Gestão de Tarefas em Sistemas Operacionais

Uma introdução aos conceitos fundamentais de gerenciamento de tarefas, estados de processos e evolução histórica dos sistemas operacionais multitarefa.

Conteúdo Programático

01

Conceito de Tarefa

Definição, diferenças entre programa e tarefa, e a importância da gestão de tarefas

02

Evolução Histórica

Sistemas monotarefa, monitores de sistema e a transição para sistemas multitarefa

03

Ciclo de Vida das Tarefas

Estados possíveis de uma tarefa e transições entre estados

04

Sistemas de Tempo Compartilhado

Conceito de preempção, quantum de tempo e escalonamento

05

Implementação Prática

Visualização e monitoramento de tarefas em sistemas operacionais reais

Objetivos de Aprendizagem

Ao final desta apresentação, você será capaz de:

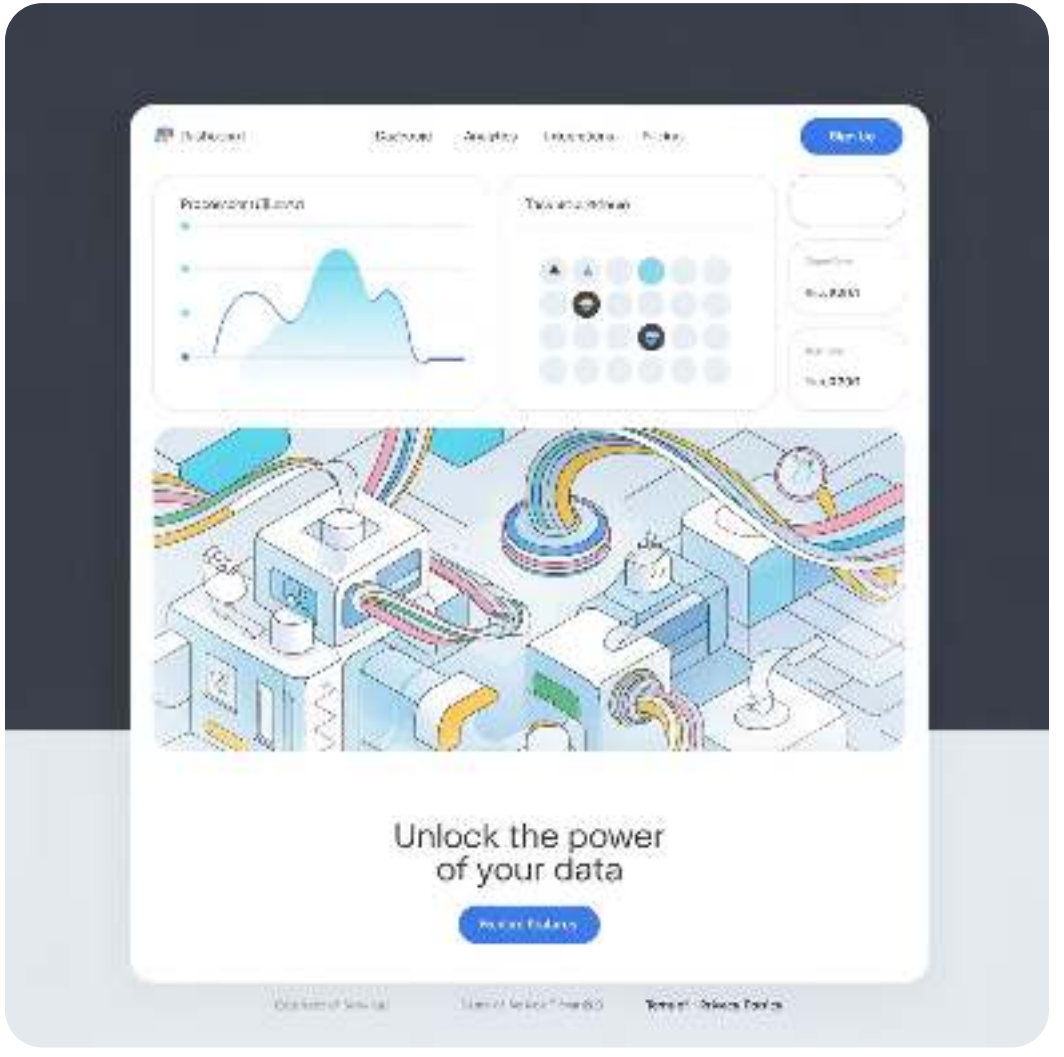
- Compreender o conceito de tarefa e sua importância em sistemas operacionais
- Diferenciar programas e tarefas em um sistema computacional
- Entender a evolução histórica dos sistemas de gestão de tarefas
- Identificar os estados possíveis de uma tarefa e suas transições
- Analisar o funcionamento de sistemas de tempo compartilhado

O Problema Fundamental

Em um sistema de computação moderno:

- Usuários desejam executar **várias tarefas simultaneamente**
- Um processador convencional trata apenas **um fluxo de instruções** por vez
- Mesmo computadores com múltiplos processadores, cores ou tecnologia hyper-threading têm **mais tarefas que unidades de processamento**

Como fazer para atender simultaneamente as múltiplas necessidades de processamento?



Exemplos de Cenários Multitarefa

Computador Pessoal

- Edição de imagem
- Impressão de relatório
- Reprodução de música
- Download de software

Servidor de E-mails

- Milhares de usuários conectados
- Envio e recebimento simultâneo de mensagens
- Filtragem de spam
- Backup automático

Navegador Web

- Busca de elementos da página
- Renderização de HTML e gráficos
- Animação de interface
- Resposta a comandos do usuário

Todos estes cenários exigem que o sistema operacional gerencie eficientemente o uso dos processadores disponíveis.

Soluções para o Compartilhamento de Processador

Solução ingênua:

Equipar o sistema com um processador para cada tarefa

❌ **Problemas:** Inviável econômica e tecnicamente

Solução viável:

Multiplexar o processador entre várias tarefas

✅ **Vantagens:** Compartilha recursos de forma eficiente e atende às necessidades de processamento de múltiplas tarefas

O Conceito de Tarefa

> Uma **tarefa** é definida como a execução de um fluxo sequencial de instruções, construído para atender uma finalidade específica: realizar um cálculo complexo, editar um gráfico, formatar um disco, etc.

A tarefa é o conceito fundamental na gestão de processamento em sistemas operacionais. É o que efetivamente consome os recursos do processador para realizar ações úteis para os usuários.

Diferença entre Programa e Tarefa

Programa

- Conjunto de instruções escritas para resolver um problema
- Conceito **estático**
- Sem estado interno definido
- Sem interações com outras entidades
- Exemplos: C:\Windows\notepad.exe, /usr/bin/vi

Tarefa

- Execução sequencial das instruções de um programa
- Conceito **dinâmico**
- Estado interno bem definido (variáveis, posição)
- Interage com usuário, periféricos e outras tarefas
- Implementadas como processos ou threads

Um programa pode originar várias tarefas simultâneas, cada uma com seu próprio estado.

Analogia: Programa vs. Tarefa

Programa = Receita de Torta

- Instruções escritas em um livro
- Lista de ingredientes (entradas)
- Modo de preparo (algoritmo)
- Guardada em uma estante (disco)
- Estática - não muda enquanto não for usada

Tarefa = Execução da Receita

- Ação de seguir os passos da receita
- Cozinheira (processador) seguindo instruções
- Ingredientes e utensílios em uso (estado)
- Evolui com o tempo
- Produz resultado (saída)

Tarefas de um Navegador Web

Um único programa (navegador) define várias tarefas que devem ser executadas simultaneamente:

- Buscar via rede os elementos da página Web
- Receber, analisar e renderizar o código HTML e gráficos
- Animar elementos da interface
- Receber e tratar eventos do usuário

Evolução Histórica da Gestão de Tarefas

1

Anos 40-50: Sistemas Monotarefa

Uma tarefa por vez, carregada do início ao fim. Operações manuais de carga e descarga de programas.

2

Anos 50-60: Monitores de Sistema

Programa permanente que gerencia automaticamente a carga e execução de programas em sequência.

3

Anos 60: Sistemas Multitarefa

Capacidade de suspender tarefas aguardando recursos e executar outras enquanto isso.

4

Anos 60-70: Sistemas de Tempo Compartilhado

Introdução do conceito de quantum de tempo e preempção, permitindo interatividade.

5

Anos 70-Presente: Sistemas Modernos

Combinação de multitarefa, tempo compartilhado e multiprocessamento.

Processo vs. Tarefa: Entendendo as Nuances

Embora os termos "processo" e "tarefa" sejam frequentemente usados de forma intercambiável na computação, eles possuem nuances importantes que os distinguem, especialmente no contexto dos sistemas operacionais.

Similaridades Chave

- Ambos representam a execução dinâmica de um programa de computador.
- Possuem um estado interno próprio (variáveis, ponteiro de instrução, etc.).
- Passam pelos mesmos estados do ciclo de vida (novo, pronto, em execução, bloqueado, terminado).
- São gerenciados diretamente pelo sistema operacional.

Diferenças Essenciais

- **Tarefa:** É um termo mais genérico e acadêmico, frequentemente usado em contextos teóricos ou em sistemas mais simples (como os monotarefa iniciais). Pode se referir a uma unidade de trabalho discreta.
- **Processo:** É um termo mais específico e prático, utilizado pelos sistemas operacionais modernos. Um processo inclui um espaço de memória isolado, um Identificador de Processo (PID) único e um conjunto de recursos (arquivos abertos, dispositivos, etc.) associados.

Na prática, quando você abre o Gerenciador de Tarefas no Windows ou executa o comando `ps` no Linux, você está visualizando **processos**. Cada processo é essencialmente uma "tarefa" em execução com seu próprio ambiente isolado e recursos dedicados, garantindo estabilidade e segurança entre diferentes aplicações.

Anatomia de um Processo/Tarefa

Um processo, ou tarefa, no contexto dos sistemas operacionais, é muito mais do que apenas um programa em execução. Ele é uma entidade complexa e gerenciada, que encapsula uma série de informações e recursos essenciais para sua operação isolada e eficiente. É como uma "cápsula" que o sistema operacional constrói e monitora.

PID (Process ID)

Identificador único atribuído pelo sistema operacional para cada processo em execução.

Estado Atual

Representa a fase em que o processo se encontra (novo, pronto, executando, suspenso, terminado).

Espaço de Memória

Área de memória dedicada que inclui código, dados, heap (memória dinâmica) e stack (pilha de execução).

Registradores do Processador

Valores dos registradores da CPU (ex: Program Counter - PC, Stack Pointer - SP) salvos durante troca de contexto.

Arquivos Abertos e Descritores

Lista de arquivos, sockets e outros recursos de I/O que o processo está utilizando.

Permissões e Credenciais

Informações de segurança que definem os direitos de acesso do processo a recursos do sistema.

Prioridade e Escalonamento

Dados que influenciam a ordem e a forma como o processo será escalonado para uso da CPU.

Recursos Alocados

Outros recursos do sistema (além de memória) alocados ao processo, como dispositivos ou semáforos.

Variáveis de Ambiente

Pares chave-valor que fornecem configurações específicas para o ambiente de execução do processo.

Diretório de Trabalho

O caminho do diretório atual a partir do qual o processo foi iniciado ou que ele está utilizando.

Relacionamentos

Referências ao processo pai e à lista de processos filhos, criando uma hierarquia de processos.

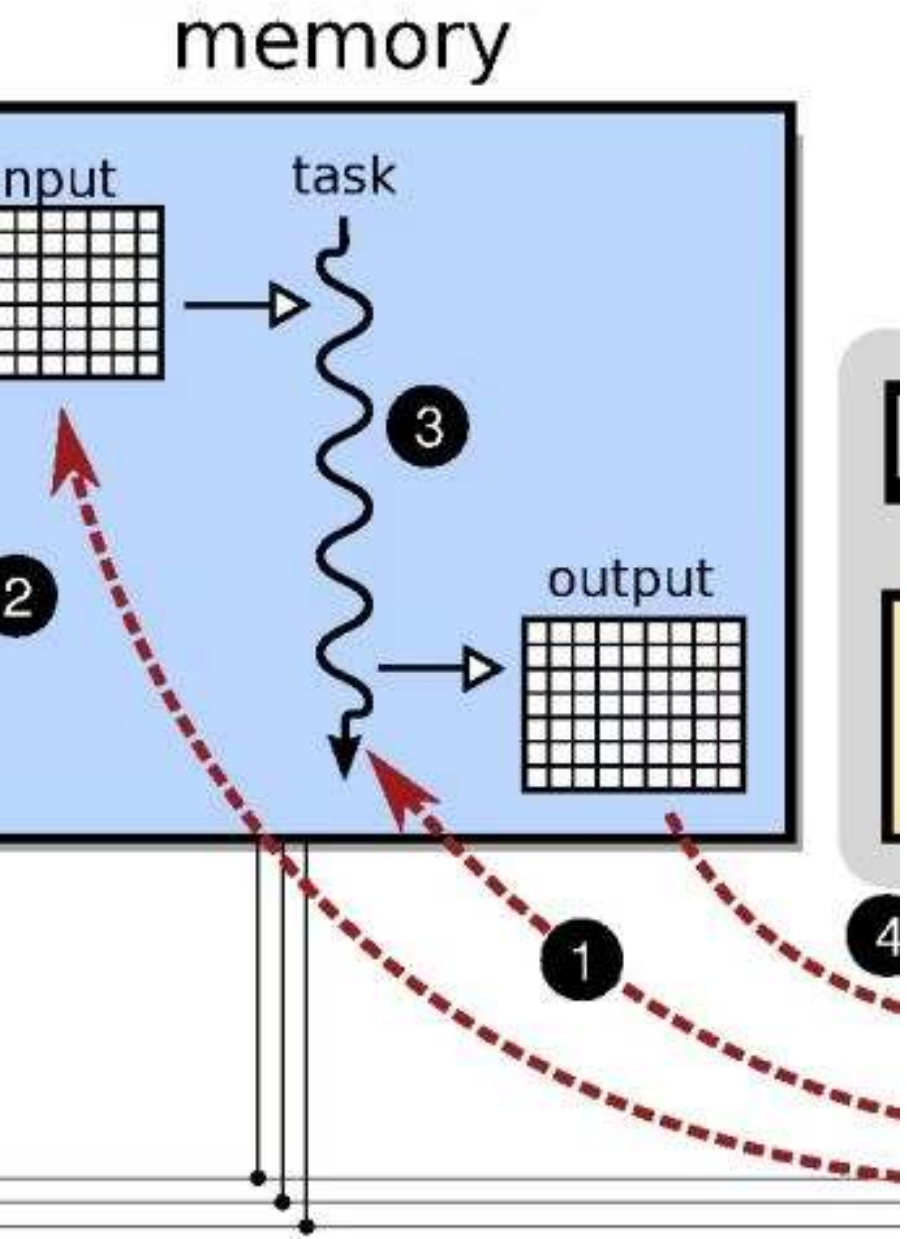
O sistema operacional gerencia todas essas propriedades, garantindo que cada processo opere de forma isolada e que seus recursos sejam alocados e liberados corretamente, mantendo a estabilidade e a segurança do sistema.

Sistemas Monotarefa

Nos primeiros sistemas de computação (anos 40):

- Apenas uma tarefa era executada por vez
- Cada programa era carregado, executado e descarregado
- Transferências entre disco e memória coordenadas por operador humano
- Uso principalmente para cálculos numéricos (militares)

Este modelo simples foi a base para a evolução dos sistemas operacionais modernos, mas apresentava sérias limitações de eficiência.



Execução em Sistemas Monotarefa

As etapas da execução de uma tarefa em um sistema monotarefa:

1. Carga do código do programa na memória
2. Carga dos dados de entrada na memória
3. Processamento (execução propriamente dita)
4. Descarga dos resultados no disco após a conclusão

Todas estas operações eram manuais, consumindo tempo e reduzindo a eficiência do sistema.

Estados da Tarefa em Sistemas Monotarefa

Neste modelo simplificado, uma tarefa passa apenas por três estados fundamentais:

Nova

A tarefa está sendo carregada na memória e preparada para execução

Executando

A tarefa está sendo processada pelo processador

Terminada

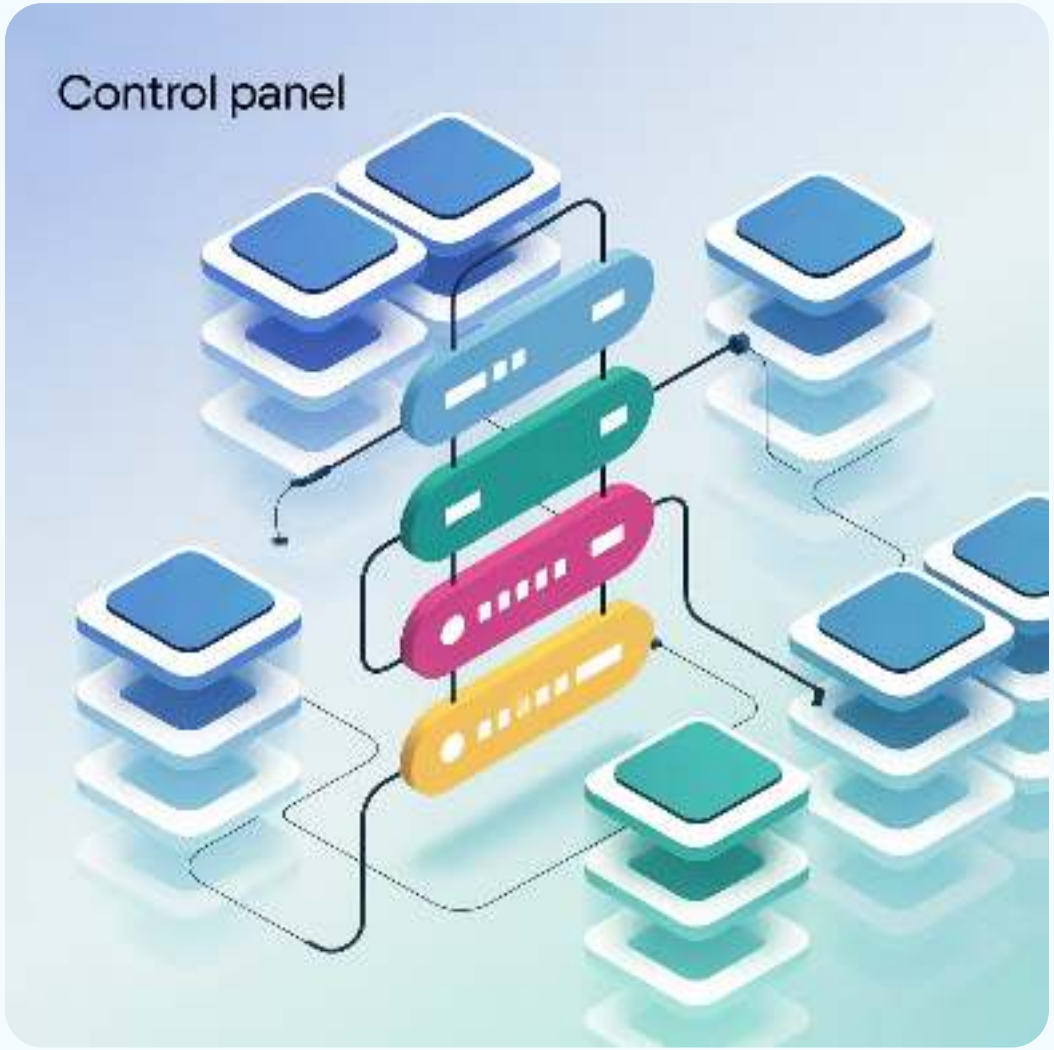
A tarefa concluiu sua execução e seus resultados estão sendo descarregados

O Monitor de Sistema

Para resolver o problema da ineficiência nas operações manuais, foi criado o **programa monitor**:

- Carregado na memória no início da operação
- Gerencia a execução dos demais programas
- Mantém uma fila de programas a executar
- Automatiza as operações de carga/descarga
- Oferece biblioteca de funções para acesso ao hardware

O monitor de sistema é o **precursor dos sistemas operacionais modernos**.



Funcionamento do Monitor de Sistema

Carregar um programa do disco para a memória

O monitor busca o próximo programa na fila e o carrega na memória principal

Carregar os dados de entrada do disco para a memória

Os dados necessários para a execução do programa são preparados

Transferir a execução para o programa recém carregado

O controle do processador é passado do monitor para o programa

Aguardar o término da execução do programa

O monitor fica inativo enquanto o programa executa completamente

Escrever os resultados gerados no disco

Após a conclusão, os resultados são salvos e o ciclo recomeça

Limitações do Monitor de Sistema

Problema Principal:

Mesmo com o monitor, o processador ficava ocioso durante as operações de entrada/saída.

> A velocidade de processamento era muito maior que a velocidade de comunicação com dispositivos de E/S, resultando em tempos de espera significativos.

Por exemplo: durante a leitura de uma fita magnética, o processador podia ficar parado por vários minutos!

Consequência: Custo elevado de equipamentos subutilizados

Solução: Sistemas Multitarefa

Para resolver o problema da ociosidade do processador, foi desenvolvido o conceito de **multitarefa**:

Suspensão de Tarefas

O monitor pode suspender uma tarefa que esteja aguardando dados externos

Alternância entre Tarefas

Enquanto uma tarefa está suspensa, outra pode utilizar o processador

Retomada Posterior

Quando os dados necessários estiverem disponíveis, a tarefa suspensa pode ser retomada

Esta solução exige mais memória (para manter várias tarefas simultaneamente) e mecanismos mais sofisticados no monitor.

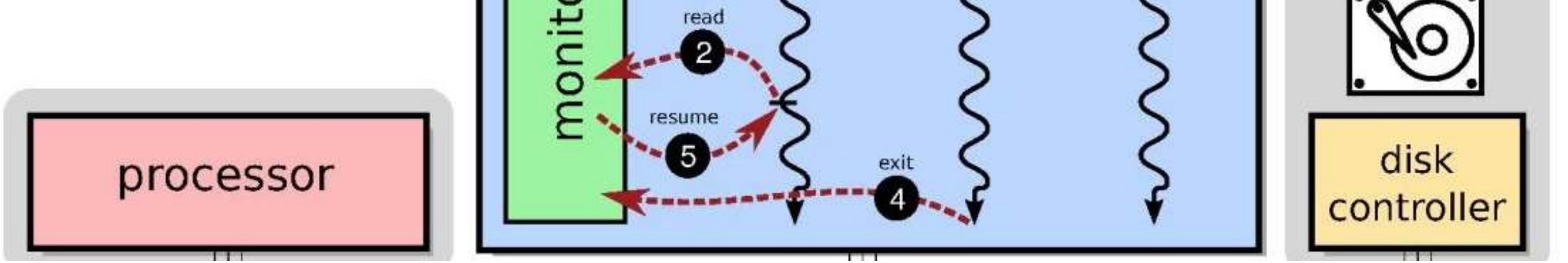
Implementação da Multitarefa

Uma forma simples de implementar a suspensão e retomada de tarefas:

Biblioteca Padronizada de E/S

- Monitor fornece rotinas padronizadas para operações de entrada/saída
- Tarefas usam estas rotinas em vez de acessar diretamente o hardware
- Rotinas podem suspender a execução quando necessário
- Controle retorna ao monitor, que pode executar outra tarefa

Este sistema permite **compartilhar o processador de forma transparente** para as aplicações.



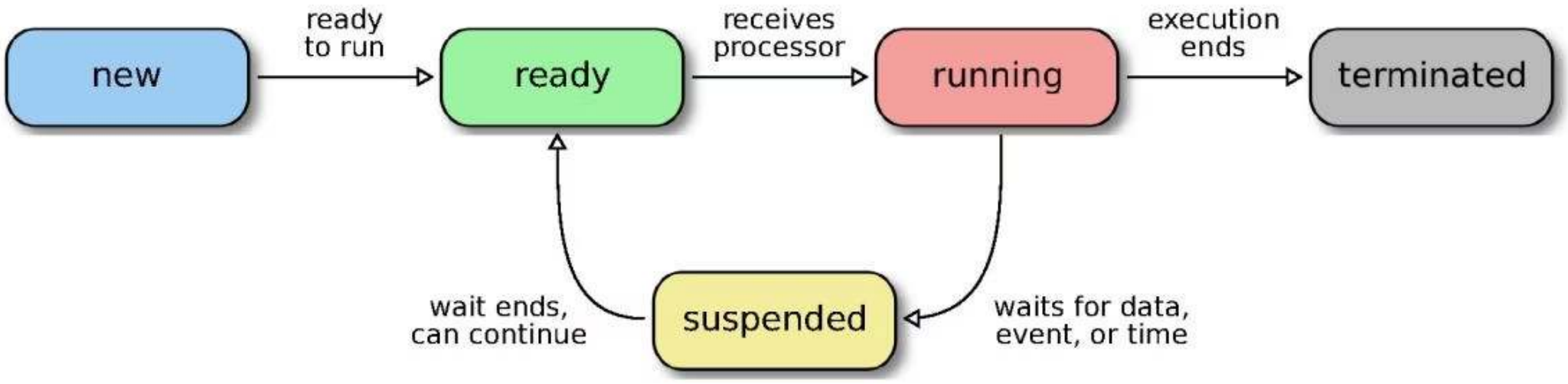
Funcionamento de um Sistema Multitarefa

Os passos numerados representam:

- **start:** início da execução de uma tarefa
- **read:** suspensão da tarefa ao solicitar leitura de dados
- **resume:** retomada da tarefa após os dados estarem disponíveis
- **exit:** conclusão da tarefa

Na figura, a tarefa A é suspensa ao solicitar uma leitura de dados, sendo retomada mais tarde, após a execução da tarefa B.

Estados da Tarefa em Sistemas Multitarefa



O diagrama mostra a evolução do modelo de estados:

- Nova
- Pronta
- Executando
- Suspensa
- Terminada

Agora a tarefa pode alternar entre os estados "Executando" e "Suspensa", permitindo que o processador seja usado por outras tarefas enquanto aguarda recursos.

Limitação dos Sistemas Multitarefa

Problema: Tarefas que não solicitam E/S

O código ao lado contém um **laço infinito** que nunca terminará nem solicitará operações de E/S.

Consequências:

- A tarefa monopoliza o processador indefinidamente
- O controle nunca volta ao monitor
- Outras tarefas não conseguem executar

Além disso, o modelo não era adequado para aplicações **interativas**, que precisam de respostas rápidas aos comandos do usuário.

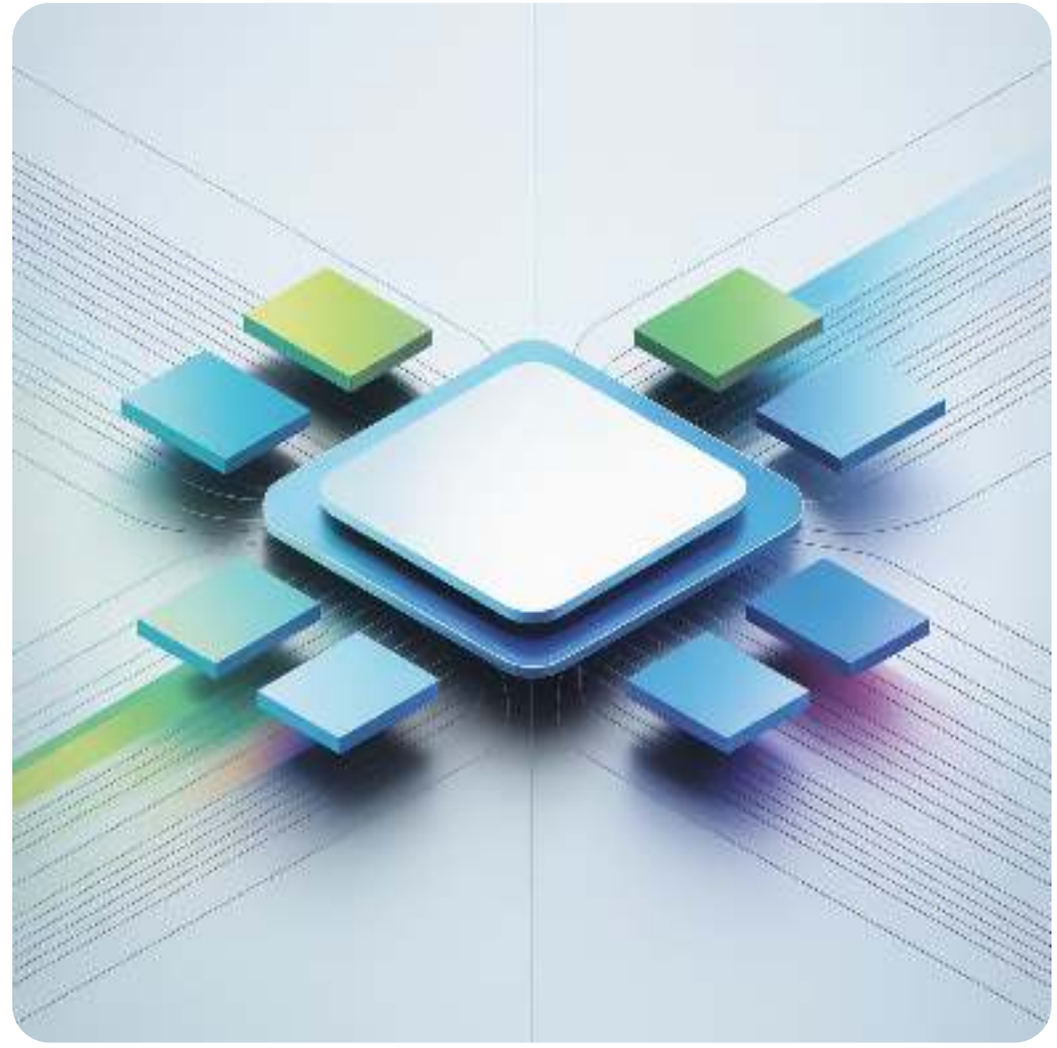
```
#include

int main() {
  int i = 0, soma = 0;

  while (i <= 1000)
    soma += i;
    // erro: contador i não incrementado

  printf("A soma vale %d\n", soma);
  exit(0);
}
```

Exemplo de código com laço infinito

Solução: Sistemas de Tempo Compartilhado

Para resolver estes problemas, foi introduzido o conceito de **compartilhamento de tempo** (time-sharing):

- Para cada tarefa que recebe o processador, é definido um prazo máximo de processamento: **fatia de tempo** ou **quantum**
- Esgotado seu quantum, a tarefa perde o processador mesmo que não tenha terminado
- A tarefa volta para uma fila de tarefas "prontas"
- Outra tarefa é ativada



Introduzido pelo sistema CTSS (Compatible Time-Sharing System) no início dos anos 60

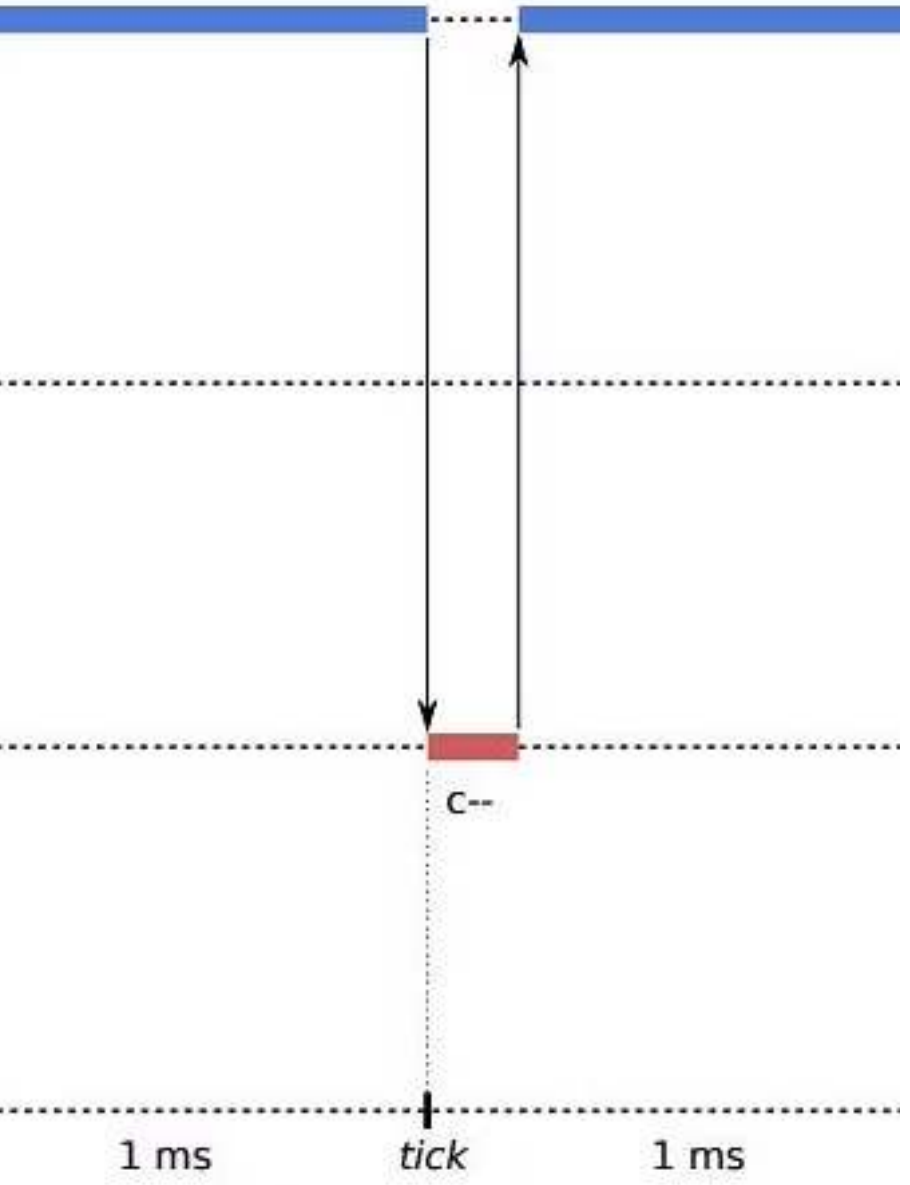
Preempção por Tempo

> O ato de retirar um recurso "à força" de uma tarefa (neste caso, o processador) é denominado **preempção**. Sistemas que implementam esse conceito são chamados **sistemas preemptivos**.

A preempção por tempo é implementada usando interrupções geradas por um temporizador programável do hardware:

- Temporizador gera interrupções em intervalos regulares (ticks)
- Ao receber o processador, a tarefa recebe um quantum (número de ticks)
- A cada tick, o contador é decrementado
- Quando chegar a zero, a tarefa perde o processador

No Linux, o quantum varia de 10 a 200 milissegundos, dependendo do tipo e prioridade da tarefa.

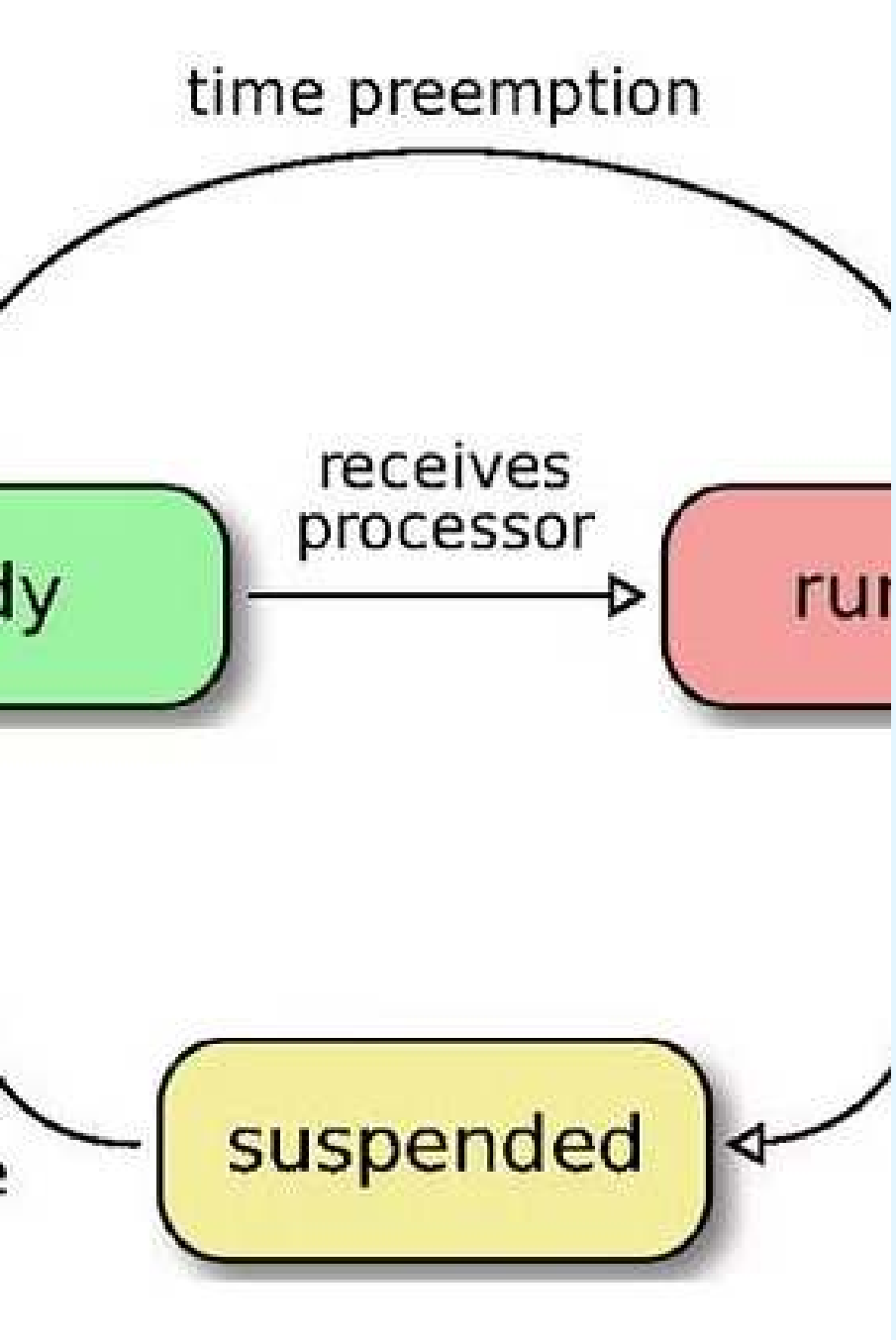


Dinâmica da Preempção por Tempo

A figura ilustra como o sistema operacional gerencia a preempção por tempo:

1. O temporizador gera interrupções periódicas (ticks)
2. A cada tick, o contador de quantum da tarefa em execução é decrementado
3. Quando o contador chega a zero, o escalonador seleciona outra tarefa
4. O processador é atribuído à nova tarefa selecionada

Este mecanismo garante que todas as tarefas tenham oportunidade de utilizar o processador, mesmo que algumas nunca solicitem operações de E/S.



Estados da Tarefa em Sistemas de Tempo Compartilhado

O diagrama agora inclui a preempção por tempo, que permite que uma tarefa em execução seja forçada a voltar ao estado "Pronta" quando seu quantum se esgota, mesmo sem solicitar E/S.

Este modelo é a base dos sistemas operacionais modernos, permitindo:

- Execução de múltiplas tarefas simultaneamente
- Distribuição justa do tempo de processador
- Interatividade com o usuário
- Prevenção de monopolização de recursos

Ciclo de Vida das Tarefas: Estados

Nova

A tarefa está sendo criada: código carregado em memória, bibliotecas necessárias anexadas e estruturas de dados do núcleo atualizadas.

Pronta

A tarefa está em memória, pronta para iniciar ou retomar sua execução, apenas aguardando a disponibilidade do processador.

Executando

O processador está dedicado à tarefa, executando suas instruções e fazendo avançar seu estado.

Suspensa

A tarefa não pode executar porque depende de dados externos, aguarda sincronização ou espera o tempo passar.

Terminada

O processamento da tarefa foi encerrado e ela pode ser removida da memória do sistema.

Ciclo de Vida das Tarefas: Transições (1/2)

... → Nova

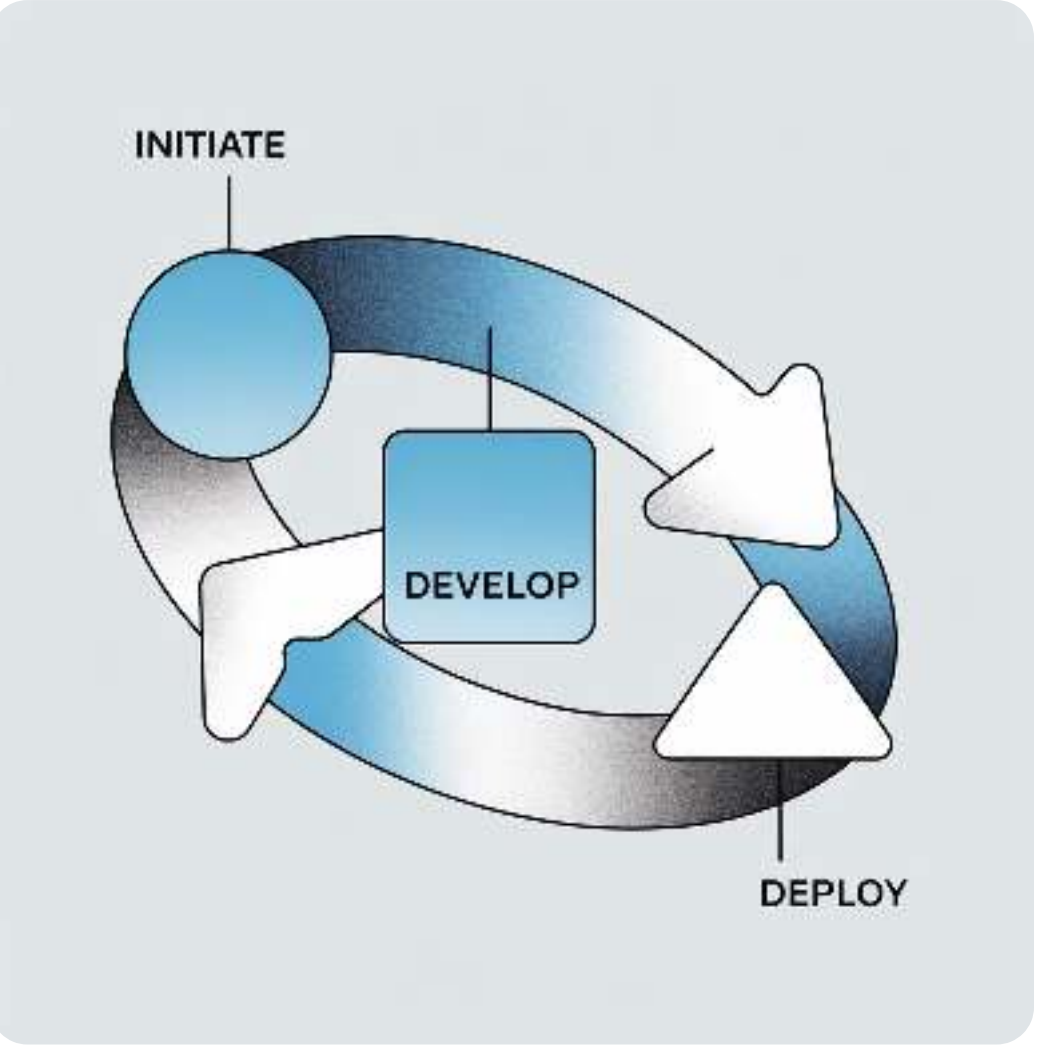
Ocorre quando uma nova tarefa é admitida no sistema e começa a ser preparada para executar.

Nova → Pronta

Ocorre quando a nova tarefa termina de ser carregada em memória, juntamente com suas bibliotecas e dados, estando pronta para executar.

Pronta → Executando

Ocorre quando a tarefa é escolhida pelo escalonador para ser executada (ou para continuar sua execução), dentre as demais tarefas prontas.



Ciclo de Vida das Tarefas: Transições (2/2)

Executando → Pronta

Ocorre quando se esgota a fatia de tempo (quantum) destinada à tarefa; ela volta à fila de tarefas prontas até receber o processador novamente.

Executando → Suspensa

Caso a tarefa solicite acesso a um recurso não disponível, como dados externos ou alguma sincronização, ela abandona o processador e fica suspensa até o recurso ficar disponível.

Suspensa → Pronta

Quando o recurso solicitado pela tarefa se torna disponível, ela volta ao estado de pronta para aguardar o processador.

Executando → Terminada

Ocorre quando a tarefa encerra sua execução ou é abortada em consequência de algum erro (acesso inválido à memória, instrução ilegal, divisão por zero, etc.).

Terminada → ...

Uma tarefa terminada é removida da memória e seus registros e estruturas de controle no núcleo são liberados.

Diferentes Interpretações do Ciclo de Vida

Modelo apresentado (Silberschatz)

- Nova
- Pronta
- Executando
- Suspensa (único estado)
- Terminada

Modelo Tanenbaum

- Nova
- Pronta
- Executando
- Bloqueada (aguardando evento)
- Suspensa (movida para área de troca)
- Terminada

A distinção de Tanenbaum entre "bloqueada" e "suspensa" não faz mais sentido nos sistemas operacionais atuais baseados em memória paginada, pois neles os processos podem executar mesmo estando somente parcialmente carregados na memória.

Verificando o Estado das Tarefas no Linux

No Linux, diversos utilitários permitem visualizar o estado das tarefas em execução, como o comando `top`:

- R (Running): Executando ou pronta para executar
- S (Sleeping): Suspensa (sono interruptível)
- D (Disk Sleep): Suspensa em sono não-interruptível
- Z (Zombie): Terminada, mas ainda presente na tabela de processos
- T (Stopped): Parada por sinal de controle

```
top - 16:58:06 up 8:26, 1 user, load average: 6,04, 2,36, 1,08
Tarefas: 218 total, 7 executando, 211 dormindo, 0 parado, 0 zumbi
%Cpu(s): 49,7 us, 47,0 sy, 0,0 ni, 3,2 id, 0,0 wa, 0,0 hi, 0,1 si, 0,0 st
KiB Mem : 16095364 total, 9856576 free, 3134380 used, 3104408 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 11858380 avail Mem

PID USUÁRIO PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
32703 maziero 20 0 2132220 432628 139312 S 44,8 2,7 0:53.64 Web Content
2192 maziero 20 0 9617080 686444 248996 S 29,8 4,3 20:01.81 firefox
11650 maziero 20 0 2003888 327036 129164 R 24,0 2,0 1:16.70 Web Content
```

Verificando o Estado das Tarefas no Windows

No Windows, o Gerenciador de Tarefas permite visualizar e gerenciar os processos em execução:

- Em execução
- Suspenso
- Não responde

O Windows não mostra todos os estados internos, mas permite ver informações como uso de CPU, memória e disco por cada processo.

Tanto no Linux quanto no Windows, uma tarefa é considerada "em execução" se estiver usando ou esperando por um processador.

Conceitos Fundamentais Revisados

Tarefa vs. Programa

Programa é estático (conjunto de instruções), tarefa é dinâmica (execução com estado)

Evolução Histórica

De sistemas monotarefa até sistemas de tempo compartilhado preemptivos

Estados da Tarefa

Nova, pronta, executando, suspensa e terminada

Preempção

Capacidade de interromper uma tarefa para dar lugar a outra

Estes conceitos formam a base da gestão de tarefas em todos os sistemas operacionais modernos.

Exercício 1: Time Sharing

O que significa time sharing e qual a sua importância em um sistema operacional?

Resposta: Time sharing (compartilhamento de tempo) é uma técnica que permite a múltiplas tarefas utilizarem o processador de forma alternada, com cada tarefa recebendo uma fatia de tempo (quantum). Sua importância está em:

- Permitir que múltiplos usuários/aplicações compartilhem os recursos computacionais simultaneamente
- Evitar que uma tarefa monopolize o processador indefinidamente
- Possibilitar aplicações interativas com tempo de resposta aceitável
- Melhorar a utilização dos recursos do sistema

Exercício 2: Duração do Quantum

Como e com base em que critérios é escolhida a duração de um quantum de processamento?

Resposta: A duração do quantum é escolhida buscando um equilíbrio entre:

- **Quantum curto:** Melhor interatividade e tempo de resposta para o usuário, mas gera sobrecarga com trocas frequentes de contexto
- **Quantum longo:** Melhor eficiência do processador, mas prejudica a interatividade

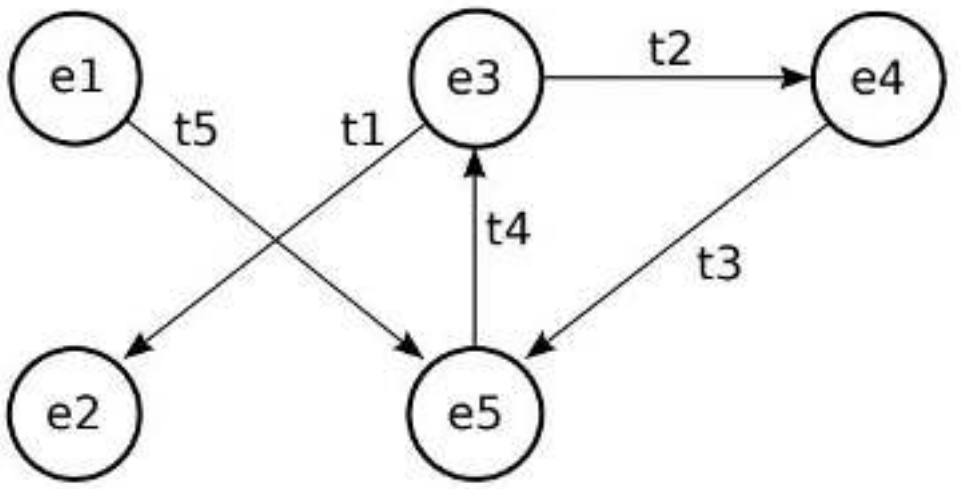
Sistemas como Linux usam quantums variáveis (10-200ms) considerando:

- Tipo de tarefa (interativa ou em lote)
- Prioridade da tarefa
- Histórico de uso do processador

Exercício 3: Diagrama de Estados

Considerando o diagrama de estados apresentado, complete o diagrama com a transição de estado que está faltando (t_6) e apresente o significado de cada um dos estados e transições.

Resposta: A transição t_6 que falta é **Suspenso → Terminado**, que ocorre quando uma tarefa suspensa é abortada pelo sistema ou por um sinal externo.



Os estados são: Nova (sendo criada), Pronta (aguardando processador), Executando (utilizando processador), Suspensa (aguardando evento) e Terminada (finalizou execução).

Exercício 4: Transições Possíveis (1/2)

Indique se cada uma das transições de estado de tarefas a seguir definidas é possível ou não:

E → P

✅ **Possível:** Ocorre quando se esgota o quantum de tempo da tarefa em execução, forçando-a a voltar para a fila de prontas.

S → T

✅ **Possível:** Ocorre quando uma tarefa suspensa é abortada pelo sistema ou por um sinal externo (ex: kill em um processo bloqueado).

E → T

✅ **Possível:** Ocorre quando a tarefa em execução termina normalmente ou é abortada por um erro (divisão por zero, acesso inválido à memória).

E → S

✅ **Possível:** Ocorre quando a tarefa em execução solicita um recurso não disponível (operação de E/S, aguarda semáforo).

N: Nova, P: Pronta, E: Executando, S: Suspensa, T: Terminada

Exercício 4: Transições Possíveis (2/2)

Indique se cada uma das transições de estado de tarefas a seguir definidas é possível ou não:

S → E

❌ **Impossível:** Uma tarefa suspensa não pode ir diretamente para execução; deve primeiro passar pelo estado "Pronta" (S → P → E).

N → S

❌ **Impossível:** Uma tarefa nova não pode ir diretamente para suspensa; deve primeiro passar pelos estados "Pronta" e "Executando" (N → P → E → S).

P → S

❌ **Impossível:** Uma tarefa pronta não pode ir diretamente para suspensa; deve primeiro passar pelo estado "Executando" (P → E → S).

P → N

❌ **Impossível:** Uma tarefa pronta não pode voltar ao estado "Nova"; o processo de criação é unidirecional.

N: Nova, P: Pronta, E: Executando, S: Suspensa, T: Terminada

Exercício 5: Estados das Tarefas

Relacione as afirmações aos respectivos estados no ciclo de vida das tarefas:

01
10

Nova (N)

- O código da tarefa está sendo carregado

Pronta (P)

- As tarefas são ordenadas por prioridades
- A tarefa vai a este estado ao terminar seu quantum
- A tarefa só precisa do processador para poder executar

Executando (E)

- A tarefa sai deste estado ao solicitar uma operação de E/S
- A tarefa pode criar novas tarefas
- Há uma tarefa neste estado para cada processador do sistema

√4

Suspensa (S)

- O acesso a um semáforo em uso pode levar a tarefa a este estado
- A tarefa aguarda a ocorrência de um evento externo

5

Terminada (T)

- Os recursos usados pela tarefa são devolvidos ao sistema

Resumo e Conclusão

Nesta apresentação, exploramos:

- O conceito fundamental de **tarefa** e sua diferença de **programa**
- A **evolução histórica** dos sistemas de gestão de tarefas: de monotarefa para sistemas de tempo compartilhado
- Os **estados possíveis** de uma tarefa: nova, pronta, executando, suspensa e terminada
- As **transições** entre estes estados e seus significados
- A importância da **preempção** e do conceito de **quantum** de tempo
- Como **visualizar** o estado das tarefas em sistemas reais

Estes conceitos são fundamentais para a compreensão do funcionamento dos sistemas operacionais modernos e sua capacidade de gerenciar múltiplas tarefas simultaneamente.