Conceitos básicos de sistemas operacionais

Vamos tratar da conceituação dos elementos de um sistema operacional, desde a sua evolução histórica, passando pela estrutura básica, até a exemplificação da instalação e utilização do sistema operacional Linux, sob o ponto de vista do usuário.

Prof. Diogo Tavares Robaina

Objetivos

- Descrever a evolução histórica dos sistemas operacionais.
- Identificar os tipos de sistemas operacionais.
- Compreender a estrutura do SO: kernel, system calls, modos de acesso.
- Analisar a arquitetura, instalação do Linux e comandos básicos.

Introdução

Qualquer sistema computacional é composto pelo conjunto de usuários, hardware e software. Os softwares que são utilizados podem ser separados em aplicativos que são disponibilizados pelos usuários e os sistemas operacionais. Os sistemas operacionais (SO) oferecem uma interface entre os aplicativos e o hardware, tornando a vida dos usuários e dos desenvolvedores mais simples. Além disso, o sistema operacional permite um uso mais eficiente e eficaz do hardware.

Para conseguirmos reconhecer o papel do sistema operacional, iremos descobrir como eles surgiram e evoluíram junto aos próprios sistemas computacionais. Iremos aprender os tipos de sistemas operacionais, bem como compreender a estrutura básica do sistema operacional, composto, entre outros elementos, do kernel, chamadas de sistemas e modos de acesso ao núcleo. Por fim, vamos verificar como é a utilização básica de um sistema operacional, aplicando conceitos ao uso do SO Linux.

No vídeo a seguir, conheceremos os componentes de um sistema computacional, destacando a função dos sistemas operacionais como interface entre aplicativos e hardware. Também exploraremos a evolução dos SO, seus tipos, a estrutura básica, incluindo kernel e chamadas de sistema, e a utilização prática do SO Linux, aplicando conceitos fundamentais.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Conceitos

O conhecimento teórico sobre sistemas operacionais é fundamental para qualquer profissional de computação, pois proporciona uma compreensão profunda sobre como o hardware e o software interagem para realizar tarefas complexas. Com uma base sólida em conceitos de sistemas operacionais, os profissionais podem otimizar o uso dos recursos de hardware, garantir a segurança e a eficiência das operações e desenvolver soluções robustas para problemas computacionais.

Neste vídeo, conheceremos os componentes de um sistema computacional, a evolução histórica dos sistemas operacionais, seus tipos, e a estrutura básica incluindo kernel e chamadas de sistema. Também veremos a utilização prática do SO Linux e conceitos fundamentais.

Conteúdo interativo

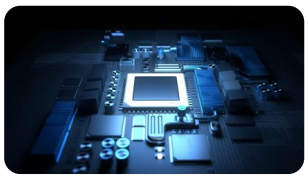
Acesse a versão digital para assistir ao vídeo.

Estamos prestes a realizar o nosso estudo em sistemas operacionais. Esse assunto é importante para todos os profissionais que buscam exercer plenamente suas atividades nas diversas áreas da computação, como administradores de sistemas, programadores de aplicações concorrentes, gerentes de segurança e administradores de rede (devido aos sistemas operacionais nas redes de comunicação de dados).

Vamos começar verificando quais são os componentes básicos de um sistema computacional (ou sistema de computação ou SC):

Hardware

Fornece recursos básicos de computação CPU, memória, dispositivos de E/S.



Aplicativos

Definem as maneiras como os recursos são usados, para resolver os problemas de computação dos usuários, como compiladores, banco de dados, jogos de videogames, programas comerciais e outros.

Usuários

São as pessoas, máquinas ou outros computadores.

Sistema operacional

Controla e coordena o uso do hardware entre os vários programas de aplicação, para os diversos usuários.

O sistema computacional é um sistema complexo demais. Não apenas para ser entendido nos mínimos detalhes, como também para realizar a gerência de todos os seus componentes e usá-los de modo otimizado. Por esse motivo é que os computadores possuem um software denominado **sistema operacional**.

Uma das definições possíveis para um sistema operacional (SO ou S.O.) é ser constituído por um conjunto de rotinas de computação elaborado para propósitos específicos, de forma semelhante com o que ocorre com um aplicativo de computador que utilizamos no dia a dia, por exemplo, uma planilha eletrônica.

> Uma particularidade do sistema operacional é que ele se diferencia de um aplicativo de usuário ao atuar como um intermediário entre o usuário e o hardware de um computador, tornando a utilização deste mais simples, rápida e segura.

Você já deve ter utilizado sistemas operacionais tais como o Windows, o Linux, o Mac OS X ou o Android, mas as aparências podem enganar:

Atenção

O programa que serve para a interação dos usuários na realidade não faz parte do sistema operacional em si, embora use o SO para realizar seu trabalho. Na realidade, o que usuário utiliza diretamente é uma interface de acesso ao sistema operacional. Essa interface pode ser baseada em texto (shell, ou interpretador de comandos), ou baseada em interface gráfica com ícones (GUI - Graphical User Interface).

Em um computador, os programas podem ser executados em modo usuário ou kernel. Conheça a diferença entre eles:

Modo usuário

Os softwares têm acesso limitado ao hardware e normalmente os programas e aplicativos são utilizados diretamente pelos usuários.

Modo kernel

O SO é o único programa executado em modo núcleo, ou kernel. Significa que o SO possui o acesso completo ao hardware e consegue executar qualquer instrução possível.

Além disso, o sistema operacional é um **programa de controle que coordena** a execução dos programas do usuário e as operações dos dispositivos de E/S (entrada e saída, ou periféricos); e também é um **gerenciador de recursos** de hardware, que gerencia e aloca as partes de todo um sistema complexo.

Comentário

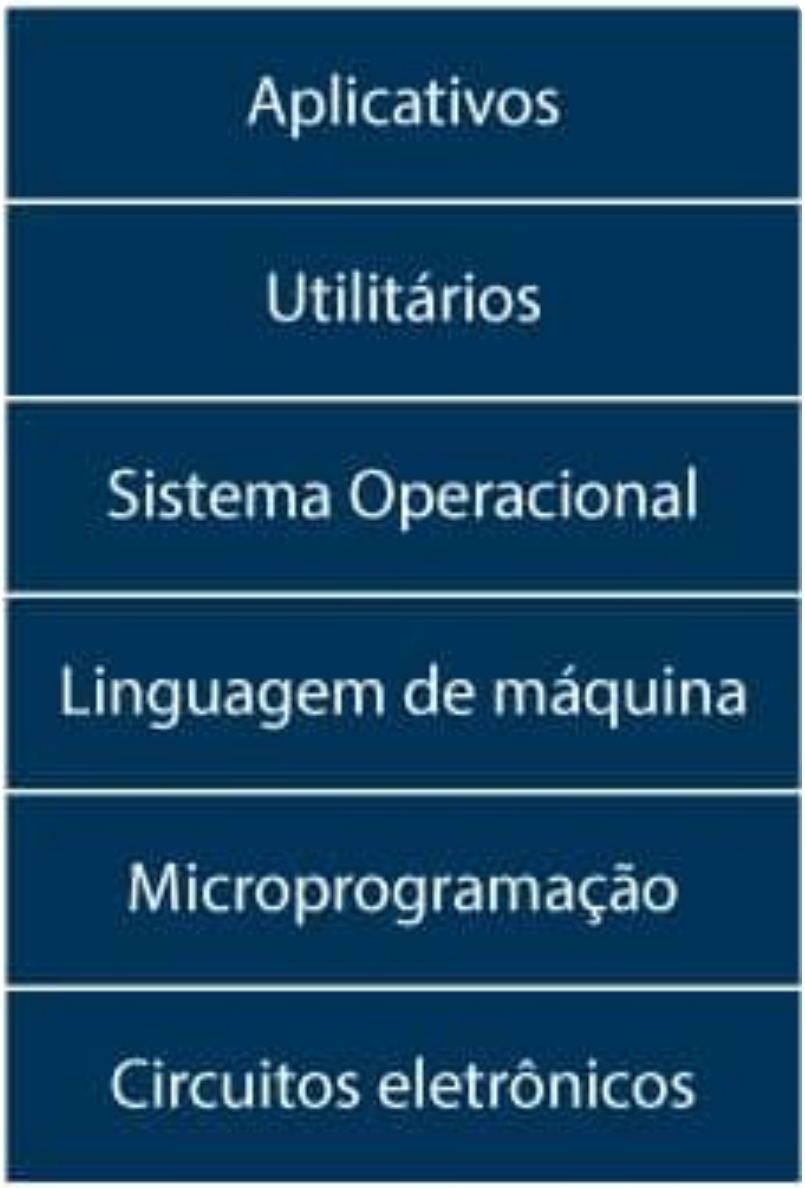
Nos primeiros computadores, a programação era toda feita com painéis físicos, exigindo do programador um grande conhecimento do hardware, mas o hardware em si possuía pouca utilidade para o usuário devido à sua complexidade. O surgimento do sistema operacional permitiu que o hardware pudesse ser utilizado de forma mais eficiente, pois o SO consegue disponibilizar os serviços aos usuários com mais eficiência, modularizando e abstraindo a visão do usuário.

O uso de softwares, incluindo o sistema operacional, possibilita oferecer ao usuário uma visão daquilo que interessa a ele, ou seja, o software modulariza aquilo que o usuário vê e usa.

O uso de um SO também permite **abstrair** não apenas o hardware como também rotinas de outros softwares, ou seja, é possível representar o funcionamento daquilo que está "abaixo" de um determinado aplicativo ou sistema operacional.

Essas visões do usuário, abaixo ou acima, que representam conjuntos de serviços ou funções, podem ser representadas em um modelo chamado **máquina de níveis**, ou **máquina de camadas**, conforme mostrado na imagem a seguir.

Particularmente, a abstração do hardware, ou seja, qualquer camada acima do hardware, pode ser chamada de **máquina virtual**.



Representação da máquina de níveis.

Atividade 1

Em um sistema computacional, o sistema operacional é essencial para o gerenciamento dos recursos e para a interação entre o hardware e os programas de aplicação. Ele garante que os diversos componentes do sistema funcionem de forma harmoniosa e eficiente, proporcionando aos usuários uma experiência integrada e

otimizada. Qual das afirmações a seguir descreve corretamente a função do sistema operacional em um sistema computacional?

A

O sistema operacional serve apenas para gerenciar a memória do computador.

B

O sistema operacional é responsável por controlar e coordenar o uso do hardware entre os vários programas de aplicação.

C

O sistema operacional funciona apenas como um aplicativo de usuário interconectando os vários programas de aplicação.

D

O sistema operacional não tem acesso ao hardware do computador facilitando o controle de vírus e softwares piratas.

E

O sistema operacional é utilizado apenas para acessar a internet.

A alternativa B está correta.

A função principal é atuar como um intermediário entre o hardware do computador e os programas de aplicação, gerenciando e coordenando o uso dos recursos de hardware de maneira eficiente. Ele garante que os diferentes programas possam executar suas tarefas de forma otimizada e segura, proporcionando uma interface que facilita a interação entre os usuários e o sistema computacional.

Histórico dos sistemas computacionais

Para entendermos como a computação chegou ao estágio atual, precisamos estudar o conhecimento teórico sobre a evolução dos sistemas operacionais. Esse estudo proporciona insights sobre as tecnologias e conceitos que moldaram o desenvolvimento dos sistemas operacionais ao longo das gerações, permitindo que os profissionais da área aproveitem ao máximo os recursos disponíveis hoje.

Neste vídeo, conheceremos a evolução histórica dos sistemas operacionais desde a primeira até a quarta geração, destacando os avanços tecnológicos e inovações em hardware e software, como o surgimento dos transistores, circuitos integrados, e sistemas de multiprocessamento e redes distribuídas.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

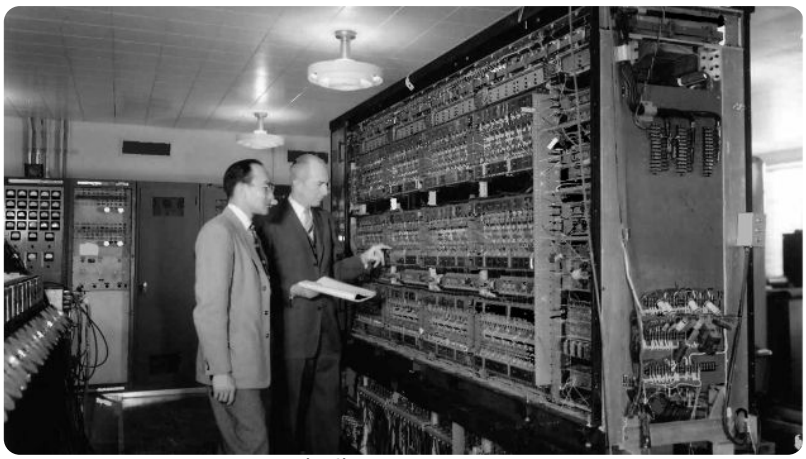
Um sistema operacional está intimamente ligado ao hardware do computador no qual ele é executado, estendendo o conjunto de instruções do computador e gerenciando seus recursos. Assim, o sistema

operacional deve possuir grande conhecimento sobre o hardware, ao menos a partir do ponto de vista do programador.

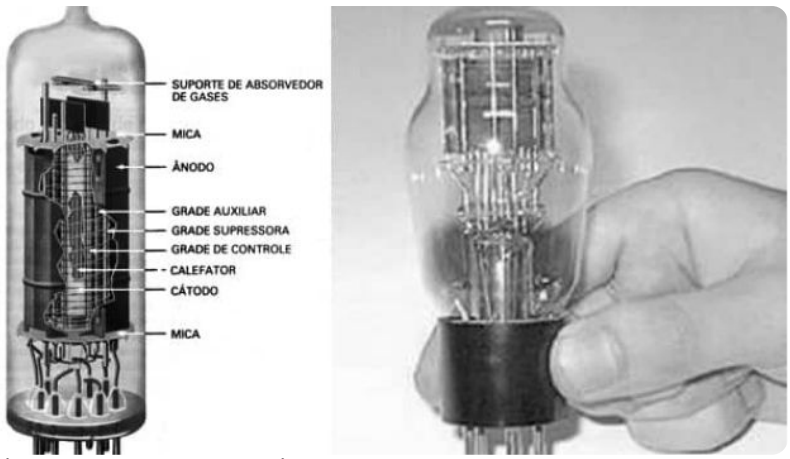
Os sistemas operacionais evoluíram gradualmente, e o processo temporal dessa evolução pode ser dividido em fases. A história dos sistemas operacionais é bastante relacionada à arquitetura de computadores. Sendo assim, veremos brevemente as várias gerações de computadores com o objetivo de entender as primeiras versões de sistemas operacionais.

Primeira geração

O período entre aproximadamente 1945 até 1955 corresponde ao surgimento da primeira geração dos computadores. Nessa época, a programação era feita em painéis de programação, e os componentes de hardware eram mais primitivos, por exemplo, empregando válvulas na composição dos circuitos lógicos.



Programação realizada em painéis físicos.



Válvula utilizada nos circuitos lógicos.

Segunda geração

Na segunda geração de computadores, compreendida entre aproximadamente 1955 até 1965, houve a adoção dos transistores no lugar das válvulas para a composição dos circuitos lógicos, e a aplicação de sistemas de computação em lote, como exemplificado na imagem a seguir:

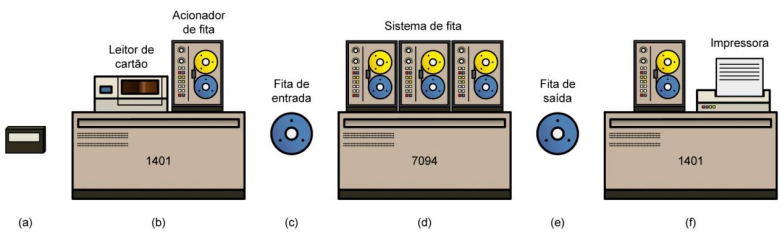


Imagem ilustrativa do sistema em lote (batch) da segunda geração.

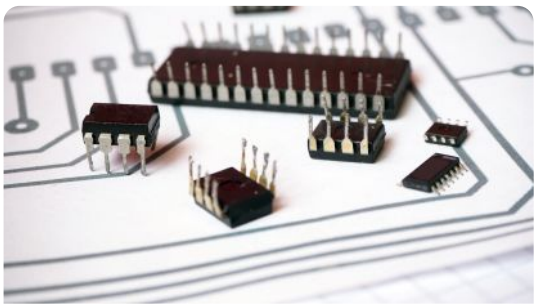
As etapas da imagem representam o seguinte:

- (a) Os programadores levavam os cartões para o leitor de cartões.
- (b) O leitor de cartões gravava o lote de tarefas em fita.
- (c) O operador levava a fita para o computador (7094) processar as informações.
- (d) O computador executava o processamento.
- (e) O operador levava a fita de saída para um dispositivo capaz de ler o conteúdo e enviar para a impressora.
- (f) A impressora gerava as saídas.

Terceira geração

Foi na terceira geração, entre 1965 a 1980, que o conceito de sistema operacional foi plenamente estabelecido, trazendo inovações como multiprocessamento, multiprogramação, time-sharing, spooling e memória virtual.

Foi nessa geração que surgiram também os circuitos integrados (CIs) e o sistema operacional UNIX.



Circuitos integrados no desenvolvimento de hardware.

Quarta geração

Pode-se dizer que estamos na quarta geração de computadores, iniciada em 1980 e durando até o momento no qual este material foi escrito. Entretanto, alguns autores podem chamar os sistemas atuais de quinta geração.

Nesse período, surgiram uma série de novidades e eventos relacionados aos computadores e aos sistemas operacionais, alguns listados a seguir:

- Surgimento dos computadores pessoais
- Integração em larga escala (LSI e VLSI) – Miniaturização e barateamento dos componentes eletrônicos
- Intel produz seu primeiro microprocessador – Intel 4004
- Intel 8080 – Primeiro microcomputador
- 1976: Apple II (8 bits)
- Surgimento da Microsoft
- Redes distribuídas
- Protocolos de redes
- Redes locais
- SO intimamente relacionados com o software de rede
- Surgimento da linguagem Pascal e C
- Surgimento do IBM PC – Intel 8088 de 16 bits com DOS (Disk Operating System)
- Sistemas multiusuários foram impulsionados
- Protocolos da internet TCP/IP
- Surgimento das estações de trabalho
- 1982: Surgimento da Sun Microsystems e das primeiras estações RISC
- Surgimento das interfaces gráficas
- Avanços de hardware, software e telecomunicações
- Processadores e dispositivos de E/S mais rápidos e menores
- Integração em ultra larga escala (ULSI)
- Internet: Problemas de gerência, segurança e desempenho
- Arquitetura Cliente – Servidor
- Software aberto (Open Source)
- Demanda cada vez maior de processamento
- Arquiteturas paralelas
- Processamento distribuído
- SO em celulares, tablets e outros

Tabela 1: Lista de inovações referentes a computadores e sistemas operacionais da quarta geração.
Fabio Henrique Silva.

Atividade 2

A evolução dos computadores e sistemas operacionais passou por várias gerações, cada uma marcada por inovações tecnológicas significativas. Entender os eventos que marcaram essas transições ajuda a compreender como chegamos aos sistemas avançados que usamos atualmente. Nesse contexto, qual evento marcou a transição da segunda para a terceira geração de computadores e sistemas operacionais?

A

Introdução dos computadores pessoais a partir do desenvolvimento de microprocessadores.

B

Surgimento dos transistores para substituir válvulas.

C

Estabelecimento do conceito de multiprocessamento e multiprogramação.

D

Desenvolvimento do primeiro microprocessador pela Intel.

E

Criação da linguagem de programação C.

A alternativa C está correta.

O estabelecimento desses conceitos permitiu que os sistemas operacionais gerenciassem eficientemente múltiplos processos de forma simultânea, aumentando a eficiência e a capacidade de utilização dos recursos de hardware. Essa fase também viu a introdução de circuitos integrados e o desenvolvimento do sistema operacional UNIX, que teve um impacto significativo na evolução dos sistemas operacionais modernos.

MS Windows e Unix

A história e a evolução dos sistemas operacionais Windows e Unix são fundamentais para a formação de qualquer profissional de TI. Esses sistemas tiveram um impacto significativo no desenvolvimento da tecnologia de computação, influenciando a forma como os sistemas operacionais modernos foram concebidos e implementados. Conhecer suas origens, características e evoluções permite uma melhor compreensão das tecnologias que utilizamos hoje.

Neste vídeo, apresentamos a história e a evolução dos sistemas operacionais Windows e Unix, desde suas origens até as versões modernas, destacando as características principais de cada sistema, suas diferenças, e como ambos influenciaram a tecnologia de computação atual.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Historicamente, há diversos tipos e denominações de sistemas operacionais. Dentre esses vários, dois se destacam por serem bastante conhecidos e utilizados, e por ainda exercerem uma grande influência na evolução de todo o contexto tecnológico relacionado aos computadores: Windows e Unix.

Windows

Vamos, a seguir, conhecer a história do Windows, um dos sistemas operacionais mais conhecidos:

1

MS-DOS

A história do Windows começa com o MS-DOS (Disk Operating System), um sistema operacional de 16 bits monoprogramável, monousuário e com interface de linha de comando, lançado pela Microsoft em 1981 para o IBM-PC.

2MS Windows 1.0

Em 1985, foi lançado o MS Windows 1.0, introduzindo uma interface gráfica para o MS-DOS. As versões seguintes continuaram mantendo o MS-DOS como núcleo de SO (tais como Windows 3.0, Windows 95, Windows 98 e Windows Me).

3

Windows NT

Paralelamente às versões baseadas no MS-DOS, o Windows NT foi lançado em 1993. Com sistema de 32 bits, possuía multitarefa preemptiva, multithread (SO executa threads (tarefas) de forma simultânea, sem uma interferir na outra), memória virtual e suporte a múltiplos processadores simétricos, nas versões desktop e servidores. Possuía um núcleo completamente novo, mas com compatibilidade parcial com aplicações legadas dos sistemas MS-DOS, além de mesma interface gráfica das versões existentes.

4

Windows 2000

O Windows 2000 foi uma evolução do Windows NT 4. Sua versão utilizava a mesma arquitetura interna do Windows NT4, apresentando, entretanto, algumas novas funções, tais como o plug and play e o serviço de diretórios Active Directory.

5

Windows XP

O Windows XP foi lançado no ano de 2001. A partir do seu lançamento, a Microsoft começou a descontinuar as famílias DOS-Windows e Windows NT/2000, integrando as duas linhas de sistemas operacionais.

6

Outras versões

Desde então, outras versões foram lançadas direcionadas para desktops (Windows Vista, Windows 7, Windows 8, Windows 10) e servidores (Windows Server 2003, Windows Server 2008, Windows Server 2012, Windows Server 2016, Windows Server 2019).

Unix

O Unix veio de outra vertente dos sistemas operacionais. Como já comentado anteriormente, uma das características da geração de computadores existentes na década de 1960 eram os sistemas batch. Acompanhe a seguir a sua história:

1965

O MIT, a Bell Labs e a General Electric se uniram para desenvolver o MULTICS (MULTiplexed Information and Computing Service), um esforço para desenvolver um verdadeiro sistema operacional de tempo compartilhado.

1969

Após a Bell Labs se retirar do projeto, Ken Thompson, pesquisador envolvido no projeto do MULTICS, desenvolveu sua própria versão em Assembly para um minicomputador PDP-7, chamada de UNICS (UNiplexed Information and Computing Service) e, mais para frente, de Unix. Para torná-lo mais fácil de ser portado para outras plataformas, Thompson desenvolveu a linguagem B e reescreveu o código do sistema nessa nova linguagem. Thompson e Dennis Ritchie evoluíram a linguagem, criando o C.

1973

O Unix foi reescrito em C e portado para um minicomputador PDP-11. Depois disso várias universidades começaram a licenciar o Unix, como a Universidade de Berkeley, na Califórnia, que desenvolveu sua própria versão do sistema, chamada BSD (Berkeley Software Distribution), com várias melhorias, tais como a memória virtual, C shell, Fast File System, sockets, e o protocolo TCP/IP.

1991

O finlandês Linus Torvalds iniciou o desenvolvimento do Linux, baseado no Minix, que correspondia a uma variante do Unix. O Linux foi desenvolvido pelo professor Andrew Tanenbaum, da Holanda, com finalidade educacional.

O Linux começou a evoluir com a ajuda de vários programadores voluntários. Assim, várias versões do Unix podem ser encontradas.

Comentário

A mais importante tentativa de unificação das diversas versões do Unix foi feita pelo IEEE através do seu comitê POSIX (Portable Operating System Unix), que resultou em uma biblioteca padrão de chamadas e um conjunto de utilitários que todo sistema Unix deveria oferecer.

Atividade 3

A evolução dos sistemas operacionais é marcada por várias inovações tecnológicas que moldaram o uso e o desenvolvimento de software ao longo do tempo. O Unix, em particular, foi muito importante nessa evolução, introduzindo conceitos que ainda são relevantes e influentes nos sistemas operacionais modernos. Qual das seguintes afirmações descreve corretamente a evolução do sistema operacional Unix?

A

O Unix foi desenvolvido pela Microsoft como sucessor do MS-DOS.

B

O Unix foi inicialmente desenvolvido em Assembly e depois reescrito em C.

C

O Unix sempre foi um sistema monoprogramável e monousuário.

D

O Unix não influenciou o desenvolvimento de outros sistemas operacionais.

E

O Unix foi desenvolvido exclusivamente para computadores pessoais.

A alternativa B está correta.

O sistema operacional Unix foi inicialmente desenvolvido por Ken Thompson em Assembly para um minicomputador PDP-7 e mais tarde reescrito em C para facilitar sua portabilidade e manutenção. Esse desenvolvimento permitiu que o Unix fosse facilmente portado para diversas plataformas, influenciando significativamente outros sistemas operacionais, incluindo o Linux. O Unix introduziu conceitos importantes como multitarefa e suporte a múltiplos usuários, e sua evolução resultou em diversas variantes amplamente utilizadas em diferentes contextos de computação.

Tipos de sistemas operacionais: características

Cada sistema operacional possui características únicas que influenciam diretamente o seu desempenho e eficiência. A classificação em sistemas monoprogramáveis, multiprogramáveis e com múltiplos processadores ajuda a entender como os recursos de hardware são gerenciados e otimizados.

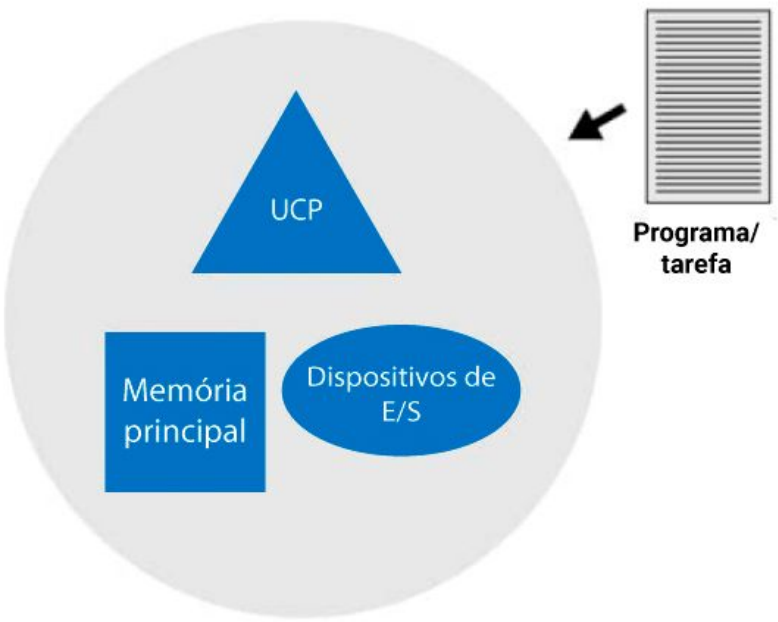
Neste vídeo, apresentaremos os diferentes tipos de sistemas operacionais: monoprogramáveis/monotarefa, multiprogramáveis/multitarefa e sistemas com múltiplos processadores, destacando suas características, funcionamento e vantagens.

> Conteúdo interativo
>
> Acesse a versão digital para assistir ao vídeo.

Vamos começar observando, na imagem a seguir, como os sistemas operacionais podem ser classificados:

Tipos de sistemas operacionais.

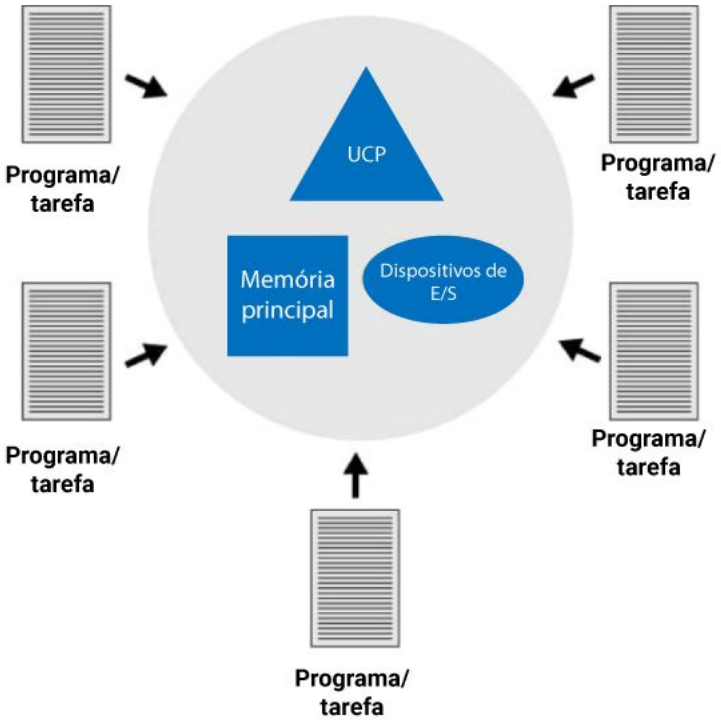
Os **sistemas monoprogramáveis** também são chamados de **sistemas monotarefa**. Nesses sistemas, os principais módulos computacionais (processador, memória e periféricos) ficam alocados exclusivamente para a execução de um único programa, sendo que qualquer outra aplicação deve esperar para poder ser processada. Veja a configuração desse tipo de sistema na imagem a seguir:



Representação de um sistema monoprogramável.

Já nos **sistemas multiprogramáveis** ou **multitarefa**, os recursos computacionais passam a ser compartilhados entre usuários e aplicações, permitindo que muitas aplicações compartilhem os mesmos recursos.

Por exemplo, alguns aplicativos podem usar o processador enquanto um determinado programa aguarda o desfecho de um comando de leitura/gravação em mídia de armazenamento. Veja, na imagem a seguir, como esse sistema funciona:



Representação de um sistema multiprogramável.

Vamos conhecer a seguir como os sistemas multiprogramáveis podem ser classificados. Na imagem, podemos notar que o sistema operacional é responsável por gerenciar vários desses programas ao mesmo tempo:

Tipos de sistemas multiprogramáveis.

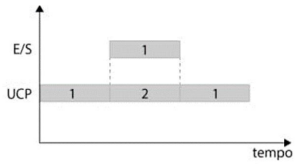
Um dos problemas que ocorriam nos sistemas monoprogramáveis é que havia um desperdício na utilização do processador, o que já não ocorre nos sistemas multiprogramáveis. Veja, a seguir, a diferença:

Monoprogramável

Neste sistema, enquanto uma leitura em disco era realizada, o processador permanecia ocioso, e o tempo de espera até que a leitura em disco fosse realizada era relativamente longo, pois as operações com dispositivos de E/S são muito lentas quando comparadas com a velocidade do processador para a execução de instruções.

Multiprogramável

Já neste sistema, o processador é utilizado por diversos programas, ou processos, de forma concorrente, ou seja, quando um processo tem que fazer uma operação de E/S, outro processo pode utilizar o processador.



A utilização da CPU como função da quantidade de processos carregados na memória simultaneamente é chamada grau de multiprogramação. A imagem a seguir apresenta um gráfico representando a utilização da CPU *versus* o grau de multiprogramação:

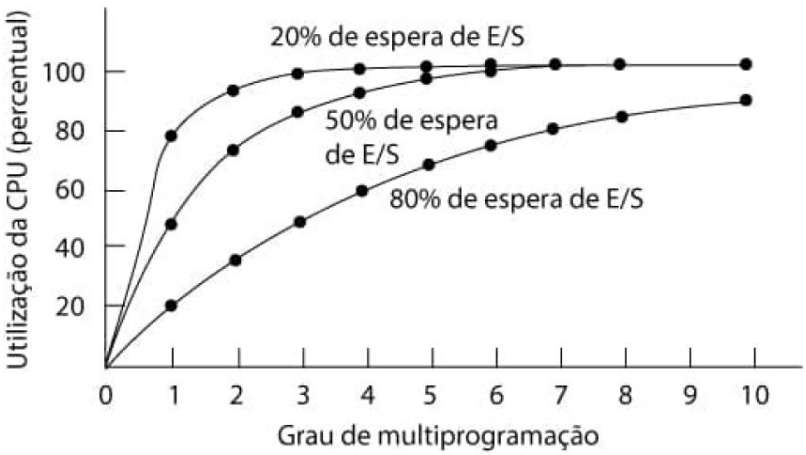


Gráfico: Utilização da CPU como função do número de processos na memória.

Pode-se perceber, pela imagem anterior, que, para a curva de “80% de espera de E/S”, se os processos passarem 80% do seu tempo esperando por dispositivos de E/S, pelo menos 10 processos devem estar na memória simultaneamente, para que a CPU desperdice menos de 10%.

A utilização da CPU é dada pela fórmula $U_{cpu} = 1 - p^n$, onde:

- $p =$ Tempo de espera de (dispositivos de) E/S.

- $n =$ Número de processos carregados na memória.

Por exemplo: Se $p = 65\%$ e $n = 3 \rightarrow U_{cpu} = 1 - 0,65^3 = 0,72$ ou 72%.

Por fim, os **sistemas com múltiplos processadores** possuem dois ou mais processadores atuando juntos, oferecendo vantagens como:

Escalabilidade

Aumenta a capacidade de processamento.

Disponibilidade

Se um processador falhar, outro pode assumir a carga de trabalho.

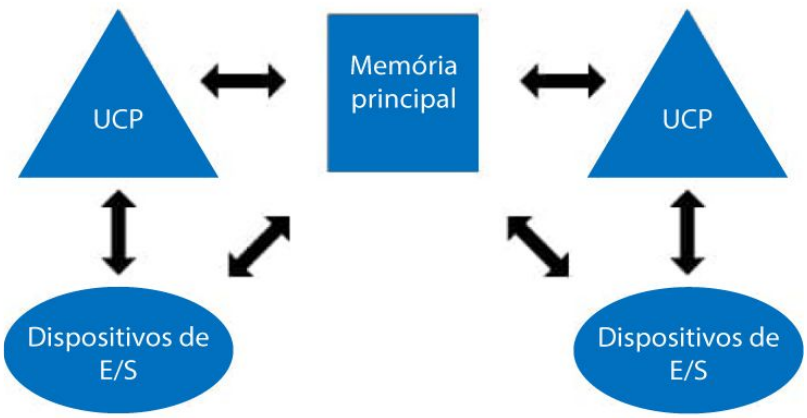
Balanceamento de carga

Distribui a carga de trabalho entre os processadores.

Em sistemas com múltiplos processadores, os processadores podem se comunicar de duas maneiras:

- nos sistemas fortemente acoplados
- nos sistemas fracamente acoplados

Nos **sistemas fortemente acoplados**, os processadores compartilham somente a memória principal e os periféricos são gerenciados por um único sistema operacional. Veja como esse sistema funciona:



Representação do sistema fortemente acoplado.

Os sistemas fortemente acoplados ainda podem ser divididos em:

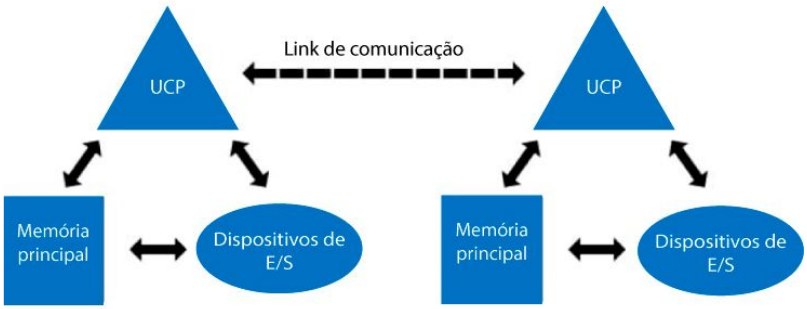
SMP - Symmetric Multiprocessors

Os processadores acessam a memória uniformemente ao longo do tempo.

NUMA - Non-Uniform Memory Access

Como existem conjuntos de processadores e memória principal, e um conjunto se conecta aos outros por meio de uma rede de interconexão, o tempo de acesso dos processadores à memória varia conforme a localização física.

Já nos sistemas **fracamente acoplados**, dois ou mais sistemas computacionais independentes (cada um possui o seu próprio sistema operacional e gerencia seus próprios recursos, como processador, memória e periféricos) estão conectados por linhas de comunicação. Veja como isso acontece:



Representação do sistemas fracamente acoplado.

Os sistemas com múltiplos processadores também podem ser chamados de **sistemas avançados de processamento**. Eles podem ser constituídos de computadores com vários processadores, processadores com vários núcleos e sistemas distribuídos. O SO deve estar adaptado para operar esses sistemas.

Atividade 1

A ilustração gráfica a seguir representa um sistema que utiliza uma técnica na qual as solicitações de entrada/saída de dados e as execuções de tarefas de um programa devem ser executadas pela CPU em alternância de tempo. Nesta técnica, pode ocorrer um desperdício na utilização do processador, por exemplo, durante operações de leitura em disco.

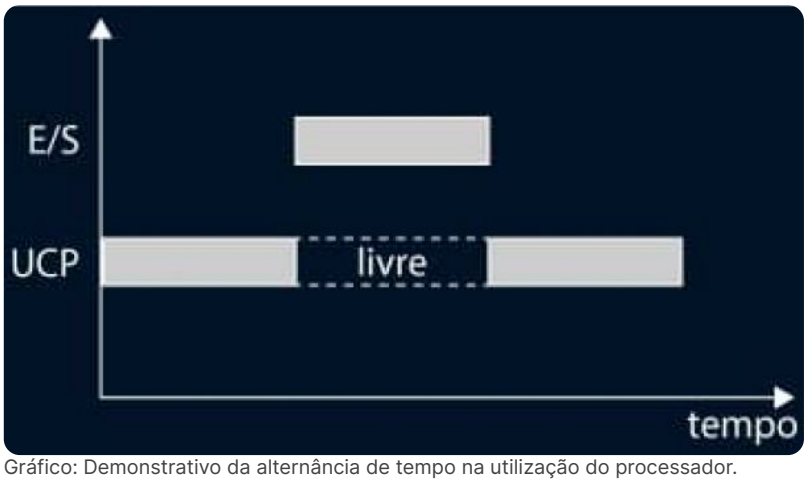


Gráfico: Demonstrativo da alternância de tempo na utilização do processador.

Essa técnica é conhecida como:

A

Multiprogramação

B

Multiusuário

C

Monoprogramação

D

Monousuário

E

Hiperusuário

A alternativa C está correta.

Nos sistemas monoprogramáveis, também chamados sistemas monotarefa, os principais módulos computacionais (processador, memória e periféricos) ficam alocados exclusivamente para a execução de um único programa, sendo que qualquer outra aplicação deveria esperar para poder ser processada.

Outras classificações dos sistemas operacionais

Entender as diversas classificações de sistemas operacionais é relevante para qualquer profissional de TI, pois cada sistema oferece um conjunto específico de serviços que atendem a diferentes necessidades e aplicações. Desde o processamento em lote até os sistemas de tempo real, cada tipo de sistema operacional possui características únicas que influenciam diretamente o desempenho e a eficiência das operações.

Neste vídeo, apresentaremos os diferentes tipos de sistemas operacionais, incluindo sistemas de grande porte e sistemas de tempo real, com destaque para os serviços que cada um oferece, como processamento simultâneo, manejo de E/S e execução de tarefas em tempo compartilhado.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Dentro desses mais de 50 anos de existência dos sistemas operacionais, existiu um desenvolvimento de uma grande variedade deles, alguns dos quais não foram bem conhecidos.

Dentro dos tipos que existiram ao longo do tempo, um SO foi elaborado oferecendo um **conjunto de serviços suportados**.

Exemplo

Alguns dos serviços que podem ser especificamente oferecidos pelos sistemas operacionais de computadores de grande porte são o processamento simultâneo de muitas tarefas, o suporte ao manejo de grandes quantidades de E/S e o oferecimento dos serviços de processamento em lote (batch), de processamento de transações e de execução de tarefas em tempo compartilhado.

A seguir, vamos conhecer um pouco mais sobre alguns desses serviços oferecidos pelos sistemas operacionais de computadores de grande porte:

Sistemas em lote

O sistema em lote (batch) processa tarefas de rotina sem a presença interativa do usuário. Por exemplo: processamento de apólices de companhia de seguro; relatório de vendas de uma cadeia de lojas.

Sistemas de processamento de transações

Os sistemas de processamento de transações administram grandes quantidades de pequenas requisições. Cada unidade de trabalho é pequena, mas o sistema precisa tratar centenas ou milhares delas por segundo.

Por exemplo: processamento de verificações em um banco ou em reservas de passagens aéreas.

Sistemas de tempo compartilhado

Os sistemas de tempo compartilhado permitem que múltiplos usuários remotos executem suas tarefas simultaneamente no computador. Por exemplo: realização de consultas a um banco de dados.

Os serviços de **sistemas de tempo real** são aqueles que possuem o **tempo** como parâmetro fundamental. Podemos citar, como exemplo, a linha de montagem de um carro. Esses sistemas podem ser de dois tipos:

Sistema de tempo real crítico

Neste sistema, as ações precisam ocorrer em determinados instantes. Como exemplos, podemos citar processos industriais, de aviônica e militares.

Sistema de tempo real não crítico

Neste tipo de sistema, o descumprimento de um prazo não causa dano permanente. Como exemplos, podemos citar sistema de áudio digital, multimídia, telefones digitais.

Alguns tipos de sistemas operacionais, assim como algumas das funcionalidades específicas relacionadas a cada tipo, podem deixar de existir, enquanto outros novos tipos e funcionalidades podem surgir, conforme acontece o avanço das tecnologias. Certos tipos de SOs e de rotinas que já existiram em algum momento no passado caem em desuso e depois voltam a ser utilizadas no tempo presente.

Tanenbaum (2010) diz que cada nova “espécie” de computador parece passar pelo mesmo desenvolvimento de seus ancestrais, tanto no que se refere ao hardware quanto ao software. Muitas vezes, uma alteração tecnológica torna alguma ideia obsoleta e ela desaparece rapidamente, mas outra mudança tecnológica pode reavivá-la.

Exemplo

O uso da memória cache nas CPUs quando se tornam mais velozes que as memórias é um caso típico. Se alguma nova tecnologia de memória tornar as memórias mais velozes que a CPU, as memórias caches irão desaparecer. E, no caso de uma nova tecnologia de CPU torná-las novamente mais rápidas que as memórias, as memórias caches reaparecerão.

Os sistemas operacionais variam quanto aos tipos de licenças, ou seja, o conjunto de ações que o usuário do SO pode ou não fazer. Vamos conhecer a seguir alguns tipos:

Software proprietário

- Licenciado sob direitos legais exclusivos – copyright.
- Usualmente o código-fonte total ou parcial não é disponibilizado para modificação por qualquer pessoa, apenas a fabricante possui o acesso.

Software livre (Free software)

Software livre se refere à **liberdade** dos usuários executarem, copiarem, distribuírem, estudarem, modificarem e aperfeiçoarem o software.

Por exemplo, a licença GNU da Free Software Foundation (FSF) possui as seguintes liberdades:

- Liberdade nº 0: Executar o programa, para qualquer propósito.
- Liberdade nº 1: Liberdade de estudar como o programa funciona, e adaptá-lo para as suas necessidades. Acesso ao código-fonte é um pré-requisito para esta liberdade.
- Liberdade nº 2: Liberdade de redistribuir cópias de modo que você possa ajudar ao seu próximo.
- Liberdade nº 3: Liberdade de aperfeiçoar o programa e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie. Acesso ao código-fonte é um pré-requisito para esta liberdade.

Software de código aberto (Open source)

O código-fonte é disponibilizado por meio de uma licença de código aberto para modificação ou melhoria por qualquer pessoa. O termo "código aberto" foi criado pela OSI (Open Source Initiative).

O software de código aberto difere-se de um software livre por não respeitar as 4 liberdades definidas pela Free Software Foundation (FSF), compartilhadas também pelo projeto Debian em "Debian Free Software Guidelines (DFSG)". Vale ressaltar que qualquer licença de software livre é também uma licença de código aberto (Open Source), mas o contrário nem sempre é verdade.

São exemplos de licenças de código aberto:

- Apache License.
- MIT License.
- Mozilla Public License.
- Common Development and Distribution License.
- Eclipse Public License.

Atividade 2

Em diversas indústrias, a precisão no tempo de execução das tarefas é fundamental para garantir a segurança e a eficiência dos processos. Sistemas de tempo real críticos são projetados para atender a essas necessidades, nas quais o cumprimento de prazos específicos é importante. Qual das seguintes características é associada aos sistemas de tempo real críticos?

A

Permitem a execução de várias tarefas simultaneamente.

B

Utilizam múltiplos processadores para tarefas paralelas.

C

As ações precisam ocorrer em determinados instantes específicos.

D

São exclusivos para sistemas de grande porte.

E

Oferecem serviços de processamento em lote e transações.

A alternativa C está correta.

Os sistemas de tempo real críticos são caracterizados pela necessidade de que as ações ocorram em momentos específicos. Esses sistemas são comumente utilizados em aplicações em que o descumprimento de um prazo pode causar danos significativos, como em processos industriais, aviônicos e militares. Por isso, é importante entender o tempo como um parâmetro fundamental nesses sistemas para o desenvolvimento e implementação eficaz de soluções em áreas críticas.

Kernel, system calls, modos de acesso

O entendimento da estrutura de um sistema operacional é fundamental para qualquer profissional de TI, pois o kernel, as system calls e os modos de acesso são elementos que garantem o funcionamento eficiente e seguro do sistema. Conhecer esses componentes ajuda a compreender como o SO gerencia recursos, processos e a comunicação entre diferentes partes do sistema.

Neste vídeo, vamos apresentar os componentes principais de um sistema operacional, incluindo o kernel, as system calls e os modos de acesso. Explicaremos as funções do kernel, como tratamento de interrupções, gerência de memória e suporte a redes, além do funcionamento das chamadas ao sistema.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Estrutura do SO: kernel, system calls, modos de acesso

O sistema operacional é formado por um conjunto de rotinas que oferecem serviços aos usuários, às suas aplicações e ao próprio sistema. Esse conjunto de rotinas é denominado **núcleo** do sistema ou **kernel**, cuja localização na máquina de níveis está ilustrada na imagem a seguir:

Kernel do SO na máquina de níveis.

O sistema operacional funciona com algumas particularidades quanto à execução das rotinas:

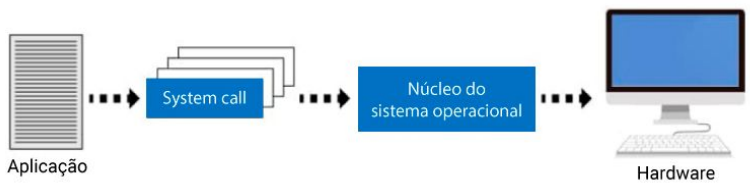
- Tarefas não sequenciais, ou seja, as rotinas são executadas concorrentemente.

- Sem ordem predefinida.
- Eventos assíncronos.
- Eventos relacionados ao hardware e eventos relacionados às tarefas internas do próprio SO.

Já o núcleo do SO deve atuar considerando essas particularidades, e para isso implementa funções como:

- Tratamento de interrupções e exceções.
- Criação e eliminação de processos e threads.
- Sincronização e comunicação entre processos e threads.
- Escalonamento e controle de processos e threads.
- Gerência de memória.
- Gerência do sistema de arquivos.
- Gerência dos dispositivos de E/S.
- Suporte a redes locais e distribuídas.
- Contabilização do uso do sistema.
- Auditoria e segurança do sistema.

A execução das rotinas no sistema operacional é controlada pelo mecanismo denominado **system call** ou **chamada ao sistema**, que é um mecanismo de proteção por software que garante a execução somente de rotinas previamente autorizadas. Veja como ele funciona na imagem a seguir:



Representação do mecanismo de system call.

Atividade 1

O núcleo de um sistema operacional, conhecido como kernel, é responsável por gerenciar os recursos básicos do sistema e garantir que todas as operações ocorram de maneira eficiente e segura. Compreender as funções do kernel é fundamental para entender como os sistemas operacionais operam e controlam o hardware e o software do computador. Nesse contexto, qual das seguintes funções é desempenhada pelo kernel de um sistema operacional?

A

Processamento de dados em lote.

B

Criação e eliminação de processos e threads.

C

Execução de aplicativos de usuário.

D

Atualização de software e drivers.

E

Acesso direto ao hardware pelo usuário.

A alternativa B está correta.

O kernel de um sistema operacional desempenha funções importantes para o funcionamento do sistema, como a criação e a eliminação de processos e threads. Ele também é responsável pela gerência de memória, tratamento de interrupções, escalonamento de processos e suporte a redes, entre outras funções. Entender essas responsabilidades do kernel é essencial para compreender como um sistema operacional gerencia os recursos do computador de forma eficiente e segura.

System calls

Entender como as chamadas de sistema (system calls) funcionam é crucial para quem deseja se aprofundar no funcionamento dos sistemas operacionais. As system calls são a principal interface entre os programas de aplicação e o núcleo do sistema, permitindo o acesso controlado aos recursos do sistema. Conhecer esses mecanismos é essencial para garantir que as operações sejam realizadas de maneira segura e eficiente.

Neste vídeo, apresentaremos os conceitos e funcionamento das system calls, como elas permitem a comunicação entre programas de aplicação e o kernel, os passos envolvidos em uma chamada de sistema "read" e exemplos de diferentes tipos de chamadas em sistemas operacionais como UNIX e Windows.

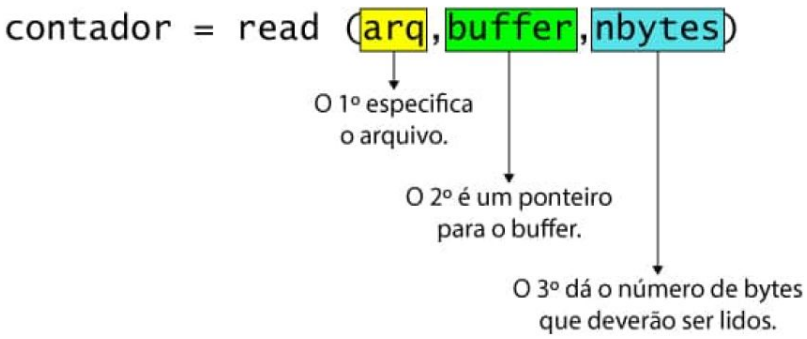
Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Quando qualquer aplicação deseja chamar uma rotina do sistema operacional, ela aciona a system call. O sistema operacional verifica se a aplicação possui os privilégios suficientes para a execução da rotina desejada. Se não tiver, o sistema operacional impede o desvio para a rotina do sistema e informa ao programa de origem que a operação não será executada.

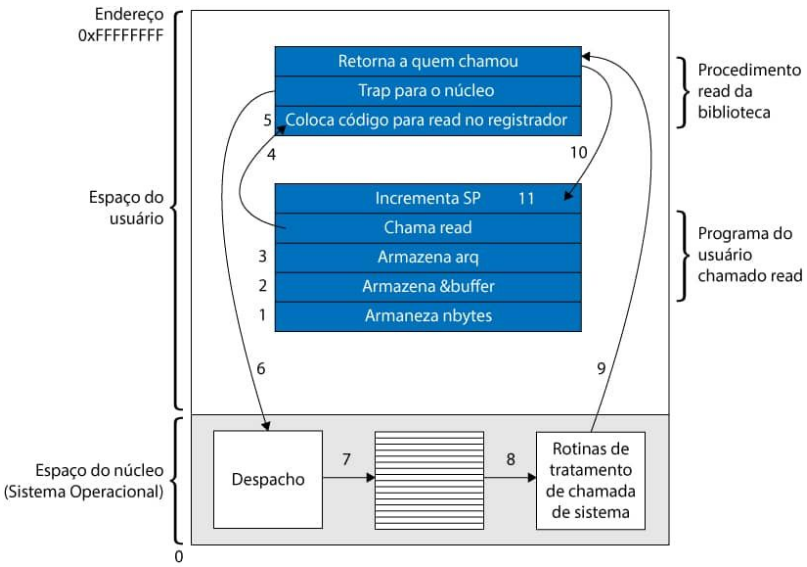
> A system call é uma “porta de entrada” para o acesso ao núcleo do SO e aos seus serviços. Cada serviço possui uma system call associada, e cada SO possui seu próprio conjunto de chamadas, com nomes, parâmetros e formas de ativação específicas.

A chamada ao sistema "read", apresentada na imagem a seguir, possui três parâmetros:



Chamada ao sistema "read" e seus parâmetros.

Observe agora os passos para a realização da system call "read" :



Apresentação dos 11 passos para a realização da chamada "read"

Acompanhe a seguir uma breve descrição dos passos apresentados na imagem anterior:

1

Passo 1 a 3

O programa que se chama read primeiramente armazena os parâmetros na pilha.

2

Passos 4 e 5

Segue-se para a chamada real à rotina de biblioteca (passo 4) que, em geral, coloca o número da chamada de sistema em um local esperado pelo SO, por exemplo, em um registrador (o passo 5).

3

Passo 6

É executada uma instrução TRAP para passar do modo usuário para o modo núcleo, iniciando a execução em um determinado endereço no núcleo.

4 Passo 7
O código do núcleo, iniciado após a instrução TRAP, verifica o número da chamada de sistema e despacha para a rotina correta de tratamento dessa chamada.

5
Passo 8
É a execução da rotina de tratamento da chamada de sistema.

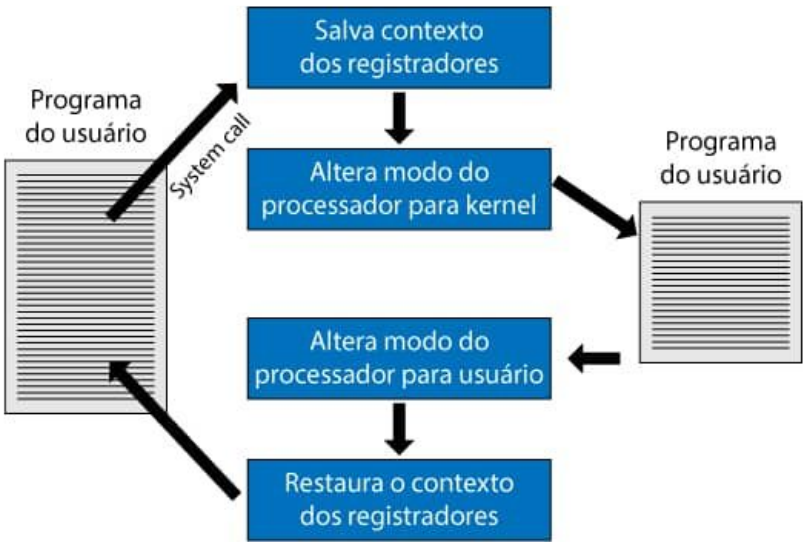
6
Passos 9 e 10
Após o término dessa rotina, pode-se retornar o controle para a rotina de biblioteca no espaço do usuário, na instrução seguinte à instrução TRAP (passo 9), em geral do mesmo modo no qual são feitas as chamadas de rotina (passo 10).

7
Passos 11
O programa do usuário deve limpar a pilha. O programa agora está liberado para fazer o que quiser.

Em geral, para qualquer situação, o funcionamento da system call consiste:

1. Na realização da chamada ao sistema pela aplicação que deseja executar rotina(s).
2. No processamento da solicitação pelo kernel.
3. No retorno, para a aplicação solicitante, de uma resposta, com um estado de conclusão, indicando se houve algum erro.

Na imagem a seguir, podemos observar o funcionamento da call system:



Chamada a uma rotina de sistema.

Um exemplo de conjunto de chamadas de sistema é o **POSIX** (Portable Operating System Interface for Unix), padrão internacional ISO/IEC/IEEE 9945 que foi uma tentativa de padronizar as system calls.

O objetivo inicial foi unificar as diversas versões existentes do sistema operacional UNIX, permitindo um aplicativo utilizando o padrão POSIX teoricamente poder ser executado em qualquer SO que possua suporte. É incorporado, atualmente, pela maioria dos sistemas operacionais modernos.

> Já o conjunto de chamadas de sistemas Win32 é suportada desde o Windows 95. Cada nova versão do Windows traz novas chamadas que não existiam anteriormente, buscando não invalidar os programas existentes.

A API Win32 possui um grande número de chamadas para gerenciar aspectos da interface gráfica (como janelas, figuras geométricas, textos, fontes de caracteres, caixas de diálogos, menus etc.) e possui diferenças em relação ao POSIX. Veja algumas de suas características:

Desacoplamento de chamadas

Uma característica é o desacoplamento das chamadas de biblioteca e das chamadas reais ao sistema, o que pode dificultar a identificação do que é uma chamada ao sistema, e do que é uma chamada de biblioteca no espaço do usuário.

Acesso aos serviços do SO

A API Win32 também permite aos programadores acesso aos serviços do SO mesmo quando trabalhando em uma versão diferente com chamadas ao sistema modificadas. As chamadas Win32 chegam a milhares.

A tabela a seguir exibe algumas chamadas da API Win32 que correspondem, aproximadamente, às chamadas do UNIX.

UNIX	Win32	Descrição
fork	CreateProcess	Cria um novo processo
waitpid	WaitForSingleObject	Pode esperar que um processo saia
execve	(nenhuma)	CreateProcess = fork + execve
exit	ExitProcess	Conclui a execução
open	CreateFile	Cria um arquivo ou abre um arquivo existente
close	CloseHandle	Fecha um arquivo
read	ReadFile	Lê dados a partir de um arquivo
write	WriteFile	Escreve dados em um arquivo
iseek	SetFilePointer	Move o ponteiro do arquivo
stat	GetFileAttributesEx	Obtém vários atributos do arquivo
mkdir	CreateDirectory	Cria um novo diretório
rmdir	RemoveDirectory	Remove um diretório vazio
link	(nenhuma)	Win32 não dá suporte a links
unlink	DeleteFile	Destrói um arquivo existente

mount	(nenhuma)	Win32 não dá suporte a links
umount	(nenhuma)	Win32 não dá suporte a links
chdir	SetCurrentDirectory	Altera o diretório de trabalho atual
chmod	(nenhuma)	Win32 não dá suporte à segurança (embora o NT suporte)
kill	(nenhuma)	Win32 não dá suporte a sinais
time	GetLocalTime	Obtém o tempo atual

Tabela: Correspondência aproximada entre as chamadas da API Win32 e as chamadas do POSIX.
Adaptado de Tanenbaum, 2010.

Podemos classificar os tipos de chamadas ao sistema em: chamadas para o gerenciamento de **processos**, **arquivos** e **diretórios**. A tabela a seguir mostra alguns exemplos desses tipos de chamadas.

Gerenciamento de processos	
Chamada	Descrição
Gerenciamento de arquivos	
Chamada	Descrição
Gerenciamento do sistema de diretório e arquivo	
Chamada	Descrição
Diversas	
Chamada	Descrição
pid=fork()	Cria um processo filho idêntico ao pai
pid=waitpid(pid, &statloc, options)	Espera que um processo filho seja concluído
fd=open(file, how, ...0)	Abre um arquivo para leitura, escrita ou ambos
s=close(fd)	Fecha um arquivo aberto
n=read(fd, buffer, nbytes)	Lê dados a partir de um arquivo em um buffer
s=mkdir(name, mode)	Cria um novo diretório
s=rmdir(name)	Remove um diretório vazio
s=mount(special, name, flag)	Monta um sistema de arquivos
s=chmod(name, mode)	Altera os bits de proteção de um arquivo
seconds=time(&seconds)	Obtém o tempo decorrido desde 1 de janeiro de 1970

Tabela: Exemplos de diferentes tipos de chamadas ao sistema.
Adaptado de Tanenbaum, 2010.

Atividade 2

Uma determinada aplicação faz uso da system call mostrada a seguir, para esperar pela conclusão de um processo filho:

num_proc = waitpid(pid, &statloc, options)

Dentre as opções a seguir, marque aquela que faz o melhor enquadramento ao tipo da chamada de sistema mostrada.

A

Gerenciamento de arquivos.

B

Gerenciamento de processos.

C

Gerenciamento do sistema de diretório e arquivo.

D

Gerenciamentos diversos.

E

Gerenciamento do terminal.

A alternativa B está correta.

A imagem Chamada a uma rotina de sistema, do system call, apresentada anteriormente, mostra alguns exemplos desses tipos de chamadas. Essa chamada ao sistema é do tipo destinado ao gerenciamento de processos.

Modos de acesso

Compreender os modos de acesso em um sistema operacional é importante para a segurança e a eficiência na execução de programas.

Os modos de acesso determinam como as instruções são executadas e quais recursos podem ser acessados, protegendo o sistema contra operações não autorizadas. Conhecer a diferença entre o modo kernel e o modo usuário é válido para qualquer profissional de TI.

Neste vídeo, apresentaremos os modos de acesso em sistemas operacionais, destacando as diferenças entre o modo kernel e o modo usuário, como as aplicações interagem com o sistema por meio das system calls, e os mecanismos de proteção que garantem a segurança do sistema.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

As instruções executadas pelas aplicações podem ser dos seguintes tipos:

- **Privilegiadas**: incluem as instruções que de algum modo podem comprometer o sistema.
- **Não privilegiadas**: são as instruções que não oferecem riscos ao sistema.

Essas instruções podem acessar o sistema de dois modos distintos, apresentados a seguir:

Modo kernel ou supervisor

Uma aplicação possui o modo de acesso supervisor quando é constituída de instruções privilegiadas. Ela pode executar o conjunto total de instruções. O modo kernel também protege a área do SO localizada na memória.

Modo de acesso usuário

Uma aplicação possui o modo de acesso usuário quando é constituída de instruções não privilegiadas.

Quando a aplicação chama a rotina do SO através da system call, o sistema verifica se ela tem os privilégios necessários (autorizada pelo administrador do sistema). Se não possuir, o SO impede o desvio para a rotina do sistema através do **mecanismo de proteção por software**.

> Uma aplicação sempre deve executar com o processador em modo usuário.

Se uma aplicação tentar executar diretamente uma instrução privilegiada **sem ser por intermédio de uma chamada à rotina do sistema, um mecanismo de proteção por hardware** garantirá a segurança do sistema. O hardware do processador irá sinalizar com um erro.

Atividade 3

Em um sistema operacional, há diferentes modos de operação que determinam o nível de acesso às instruções e aos recursos do sistema. O modo de acesso kernel é importante para garantir a segurança e a estabilidade do sistema, controlando o acesso aos recursos críticos. Qual das seguintes afirmações descreve corretamente o modo de acesso kernel em um sistema operacional?

A

Permite a execução de instruções não privilegiadas apenas.

B

É utilizado exclusivamente por aplicações de usuário comum.

C

Protege a área do sistema operacional na memória.

D

Impede a comunicação entre aplicações e o kernel.

E

Executa instruções diretamente sem verificar privilégios.

A alternativa C está correta.

O modo kernel, também conhecido como modo supervisor, permite a execução de instruções privilegiadas e protege a área do sistema operacional localizada na memória. Esse modo garante que somente o kernel e as instruções autorizadas possam acessar recursos críticos do sistema, proporcionando uma camada de segurança essencial para evitar operações não autorizadas e potencialmente prejudiciais. Quando uma aplicação precisa executar uma rotina do sistema, ela deve utilizar uma system call, permitindo ao sistema operacional verificar e controlar os privilégios necessários.

Arquiteturas de kernel

Entender as diferentes arquiteturas de kernel é essencial para qualquer profissional de TI, pois elas determinam como o sistema operacional gerencia recursos e processos. Cada arquitetura possui vantagens e desvantagens que influenciam diretamente o desempenho, a segurança e a manutenção do sistema. Por isso, é importante conhecer essas variações para escolher a melhor abordagem para diferentes necessidades tecnológicas.

Neste vídeo, apresentaremos as principais arquiteturas de kernel: monolítica, de camadas, máquina virtual, microkernel e exokernel. Vamos discutir suas características, vantagens, desvantagens e exemplos práticos de uso, além de como essas arquiteturas impactam o desempenho e a segurança dos sistemas operacionais.

Conteúdo interativo

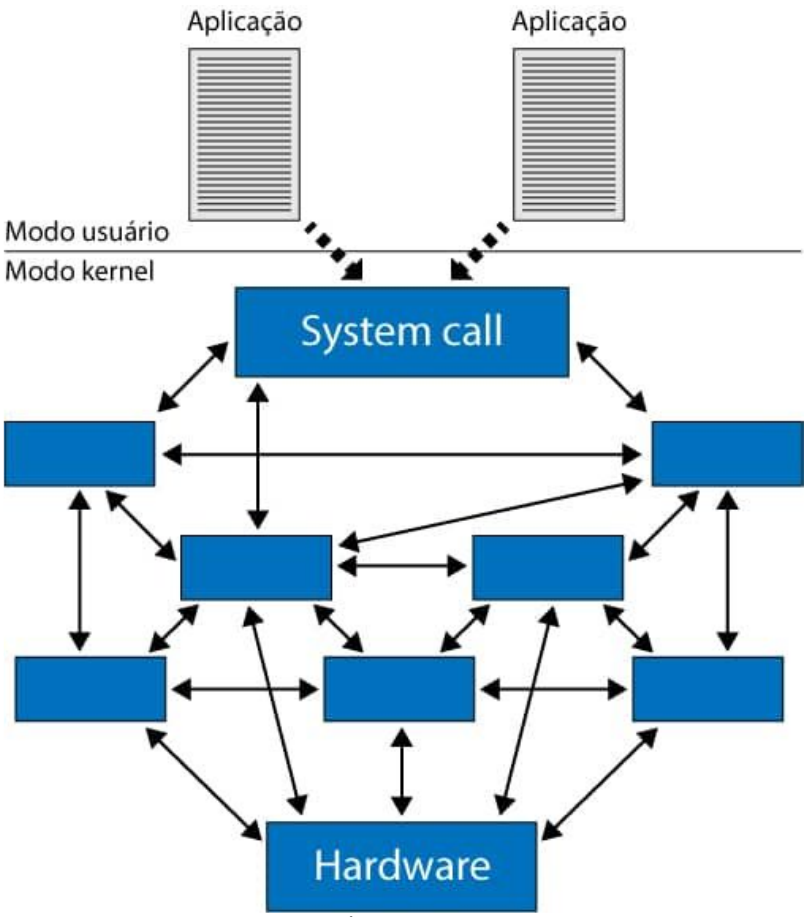
Acesse a versão digital para assistir ao vídeo.

Existem variações quanto à concepção da estrutura do núcleo do sistema operacional. Vamos conhecer as principais arquiteturas dos SO.

Arquitetura monolítica (mono-kernel)

Podemos comparar a arquitetura de núcleo monolítico com uma aplicação composta por vários módulos, que são compilados separadamente e depois **linkados** (ligados), constituindo um grande e único programa executável, no qual os módulos podem interagir livremente.

Na imagem a seguir, observe que, nessa abordagem, o programa-objeto real do sistema operacional é construído compilando todas as rotinas individualmente e depois juntando todas em um único arquivo-objeto usando o ligador (linker) do sistema. Todas as rotinas são visíveis umas às outras, não existindo, em essência, uma ocultação de informação.



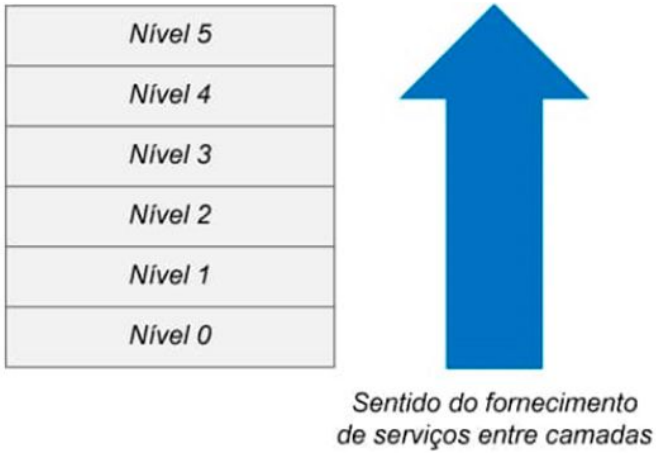
Representação da arquitetura monolítica.

Arquitetura de camadas

Nesta arquitetura, o sistema é dividido em várias camadas, que oferecem um conjunto de funções. Cada camada oferece suas funções para a camada imediatamente superior. Conheça as suas vantagens e a sua desvantagem:

Vantagens	Desvantagem
Possui as vantagens de isolar as funções do SO, facilitando sua manutenção e depuração, e de criar uma hierarquia de níveis de modo de acesso, protegendo as camadas mais internas.	Possui a desvantagem de poder ter o desempenho comprometido (a segregação implica em mais rotinas para a comunicação entre as camadas).

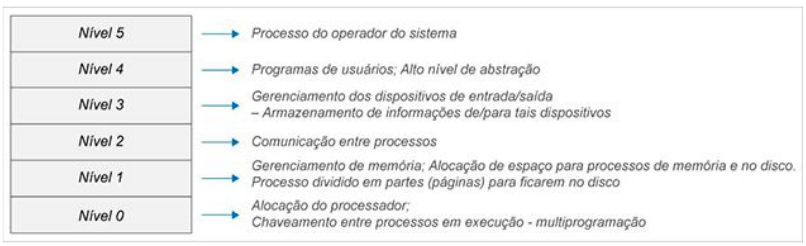
Atualmente, a maioria dos SO comerciais para workstations, em geral, usa duas camadas. Veja a seguir a arquitetura de camadas:



Arquitetura de camadas com níveis de funções.

Conteúdo interativo

Acesse a versão digital para ver mais detalhes da imagem abaixo.

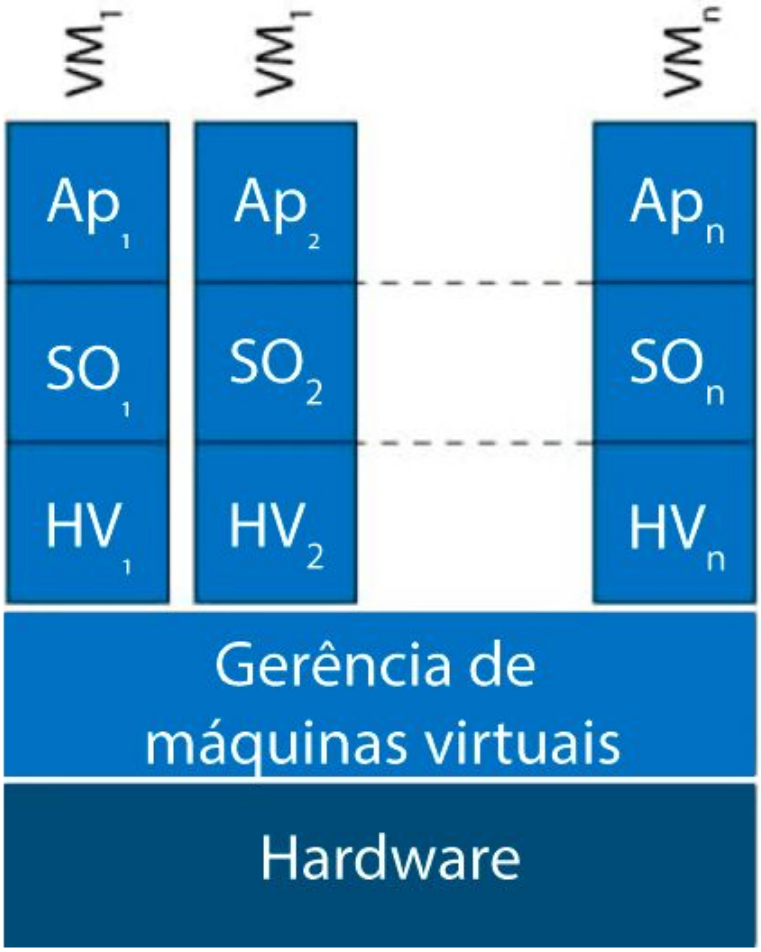


Descrição dos níveis da arquitetura de camadas.

Máquina virtual ou virtual machine (VM)

Cria um nível intermediário entre o hardware e o sistema operacional, chamado de gerenciador de máquinas virtuais. Este nível possibilita a criação de máquinas virtuais independentes, sendo que cada uma oferece uma cópia virtual do hardware (**virtualização**), incluindo os modos **kernel** e usuário, as interrupções e dispositivos de E/S.

O termo máquina virtual também pode ser definido como a ocultação de certas características computacionais através de uma camada de abstração. Veja a seguir a estrutura da máquina virtual:



Representação da VM.

Nos últimos anos, as máquinas virtuais voltaram a ser bastante utilizadas. Muitas organizações executavam seus servidores em computadores separados às vezes com sistemas operacionais distintos. Com a virtualização, é possível executar todos eles na mesma máquina sem que uma falha em um servidor afete os outros. Veja as vantagens obtidas pelos clientes que alugam uma máquina virtual:

Custo

Uma mesma máquina física suporta muitas máquinas virtuais simultaneamente. Logo, os clientes poderão executar os sistemas operacionais e softwares que quiserem por uma parte do custo de um servidor dedicado.

Uso simultâneo de SOs

Os clientes podem usar a virtualização também para executar dois ou até mais sistemas operacionais ao mesmo tempo.

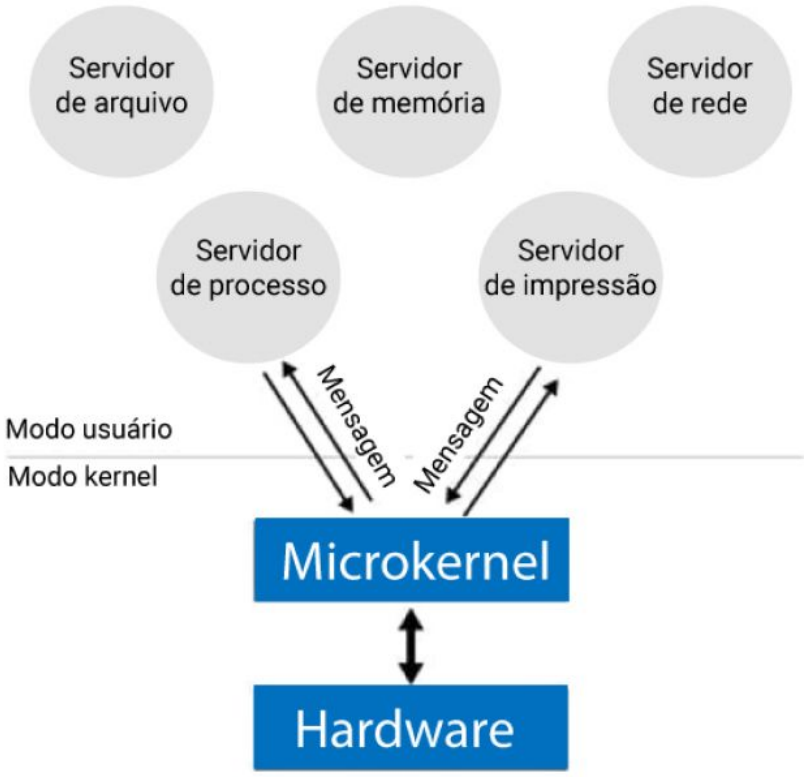
Não devemos esquecer que, para executar o software de máquina virtual em um computador, a CPU deve ser virtualizável.

Comentário

Em CPUs mais antigas, as tentativas de executar instruções não privilegiadas no modo usuário são ignoradas, o que impossibilitou a existência de máquinas virtuais nesse hardware. Essa situação deixou de existir ao substituir os hypervisors de tipo 1, executados diretamente no hardware, pelos hypervisors de tipo 2, executados como programas aplicativos na camada superior do sistema operacional, conhecida como sistema operacional hospedeiro. O hypervisor é o monitor de máquina virtual. O hypervisor de tipo 2 instala o sistema operacional hóspede em um disco virtual, que é apenas um arquivo grande no sistema de arquivos do sistema operacional hospedeiro. Quando o sistema operacional hóspede é inicializado, faz o mesmo procedimento que é feito no verdadeiro hardware, iniciando algum processo subordinado e depois uma interface gráfica.

Arquitetura microkernel

Visa a tornar o núcleo do sistema operacional o menor e mais simples possível, disponibilizando no núcleo apenas os serviços do sistema que oferecem funções específicas, como gerência de arquivos, gerência de processos, gerência de memória e escalonamento. Os demais processos são executados **fora do núcleo**. Veja como isso se dá na imagem a seguir:



Representação da arquitetura microkernel.

Essa arquitetura é de difícil implementação na prática, sendo usualmente feita uma combinação **híbrida**, ou seja, do modelo de camadas com a arquitetura microkernel.

Atenção

O princípio do microkernel é obter alta confiabilidade ao dividir o sistema operacional em pequenos módulos e bem definidos. Apenas o micronúcleo pode ser executado no modo núcleo. Os outros módulos são executados como processos de usuário comuns.

Quando, por exemplo, há um erro em um driver de dispositivo executado como um processo de usuário separado, somente o componente correspondente é quebrado, enquanto o sistema continua funcionando inteiramente.

Exonúcleo

No lugar da clonagem da máquina real, como é no caso das máquinas virtuais, uma estratégia é dividi-la, ou seja, fornecer um subconjunto de recursos para cada usuário. Por exemplo, uma máquina virtual pode obter os blocos 0 a 1.023 do disco, outra os blocos 1.024 a 2.047 etc.

> O exonúcleo é um programa em execução em modo núcleo na camada mais inferior, que aloca recursos às máquinas virtuais e verifica as tentativas de uso para garantir que uma máquina não tente usar recursos de outra.

A vantagem é que o exonúcleo poupa uma camada de mapeamento: ao invés de cada máquina virtual pensar que possui seu próprio disco (com blocos indo de 0 até o limite), fazendo com que o hypervisor mantenha tabelas para remapear os endereços de disco, com o exonúcleo, esse mapeamento não é mais necessário. Basta manter o registro de para qual máquina virtual foi atribuído qual recurso.

Atividade 4

Considere que você está projetando um sistema operacional que precisa ter alta confiabilidade e segurança. Você decide utilizar uma arquitetura de kernel que mantenha os serviços essenciais no núcleo e execute outros processos, como drivers de dispositivos, fora do núcleo. Qual arquitetura de kernel você escolheria e por quê?

A

Arquitetura monolítica, pois todos os serviços do sistema são integrados em um único módulo.

B

Arquitetura de camadas, pois o sistema é dividido em várias camadas independentes.

C

Arquitetura microkernel, pois apenas os serviços essenciais são executados no núcleo.

D

Arquitetura de máquina virtual, pois cria máquinas virtuais para cada aplicação.

E

Arquitetura exokernel, pois fornece um subconjunto de recursos diretamente.

A alternativa C está correta.

A escolha da arquitetura microkernel seria a mais adequada para um sistema operacional que precisa de alta confiabilidade e segurança. Nesta arquitetura, apenas os serviços essenciais, como a gerência de processos e memória, são executados no núcleo, enquanto outros serviços, como drivers de dispositivos, são executados como processos de usuário separados. Isso significa que erros em serviços não essenciais não comprometem o núcleo do sistema, aumentando a estabilidade e a segurança. Além disso, a modularidade dessa arquitetura facilita a manutenção e a extensão do sistema operacional.

Instalação

O Linux, um sistema operacional "Unix-like", oferece flexibilidade, segurança e é amplamente utilizado em servidores, desktops e dispositivos embarcados.

Neste vídeo, apresentaremos a arquitetura do Linux, o processo de instalação do Ubuntu e uma introdução aos comandos básicos do sistema. Abordaremos desde o download do arquivo ISO até a configuração inicial e execução de comandos essenciais no terminal.

> **Conteúdo interativo**
>
> Acesse a versão digital para assistir ao vídeo.

O Linux é um sistema operacional moderno e gratuito desenvolvido inicialmente em 1991 como um kernel pequeno e autocontido por Linus Torvalds, com os objetivos de ser baseado nos padrões UNIX e manter a compatibilidade com o mesmo. Sua história tem sido de colaboração por muitos usuários do mundo inteiro se comunicando principalmente através da Internet.

> O Linux é um sistema operacional dito “Unix-like”, por possuir comportamento similar ao do sistema operacional Unix (multitarefa e multiusuário).

Uma distribuição Linux é constituída por uma coleção de aplicativos e o kernel (núcleo) do sistema operacional. As distribuições podem ser:

- Utilizadas por meio da instalação em disco no terminal do usuário.
- Usadas por meio das distribuições do tipo live (oferecendo recursos mais limitados), que geralmente são executados a partir de dispositivos de armazenamento como discos removíveis, e carregados na memória RAM sem a necessidade de instalação no host.

Vamos agora ver como podemos instalar o Linux para darmos os primeiros passos nesse sistema operacional.

Para isso, iremos utilizar o **Ubuntu**, uma distribuição popular baseada no Debian, que servirá como base para as demonstrações exibidas neste módulo, especificamente a versão "Ubuntu desktop".

Entretanto, você poderá optar por usar outras distribuições de sua escolha. O download poderá ser realizado na página do Ubuntu.

Cada versão do Ubuntu recebe uma numeração e um codinome, que seguem a lógica mostrada na tabela a seguir:

Imagem representativa do Ubuntu.

Denominação da versão	20.	04	LTS	Focal Fossa
Significado	Ano (2020)	Mês (04 – abril)	Long-Term Support (Suporte de longo prazo - 5 anos de suporte (atualizações))	Codinome

Tabela: Exemplo da denominação de uma versão do Ubuntu.
Fabio Henrique Silva.

Na sequência, você deverá realizar o download do arquivo de imagem “iso" correspondente à versão atual do Ubuntu, acessando a página do Ubuntu.

Comentário

Você poderá melhorar a utilização do SO no VirtualBox instalando o “Adicionais para convidado”, acessando o menu “Dispositivos” → “Inserir Imagem de CD dos Adicionais para Convidado”. O “CD” será automaticamente montado e deverá aparecer como execução automática.

Atividade 1

O Linux é um sistema operacional de código aberto que vem em diversas versões conhecidas como distribuições. Cada distribuição oferece um conjunto específico de aplicativos, ferramentas e o kernel do sistema operacional, adaptando-se a diferentes necessidades e preferências dos usuários. Qual é a principal característica que define uma distribuição Linux?

A

Um conjunto de aplicativos e o kernel do sistema operacional.

B

Um sistema operacional desenvolvido exclusivamente por Linus Torvalds.

C

Uma versão do Windows compatível com Linux.

D

Um sistema operacional que não suporta multitarefa.

E

Um software comercial sem suporte à comunidade.

A alternativa A está correta.

Uma distribuição Linux é definida como uma coleção de aplicativos e o kernel do sistema operacional. Diferentes distribuições podem oferecer variações no conjunto de aplicativos e ferramentas disponíveis, mas todas compartilham o núcleo do Linux. Esse conjunto inclui tudo o que é necessário para instalar e usar o sistema operacional, desde o ambiente gráfico até as ferramentas de linha de comando. As distribuições podem ser instaladas diretamente no disco ou executadas a partir de dispositivos de armazenamento removíveis, oferecendo flexibilidade e conveniência para os usuários.

Estrutura de diretórios no Linux

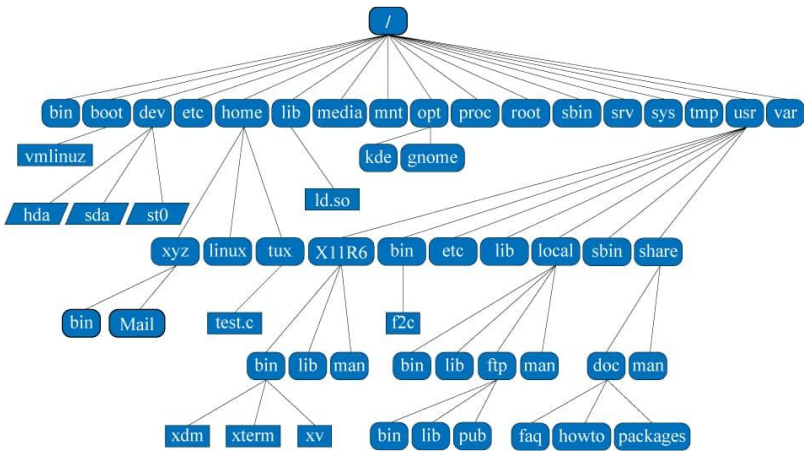
A estrutura de diretórios no Linux é essencial para navegar e administrar o sistema eficientemente. Cada diretório possui uma função específica e conhecer essa organização ajuda a localizar arquivos e configurar o sistema de maneira adequada.

Neste vídeo, apresentaremos a estrutura hierárquica de diretórios no Linux, destacando os principais diretórios como /home, /bin, /etc, /var, suas respectivas funções e conteúdos. Vamos aprender como essa organização facilita a administração do sistema.

Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

O Linux possui uma estrutura única de pastas que partem de uma única raiz. Todos os dispositivos, internos ou externos, fixos ou removíveis, são montados em algum ponto abaixo da raiz. Veja essa estrutura na imagem a seguir:



Estrutura de pastas do Linux.

Existem regras para nomes de arquivos e pastas. O Linux é case-sensitive, ou seja, as letras MAIÚSCULAS e minúsculas fazem diferença no nome de um arquivo.

Exemplo

“texto", "Texto" e "TEXTO" são três arquivos distintos. Quando for escrever um caminho de diretórios, use o separador "/". Da seguinte forma: /tmp/Texto.txt.

As principais pastas do Linux e seus significados resumidos estão mostrados na tabela a seguir:

/	Pasta Raiz	/usr	Programas de Usuário
/bin	Executáveis Binários	/home	Pasta Pessoal
/sbin	Sistema Binário	/boot	Arquivos de Inicialização
/etc	Arquivos de Configuração	/lib	Bibliotecas do Sistema
/dev	Arquivos de Dispositivos	/opt	Aplicações Opcionais
/proc	Informação de Processo	/mnt	Pasta de Montagem
/var	Arquivos Variáveis	/media	Dispositivos Removíveis
/tmp	Arquivos Temporários	/srv	Serviço de Dados

Tabela: Diretórios e seus significados.
Fabio Henrique Silva.

O **Super Usuário** é um usuário especial predefinido, com poderes especiais de administração. No Windows, é chamado de administrador (administrator), e no Linux (Unix) é denominado root.

Atividade 2

Um usuário está usando o terminal do Linux, sem ter a intenção de realizar qualquer tipo de comando ou procedimento específico, quando percebe a linha de comandos escrita da seguinte maneira:

root@sistema-servidor:/#

Assinale a alternativa correta quanto ao significado dos elementos mostrados.

A

O nome do computador é root.

B

Está acessando o diretório sistema-servidor.

C

Está em modo de super usuário.

D

Está no modo de usuário comum.

E

Está acessando o terminal do servidor de forma logada.

A alternativa C está correta.

Quando o usuário usa o comando, poderá acessar o modo super usuário e perceberá a presença do nome de usuário root e do símbolo tralha ‘#’ na linha de comandos do terminal.

Comandos do Linux

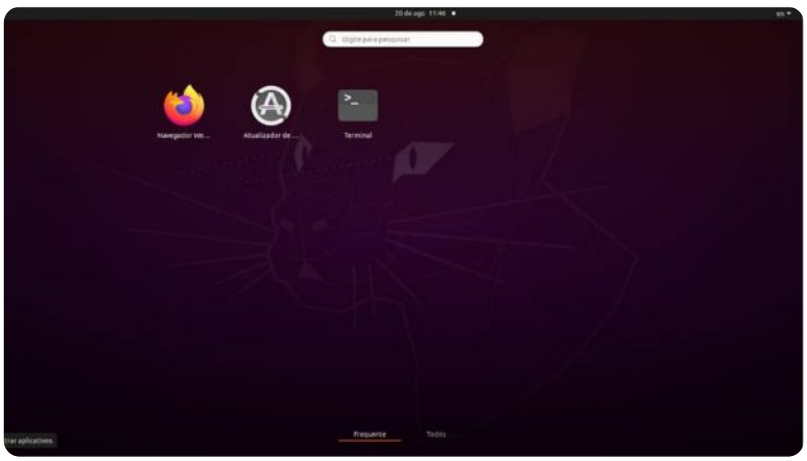
O uso do terminal é uma habilidade necessária para a administração de sistemas Linux. Compreender os comandos básicos e saber navegar pelo terminal permite que os usuários realizem uma ampla gama de tarefas, desde a manipulação de arquivos até a instalação de softwares.

Neste vídeo, apresentaremos os passos para abrir o terminal no Ubuntu Desktop e uma introdução aos comandos básicos do Linux, incluindo a navegação entre diretórios, manipulação de arquivos e execução de comandos essenciais. Vamos demonstrar como realizar operações comuns no terminal para facilitar o uso do sistema.

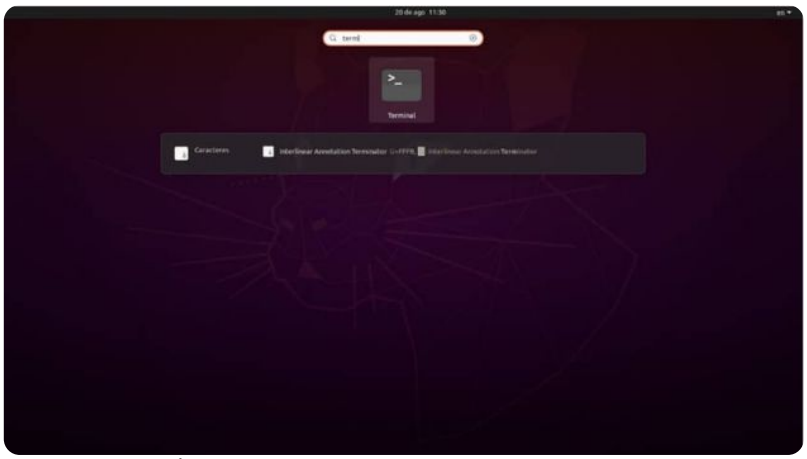
Conteúdo interativo

Acesse a versão digital para assistir ao vídeo.

Mostraremos, a seguir, os comandos básicos para o terminal do Linux. No Ubuntu Desktop, clique no ícone "mostrar aplicativos" na barra da lateral esquerda, digite "term" na caixa de pesquisa, e abra o terminal, conforme mostrado nas imagens a seguir:



Tela de abertura do terminal no Ubuntu Desktop.



Tela subsequente à tela de abertura do terminal no Ubuntu Desktop

O **terminal** (ou prompt de comando) será aberto, conforme a imagem a seguir.

Tela do terminal no Ubuntu Desktop.

Veja o significado da linha de comandos:

teste@teste-VirtualBox:~$

teste: Usuário

teste-VirtualBox: Nome da máquina

~$: Significa que o usuário está na sua pasta pessoal em /home ("/home/usuário")

Clicando no link comandos básicos para uso via terminal, você encontrará uma lista com os comandos.

Alguns aplicativos são obtidos na forma de código-fonte aberto, implicando na necessidade de compilação do código. Os passos gerais para compilação de código-fonte são:

- Realizar o download do arquivo-fonte (por exemplo, **.tar.gz**).
- Descompactar o arquivo-fonte.
- Acessar o diretório contendo os arquivos do código-fonte.

Por fim, deve-se executar os seguintes comandos:

```
plain-text

./configure
make
make install
```

Atividade 3

Suponha que você está usando o terminal do Linux com o nome de usuário "gestor".

Partindo da verificação do seu diretório de login, você deseja acessar o diretório "setor", dentro do seu diretório de login, e mover o arquivo "planilha" para o diretório "Documentos", que se encontra dentro do diretório de login. Marque a alternativa que contém a sequência de comandos a serem realizados:

A

$ pwd
$ cd ./gestor
$ mv /home/setor/gestor/planilha /home/setor/Documentos

B

$ pwd
$ cd ./setor
$ mv /home/gestor/setor/planilha /home/gestor/Documentos

C

$ head
$ mkdir ./setor
$ cd /home/gestor/setor/planilha /home/gestor/Documentos

D

$ tail
$ cd setor
$ mv ./planilha /home/gestor/Documentos

E

$ tail
$ mkdir ./setor
$ mv ./planilha /home/gestor/Documentos

A alternativa B está correta.

O comando pwd mostra em qual diretório o usuário se encontra. O comando cd permite acessar diretórios, veja que o ponto em "./setor" significa "este diretório". O comando mv move arquivos. Note que foram digitados todos os caminhos completos.

Considerações finais

- O sistema operacional surgiu para tornar o uso do computador mais fácil e conveniente para o usuário.
- A ideia de abstração em sistemas de computação é fundamental para entender os sistemas operacionais.
- O sistema operacional é um conjunto de rotinas que abstrai as camadas inferiores e o hardware.
- Essas rotinas geralmente constituem o núcleo do sistema operacional.
- Algumas rotinas podem ser executadas no espaço do usuário, dependendo da estrutura do núcleo.
- O modelo de máquina de camadas ajuda a organizar essas abstrações.
- Existem muitos sistemas operacionais diferentes ao longo da história.
- O Linux é um dos sistemas operacionais mais conhecidos e amplamente utilizados.
- O Linux oferece uma enorme variedade de distribuições para os usuários.
- A diversidade de distribuições Linux atende à diferentes necessidades e preferências.

Podcast

Escute um resumo sobre as principais características, tipos e estruturas de sistemas operacionais, kernel, chamadas de sistemas e modos de acesso ao núcleo.

Conteúdo interativo

Acesse a versão digital para ouvir o áudio.

Explore +

Para saber mais sobre os assuntos tratados neste tema, pesquise na internet:

- Comunidade Open Source — Open Source (Código Aberto).
- GuiaFoca.
- Linux From Scratch!
- O Sistema Operacional GNU e o Movimento de Software Livre, GNU.
- Open Source Initiative, Opensource.

Para saber mais sobre os assuntos tratados neste tema, pesquise na internet:

- Crash Course Ciência da Computação, Episódio 2, YouTube.
- Dentro do seu computador — Bettina Bair, YouTube.

Referências

MACHADO, F. B.; MAIA, L. P. M. **Arquitetura de sistemas operacionais**. 4. ed. Rio de Janeiro: LTC, 2007.

MACHADO, F. B.; MAIA, L. P. M. **Arquitetura de sistemas operacionais**. 5. ed. Rio de Janeiro: LTC, 2013.

MACHADO, F. B.; MAIA, L. P. M. **Arquitetura de sistemas operacionais** – Material Suplementar para Acompanhar. 5. ed. Rio de Janeiro: LTC, 2013.

SILBERSCHATZ, A. **Fundamentos de sistemas operacionais** – Princípios básicos. 1. ed. Rio de Janeiro: LTC, 2013.

TANENBAUM, A. S. **Sistemas Operacionais Modernos**. 3. ed. São Paulo: Pearson, 2010.

TANENBAUM, A. S.; BOS, H. **Sistemas Operacionais Modernos**. 4. ed. São Paulo: Pearson, 2016.