



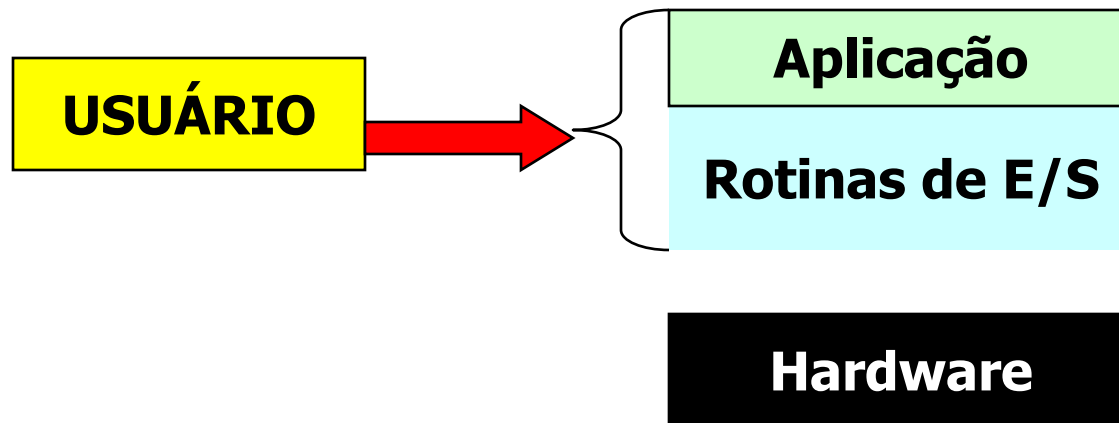
Sistemas Operacionais

Introdução

A Importância do Sistema Operacional (S.O.)

■ Sistema sem S.O.

- Gasto maior de tempo de programação
- Aumento da dificuldade
- Usuário preocupado com detalhes de hardware

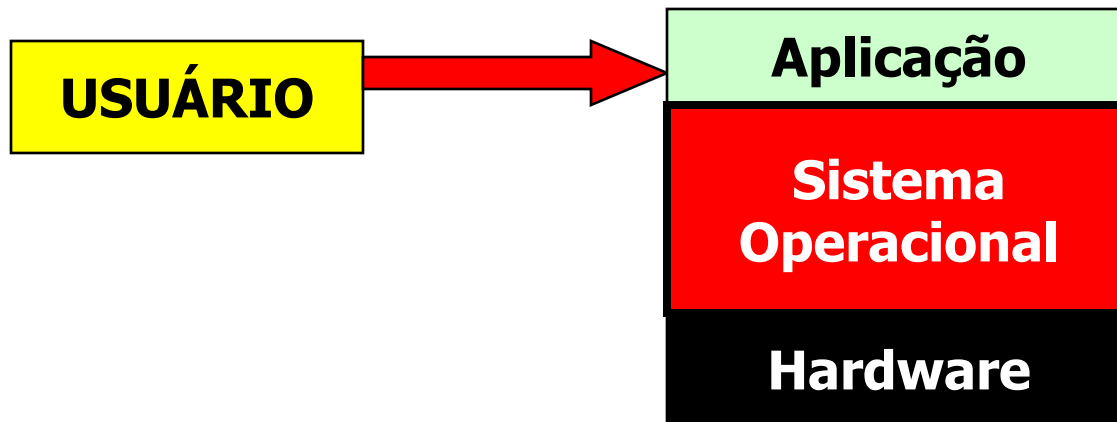


Introdução

A Importância do Sistema Operacional (S.O.)

■ Sistema com S.O.

- Maior racionalidade
- Maior dedicação aos problemas de alto nível
- Maior portabilidade



Máquinas Multinível



Introdução

Definição de Sistema Operacional

Um sistema operacional é um programa, ou conjunto de programas, interrelacionados cuja finalidade é agir como intermediário entre o usuário e o hardware. Possui várias funções, entre elas:

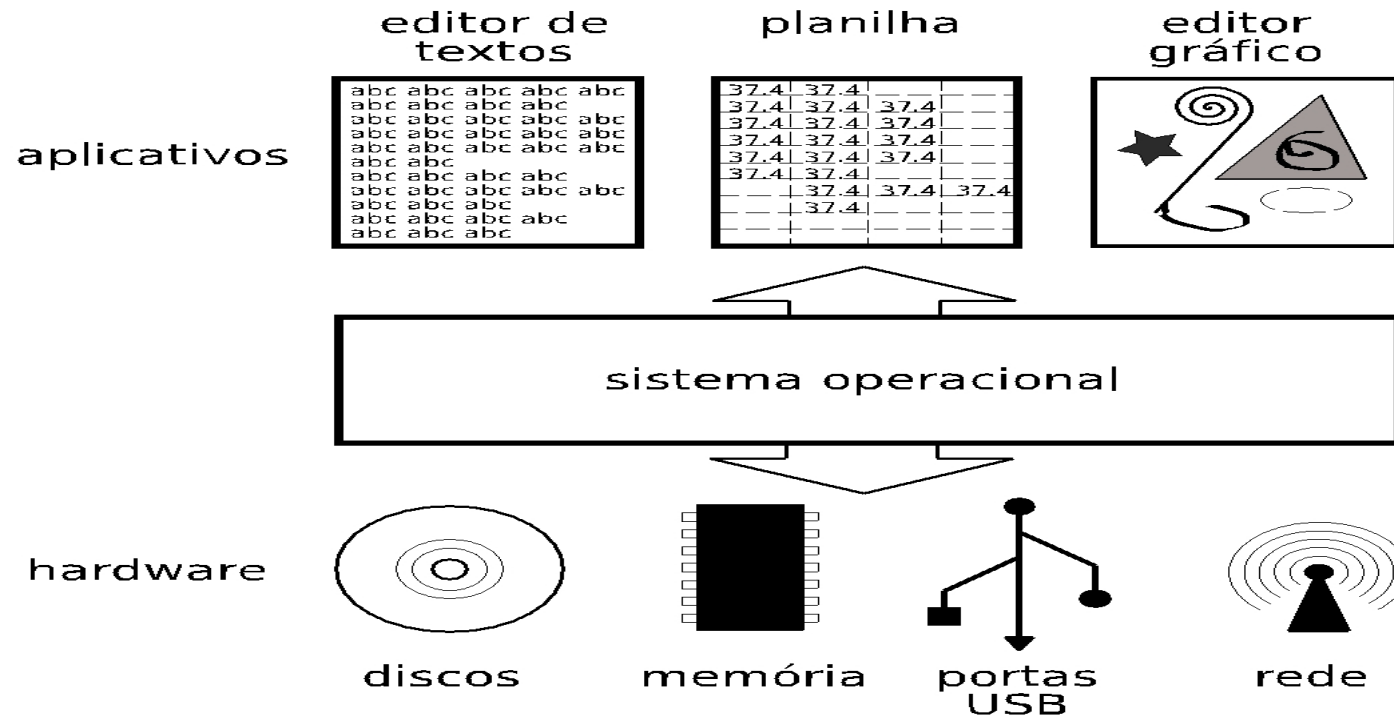
- apresentar uma máquina mais flexível;
- permitir o uso eficiente e controlado dos componentes de hardware;
- permitir o uso compartilhado e protegido dos diversos componentes de hardware e software, por diversos usuários.



SO – Conceitos básicos

- O sistema operacional é uma camada de software que opera entre o hardware e os programas aplicativos voltados ao usuário final.
- O sistema operacional é uma estrutura de software ampla, que incorpora aspectos de baixo nível (como drivers de dispositivos e gerência de memória física) e de alto nível (como programas utilitários e a própria interface gráfica).

SO – Conceitos básicos



Arquitetura geral de um sistema computacional típico.



SO – Conceitos básicos

Os objetivos básicos de um sistema operacional podem ser sintetizados em duas palavras-chave:

“abstração” e “gerência”



SO – Conceitos básicos

Abstração:

O sistema operacional deve definir interfaces abstratas para os recursos do hardware.



SO – Conceitos básicos

Abstração: Objetivos.

- Prover interfaces de acesso aos dispositivos, mais simples de usar que as interface de baixo nível, para simplificar a construção de programas aplicativos.
- Tornar os aplicativos independentes do hardware.
- Definir interfaces de acesso homogêneas para dispositivos com tecnologias distintas.



SO – Conceitos básicos

Gerência de recursos:

O sistema operacional deve definir políticas para gerenciar o uso dos recursos de hardware pelos aplicativos, e resolver eventuais disputas e conflitos.

SO – Conceitos básicos

Gerência de recursos:

Situações onde se faz necessária a gerência de recursos:

- O uso desse processador deve ser distribuído entre os aplicativos presentes no sistema, de forma que cada um deles possa executar na velocidade adequada para cumprir suas funções sem prejudicar os outros.
- A memória RAM, que deve ser distribuída de forma justa entre as aplicações.
- A impressora é um recurso cujo acesso deve ser efetuado de forma mutuamente exclusiva (apenas um aplicativo por vez).
- Impedir que todos os recursos do sistema sejam monopolizados por um só usuário (quota de uso por usuário).



SO – Conceitos básicos

Resumindo:

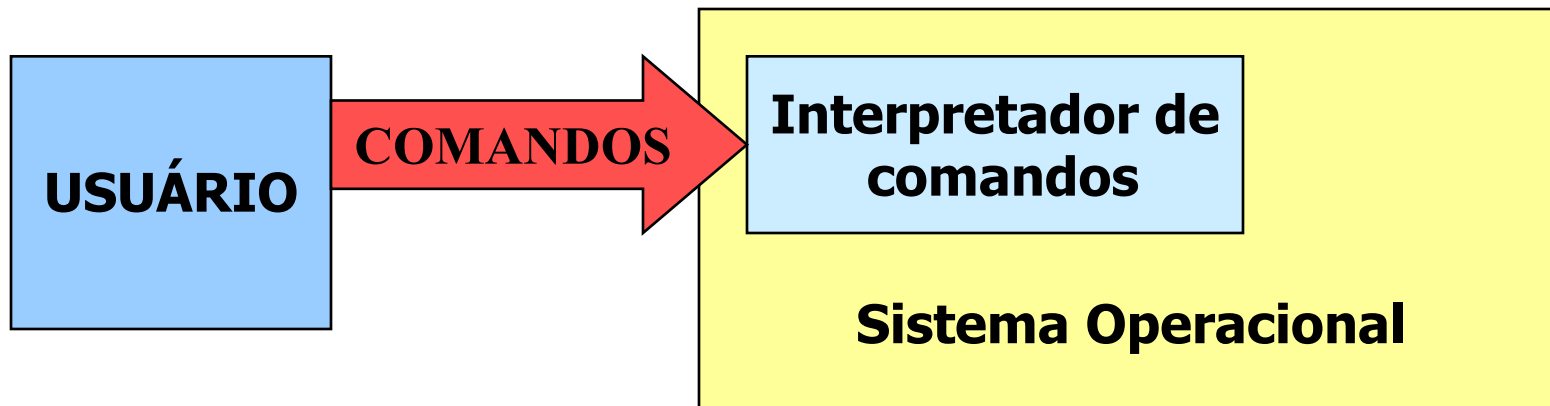
Um sistema operacional visa abstrair o hardware gerenciando seus recursos e provendo aos aplicativos um ambiente de execução abstrato, no qual o acesso aos recursos de hardware se dá através de interfaces simples, independentes das características de baixo nível do hardware, e no qual os conflitos no uso do hardware são minimizados.

Introdução

Interação com o Sistema Operacional

■ O USUÁRIO

Interage com o S.O. de maneira direta, através de comandos pertencentes à uma linguagem de comunicação especial, chamada "linguagem de comando".

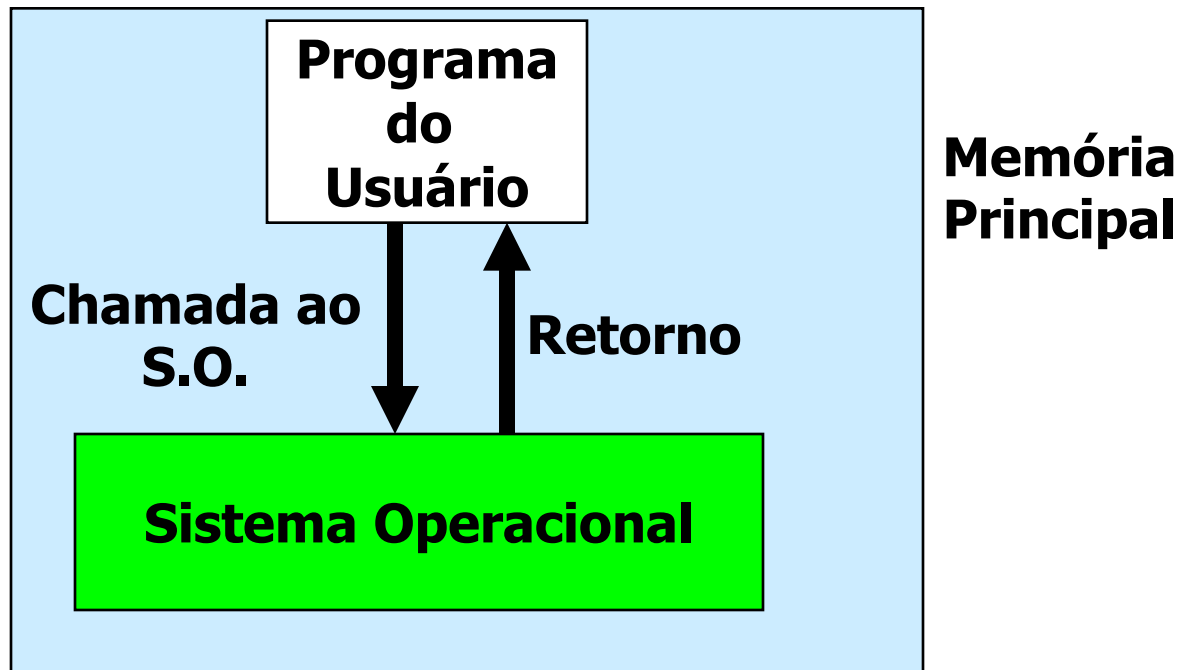


Introdução

Interação com o Sistema Operacional

■ OS PROGRAMAS DE USUÁRIO

Invocam os serviços do S.O. por meio das “chamadas ao sistema operacional”.





SO – Tipos de sistemas.

Os sistemas operacionais podem ser classificados de acordo com diversos parâmetros e perspectivas, como tamanho, velocidade, suporte a recursos específicos, acesso à rede, etc.

Introdução

A Evolução dos Sistemas Operacionais

- O Alcance e extensão dos serviços de um S.O: dependem das necessidades e características do ambiente que deve suportar.
- Um S.O. pode processar sua carga de trabalho (workload) de duas formas:
 - **Serial:** neste, os recursos são dedicados à um único programa, até o seu término.
 - **Concorrente:** os recursos são dinamicamente reassociados entre uma coleção de programas ativos, em diferentes estágios de execução.

Introdução

Tipos de Sistemas Operacionais

■ Classificação quanto ao compartilhamento de hardware

- **Sistemas Operacionais Monoprogramados**
 - Só permite um programa ativo em um dado período de tempo, o qual permanece na memória até seu término
- **Sistemas Operacionais Multiprogramados**
 - Mantém mais de um programa simultaneamente na memória principal, para permitir o compartilhamento efetivo do tempo de UCP e demais recursos

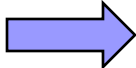


SO – Tipos de sistemas.

Batch (de lote): os sistemas operacionais mais antigos trabalhavam “por lote”, ou seja, todos os programas a executar eram colocados em uma fila, com seus dados e demais informações para a execução. O processador recebia um programa após o outro, processando-os em sequência, o que permitia um alto grau de utilização do sistema.

Introdução

■ Classificação quanto a interação permitida

- fator determinante  Tempo de resposta

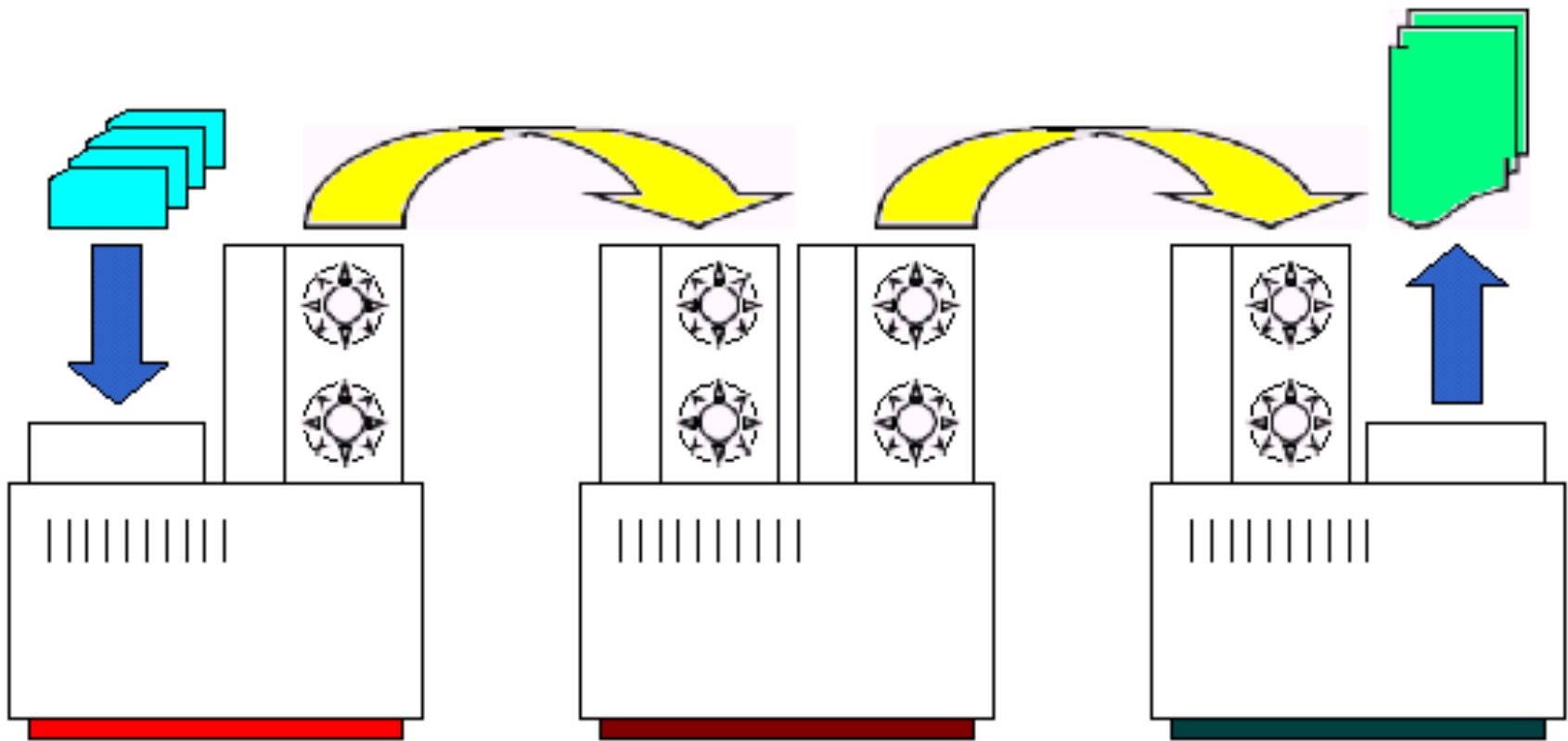
■ S.O. para processamento em Batch (lote)

- Os jobs dos usuários são submetidos em ordem sequencial para a execução
- Não existe interação entre o usuário e o job durante sua execução



Introdução

■ S.O. para processamento em Batch (lote)



Sistema Batch (processamento em lote)



SO – Tipos de sistemas.

De rede: um sistema operacional de rede deve possuir suporte à operação em rede, ou seja, a capacidade de oferecer às aplicações locais recursos que estejam localizados em outros computadores da rede, como arquivos e impressoras. Ele também deve disponibilizar seus recursos locais aos demais computadores, de forma controlada. A maioria dos sistemas atuais oferece esse tipo de funcionalidade.



SO – Tipos de sistemas.

Distribuído: em um sistema operacional distribuído, os recursos de cada máquina estão disponíveis globalmente, de forma transparente aos usuários. Ao lançar uma aplicação, o usuário interage com sua janela, mas não sabe onde ela está executando ou armazenando seus arquivos: o sistema é quem decide, de forma transparente.



SO – Tipos de sistemas.

Multi-Usuário: Um sistema operacional multiusuário deve suportar a identificação do “dono” de cada recurso dentro do sistema (arquivos, processos, áreas de memória, conexões de rede) e impor regras de controle de acesso para impedir o uso desses recursos por usuários não autorizados. Essa funcionalidade é fundamental para a segurança dos sistemas operacionais de rede e distribuídos.

Grande parte dos sistemas atuais são multiusuários.

SO – Tipos de sistemas.

Desktop: um sistema operacional “de mesa” é voltado ao atendimento do usuário doméstico e corporativo para a realização de atividades corriqueiras, como edição de textos e gráficos, navegação na Internet e reprodução de mídias simples. Suas principais características são a interface gráfica, o suporte à interatividade e a operação em rede. Exemplos de sistemas desktop são o Windows, MacOS e Linux.



SO – Tipos de sistemas.

Servidor: um sistema operacional servidor deve permitir a gestão eficiente de grandes quantidades de recursos (disco, memória, processadores), impondo prioridades e limites sobre o uso dos recursos pelos usuários e seus aplicativos. Normalmente um sistema operacional servidor também tem suporte a rede e multi-usuários.



SO – Tipos de sistemas.

Embutido: um sistema operacional é dito embutido (embedded) quando é construído para operar sobre um hardware com poucos recursos de processamento, armazenamento e energia. Aplicações típicas desse tipo de sistema aparecem em telefones celulares, controladores industriais e automotivos.



SO – Tipos de sistemas.

Tempo Real: ao contrário da concepção usual, um sistema operacional de tempo real não precisa ser necessariamente ultrarápido; sua característica essencial é ter um comportamento temporal previsível (ou seja, seu tempo de resposta deve ser conhecido no melhor e pior caso de operação). A estrutura interna de um sistema operacional de tempo real deve ser construída de forma a minimizar esperas e latências imprevisíveis, como tempos de acesso a disco e sincronizações excessivas.

SO – Tipos de sistemas.

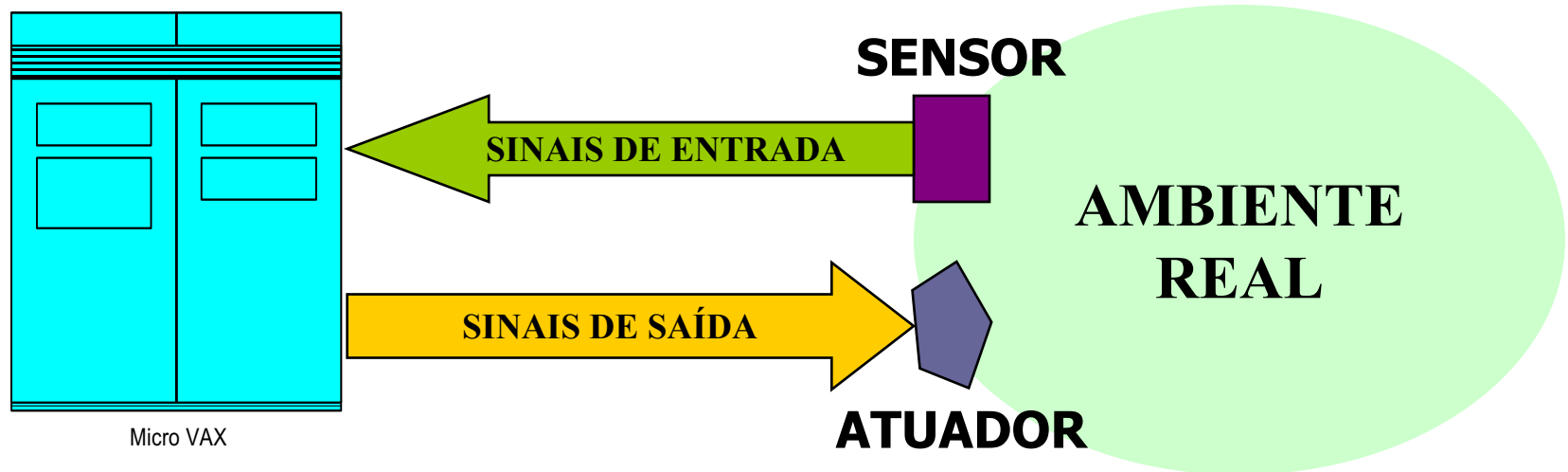
Tempo Real :

- soft real-time systems: nos quais a perda de prazos implica na degradação do serviço prestado. Um exemplo seria o suporte à gravação de CDs ou à reprodução de músicas.
- hard real-time systems: a perda de prazos pelo sistema pode perturbar o objeto controlado, com graves consequências humanas, econômicas ou ambientais. Exemplos: controle de funcionamento de uma turbina de avião a jato ou de uma caldeira industrial.

Introdução

S.O. de Tempo Real

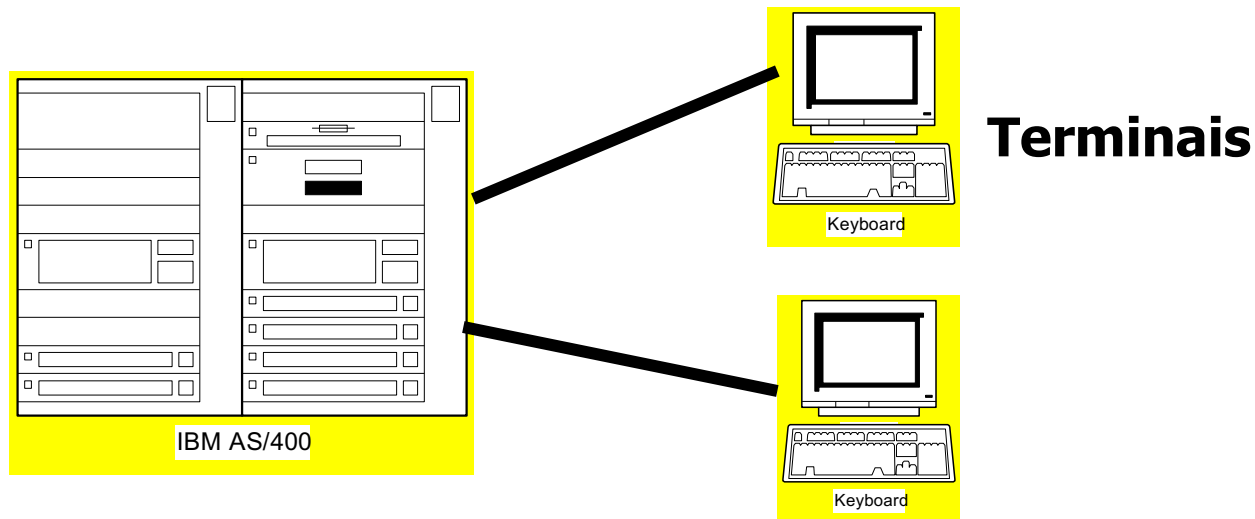
- Usados para servir aplicações que atendem processos externos, e que possuem tempos de resposta limitados
- Geralmente sinais de interrupções comandam a atenção do sistema
- Geralmente são projetados para uma aplicação específica



Introdução

S.O. Interativo

- O sistema permite que os usuários interajam com suas computações na forma de diálogo
- Podem ser projetados como sistemas mono-usuários ou multi-usuários (usando conceitos de multiprogramação e time-sharing)





SO - Funcionalidades

- Para cumprir seus objetivos de abstração e gerência, o sistema operacional deve atuar em várias frentes. Cada um dos recursos do sistema possui suas particularidades, o que impõe exigências específicas para gerenciar e abstrair os mesmos. As principais funções implementadas por um sistema operacional típico são:



SO - Funcionalidades

- **Gerência do processador:** também conhecida como gerência de processos ou de atividades, esta funcionalidade visa distribuir a capacidade de processamento de forma justa entre as aplicações, evitando que uma aplicação monopolize esse recurso e respeitando as prioridades dos usuários. Busca-se criar a abstração de “um processador para cada tarefa”.



SO - Funcionalidades

- **Gerência de memória:** tem como objetivo fornecer a cada aplicação um espaço de memória próprio, independente e isolado dos demais, inclusive do núcleo do sistema. Caso a memória RAM não seja suficiente, o sistema deve prover armazenamento secundário (espaço em disco) como complemento de memória, de forma transparente às aplicações.
- A principal abstração construída pela gerência de memória é a noção de memória virtual, que desvincula o espaço de endereços visto por cada aplicação do espaço físico.



SO - Funcionalidades

- **Gerência de dispositivos:** A função da gerência de dispositivos (também conhecida como gerência de entrada/saída) é implementar a interação com cada dispositivo por meio de drivers e criar modelos abstratos que permitam agrupar vários dispositivos distintos sob a mesma interface de acesso.



SO - Funcionalidades

- **Gerência de arquivos:** esta funcionalidade é construída sobre a gerência de dispositivos e visa criar as abstrações de arquivo e diretório, definindo também sua interface de acesso e as regras para seu uso.



SO - Funcionalidades

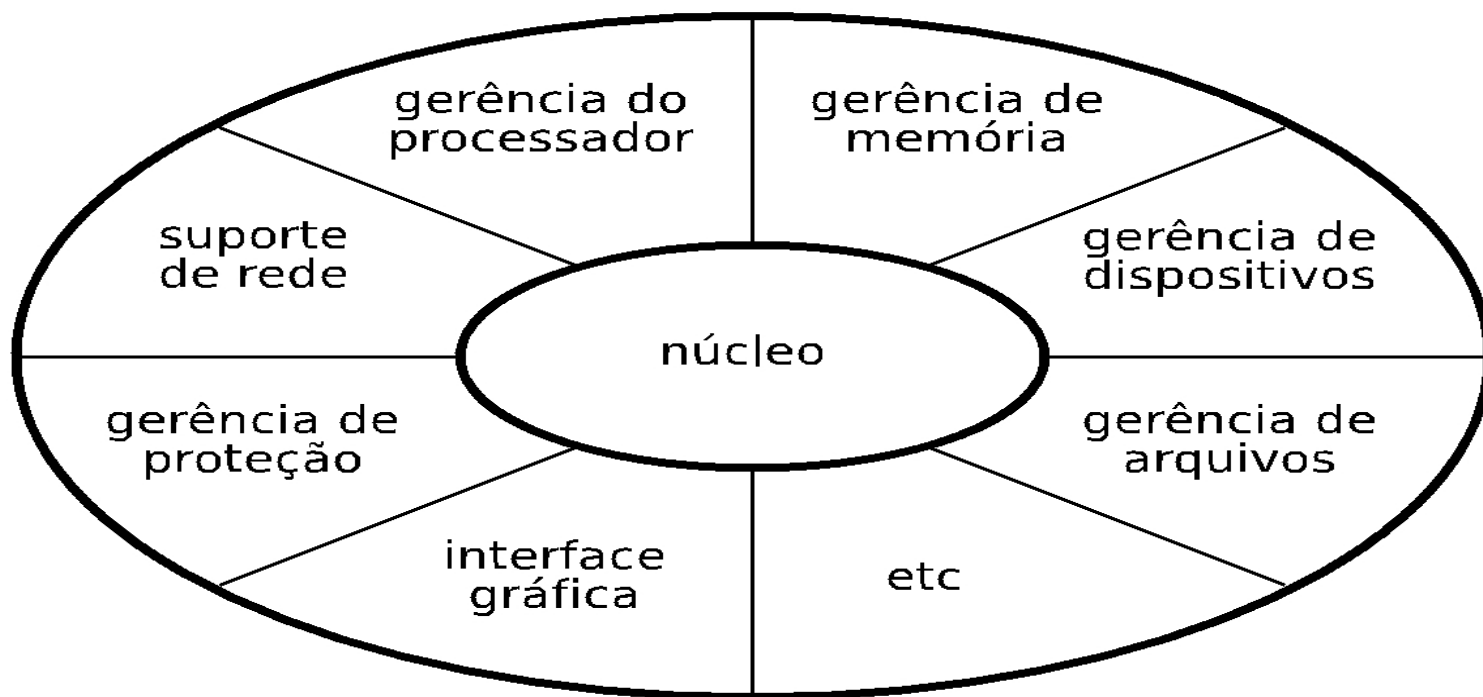
- **Gerência de proteção:** com computadores conectados em rede e compartilhados por vários usuários, é importante definir claramente os recursos que cada usuário pode acessar, as formas de acesso permitidas (leitura, escrita, etc) e garantir que essas definições serão cumpridas.



SO - Funcionalidades

Além dessas funcionalidades básicas, oferecidas pela maioria dos sistemas operacionais, várias outras vêm se agregando aos sistemas modernos, para cobrir aspectos complementares, como a interface gráfica, suporte de rede, fluxos multimídia, gerência de energia, etc.

SO - Funcionalidades





SO - Funcionalidades

Política X mecanismo



SO - Funcionalidades

Como política consideram-se os aspectos de decisão mais abstratos, que podem ser resolvidos por algoritmos de nível mais alto, como por exemplo decidir a quantidade de memória que cada aplicação ativa deve receber, ou qual o próximo pacote de rede a enviar para satisfazer determinadas especificações de qualidade de serviço.



SO - Funcionalidades

Como mecanismo consideram-se os procedimentos de baixo nível usados para implementar as políticas.

Os mecanismos devem ser suficientemente genéricos para suportar mudanças de política sem necessidade de modificações.



SO - Funcionalidades

Essa separação entre os conceitos de política e mecanismo traz uma grande flexibilidade aos sistemas operacionais, permitindo alterar sua personalidade sem ter de mexer no código que interage diretamente com o hardware.



S.O. – Estrutura do sistema

Um sistema operacional não é um bloco único e fechado de software executando sobre o hardware. Na verdade, ele é composto de diversos componentes que têm objetivos e funcionalidades complementares.

Alguns dos componentes mais relevantes de um sistema operacional típico são:



S.O. – Estrutura do sistema

Núcleo: é o coração do sistema operacional, responsável pela gerência dos recursos do hardware usados pelas aplicações. Ele também implementa as principais abstrações utilizadas pelos programas aplicativos.



S.O. – Estrutura do sistema

Drivers: módulos de código específicos para acessar os dispositivos físicos. Existe um driver para cada tipo de dispositivo, como discos rígidos, portas USB, placas de vídeo, etc. Muitas vezes o driver é construído pelo próprio fabricante do hardware e fornecido em forma binária para ser acoplado ao restante do sistema operacional.



S.O. – Estrutura do sistema

Código de inicialização: a inicialização do hardware requer uma série de tarefas complexas, como reconhecer os dispositivos instalados, testá-los e configurá-los adequadamente para seu uso posterior. Outra tarefa importante é carregar o núcleo do sistema operacional em memória e iniciar sua execução.



S.O. – Estrutura do sistema

Programas utilitários: são programas que facilitam o uso do sistema computacional, fornecendo funcionalidades complementares ao núcleo, como formatação de discos e mídias, configuração de dispositivos, manipulação de arquivos (mover, copiar, apagar), interpretador de comandos, terminal, interface gráfica, gerência de janelas, etc.



S.O. – Proteção do núcleo

Um sistema operacional deve gerenciar os recursos do hardware, fornecendo-os às aplicações conforme suas necessidades. Para assegurar a integridade dessa gerência, é essencial garantir que as aplicações não consigam acessar o hardware diretamente, mas sempre através de pedidos ao sistema operacional, que avalia e intermedia todos os acessos ao hardware.



S.O. – Proteção do núcleo

Para permitir a diferenciação de privilégio de acesso entre os diferentes tipos de software, os processadores modernos contam com dois ou mais níveis de privilégio de execução. Esses níveis são controlados por flags especiais nos processadores, e a mudança de um nível de execução para outro é controlada por condições específicas.



S.O. – Proteção do núcleo

Na forma mais simples desse esquema, podemos considerar dois níveis básicos de privilégio:

Nível núcleo: também denominado nível supervisor, sistema, monitor ou ainda *kernel space*. Para um código executando nesse nível, todo o processador está acessível: todos os registradores, portas de entrada/saída e áreas de memória podem ser acessados em leitura e escrita. Além disso, todas as instruções do processador podem ser executadas.



S.O. – Proteção do núcleo

Nível usuário (ou *userspace*): neste nível, somente um subconjunto das instruções do processador, registradores e portas de entrada/saída estão disponíveis. Instruções “perigosas” como HALT (parar o processador) e RESET (reiniciar o processador) são proibidas para todo código executando neste nível. Além disso, o hardware restringe o uso da memória, permitindo o acesso somente a áreas previamente definidas.



S.O. – Proteção do núcleo

Caso o código em execução tente executar uma instrução proibida ou acessar uma área de memória inacessível, o hardware irá gerar uma exceção, desviando a execução para uma rotina de tratamento dentro do núcleo, que provavelmente irá abortar o programa em execução.



S.O. – Proteção do núcleo

É fácil perceber que, em um sistema operacional convencional, o núcleo e os drivers operam no nível núcleo, enquanto os utilitários e as aplicações operam no nível usuário, confinados em áreas de memória distintas.



S.O. – Chamadas do sistema

A ativação de procedimentos do núcleo usando interrupções de software é denominada chamada de sistema (system call ou syscall). Os sistemas operacionais definem chamadas de sistema para todas as operações envolvendo o acesso a recursos de baixo nível (periféricos, arquivos, etc) ou abstrações lógicas (criação e finalização de tarefas, operadores de sincronização e comunicação, etc). Geralmente as chamadas de sistema são oferecidas para as aplicações em modo usuário através de uma biblioteca do sistema (**system library**), que prepara os parâmetros, invoca a interrupção de software e retorna à aplicação os resultados obtidos.



S.O. – Chamadas do sistema

A maioria dos sistemas operacionais implementa centenas ou mesmo milhares de chamadas de sistema, para as mais diversas finalidades. O conjunto de chamadas de sistema oferecidas por um núcleo define a API (Application Programming Interface) desse sistema operacional.



S.O. - Arquiteturas

As múltiplas partes que compõem o sistema podem ser organizadas de diversas formas, separando suas funcionalidades e modularizando seu projeto.



Introdução

Estrutura de Sistemas Operacionais

- **Como os sistemas operacionais são normalmente grandes e complexas coleções de rotinas de software, os projetistas devem dar grande ênfase à sua organização interna e estrutura**



S.O. - Arquiteturas

Sistemas monolíticos

Em um sistema monolítico, todos os componentes do núcleo operam em modo núcleo e se inter-relacionam conforme suas necessidades, sem restrições de acesso entre si (pois o código no nível núcleo tem acesso pleno a todos os recursos e áreas de memória).



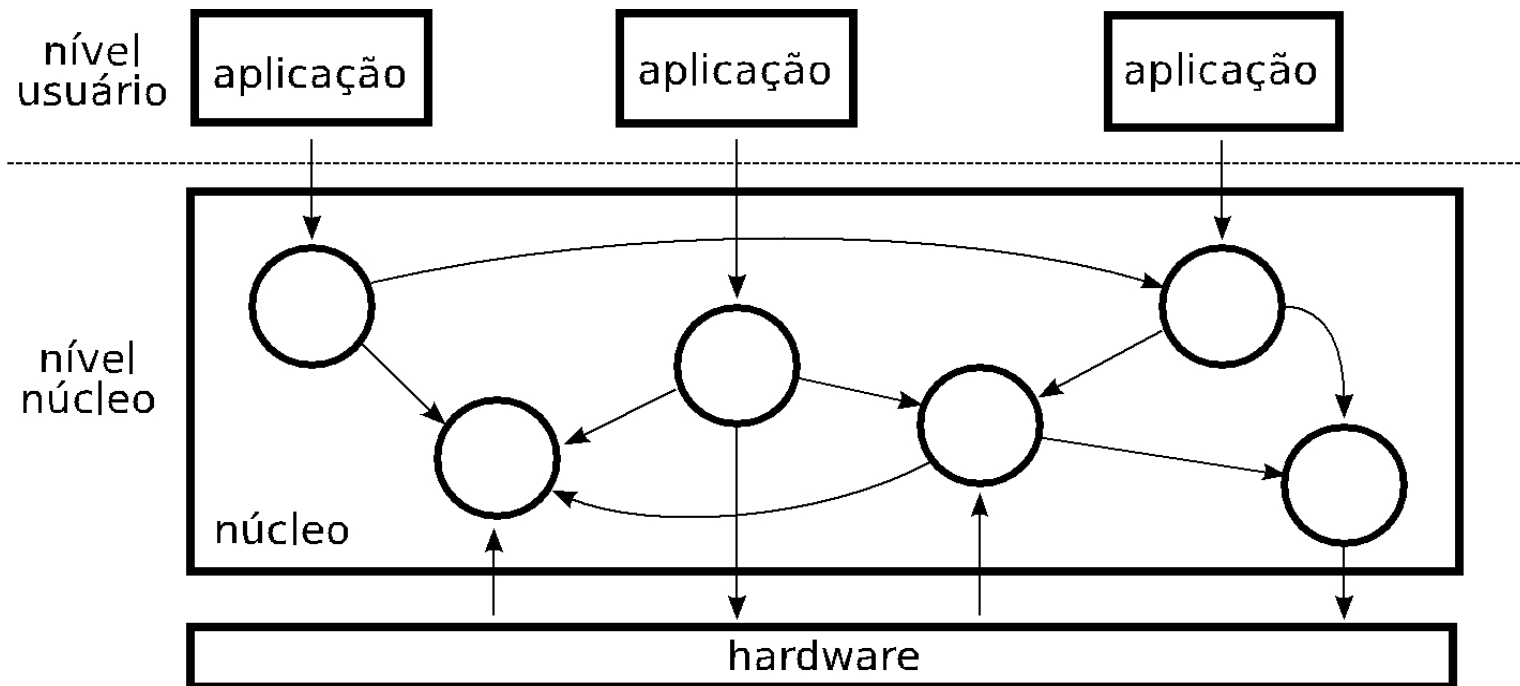
S.O. - Arquiteturas

Sistemas monolíticos

- Não há estruturação visível.
- SO é escrito como uma coleção de processos → cada processo podendo fazer chamadas a qualquer outro.
- Os serviços (system calls) são requisitados através da colocação dos parâmetros em lugares definidos (pilhas e registradores) e da execução de uma chamada de sistema especial ao kernel.

S.O. - Arquiteturas

Sistemas monolíticos





S.O. - Arquiteturas

Sistemas monolíticos

Vantagem: dessa arquitetura é seu desempenho: qualquer componente do núcleo pode acessar os demais componentes, toda a memória ou mesmo dispositivos periféricos diretamente, pois não há barreiras impedindo esse acesso. A interação direta entre componentes também leva a sistemas mais compactos.



S.O. - Arquiteturas

Sistemas monolíticos

Desvantagens:

- O mal funcionamento de uma aplicação do kernel pode se alastrar e levar o sistema ao colapso (travamento ou instabilidade)
- Manutenção mais complexa.
- Evolução mais complexa.



S.O. - Arquiteturas

Sistemas monolíticos

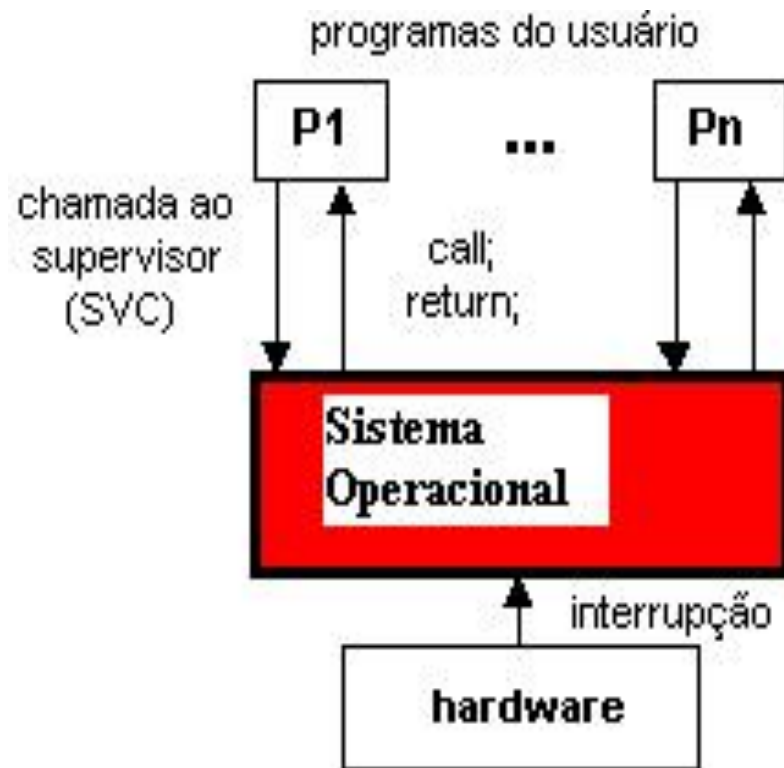
A arquitetura monolítica foi a primeira forma de organizar os sistemas operacionais. Sistemas UNIX antigos e o MS-DOS seguiam esse modelo.

Atualmente, apenas sistemas operacionais embutidos usam essa arquitetura, devido às limitações do hardware sobre o qual executam. O núcleo do Linux nasceu monolítico, mas vem sendo estruturado e modularizado desde a versão 2.0

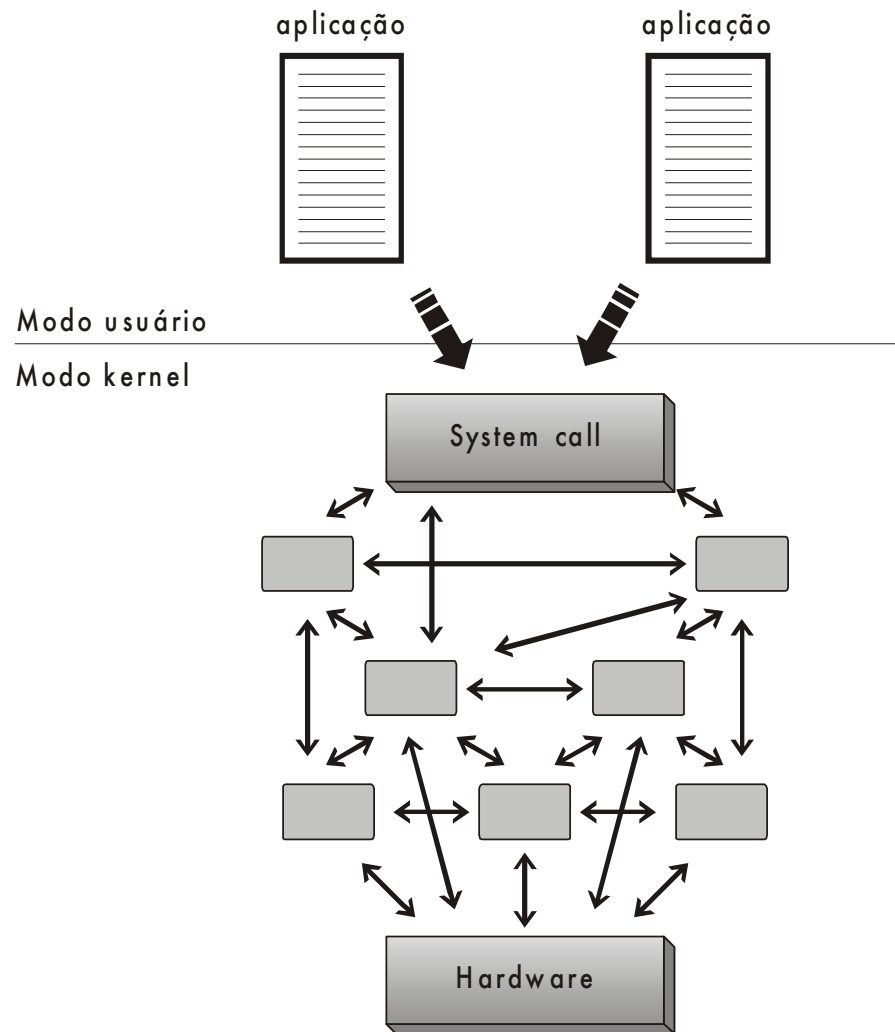
Introdução

Estrutura Monolítica

- É a forma mais primitiva de S.O.
- Consiste de um conjunto de programas que executam sobre o hardware, como se fosse um único programa.
- Os programas de usuário podem ser vistos como sub-rotinas, invocadas pelo S.O., quando este não está executando nenhuma das funções do sistema



Estrutura do Monolítica





S.O. - Arquiteturas

Sistemas em camadas

A camada mais baixa realiza a interface com o hardware, enquanto as camadas intermediárias proveem níveis de abstração e gerência cada vez mais sofisticados. Por fim, a camada superior define a interface do núcleo para as aplicações (as chamadas de sistema).



S.O. - Arquiteturas

Sistemas em camadas

Essa abordagem de estruturação de software fez muito sucesso no domínio das redes de computadores, através do modelo de referência OSI, e também seria de se esperar sua adoção no domínio dos sistemas operacionais. No entanto, alguns inconvenientes limitam sua aceitação nesse contexto.

Introdução

Sistemas de Camadas - Estrutura Hierárquica de Níveis de Abstração

Os princípios utilizados nesta abordagem são:

- | **Modularização:** divisão de um programa complexo em módulos de menor complexidade. Os módulos interagem através de interfaces bem definidas.
- | **Conceito de "Informação Escondida":** os detalhes das estruturas de dados e algoritmos são confinados em módulos. Externamente, um módulo é conhecido por executar uma função específica sobre objetos de determinado tipo.



Introdução

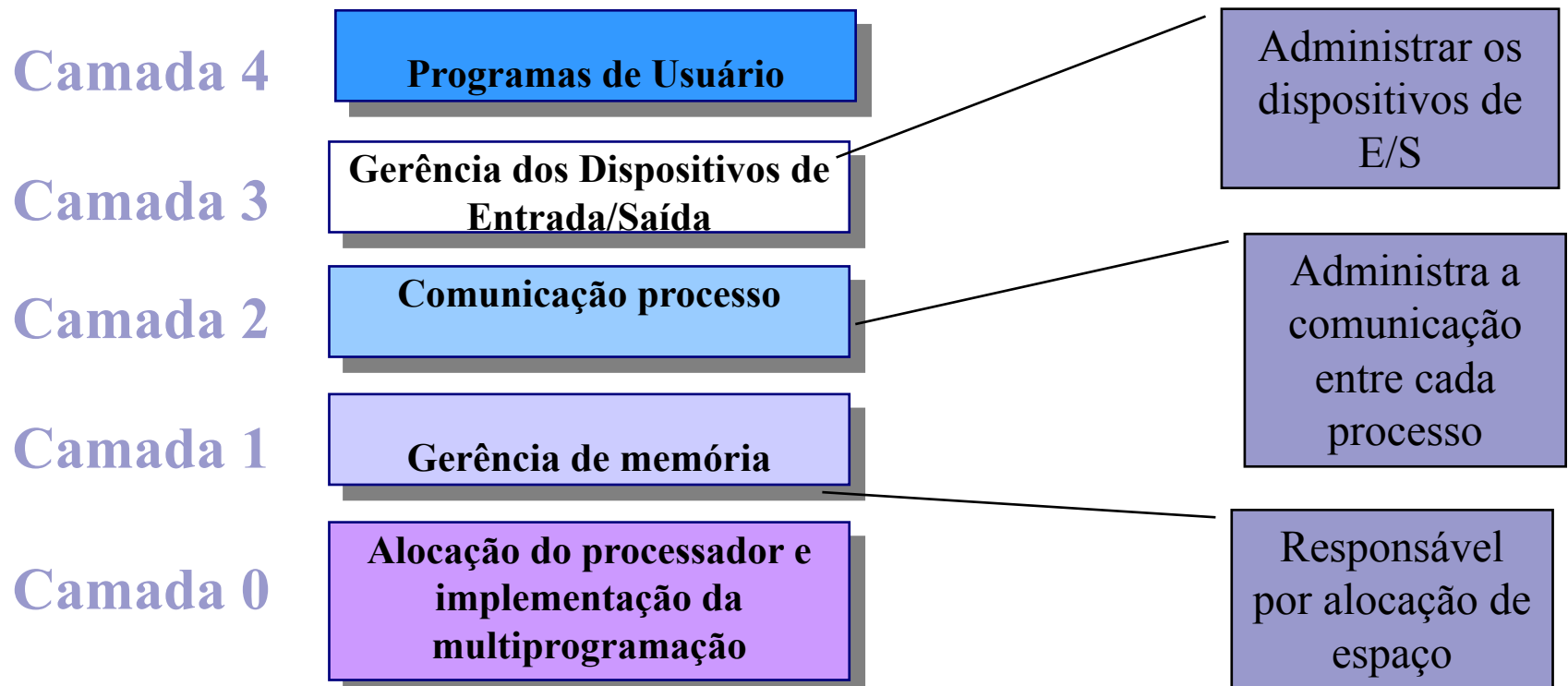
Estrutura Hierárquica de Níveis de Abstração

- **A idéia básica é criar um S.O. como uma hierarquia de níveis de abstração, de modo que, a cada nível, os detalhes de operação dos níveis inferiores possam ser ignorados. Através disso, cada nível pode confiar nos objetos e operações fornecidas pelos níveis inferiores.**

S.O. - Arquiteturas

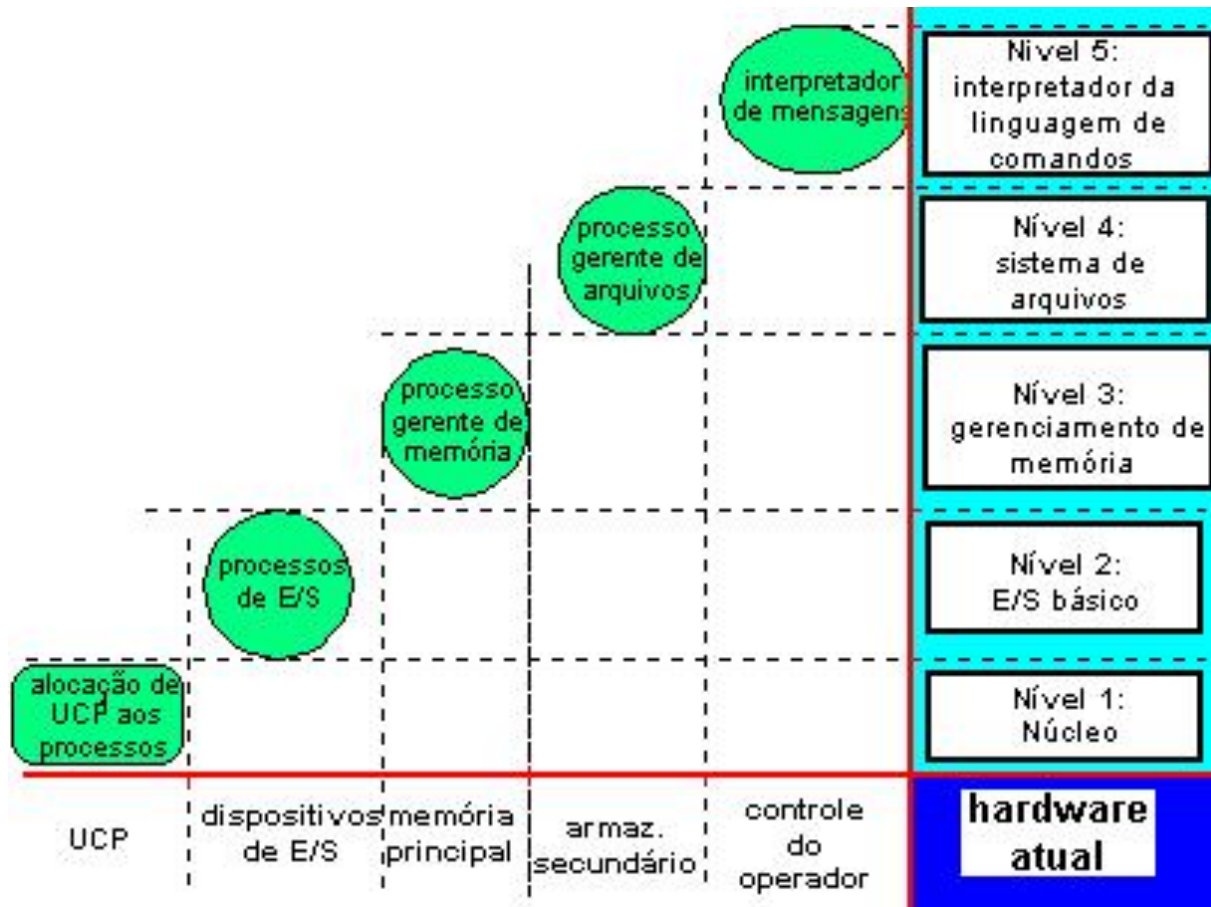
Sistemas em camadas

- Camadas sobrepostas;
- Cada módulo oferece um conjunto de funções que podem ser utilizadas por outros módulos.



Introdução

Estrutura Hierárquica de Níveis de Abstração





S.O. - Arquiteturas

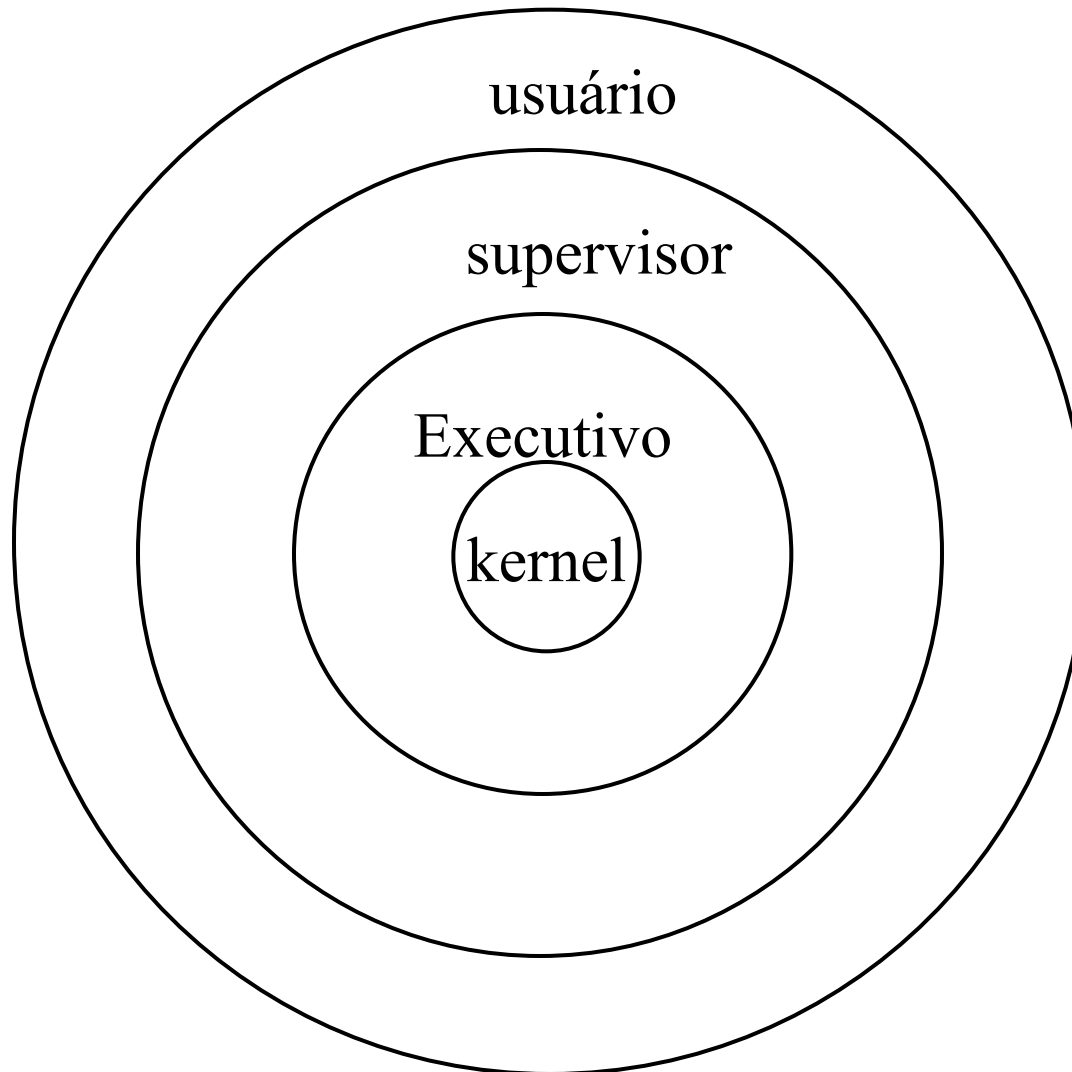
Sistemas em camadas

Sistema visto como camadas de anéis concêntricos.

- anéis mais internos são mais privilegiados que os externos.
- procedimentos de anéis externos executavam chamadas de sistema para utilizar os serviços dos anéis internos.
- proteção dos segmentos de memória.

S.O. - Arquiteturas

MULTICS
VMS





S.O. - Arquiteturas

Sistemas em camadas

Inconvenientes:

- O empilhamento de várias camadas de software faz com que cada pedido de uma aplicação demore mais tempo para chegar até o dispositivo periférico ou recurso a ser acessado, prejudicando o desempenho do sistema.
- Não é óbvio como dividir as funcionalidades de um núcleo de sistema operacional em camadas horizontais de abstração crescente, pois essas funcionalidades são interdependentes, embora tratem muitas vezes de recursos distintos.



S.O. - Arquiteturas

Sistemas em camadas

A estruturação em camadas é apenas parcialmente adotada hoje em dia. Muitos sistemas implementam uma camada inferior de abstração do hardware para interagir com os dispositivos e também organizam em camadas alguns subsistemas como a gerência de arquivos e o suporte de rede (seguindo o modelo OSI).



S.O. - Arquiteturas

Sistemas micro-núcleo (cliente/sevidor)

Uma outra possibilidade de estruturação consiste em retirar do núcleo todo o código de alto nível (normalmente associado às políticas de gerência de recursos), deixando no núcleo somente o código de baixo nível necessário para interagir com o hardware e criar as abstrações fundamentais (como a noção de atividade).



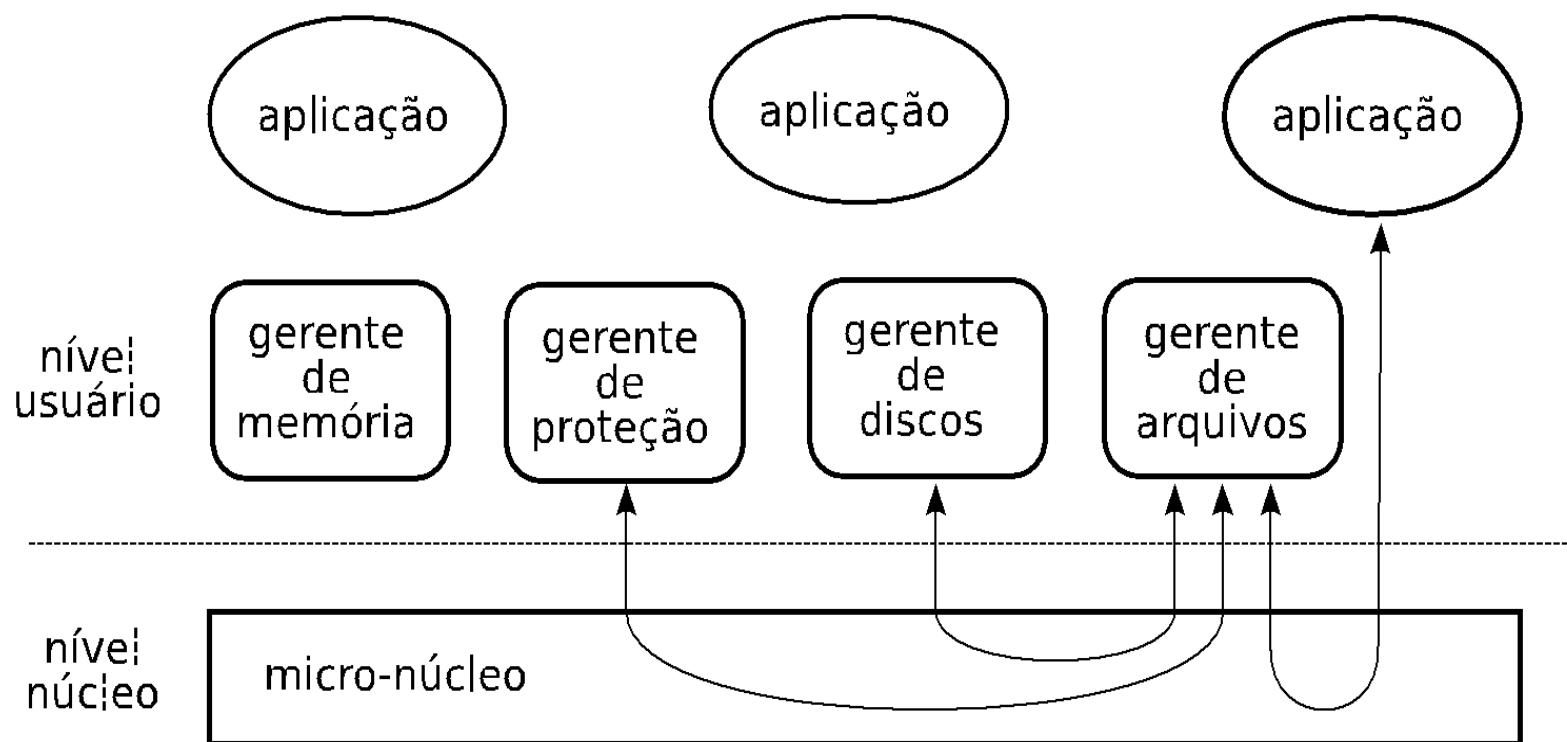
S.O. - Arquiteturas

Sistemas micro-núcleo

Por fazer os núcleos de sistema ficarem menores, essa abordagem foi denominada micro-núcleo. Um micro-núcleo normalmente implementa somente a noção de atividade, de espaços de memória protegidos e de comunicação entre atividades. Todos os aspectos de alto nível, como políticas de uso do processador e da memória, o sistema de arquivos e o controle de acesso aos recursos são implementados fora do núcleo, em processos que se comunicam usando as primitivas do núcleo.

S.O. - Arquiteturas

Sistemas micro-núcleo





S.O. - Arquiteturas

Sistemas micro-núcleo

Exemplo: usando essa abordagem o código de acesso aos blocos de um disco rígido seria mantido no núcleo, enquanto as abstrações de arquivo e diretório seriam criadas e mantidas por um código fora do núcleo, executando da mesma forma que uma aplicação do usuário.



S.O. - Arquiteturas

Sistemas micro-núcleo

Em um sistema micro-núcleo, as interações entre componentes e aplicações são feitas através de trocas de mensagens. Assim, se uma aplicação deseja abrir um arquivo no disco rígido, envia uma mensagem para o gerente de arquivos, que por sua vez se comunica com o gerente de dispositivos para obter os blocos de dados relativos ao arquivo desejado.



S.O. - Arquiteturas

Sistemas micro-núcleo

Todas as mensagens são transmitidas através de serviços do micro-núcleo. Como os processos têm de solicitar “serviços” uns dos outros (para poder realizar suas incumbências), essa abordagem também foi denominada cliente-servidor.



S.O. - Arquiteturas

Sistemas micro-núcleo

Vantagens:

Robustez e flexibilidade. Caso um sub-sistema tenha problemas, os mecanismos de proteção de memória e níveis de privilégio irão confiná-lo, impedindo que a instabilidade se alastre ao restante do sistema. Além disso, é possível **customizar** o sistema operacional, iniciando somente os componentes necessários ou escolhendo os componentes mais adequados às aplicações que serão executadas.



S.O. - Arquiteturas

Sistemas micro-núcleo

Desvantagens:

o custo associado às trocas de mensagens entre componentes pode ser bastante elevado, o que prejudica seu desempenho e diminui a aceitação desta abordagem.

S.O. - Arquiteturas

Sistemas micro-núcleo

Na prática a implementação de estruturas cliente-servidor é muito difícil devido a certas funções do sistema operacional exigirem acesso direto ao hardware;

- » exemplo: operações de e/s.

Na realidade é implementada uma combinação do modelo de camadas com o modelo cliente-servidor;

Núcleo do sistema:

- » realizar a comunicação entre cliente e servidor;
- » executar funções críticas do sistema;
- » executar funções dos device drivers.

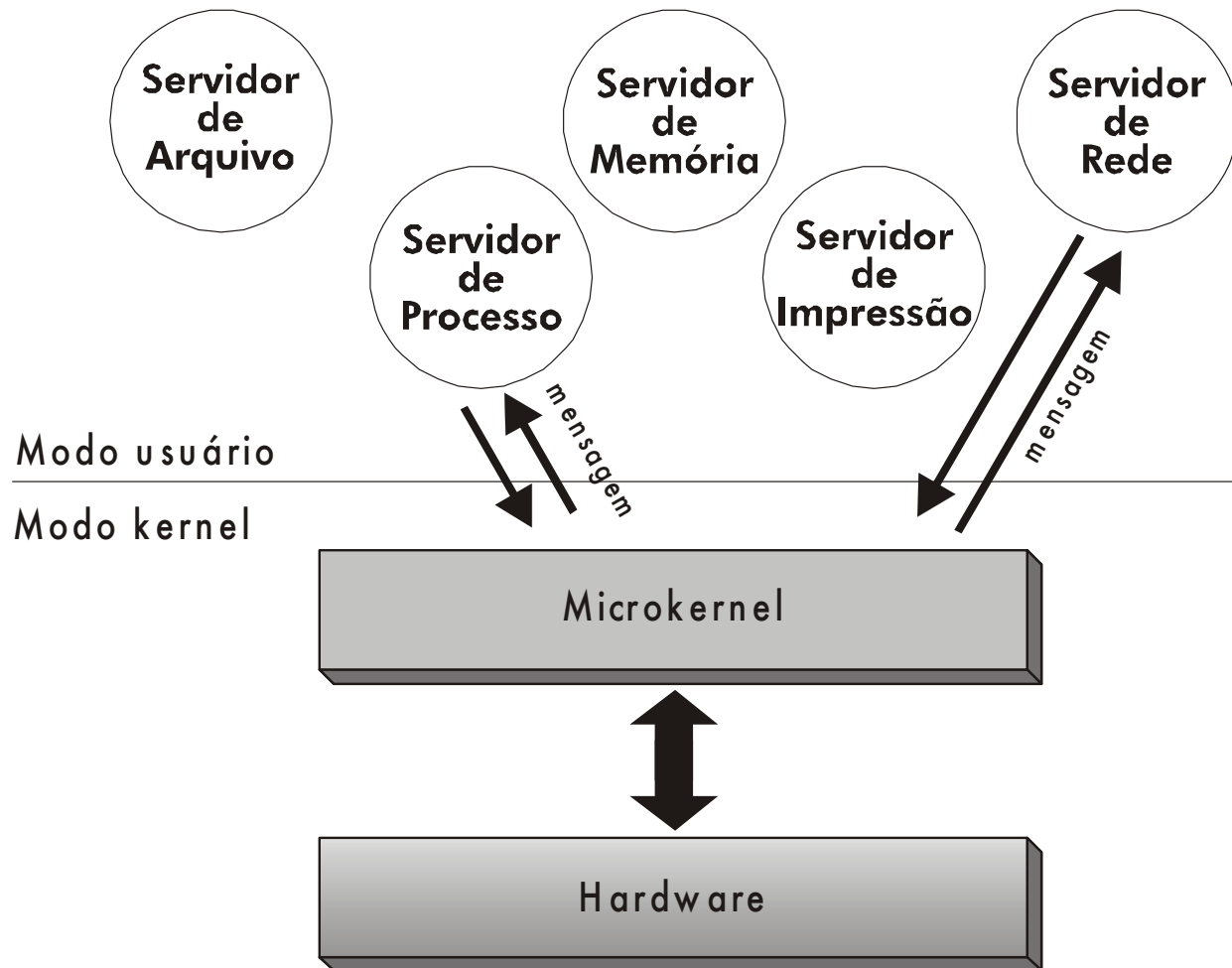


Introdução

Estrutura do MicroKernel

- **MicroNúcleo (microkernel): incorpora somente as funções de baixo nível mais vitais**
- **O microkernel fornece uma base sobre a qual é contruído o resto do S.O.**
- **A maioria destes sistemas são construídos como coleções de processos concorrentes**
- **Fornece serviços de alocação de UCP e de comunicação aos processos**

Estrutura do MicroKernel





S.O. - Arquiteturas

Máquinas virtuais

Uma máquina virtual é definida como “uma duplicata eficiente e isolada de uma máquina real”.



S.O. - Arquiteturas

Máquinas virtuais

Em uma máquina real, uma camada de software de baixo nível (por exemplo, a BIOS) fornece acesso aos vários recursos do hardware para o sistema operacional, que os disponibiliza de forma abstrata às aplicações. Em uma máquina real, quando o sistema operacional acessa os dispositivos de hardware, ele faz uso dos drivers respectivos, que interagem diretamente com a memória e os dispositivos da máquina.



S.O. - Arquiteturas

Máquinas virtuais

Um emulador é o oposto da máquina real. O emulador implementa todas as instruções realizadas pela máquina real em um ambiente abstrato, possibilitando executar um aplicativo de uma plataforma em outra, por exemplo, um aplicativo do Windows executando no Linux.



S.O. - Arquiteturas

Máquinas virtuais

Um emulador perde muito em eficiência ao traduzir cada instrução da máquina real. Além disso, emuladores são bastante complexos, pois geralmente necessitam simular a quase totalidade das instruções do processador e demais características do hardware que os circundam .



S.O. - Arquiteturas

Máquinas virtuais

A funcionalidade e o nível de abstração de uma máquina virtual encontra-se entre uma máquina real e um emulador, na medida em que abstrai somente os recursos de hardware e de controle usados pelas aplicações.



S.O. - Arquiteturas

Máquinas virtuais

Uma máquina virtual é um ambiente criado por um monitor de máquinas virtuais, também denominado “sistema operacional para sistemas operacionais”. O monitor pode criar uma ou mais máquinas virtuais sobre uma única máquina real.



S.O. - Arquiteturas

Máquinas virtuais

Enquanto um emulador fornece uma camada de abstração completa entre o sistema em execução e o hardware, um monitor abstrai o hardware subjacente e controla uma ou mais máquinas virtuais. Cada máquina virtual fornece facilidades para uma aplicação ou um “sistema convidado” que acredita estar executando sobre um ambiente normal com acesso físico ao hardware.

Máquina virtual

- O Modelo de Máquina Virtual ou *Virtual Machine* (VM), cria um nível intermediário entre o hardware e o S.O., denominado Gerência de Máquinas Virtuais.
- Este nível cria diversas máquinas virtuais independentes, onde cada uma oferece uma cópia virtual do hardware, incluindo modos de acesso, interrupções, dispositivos de E/S, etc.
- Como cada VM é independente das demais, é possível que tenha seu próprio S.O.



S.O. - Arquiteturas

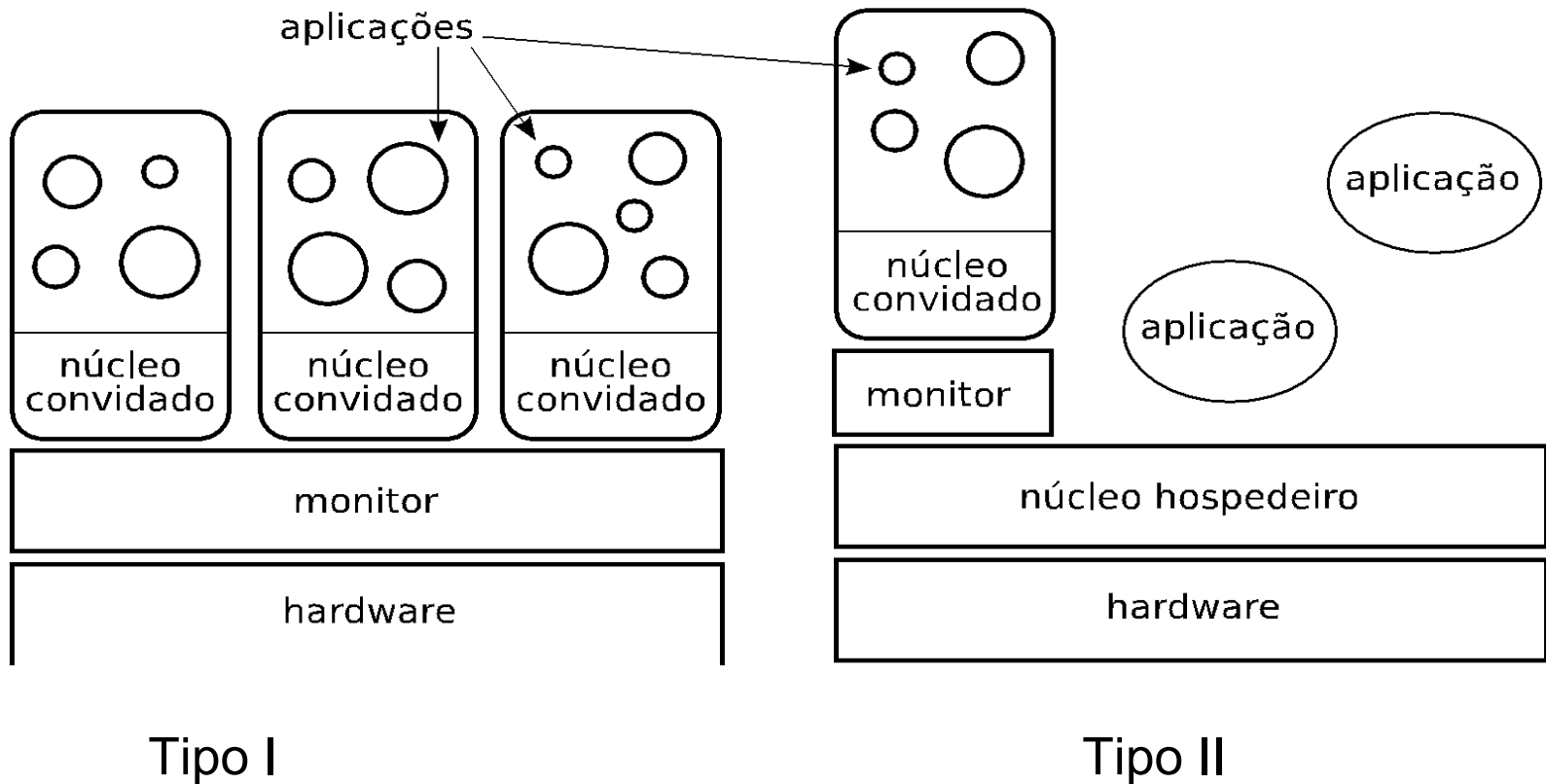
Máquinas virtuais

Existem basicamente duas abordagens para a construção de sistemas de máquinas virtuais:

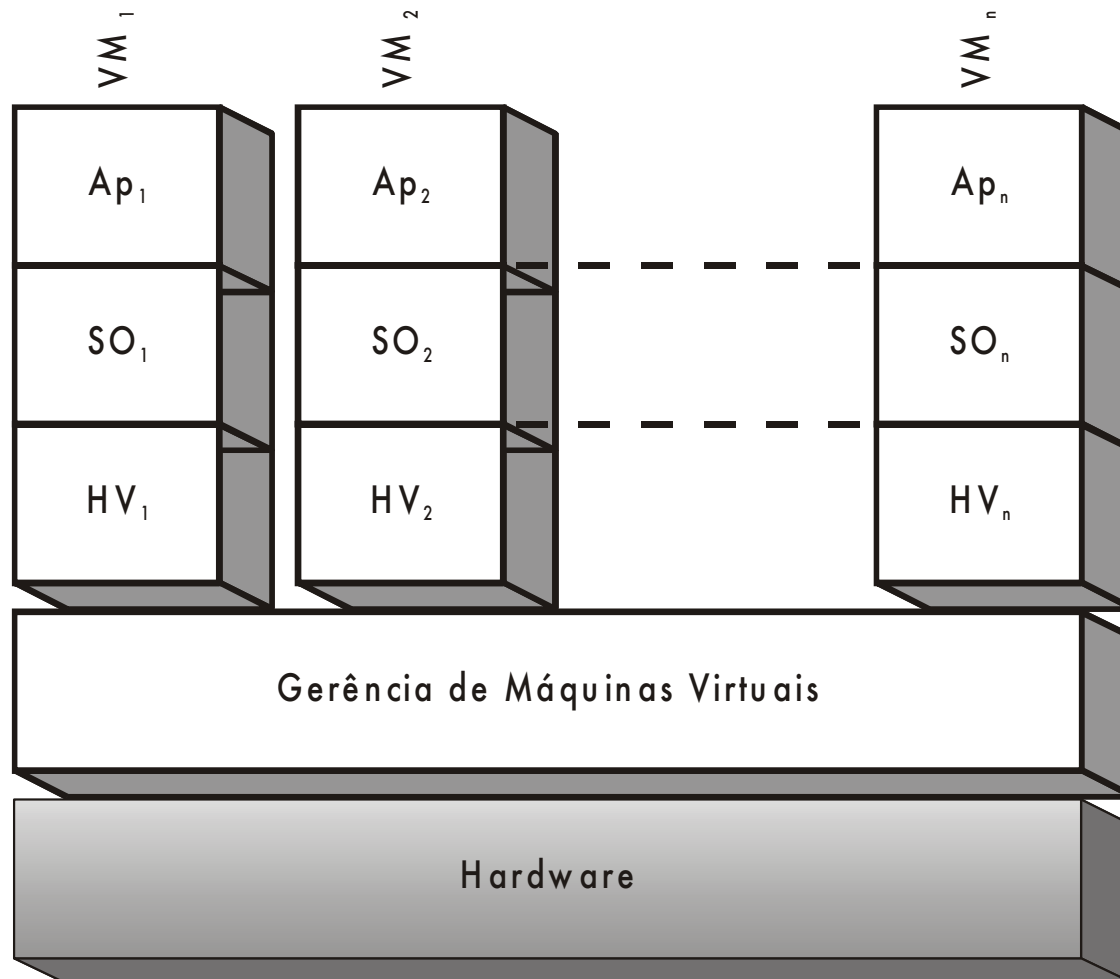
- O tipo I, onde o monitor de máquinas virtuais é implementado entre o hardware e os sistemas convidados.
- O tipo II, onde o monitor é implementado como um processo de um sistema operacional real subjacente, denominado sistema anfitrião ou sistema hospedeiro.

S.O. - Arquiteturas

Máquinas virtuais



Máquina virtual





S.O. - Arquiteturas

Máquinas virtuais

Vantagens:

- **Aperfeiçoamento e testes de novos sistemas operacionais;**
- **Ensino prático de sistemas operacionais e programação de baixo nível;**
- **Executar diferentes sistemas operacionais sobre o mesmo hardware, simultaneamente;**
- **Simular configurações e situações diferentes do mundo real, como por exemplo, mais memória disponível, outros dispositivos de E/S;**
- **Simular alterações e falhas no hardware para testes ou reconfiguração de um sistema operacional, provendo confiabilidade e escalabilidade para as aplicações;**
- **Garantir a portabilidade das aplicações legadas (que executariam sobre uma VM simulando o sistema operacional original);**
- **Desenvolvimento de novas aplicações para diversas plataformas, garantindo a portabilidade destas aplicações;**
- **Diminuir custos com hardware.**



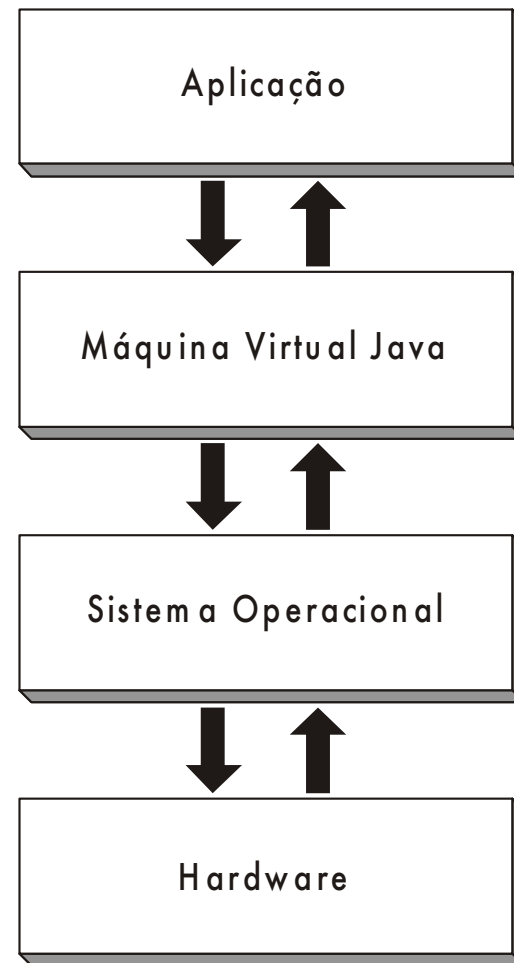
S.O. - Arquiteturas

Máquinas virtuais

Desvantagens:

A principal desvantagem do uso de máquinas virtuais é o custo adicional de execução dos processos na máquina virtual em comparação com a máquina real. Esse custo é muito variável, podendo passar de 50% em plataformas sem suporte de hardware à virtualização. Todavia, pesquisas recentes têm obtido a redução desse custo a patamares abaixo de 20%, graças sobretudo a ajustes no código do sistema hospedeiro. Esse problema não existe em ambientes cujo hardware suporta o conceito de virtualização, como é o caso dos mainframes.

Um outro exemplo de utilização desta estrutura ocorre na linguagem Java. Para executar um programa Java é necessário uma máquina virtual Java (*Java Virtual Machine* – JVM)



Introdução

Um Breve Histórico

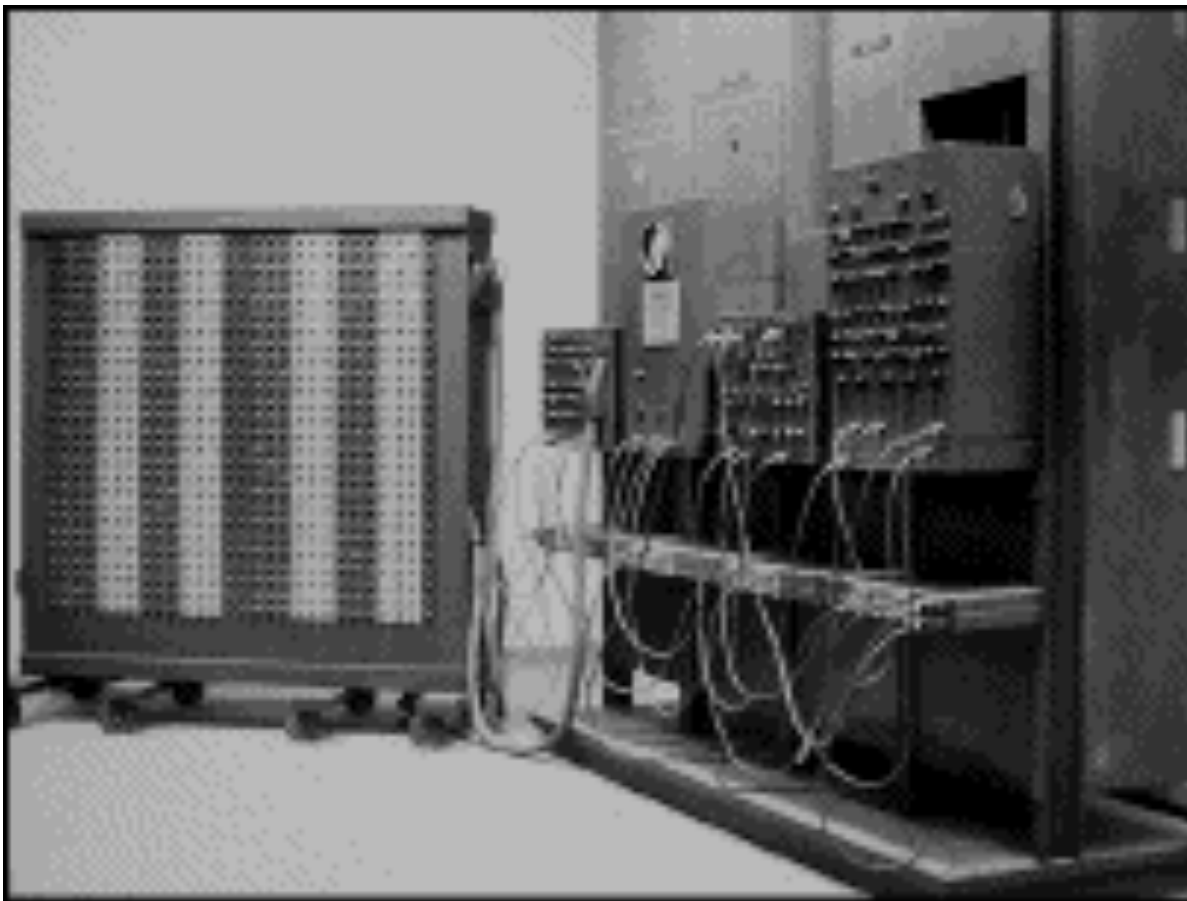
■ 1a. Geração de Computadores (1945 - 1955)

- Computadores à Válvula
- Ausência de um S.O.: a programação era feita diretamente em linguagem de máquina



Colossus Mark I

Introdução



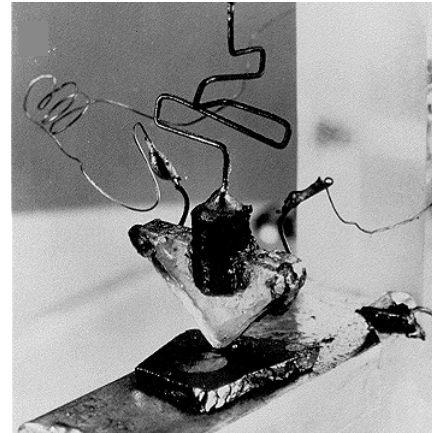
ENIAC

Introdução

Um Breve Histórico

■ 2a. Geração de Computadores (1955 - 1965)

- Invenção do Transistor (William Shockley, John Bardeen, e Walter Brattain)

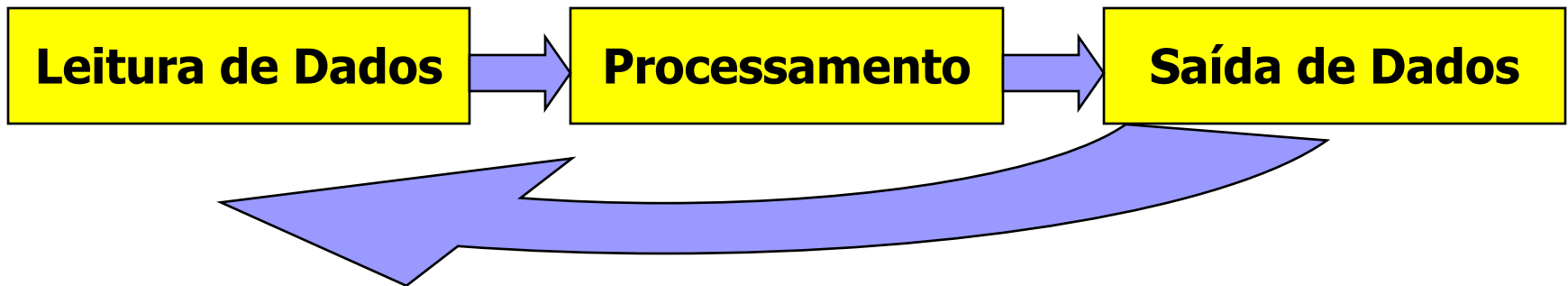


- Uso da linguagem Assembly e FORTRAN
- Soss do tipo lote (batch)

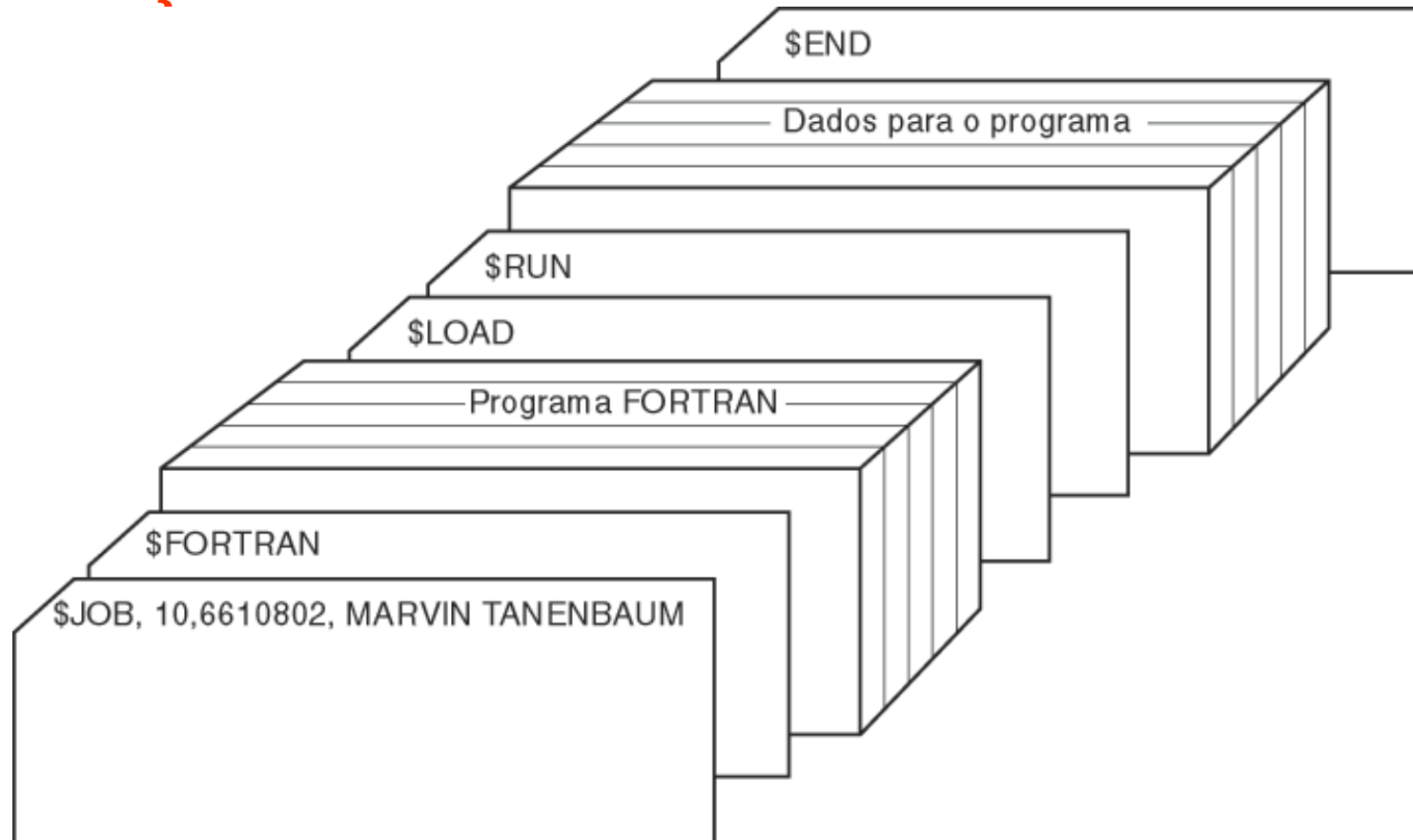
Introdução

Um Breve Histórico

- Até 1956:



Introdução

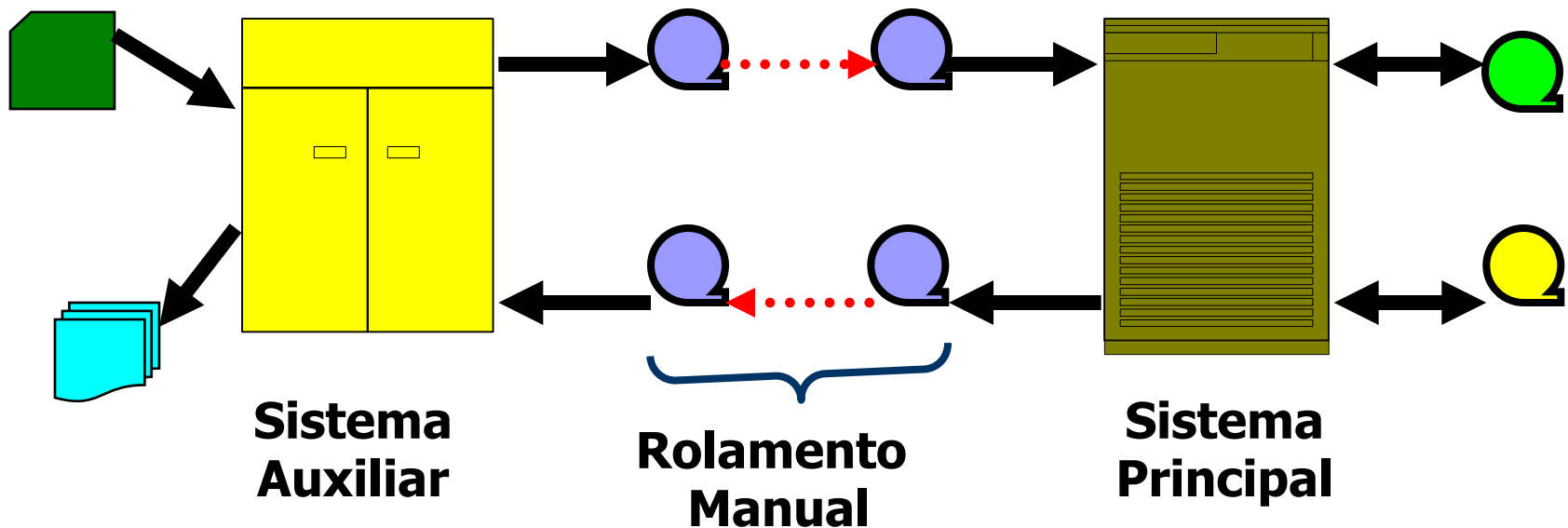


Estrutura de um job FMS típico – 2a. geração

Introdução

Um Breve Histórico

- **1957:** uso de sistema auxiliar (técnica do spooling)



Introdução

Um Breve Histórico

■ 1959:

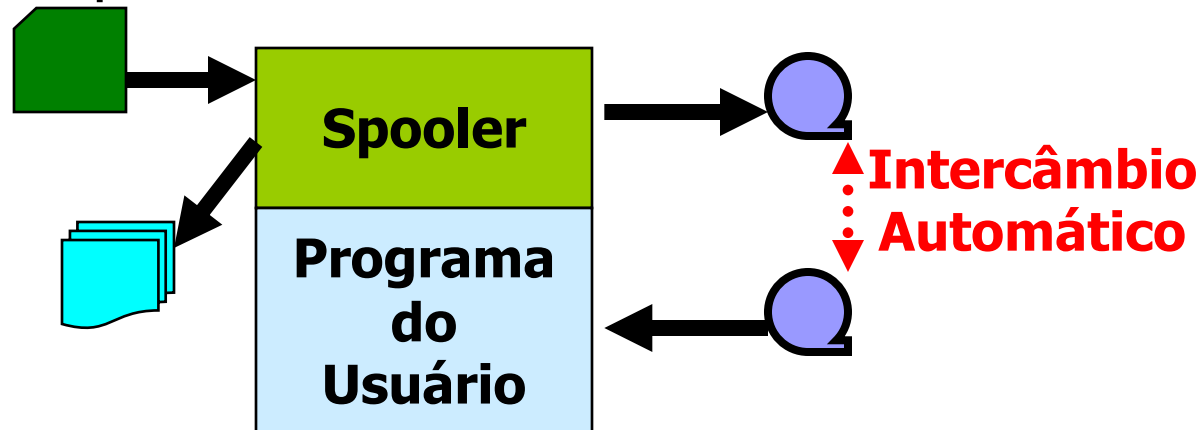
- Introdução de canais autônomos de Entrada/Saída
- Criação das Interrupções
- **Entrada/Saída em paralelo com o cálculo**

Introdução

Um Breve Histórico

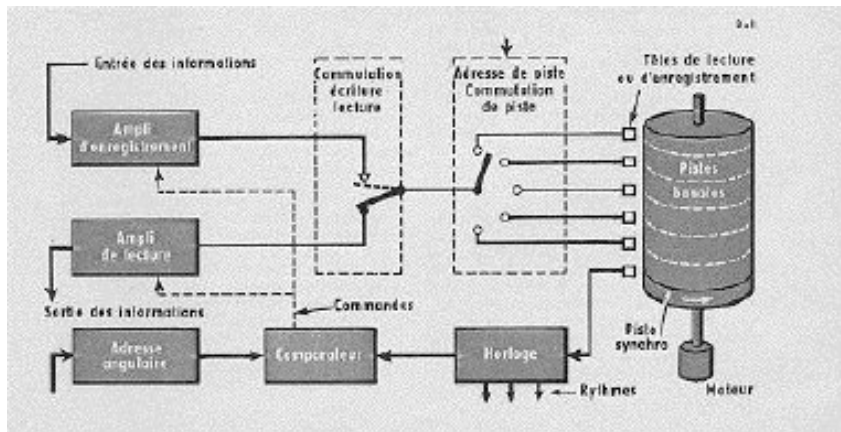
■ 1960:

- Uso de Spooler automático

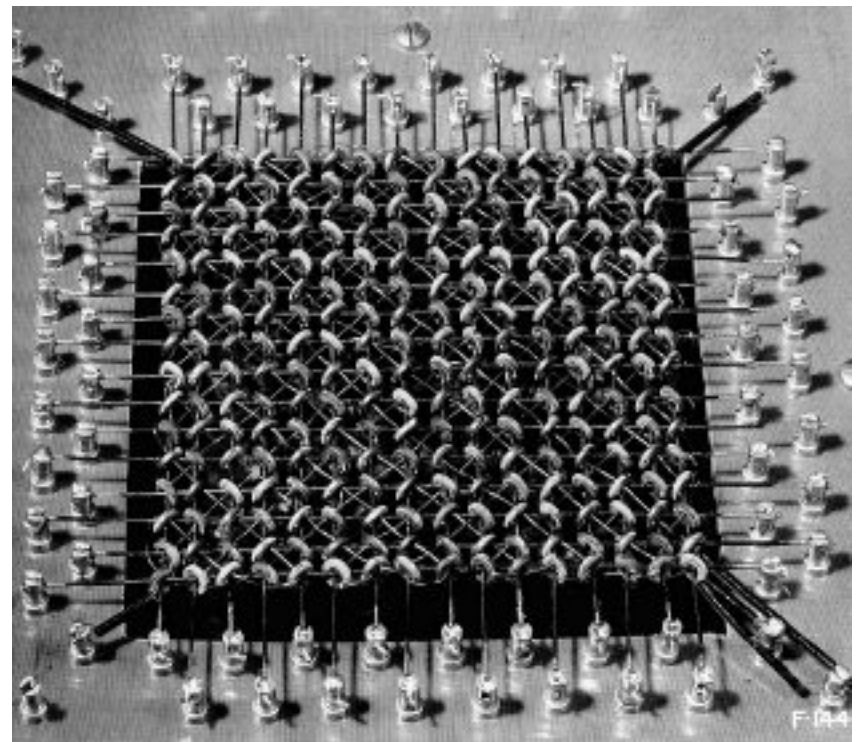


- Invenção dos discos e tambores magnéticos
- S.O.s Típicos: FMS (Fortran Monitor System) e IBSYS (da IBM)

Exemplos de tecnologia de armazenamento da 2a. geração



Tambor Magnético



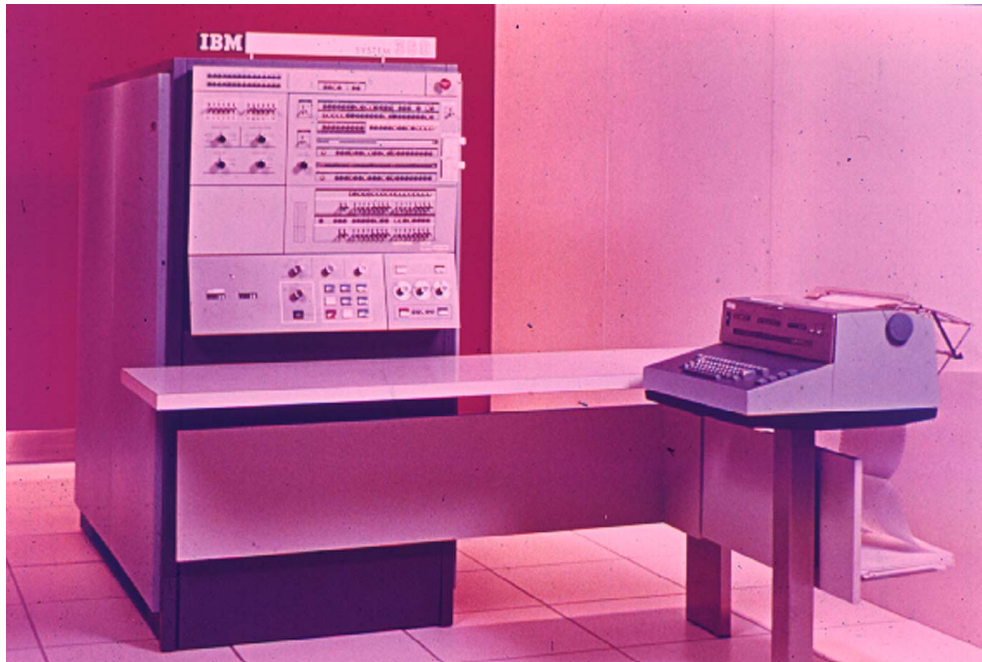
Memória de Ferrite

Introdução

Um Breve Histórico

■ 3a. Geração de Computadores (1965 - 1980)

- Invenção dos Circuitos Integrados (chips) com baixa escala de integração (SSI - Small Scale Integration)
- Sistema OS/360 (IBM): 1o. a usar circuitos SSI



Introdução

Um Breve Histórico

- Introdução dos conceitos de **Multiprogramação** e **Time Sharing**
- Proliferação dos mini-computadores (ex. PDP-1)
- Sistema **MULTICS** (**MULT**iplexed **I**nformation and **C**omputing **S**ervice)

Introdução



**Sistema
GE 625**

**(SO
Multics)**

Introdução

Um Breve Histórico

■ 4a. Geração de Computadores (1980 - Hoje)

- **Invenção dos Circuitos Integrados com alta escala de integração (LSI - Large Scale Integration)**
- **Sistemas Operacionais para Microcomputadores**
 - **CP/M (8 bits)**
 - **DOS (16 bits)**
 - **UNIX (32 bits)...**
- **Sistemas Operacionais de Rede**
- **Sistemas Operacionais Distribuídos**