Sistemas Operacionais

4º período

Professora: Michelle Hanne

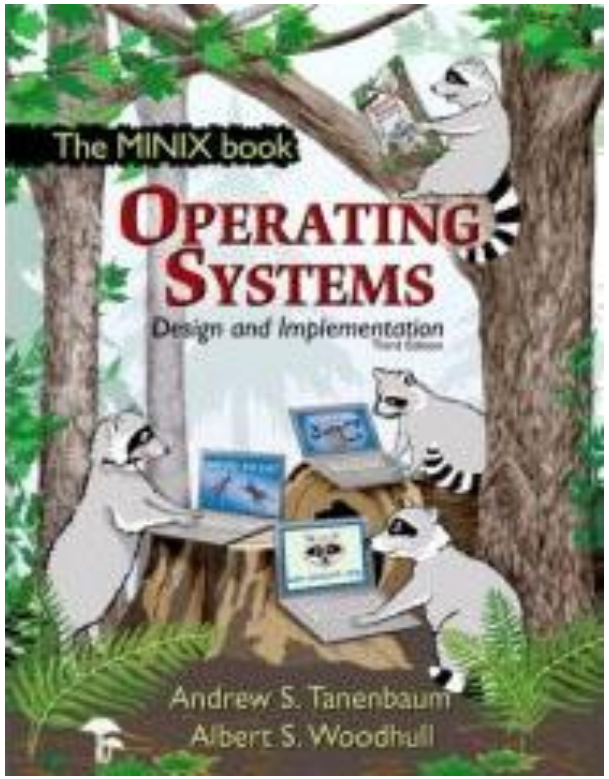
PUC Minas

Processos

SISTEMA OPERACIONAL MINIX3

- É usuário e defensor do MINIX. Ele possui um servidor web, que executa MINIX, e fornece suporte para usuários desse sistema operacional. Sua página pessoal na web está localizada lá. Você pode encontrá-la no URL http://minix1.hampshire.edu/asw/.

- O MINIX 3 é um sistema operacional gratuito e de código aberto projetado para ser altamente confiável, flexível e seguro. Ele é baseado em um minúsculo microkernel rodando no modo kernel com o resto do sistema operacional rodando como uma série de processos isolados e protegidos no modo usuário. Ele roda em CPUs x86 e ARM. https://www.minix3.org/



PUC Minas

Recapitulando

Os sistemas operacionais têm uma função muito importante em **sistemas interativos**. Esses sistemas criam diversos processos que disputam concorrentemente um tempo de execução pelo processador. **A gerência do procedimento** que decide **qual o processo e quanto ele será processado pela unidade central de processamento (CPU),** esta é uma das responsabilidades dos sistemas operacionais.

PUC Minas

Processos - Conceitos

Um sistema operacional possui o objetivo de controlar ou gerenciar tanto o hardware quanto o software. Esse recurso pode ser um dado armazenado na memória ou controlar um fluxo de um dispositivo de entrada e saída.

Um sistema multiprogramável simula um ambiente monoprogramável para cada usuário. Esse usuário tem a percepção do uso exclusivo do recurso.

Nesses sistemas, várias tarefas são executadas quase que ao mesmo tempo na CPU e diversos algoritmos determinam quando e como serão executadas.

PUC Minas

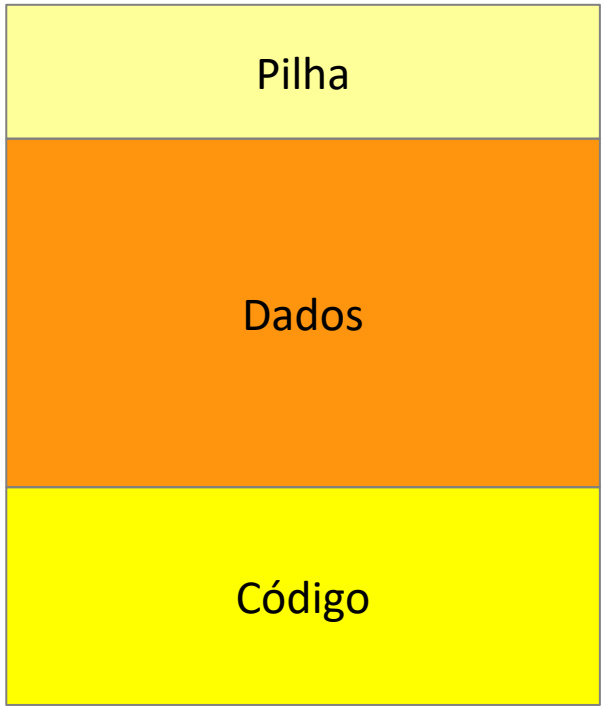
Processos - Conceitos

Um **processo** é **simplesmente um programa em execução**, incluindo os valores correntes do contador de programa, dos registradores e das variáveis. Conceitualmente, cada processo tem sua própria CPU virtual (Tanembaum, 2016).

PUC Minas

Processos - Conceitos

Estrutura básica de um processo na memória principal



PUC Minas

Bloco de Controle de Processo

- Bloco de controle do processo (PCB)
- Armazena informacoes associadas a um processo
- Estado do processo
- Registradores
- Contador de programa (PC)
- Acumuladores
- Ponteiro da pilha
- Base e limite de memoria
- Prioridades
- “Contabilidade”

Processos - Conceitos

- **Um processo precisa estar na memoria principal para ser executado.**

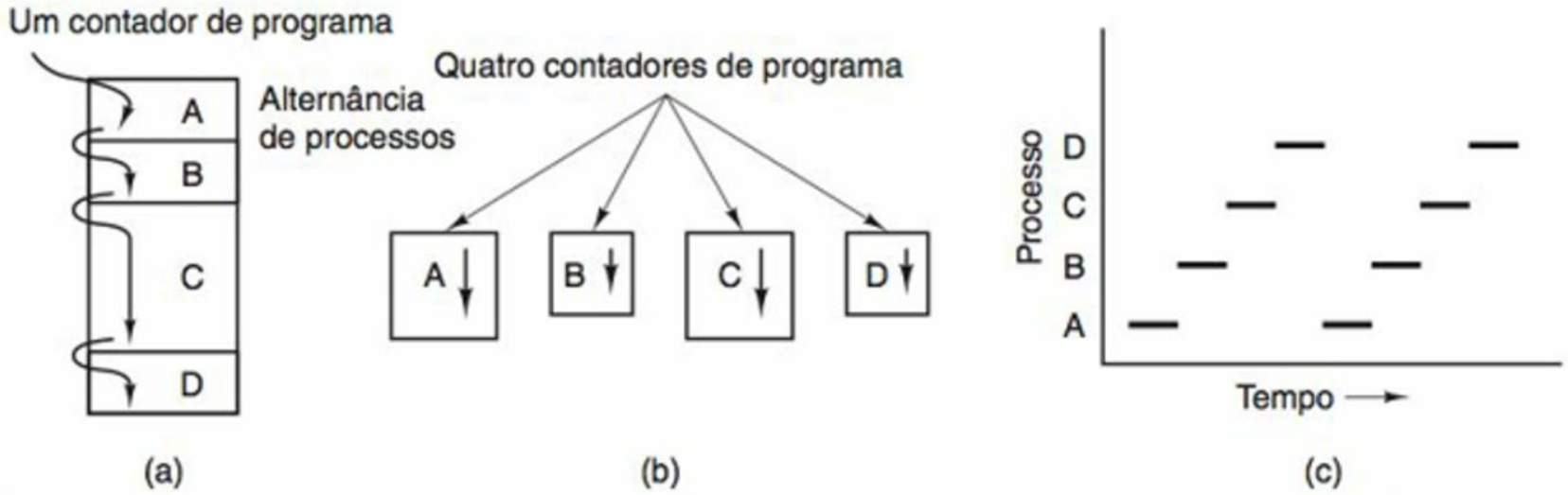


Figura 2-1 (a) Multiprogramação de quatro programas. (b) Modelo conceitual de quatro processos seqüenciais independentes. (c) Apenas um programa está ativo em dado instante.

2-1(a), temos quatro programas em memória.
2-1(b), vemos quatro processos, cada um com seu próprio fluxo de controle (isto é, seu próprio contador de programa lógico) e cada um executando independentemente dos outros.
2-1(c), vemos que, observados por um intervalo de tempo suficientemente longo, todos os processos fizeram progresso, mas em um dado instante apenas um está sendo executado.

É claro que existe apenas um contador de programa físico, de modo que, quando cada processo é executado, seu contador de programa lógico é carregado no contador de programa físico. Quando ele termina, o contador de programa físico é salvo no contador de programa lógico do processo, em memória.

Fonte: Tanembaum (2016).

PUC Minas

Modelo de Processos

Processo é a estrutura que mantém as informações necessárias a respeito de uma tarefa como por exemplo endereço de alocação na memória. A estrutura de dados que armazena as informações necessárias para tratar um processo é chamada de **Bloco de Controle de Processo** (***Process Block Control***).

Essa estrutura fica no núcleo do sistema operacional, e a partir do PCB, informações a respeito do processo como identificação, prioridade, estado corrente, recursos alocados e informações sobre o programa em execução são mantidas pelo Sistema Operacional –SO (MACHADO; MAIA, 2013).

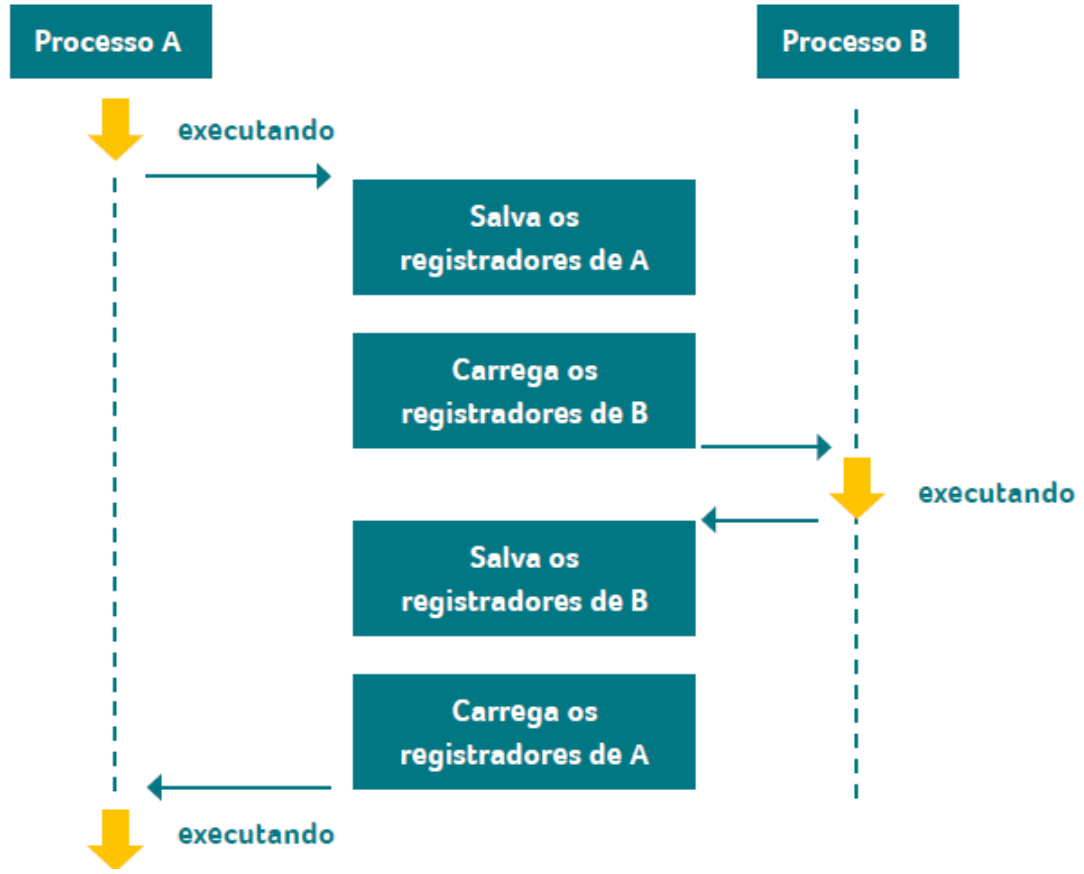
Bloco de Controle de Processo

Ponteiros
Estado do processo
Nome do processo
....
....
Limite de memória
Limite de arquivos abertos

PUC Minas

Modelo de Processos - Hardware

Contexto de Hardware formado por: **Program Counter (PC), Stack Pointer (SP).** Um registrador é uma memória RAM (Random Access Memory) que fica no processador e armazena os dados de um programa em execução. Quando um programa está em execução, as informações em memória do processo são deslocadas para os registradores, e ao término do processamento, são deslocadas novamente para memória.



PUC Minas

Modelo de Processos - Software

No contexto de software o processo mantêm as características e limites de recursos que podem ser alocados.

- Cada processo ao ser criado recebe um número de identificação do usuário ou processo que o criou, data da criação e hora da criação
- Nome do processo, proprietário, data da criação, etc.
- Quantidade de subprocessos, tamanho de alocação na memória, número máximo de arquivos abertos, número máximo de operações de entrada e saída (E/S) .
- **Esse contexto é dividido em três partes: identificação, quotas e privilégio.**

PUC Minas

Modelo de Processos – Linux e Windows

No Linux, existem dois tipos de prioridade: estática e dinâmica. A prioridade estática é definida pelo usuário e utilizada para sistemas em tempo real, e a dinâmica para os demais sistemas

Já no Windows, sistema gráfico e interativo, todo processo ganha prioridade normal e o escalonador determina a ordem de execução na CPU pelo nível de prioridade. São 32 níveis de prioridade, sendo que 16 - 31 são os níveis de prioridade mais alta

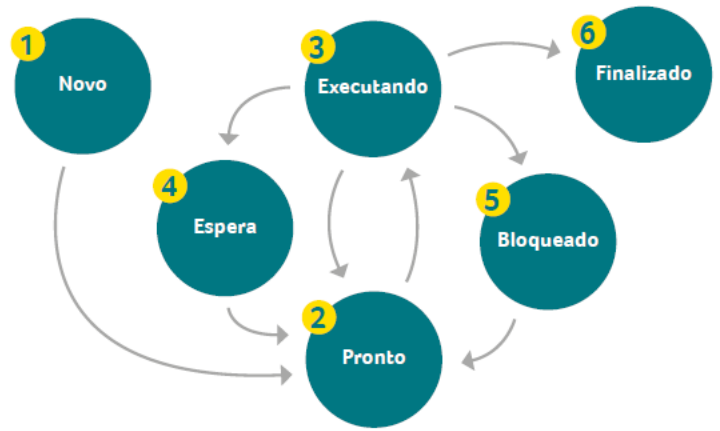
- A prioridade é determinada pela classe de prioridade dentro de cada processo (Tempo Real, Alta, Acima do Normal, Normal, Abaixo do Normal, Baixo) podendo ser alteradas pelo usuário.

PUC Minas

Estados de um Processo

Num sistema multitarefa, um processo possui vários estados pois nem sempre todo o processo está sendo processado na CPU (SILBERSCHATZ, 2015).

- **Running (execução)** – quando o processo está sendo executado na CPU.
- **Ready (pronto)** – quando o processo está em memória esperando para ser executado pela CPU.
- **WAIT (espera)** – quando o processo está aguardando por um evento externo ou recurso para ser executado na CPU, por exemplo, um evento de entrada e saída.
- **BLOCKED (bloqueado)** – quando um processo é interrompido por outro processo na CPU e aguarda por um recurso do sistema que ainda não está disponível, por exemplo, um dispositivo de entrada e saída.
- **FINISHED (finalizado)** – quando um processo é finalizado.
- **New (novo)** – quando o arquivo é criado.



Estados de um Processo

Estado	Evento
1 – Novo	O momento que o processo é criado no sistema operacional.
2- Pronto	O processo está armazenado em memória RAM, foi atribuído um determinado recurso e o seu para ser executado na CPU.
3 – Executando	Nesse momento o processo está sendo executado no processador num determinado tempo ou totalmente.
4 – Espera	O processo está armazenado em memória secundária, disco, e espera por um evento externo ou algum recurso para voltar a ser processado. Por exemplo, uma determinada data e/ou hora para poder voltar a ser processado ou por um término de uma operação de entrada/saída.
5 – Finalizado	O processo é finalizado quando sua tarefa for concluída.

PUC Minas

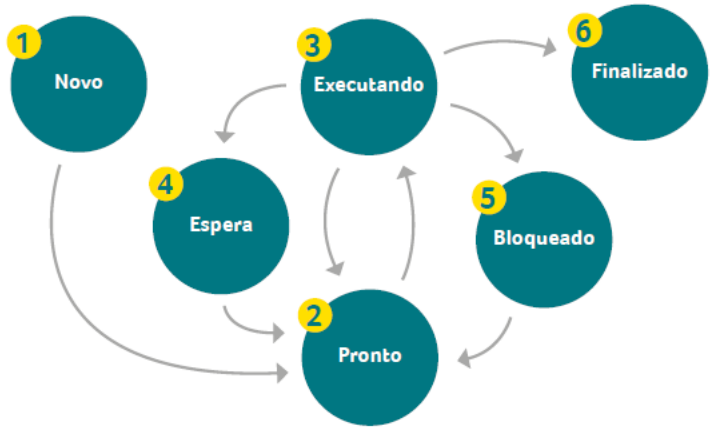
Estados de um Processo

Devido a uma técnica chamada ***swapping (troca)***, processos em estado de **pronto ou espera/bloqueado** podem momentaneamente estar em **memória secundária, e não em memória principal.**

Quando o processo está sendo executado no **estado** de **executando**, pode sofrer uma interferência externa e passar para o estado de espera.

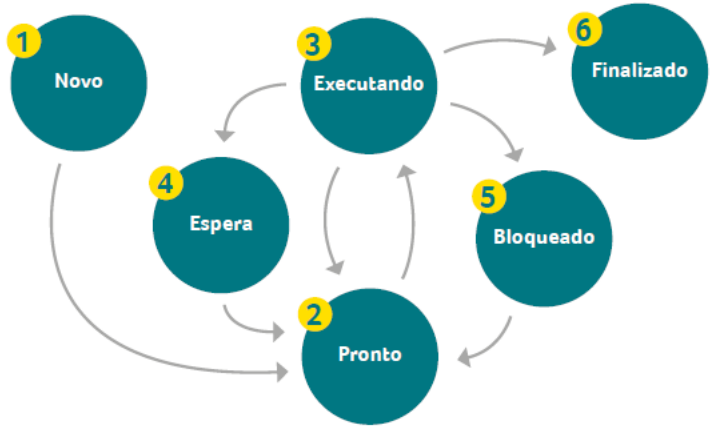
Quando o processo for interrompido por **tempo** ou por **prioridade** e ainda precisar de mais tempo na CPU para terminar sua execução, ele passa do estado de executando para **o estado de pronto**, e outro processo na **fila de pronto poderá** ser executado na CPU. Após seu tempo, ele **aloca uma nova fatia de tempo e retorna o processamento na CPU.**

Quando o processo em estado de execução termina sua tarefa, ele é finalizado e, assim, pode ser removido da memória.



Estados de um Processo

- Em um sistema **multitarefa**, um processo muda de estado várias vezes devido a eventos provocados pelo próprio processo ou pelo SO. São mudanças de estado permitidas:
- Pronto → Execução;
- Execução → Espera:
- Espera → Pronto;
- Execução → Pronto;
- Execução → Bloqueado;
- Bloqueado → Pronto.



PUC Minas

Multiprogramação

Multiprogramação

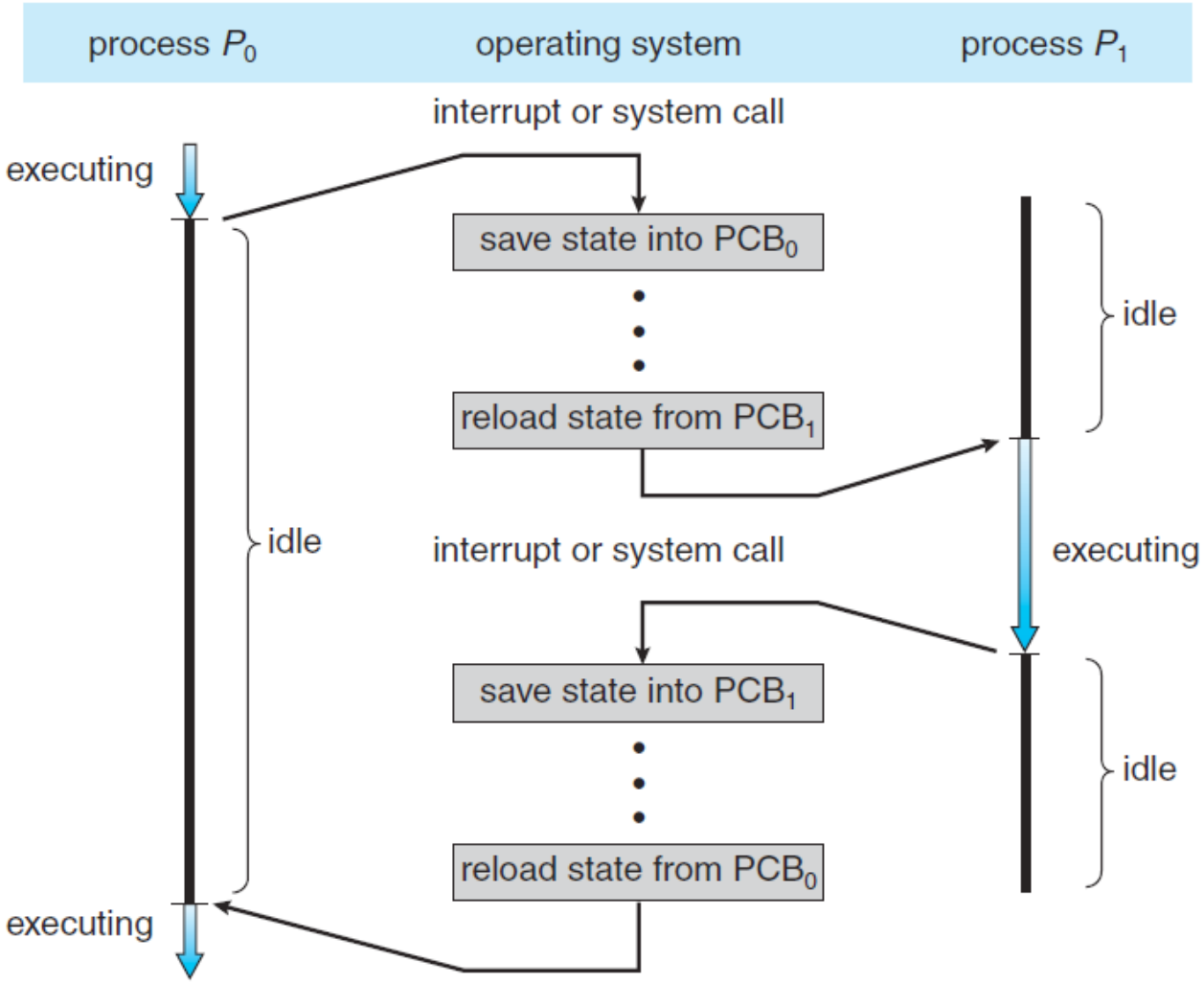
- Controle simultâneo de diversas tarefas (jobs)
- Alternância por bloqueio
- Alternância por prioridade
- Retomada de execução

A CPU também alterna de um programa para outro, executando cada um deles por dezenas ou centenas de milissegundos. Rigorosamente falando, a qualquer momento, enquanto a CPU está executando apenas um programa, durante 1 segundo ela pode trabalhar em vários programas, dando aos usuários a ilusão de paralelismo. Ou melhor, Pseudoparalelismo - para contrastar com o verdadeiro paralelismo de hardware dos sistemas.

- Paralelismo em multiprocessadores – paralelismo de hardware - que têm duas ou mais CPUs compartilhando a mesma memória física

Troca de Contexto

> Multiprogramação, alternância e troca de contextos



SILBERSCHATZ et al(2015)

Threads

Threads - Introdução

Um processo (processo-pai) pode criar outros processos (processo-filho). Esse processo-filho, agora processo pai, pode criar outros processo-filho e assim sucessivamente. Esse procedimento de criar subprocessos pelos processos existentes é denominado hierarquia de processos (SILBERSCHATZ, 2015).

Esse procedimento de criação de vários processos pode gerar um super processamento de processos na CPU (*overhead*), pois os processos são estruturas independentes e possuem seus próprios contextos.

Threads

PUC Minas

Thread: unidade basica de utilizacao da CPU

Modelo de processo tradicional:

- fluxo de controle unico → monothread

Processo multithread:

- Divisao das tarefas em fluxos independentes
- Pode realizar mais de uma tarefa concorrentemente, ou simultaneamente

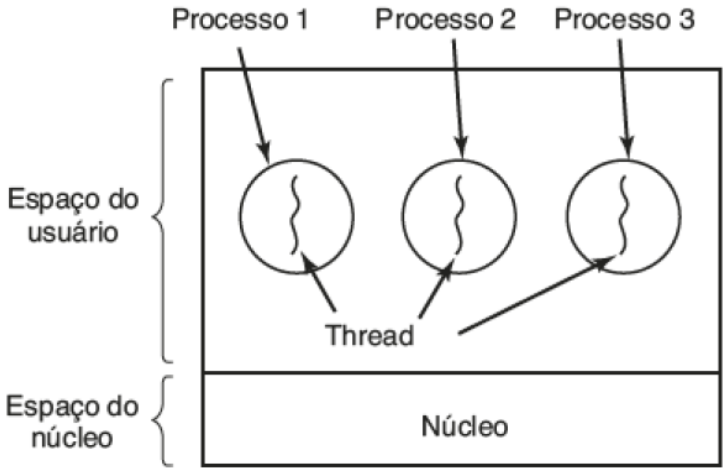
Threads são linhas de comando (programação) existentes dentro de um processo que podem executar tarefas concorrentes.

Threads

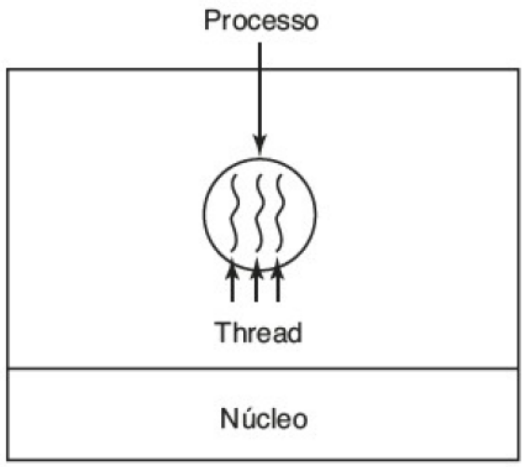
PUC Minas

- ***Threads* são divididas em dois níveis: usuário e *kernel*.**
- O **sistema operacional fornece o suporte à nível de *kernel***, e a nível de **usuário são implementadas por meio de bibliotecas de determinada linguagem de programação.**
- Como características *as* ***threads*** possuem um contexto mais simplificado, mais fáceis de criar/destruir e melhoram o desempenho em sistemas que realizam muitos IOs (Entradas e Saídas).

Threads



(a)



(b)

Adaptado de TANENBAUM; BOS (2016)

Um sistema que possui um **único processo com uma única thread, é chamada monothread**, mas o sistema que opera com vários processos, ou seja, cada processo com diversas **threads é chamado multithread**.

Por que multithreads?

- Responsividade
- Compartilhamento e economia de recursos
- Aproveitamento de arquiteturas multiprocessadas
- Ganho de tempo

Threads - Exemplos

PUC Minas

Um jogo é um excelente exemplo de sistema multithread, pois permite que uma thread fique responsável pelo gráfico e outra pelo áudio.

Threads - Exemplos

PUC Minas

As *threads* são utilizadas por **programadores para a divisão de trabalho.** Imagine um computador com dois núcleos de processamento; nesse caso, **uma thread poderia ser processada para exibição de vídeo num núcleo, enquanto a outra trataria o som no outro núcleo.**

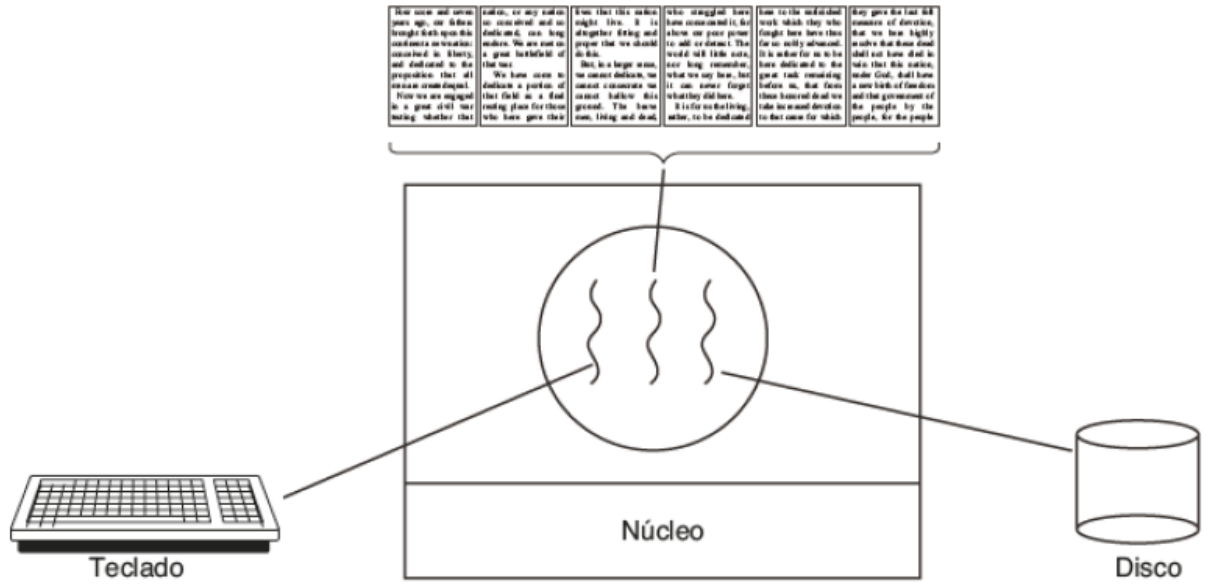
Caso não houvesse divisão de trabalho, **tudo seria realizado num único núcleo**. As *threads* consomem menos memória, **compartilham as informações do processo ao qual que estão associadas e possuem um tempo de troca de contexto mais curto.**

Sistemas operacionais Android fazem uso de *thread* para envio de *emails*, por exemplo.

Threads - Exemplos

Exemplo 1: Editor de textos

- Usuário digita o texto
- Verificação ortográfica / correção automática
- Formatação de páginas não visíveis
- Cópias de segurança

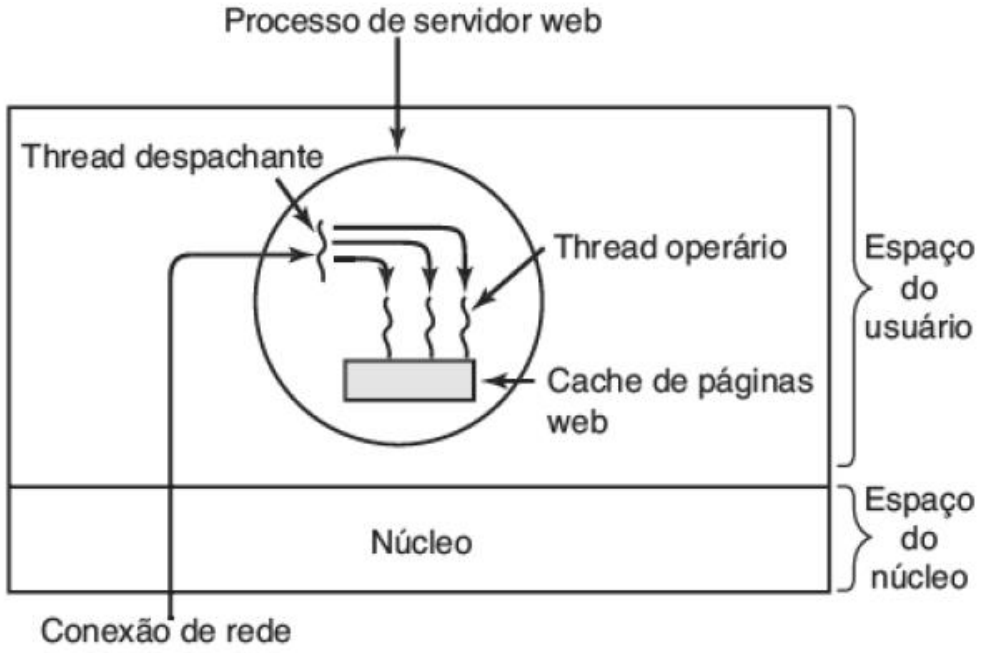


Fonte: TANENBAUM;BOS (2016)

Threads - Exemplos

Exemplo 2: Servidor web

- Despachante
- Trabalhadores



Fonte: TANENBAUM;BOS (2016)

Modelos de *Threads* no SO

- ***Threads:*** processos leves (lightweight process – LWP)
  - Ciclo de vida equivalente aos processos
  - Possuem contador de programa, registradores e pilhas próprios
  - Compartilham código e dados
  - Mais leves para criar e destruir
- ***SO deve decidir como gerenciar threads***
  - Em espaço do usuário (muitos para um)
  - Em espaço do núcleo (um para um)
  - De maneira híbrida (muitos para muitos)

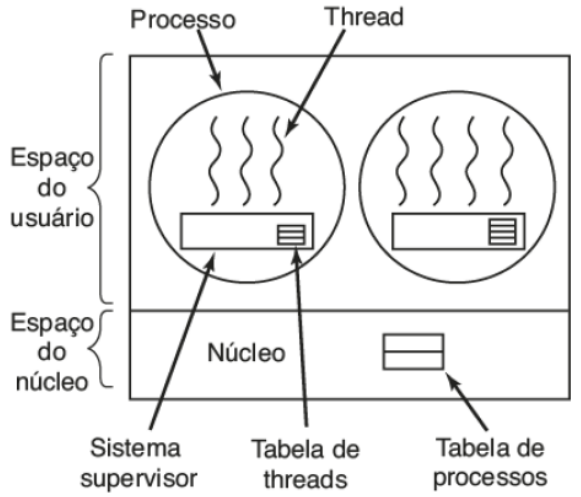
Implementação e gerência de Threads

Espaço de usuário – Muitos para um

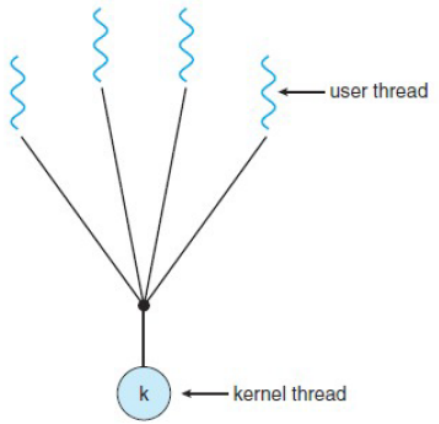
- Threads são implementadas pelo processo e gerenciadas por ele
- Bibliotecas de programação
- Kernel continua gerenciando processos

- Gerencia simplificada (e, portanto, eficiente)
- Não necessita alternar de processo para trocar de thread
- Bloqueio da thread → bloqueio do processo
- Kernel não “enxerga” a thread
- Não ha paralelismo no processo

Kernel continua gerenciando processos



Fonte: TANENBAUM;BOS (2016)



Fonte: SILBERSCHATZ et al(2015)

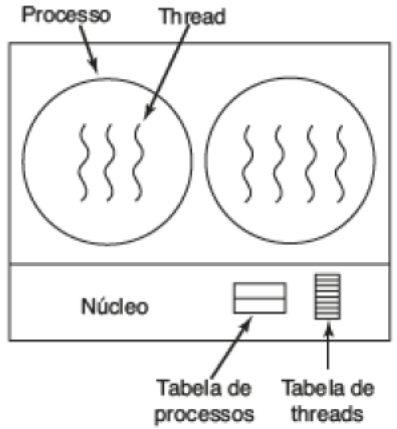
Implementação e gerência de Threads

Espaço do núcleo – Um para um

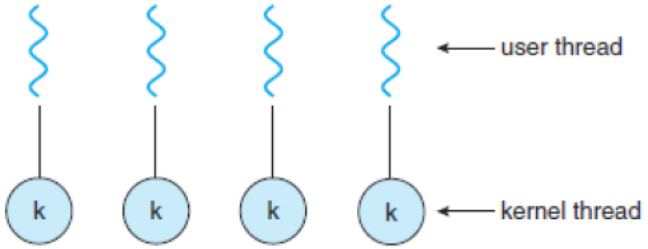
- Espaço de kernel (modo supervisor)
- Cada thread do processo é associada a uma thread de kernel
- Maior concorrência
- Paralelismo real em multiprocessadores

Cada thread do processo é associada a uma thread de kernel

- **Custo adicional de criação e gerência**
  - Atraso na criação de uma thread
  - Tabela de threads x tabela de processos
- **Gestão de um numero ilimitado(?) de threads**
  - Threads ilimitadas derrubando o sistema
- **Paralelismo real: suposta melhora de desempenho**



Fonte: TANENBAUM;BOS (2016)



Fonte: SILBERSCHATZ et al(2015)

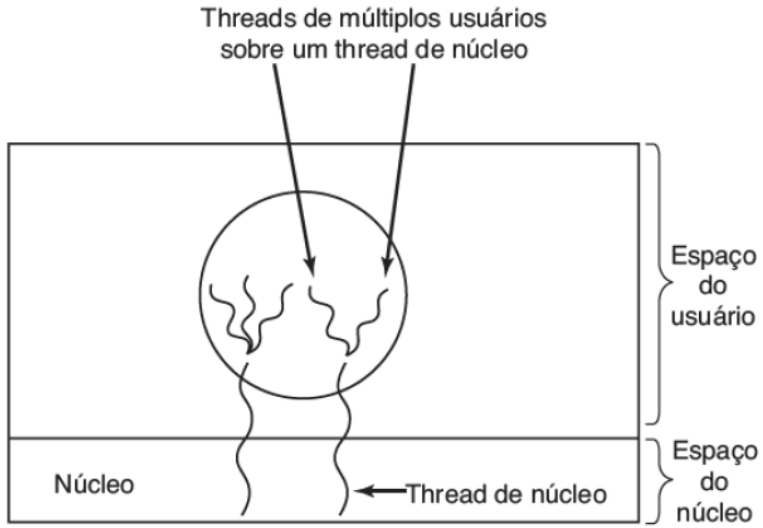
Implementação e gerência de Threads

Híbrido– Muitos para muitos

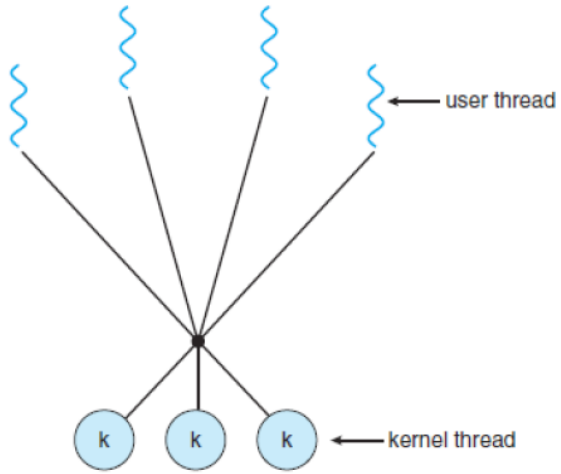
- Modelo híbrido:
- Threads de processo ilimitadas
- Alocadas para um número menor ou igual de threads no núcleo

- **Tenta resolver o problema de gestão com uma thread pool**
- **Thread pool:**
  - Criar varias threads na inicialização e aloca-las em um banco
  - Tarefas novas: “acordam” threads do banco
  - Se há falta de threads: espera
- **Agilidade**
- **Flexibilidade**
- **Controle de recursos**

› Multiplexação de threads de usuário



Fonte: TANENBAUM;BOS (2016)



Fonte: SILBERSCHATZ et al(2015)

Referências

TANENBAUM, Andrew S.; BOS, Herbert. Sistemas operacionais modernos. 4. ed. São Paulo: Pearson Education do Brasil, c2016.

MACHADO, Francis Berenger; MAIA, Luiz Paulo. Arquitetura de sistemas operacionais. 5. ed. Rio de Janeiro: LTC, 2013. 263 p. ISBN: 9788521622109.

SILBERSCHATZ, Abraham; GALVIN, Peter B.; GAGNE, Greg. Fundamentos de sistemas operacionais. 9. ed. Rio de Janeiro, RJ: LTC, c2015.

OLIVEIRA, Rômulo Silva de; CARISSIMI, Alexandre da Silva; TOSCANI, Simão Sirineo. Sistemas operacionais. 4. ed. Porto Alegre: Bookman, 2010.