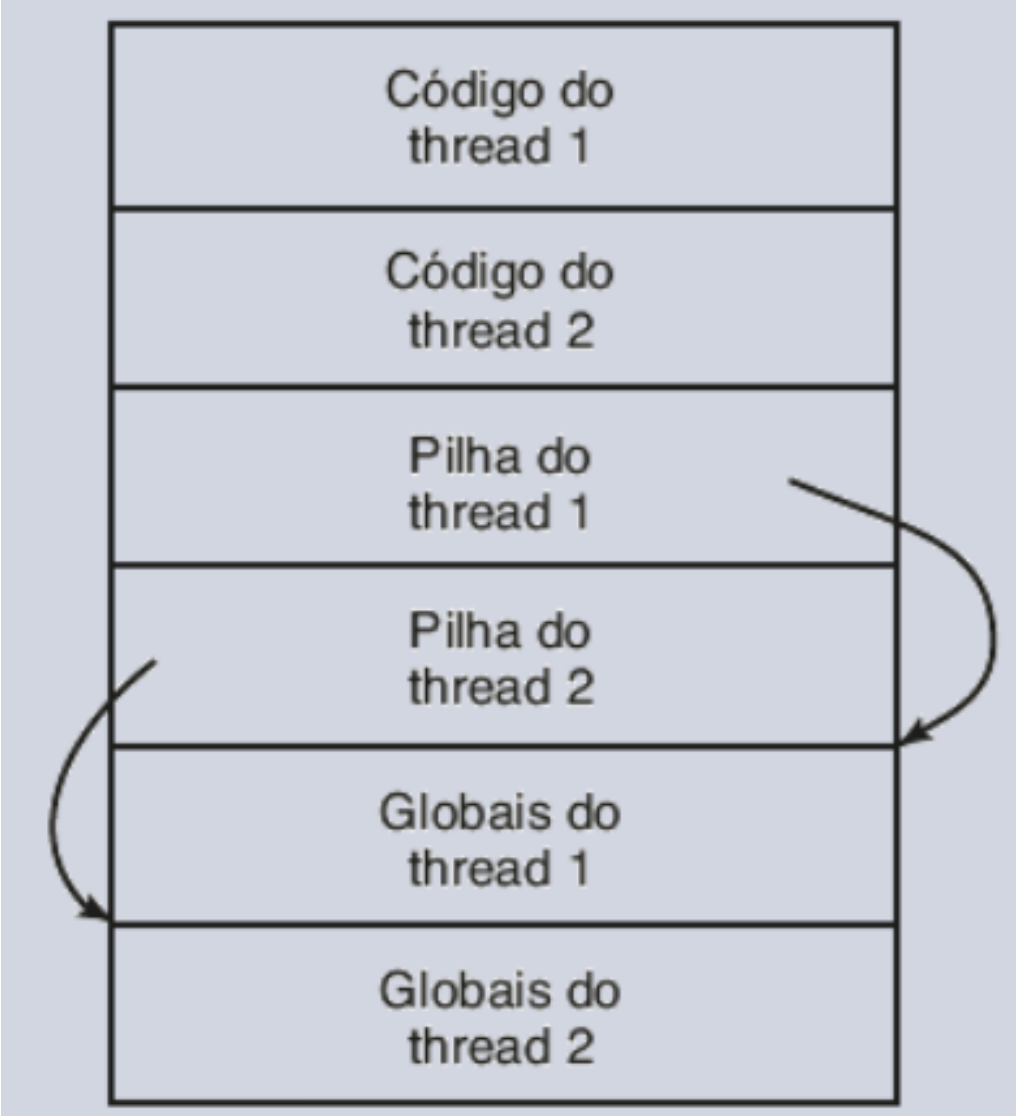
- Threads costumam ser úteis em sistemas distribuídos.
- A abordagem tradicional é ter um processo ou thread que esteja bloqueado em uma chamada de sistema receive esperando pela mensagem que chega.
- Quando uma mensagem chega, ela é aceita, aberta, seu conteúdo examinado e processada.
- Com a chegada de uma mensagem o sistema cria uma nova thread para lidar com a mensagem. Essa thread é chamado de thread pop-up.



Convertendo código de um thread em código multithread

- Muitos programas existentes foram escritos para processos monothread. Convertê-los para multithreading é muito mais complicado do que pode parecer em um primeiro momento.
- O código de um thread em geral consiste em múltiplas rotinas, exatamente como um processo.
- Essas rotinas podem ter variáveis locais, variáveis globais e parâmetros.
- Variáveis locais e de parâmetros não causam problema algum, mas variáveis que são globais para um thread, mas não globais para o programa inteiro, são um problema.

Threads podem ter variáveis globais individuais

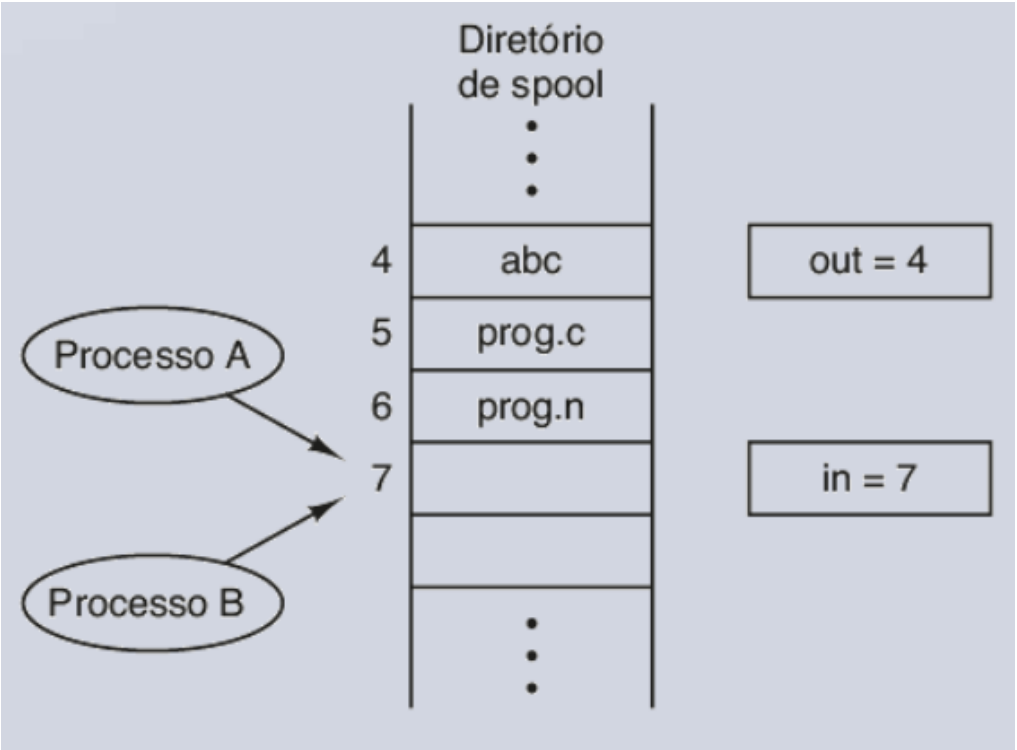


Comunicação entre processos

- Processos podem comunicar-se uns com os outros usando primitivas de comunicação entre processos.
- Por exemplo: semáforos, monitores ou mensagens.
- Essas primitivas são usadas para assegurar que jamais dois processos estejam em suas regiões críticas ao mesmo tempo.

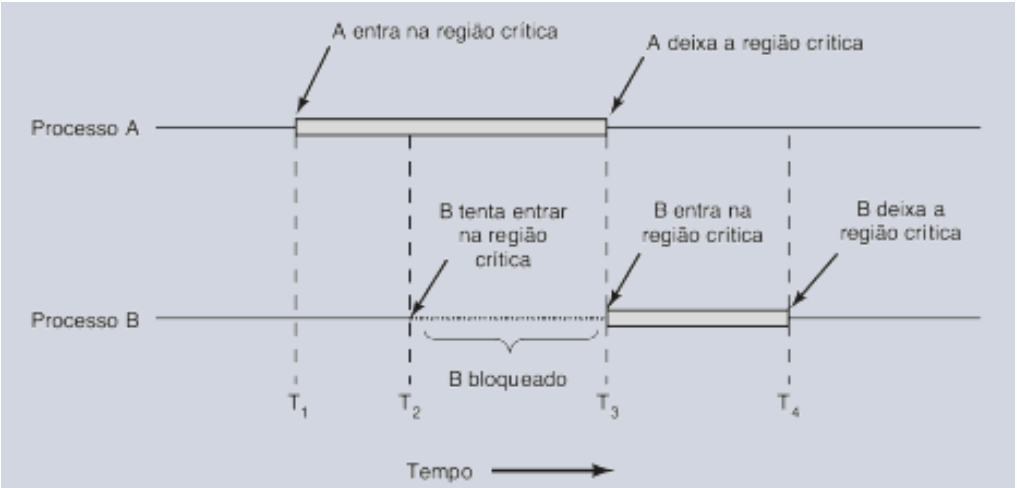
Condições de corrida

- Em alguns sistemas operacionais, processos que estão trabalhando juntos podem compartilhar de alguma memória comum que cada um pode ler e escrever.



Regiões críticas

- **Como evitar as condições de corrida?**
- A chave para evitar problemas é encontrar alguma maneira de proibir mais de um processo de ler e escrever os dados compartilhados ao mesmo tempo.
- O que precisamos é de **exclusão mútua**, isto é, alguma maneira de se certificar de que se um processo está usando um arquivo ou variável compartilhados, os outros serão impedidos de realizar a mesma coisa.



Escalonamento

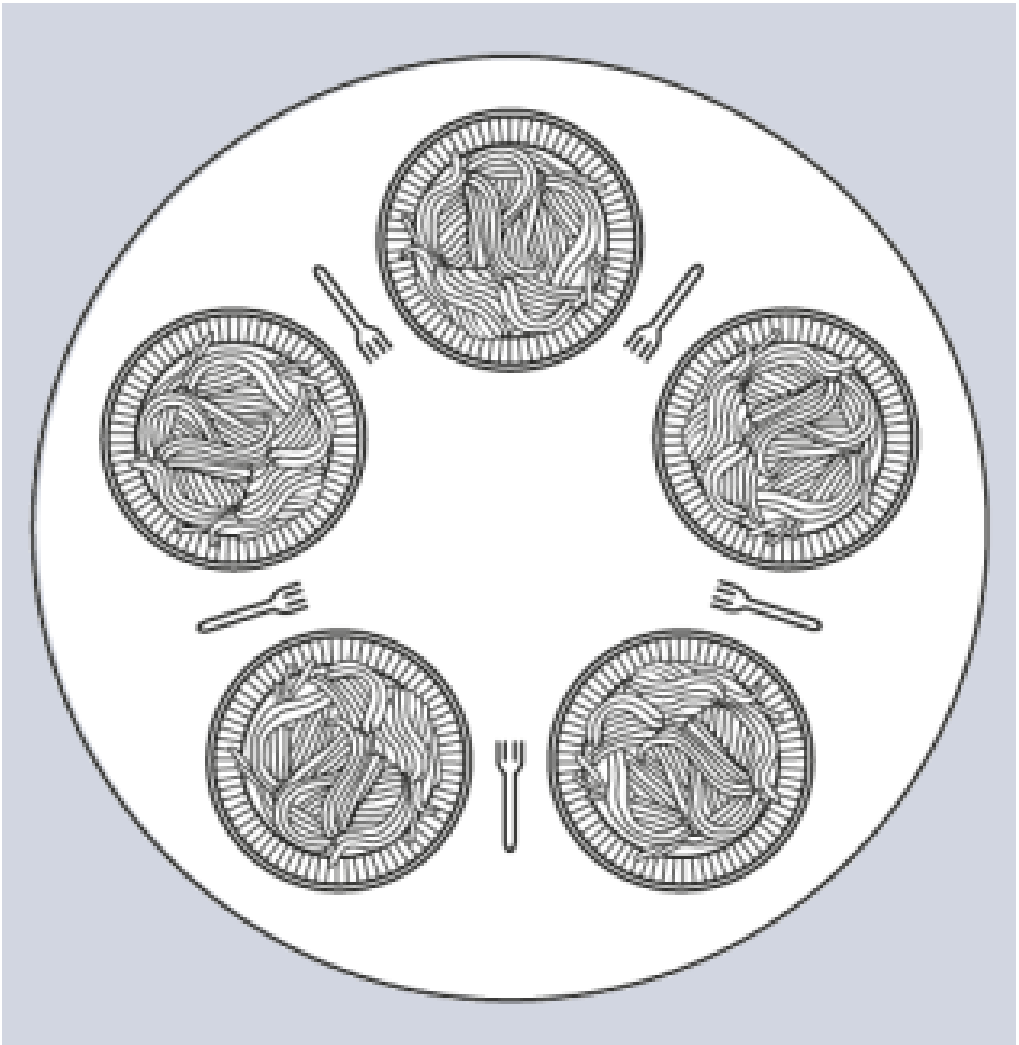
- Quando um computador é multiprogramado, ele frequentemente tem múltiplos processos ou threads competindo pela CPU ao mesmo tempo.
- Essa situação ocorre sempre que dois ou mais deles estão simultaneamente no estado pronto.
- Se apenas uma CPU está disponível, uma escolha precisa ser feita sobre qual processo será executado em seguida.
- A parte do sistema operacional que faz a escolha é chamada de escalonador, e o algoritmo que ele usa é chamado de algoritmo de escalonamento.

Problemas clássicos de IPC

- **O problema do jantar dos filósofos**
- Cinco filósofos estão sentados em torno de uma mesa circular. Cada filósofo tem um prato de espaguete. O espaguete é tão escorregadio que um filósofo precisa de dois garfos para comê-lo. Entre cada par de pratos há um garfo.
- Quando um filósofo fica suficientemente faminto, ele tenta pegar seus garfos à esquerda e à direita, um de cada vez, não importa a ordem. Se for bem-sucedido em pegar dois garfos, ele come por um tempo, então larga os garfos e continua a pensar.

Problemas clássicos de IPC

- A questão fundamental é: você consegue escrever um programa para cada filósofo que faça o que deve fazer e jamais fique travado?



Problemas clássicos de IPC

- **O problema dos leitores e escritores**
- Imagine, por exemplo, um sistema de reservas de uma companhia aérea, com muitos processos competindo entre si desejando ler e escrever.
- É aceitável ter múltiplos processos lendo o banco de dados ao mesmo tempo, mas se um processo está atualizando o banco de dados, nenhum outro pode ter acesso, nem mesmo os leitores.

Pesquisas sobre processos e threads

- Processos, threads e escalonamento, não são mais os tópicos quentes de pesquisa que já foram um dia. A pesquisa seguiu para tópicos como **gerenciamento de energia, virtualização, nuvens e segurança**.

Pesquisas sobre processos e threads

- O escalonamento de dispositivos móveis em busca da eficiência energética (YUAN e NAHRSTEDT, 2006), escalonamento com tecnologia hyperthreading (BULPIN e PRATT, 2005) e escalonamento bias-aware (KOUFATY, 2010).
- Com cada vez mais computação em smartphones com restrições de energia, alguns pesquisadores propõem migrar o processo para um servidor mais potente na nuvem, quando isso for útil (GORDON et al, 2012).
- No entanto, poucos projetistas de sistemas andam preocupados com a falta de um algoritmo de escalonamento de threads decente, então esse tipo de pesquisa parece ser mais um interesse de pesquisadores do que uma demanda de projetistas.

Obrigado!
Dúvidas?

Guilherme Henrique de Souza Nakahata

guilhermenakahata@gmail.com

https://github.com/GuilhermeNakahata/UNESPAR-2023