

DESCRIÇÃO

A 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.

PROPÓSITO

Compreender os conceitos básicos que formam os sistemas operacionais é importante para a solidificação do seu leque de conhecimentos fundamentais em sistemas de computação, possibilitando expandir a percepção de mais detalhes relacionados aos assuntos que envolvem computadores.

OBJETIVOS

MÓDULO 1

Descrever a evolução histórica dos sistemas operacionais

MÓDULO 2

Identificar os tipos de sistemas operacionais

MÓDULO 3

Compreender a estrutura do SO: Kernel, system calls, modos de acesso

MÓDULO 4

Analisar a arquitetura, instalação do Linux e comandos básicos

INTRODUÇÃO

Qualquer sistema computacional é composto pelo conjunto de usuários, hardware e software. Dentre os softwares que são utilizados, podemos separá-los nos 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 um pouco de 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, dentre outros elementos, do kernel, chamadas de sistemas e modos de acesso ao núcleo.

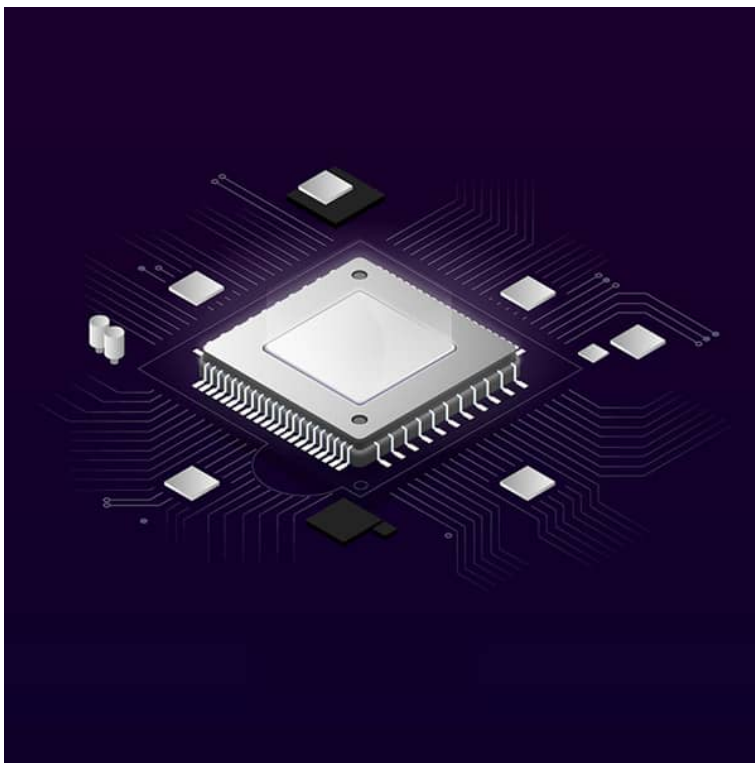
Por fim, vamos verificar como é a utilização básica de um sistema operacional, aplicando conceitos ao uso do SO Linux.

MÓDULO 1

CONCEITOS

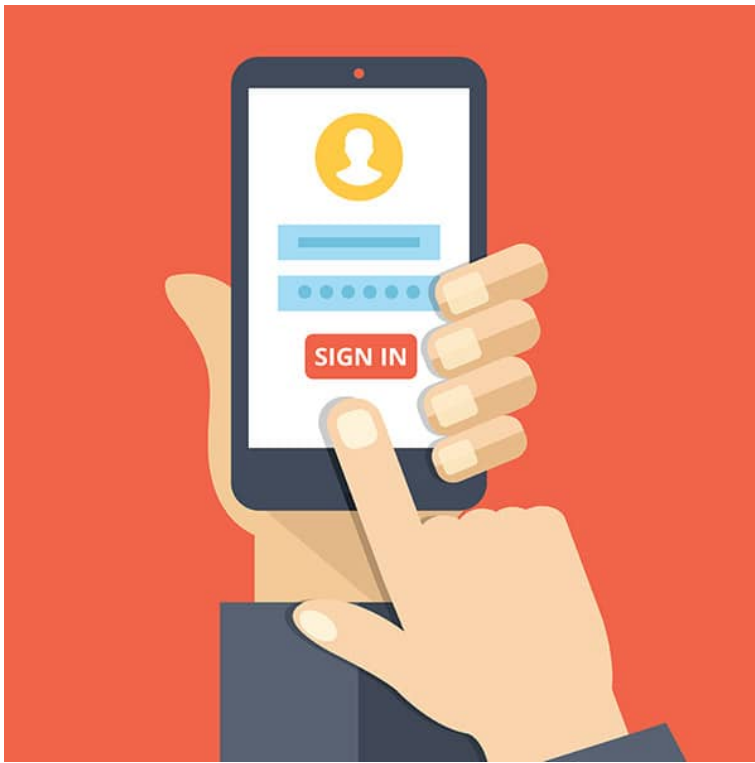
Estamos prestes a realizar o nosso estudo em sistemas operacionais. Esse assunto é importante para todos os profissionais que buscam exercer plenamente suas atividades nas diversas áreas da computação, como administradores de sistemas, programadores de aplicações concorrentes, gerentes de segurança e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



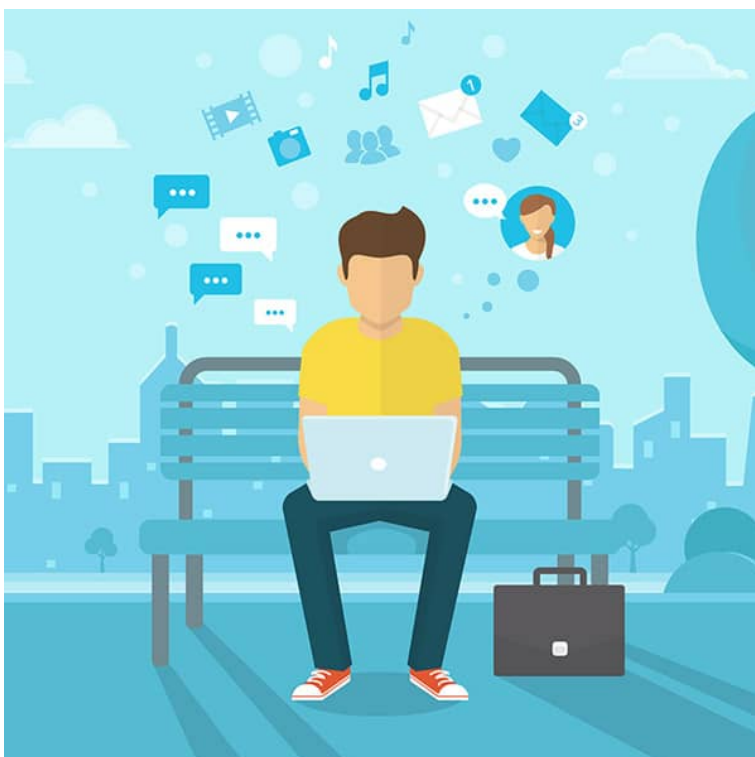
HARDWARE

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

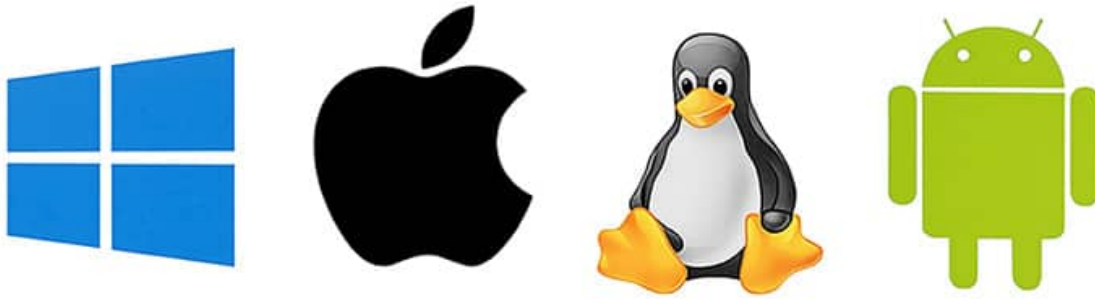
Controla e coordena o uso do hardware entre os vários programas de aplicação, para os diversos usuários.

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 (abreviado muitas vezes por 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. Porém, há uma particularidade:

O sistema operacional 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.



📷 Figura 1 – Sistemas Operacionais. Fonte: Rose Carson/Shutterstock.

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 usem 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 baseado em interface gráfica com ícones (GUI - Graphical User Interface).

Em um computador, os programas podem ser executados em modo usuário ou kernel. No modo usuário, os softwares têm acesso limitado ao hardware e normalmente estão os programas e aplicativos utilizados diretamente pelos usuários.

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.

❓ VOCÊ SABIA

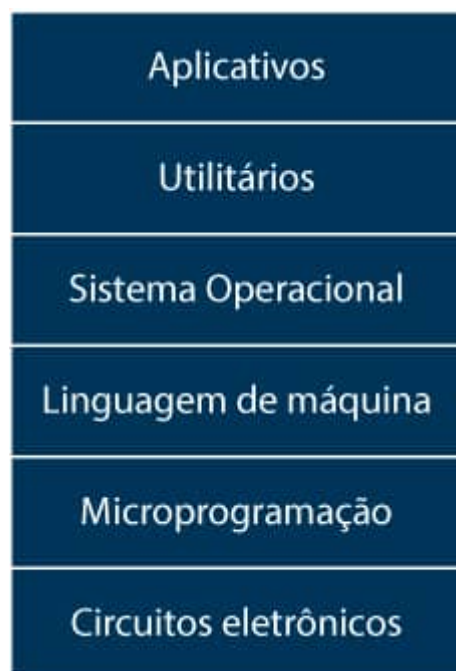
Nos primeiros computadores, a programação era toda feita através de painéis físicos, exigindo do programador um grande conhecimento do hardware, mas o hardware em si possuía pouca utilidade para o usuário devido à sua complexidade.

O surgimento do sistema operacional permitiu que o hardware pudesse ser utilizado de forma mais eficiente, levando em consideração que o SO consegue disponibilizar de forma mais eficiente os serviços aos usuários, **modularizando** e **abstraíndo** 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 figura 2.

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



📷 Figura 2 – Máquina de níveis.

Fonte: Adaptado de Machado, 2007.



HISTÓRICO DOS SISTEMAS COMPUTACIONAIS

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.

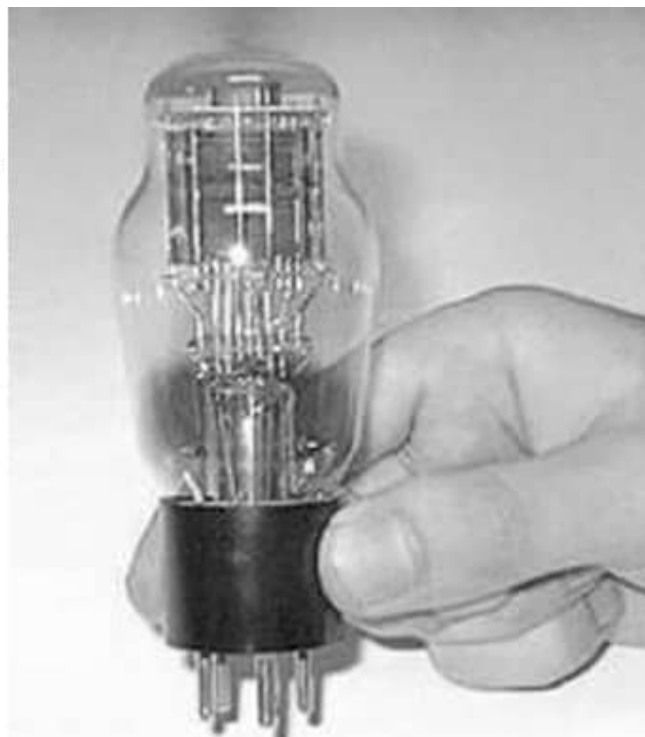
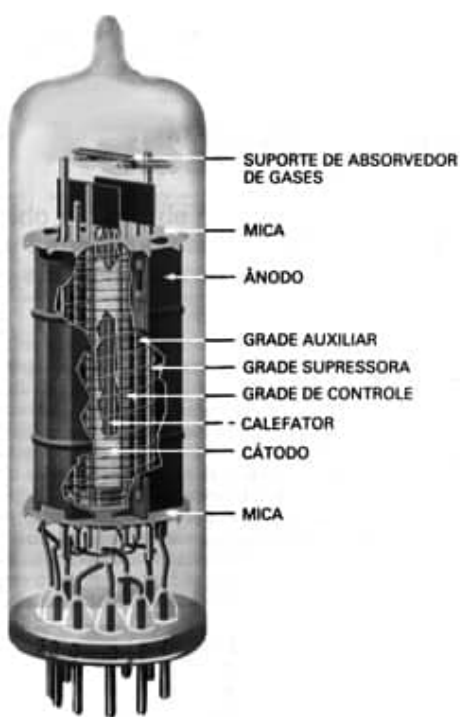
A história dos sistemas operacionais é bastante relacionada à arquitetura de computadores. Sendo assim, veremos brevemente as várias gerações de computadores com o objetivo de entender as primeiras versões de sistemas operacionais.

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 (figura 3), e os componentes de hardware eram mais primitivos, por exemplo, empregando válvulas (figura 4) na composição dos circuitos lógicos.



📷 Figura 3 – Programação realizada em painéis físicos. Fonte: Produção Virtual UFPB.



📷 Figura 4 – Válvula. Fonte: Produção Virtual UFPB.

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 figura 5. O significado das etapas da figura encontra-se a seguir:

(a) Os programadores levavam os cartões para o leitor de cartões.

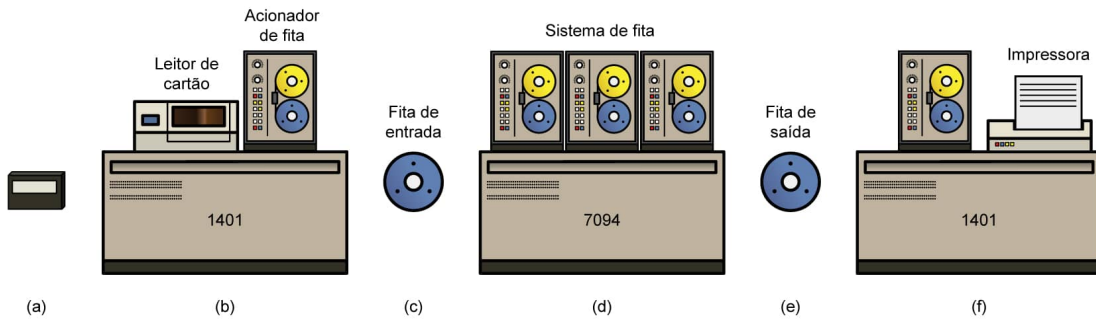
(b) O leitor de cartões gravava o lote de tarefas em fita.

(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.



📷 Figura 5 - Imagem ilustrativa do sistema em lote (batch) da segunda geração. Fonte: Adaptado de Tanenbaum, 2010.

TERCEIRA GERAÇÃO

Foi na **terceira geração**, compreendida 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.

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 dos quais exibidos na tabela 1.

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



MS WINDOWS E UNIX

Historicamente, existiram (e existem) 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



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.

MS 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 NT

Paralelo às versões baseadas no MS-DOS, o Windows NT foi lançado em 1993, sendo um sistema de 32 bits, com multitarefa preemptiva, **multithread**, memória virtual e suporte a múltiplos processadores simétricos, nas versões para desktop e servidores. O Windows NT possuía um núcleo completamente novo, mas oferecia compatibilidade parcial com aplicações legadas dos sistemas baseados em MS-DOS, e possuía a mesma interface gráfica das versões existentes.

WINDOWS 2000

O Windows 2000 foi uma evolução do Windows NT 4, utilizando a mesma arquitetura interna, e novas funções, tais como o plug and play e o serviço de diretórios Active Directory.



FOTO BASE





WINDOWS XP

A partir do Windows XP, lançado em 2001, a Microsoft começou a descontinuar as famílias DOS-Windows e Windows NT/2000, integrando as duas linhas de sistemas operacionais.

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).



MULTITHREAD

É a capacidade do sistema operacional de executar várias threads (tarefas), de forma simultânea, sem uma interferir na outra.

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.



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, que foi um dos pesquisadores envolvidos no projeto do MULTICS, desenvolveu sua própria versão em Assembly para um minicomputador PDP-7, que veio a se chamar UNICS (UNiplexed Information and Computing Service) e, mais para frente, Unix. Com o objetivo de 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.



FOTO BASE





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

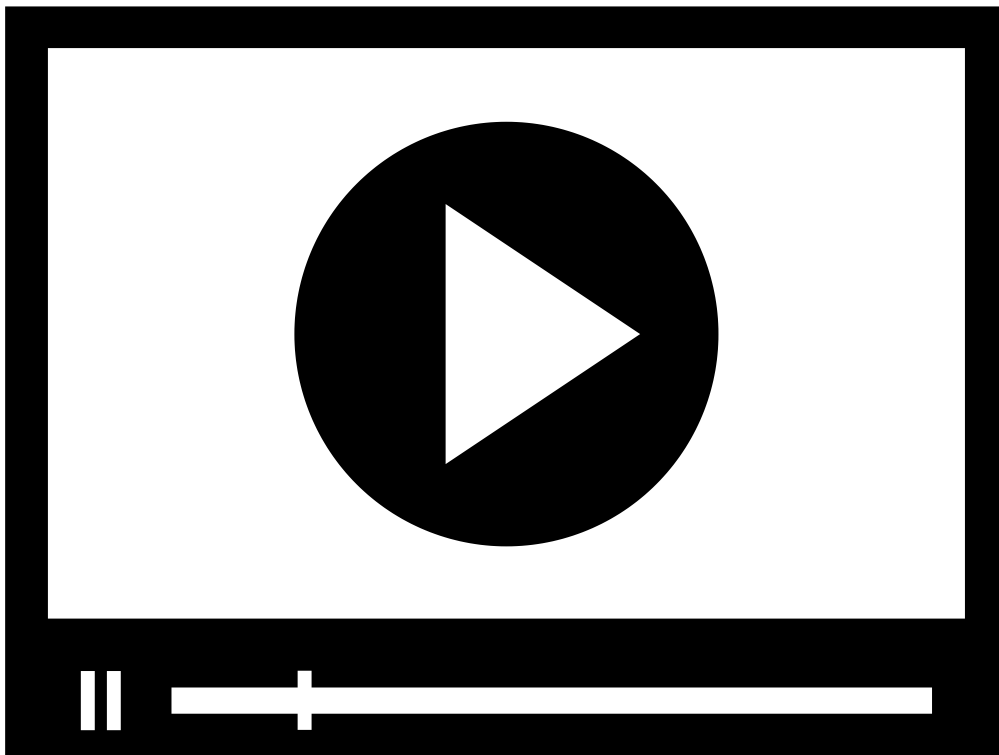
O finlandês Linus Torvalds iniciou o desenvolvimento do Linux, baseado no Minix, uma variante do Unix desenvolvido pelo professor Andrew Tanenbaum, da Holanda, com fins educacionais.



ATENÇÃO

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

A mais importante tentativa de unificação das várias 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.



Antes de finalizar o módulo, assista ao vídeo e acompanhe a evolução dos sistemas operacionais:



VERIFICANDO O APRENDIZADO

MÓDULO 2

-
- ⦿ Identificar os tipos de sistemas operacionais

TIPOS DE SISTEMAS OPERACIONAIS

Os sistemas operacionais podem ser classificados em: Sistemas operacionais monoprogramáveis/monotarefa; sistemas operacionais multiprogramáveis/multitarefa e sistemas com múltiplos processadores.



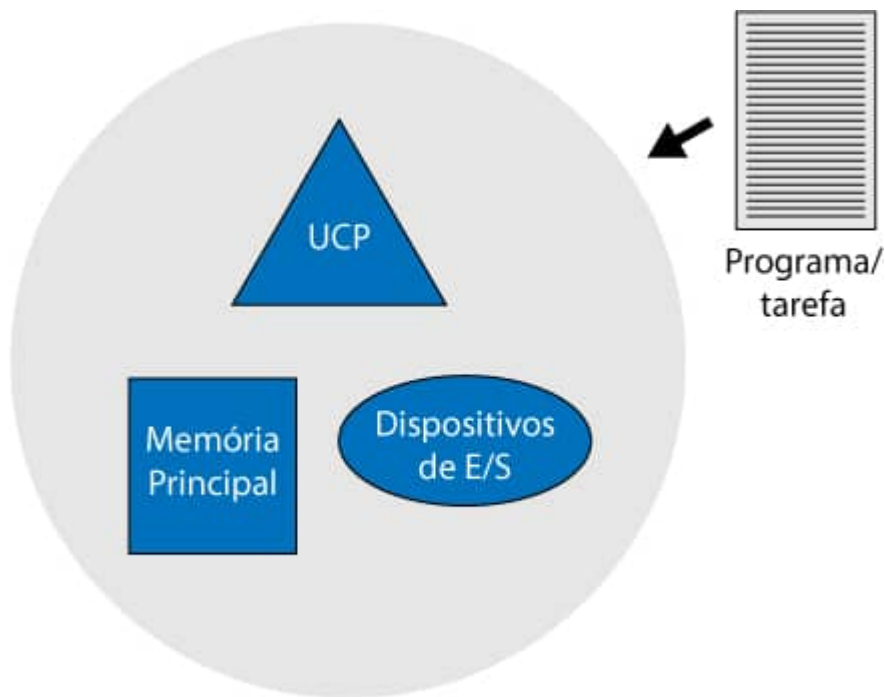
📷 Figura 6 – Tipos de sistemas operacionais.

Fonte: Adaptado de Machado, 2007.

Vejamos agora cada um deles:

SISTEMAS MONOPROGRAMÁVEIS

Nos sistemas **monoprogramáveis** (figura 7), também chamados de 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 deve esperar para poder ser processada.



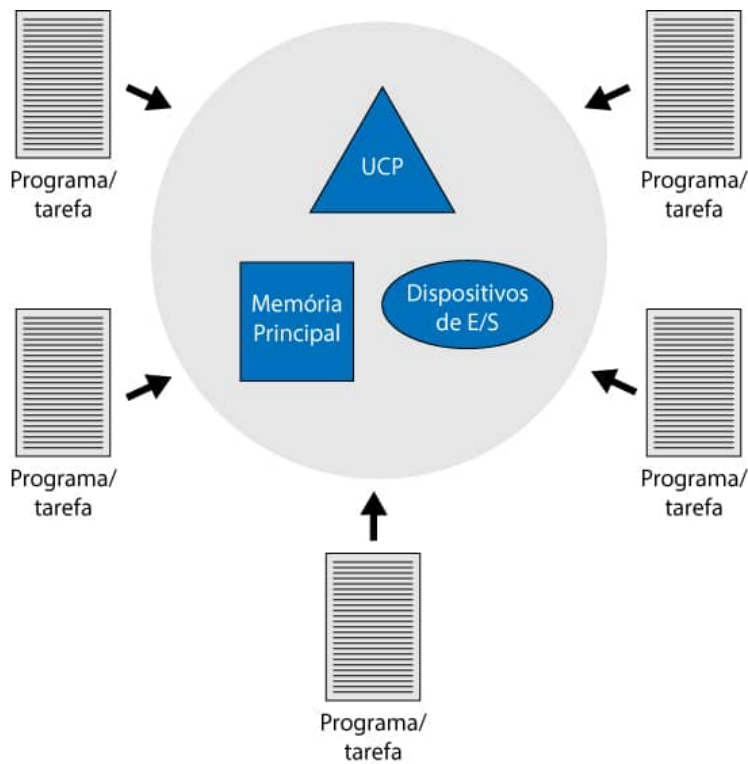
📷 Figura 7 –Sistemas monoprogramáveis/monotarefa.

Fonte: Adaptado de Machado, 2007.

SISTEMAS MULTIPROGRAMÁVEIS OU MULTITAREFA

Nos sistemas **multiprogramáveis** (figura 8), 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.



📷 Figura 8 – Sistemas multiprogramáveis/multitarefas.

Fonte: Adaptado de Machado, 2007.

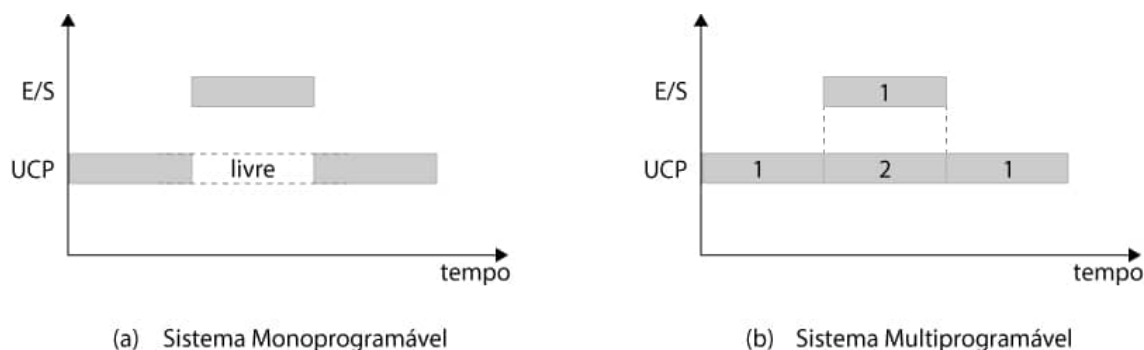
Os sistemas multiprogramáveis podem ser classificados em sistemas batch, de tempo compartilhado ou de tempo real. O sistema operacional é responsável por gerenciar vários desses programas ao mesmo tempo, conforme ilustrado na figura 9.



📷 Figura 9 – Classificação dos sistemas multiprogramáveis.

Fonte: Adaptado de Machado, 2007.

Um dos problemas que ocorriam nos sistemas monoprogramáveis é que havia um desperdício na utilização do processador (figura 10). Por exemplo, 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.

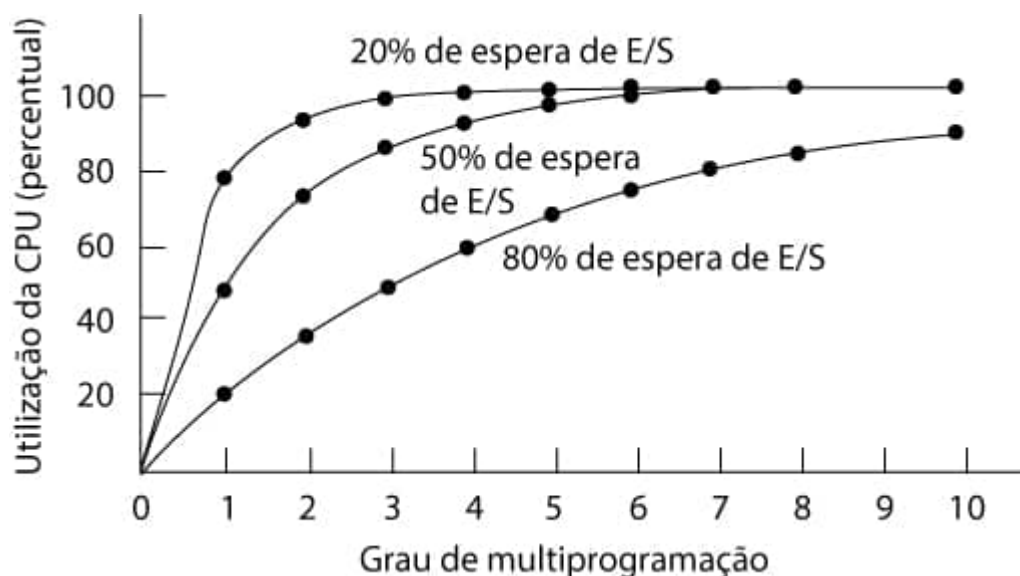


📷 Figura 10 – Sistemas monoprogramáveis x sistemas multiprogramáveis.

Fonte: Adaptado de Machado, 2007.

Em sistemas multiprogramáveis, 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 figura 11 apresenta um gráfico representando a utilização da CPU versus o grau de multiprogramação.



📷 Figura 11 – Utilização da CPU como função do número de processos na memória.

Fonte: Adaptado de Machado, 2007.

📢 ATENÇÃO

Pode-se perceber através da análise da figura 11 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%.

SISTEMAS COM MÚLTIPLOS PROCESSADORES

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.

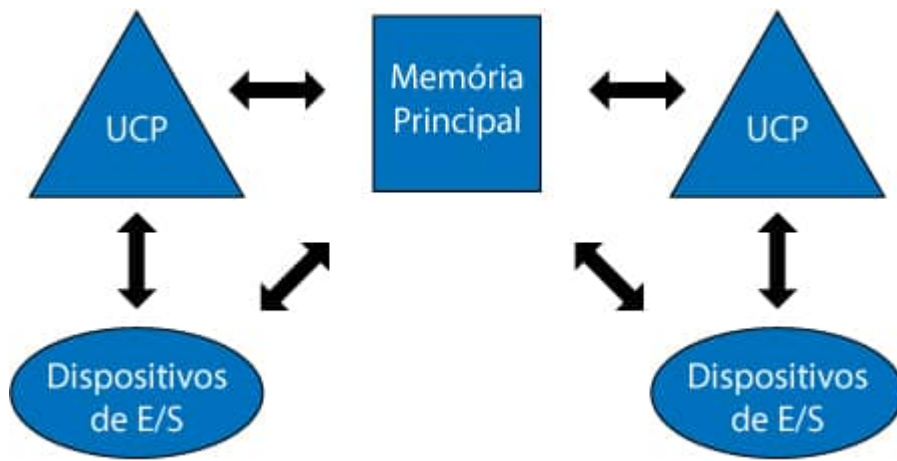
BALANCEAMENTO DE CARGA

Distribuição da carga de trabalho entre os processadores.

Em sistemas com múltiplos processadores, os processadores podem se comunicar de duas maneiras: Nos **sistemas fortemente acoplados** e **sistemas fracamente acoplados**.

SISTEMAS FORTEMENTE ACOPLADOS

Nos sistemas fortemente acoplados, os processadores compartilham somente a memória principal e os periféricos são gerenciados por um único sistema operacional.



📷 Figura 12 - Sistemas fortemente acoplados.

Fonte: Adaptado de Machado, 2007.

Os sistemas fortemente acoplados podem ser:

SMP - SYMMETRIC MULTIPROCESSORS

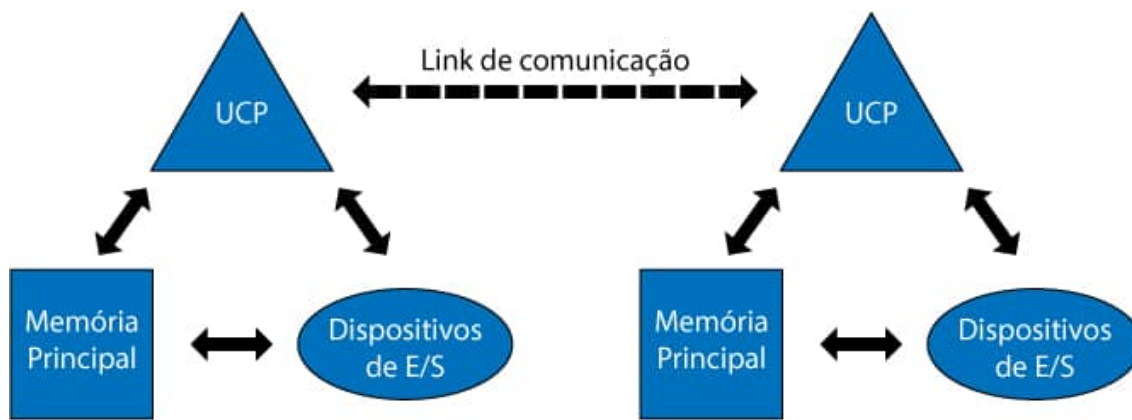
Os processadores acessam a memória uniformemente ao longo do tempo.

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.

SISTEMAS FRACAMENTE ACOPLADOS

Já nos **sistemas fracamente acoplados** (figura 13), 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.



📷 Figura 13 - Sistemas fracamente acoplados. Fonte: Adaptado de Machado, 2007.

Os sistemas com múltiplos processadores também podem ser chamados "**Sistemas Avançados de Processamento**". Eles podem ser constituídos de computadores com vários processadores, processadores com vários núcleos e sistemas distribuídos. O S.O. deve estar adaptado para operar esses sistemas.



OUTRAS CLASSIFICAÇÕES DOS SISTEMAS OPERACIONAIS

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 S.O. foi elaborado oferecendo um **conjunto de serviços suportados**. Por exemplo, alguns dos serviços que podem ser

especificamente oferecidos pelos sistemas **operacionais de computadores de grande porte** são:

(1) O processamento simultâneo de muitas tarefas.

(2) O suporte ao manejo de grandes quantidades de E/S.

(3) O oferecimento dos serviços de processamento em lote (batch), de processamento de transações e de execução de tarefas em tempo compartilhado.

Quanto aos serviços citados em (3), temos

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.

Já os serviços de **sistemas de tempo real** possuem o **tempo** como parâmetro fundamental. Por exemplo: Linha de montagem de um carro.

Os sistemas de tempo real podem ser dos tipos:



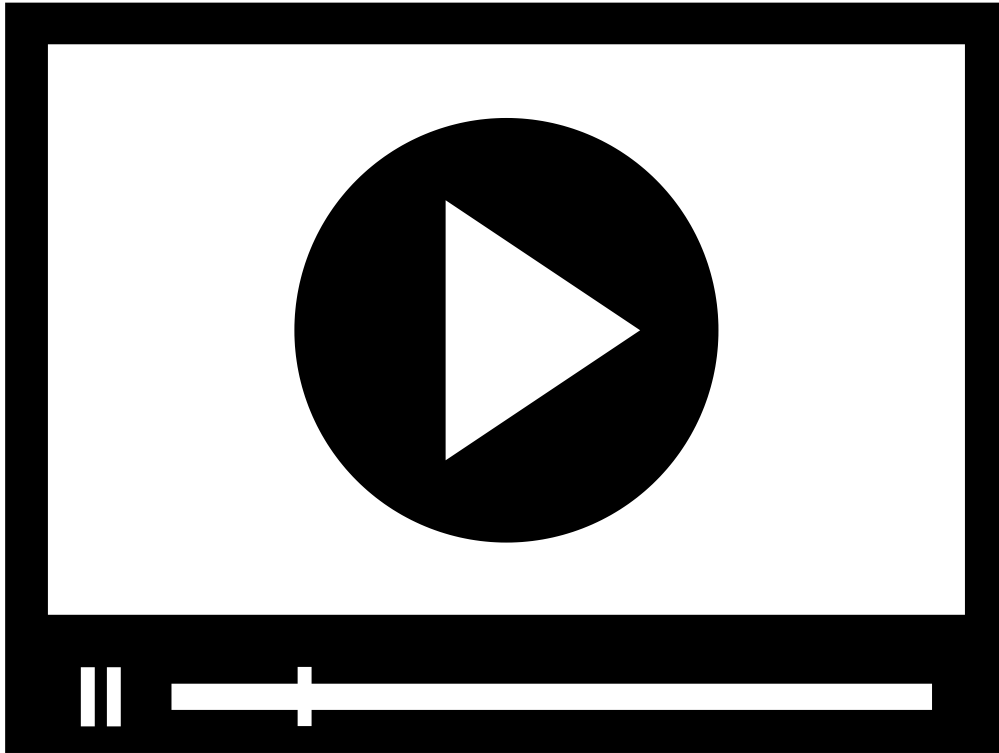
SISTEMA DE TEMPO REAL CRÍTICO

As ações precisam ocorrer em determinados instantes. Exemplo: Processos industriais, aviação, militares.



SISTEMA DE TEMPO REAL NÃO CRÍTICO

O descumprimento de um prazo não causa dano permanente. Exemplo: Sistema de áudio digital, multimídia, telefones digitais.



Existem outros tipos de sistemas que oferecem um conjunto de serviços específicos para a finalidade das quais foram concebidos.

Assista ao vídeo abaixo e conheça alguns:



COMENTÁRIO

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.

Outra situação que acontece muitas vezes é certos tipos de SOs e de rotinas que já existiram em algum momento no passado caírem em desuso e voltarem 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

Um exemplo é o uso da memória cache nas CPUs quando se tornam mais velozes que as memórias. 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.

Para terminar, os sistemas operacionais variam quanto aos tipos de licenças, ou seja, o conjunto de ações que o usuário do SO pode ou não fazer:

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.

Diferente das liberdades do software livre.

O termo "código aberto" foi criado pela OSI (Open Source Initiative).

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)".

Qualquer licença de software livre é também uma licença de código aberto (Open Source), mas o contrário nem sempre é verdade.

Exemplos de licenças de código aberto:

Apache License.

MIT License.

Mozilla Public License.

Common Development and Distribution License.

Eclipse Public License.

VERIFICANDO O APRENDIZADO

MÓDULO 3

- ⦿ Compreender a estrutura do SO: Kernel, system calls, modos de acesso

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 figura 14.



📷 Figura 14 - Kernel do SO na máquina de níveis.

Fonte: Adaptado de Machado, 2007.

SAIBA MAIS

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.

Existem eventos relacionados ao hardware e eventos relacionados às tarefas internas do próprio SO.

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.

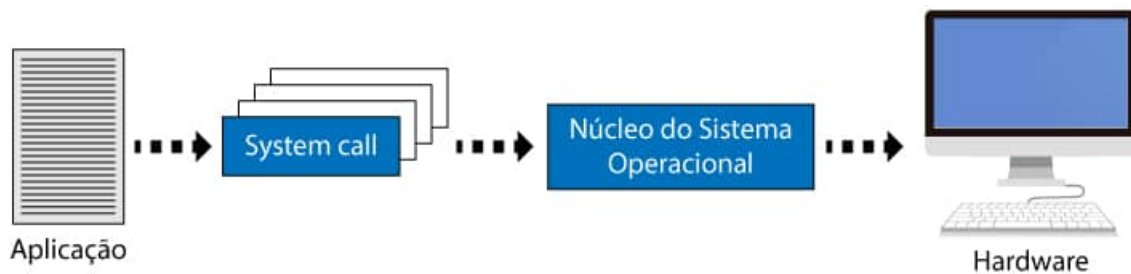
Suporte a redes locais e distribuídas.

Contabilização do uso do sistema.

Auditoria e segurança do sistema.

Etc.

A execução das rotinas no sistema operacional é controlada pelo mecanismo denominado **system call** (figura 15), ou **chamada ao sistema**, um mecanismo de proteção por software que garante a execução somente de rotinas previamente autorizadas.



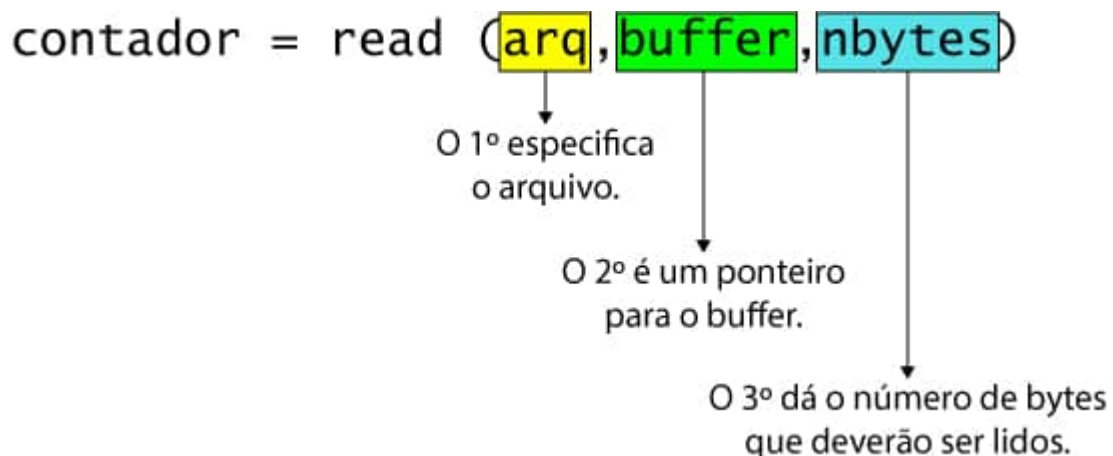
📷 Figura 15 – System call. Fonte: Adaptado de Machado, 2007.

SYSTEM CALLS

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.

Por exemplo, a chamada ao sistema "read" (figura 16) possui três parâmetros:



📷 Figura 16 – Chamada ao sistema "read". Fonte: Adaptado de Tanenbaum, 2010.

Os passos para a realização da system call "read" estão ilustrados na figura 17:

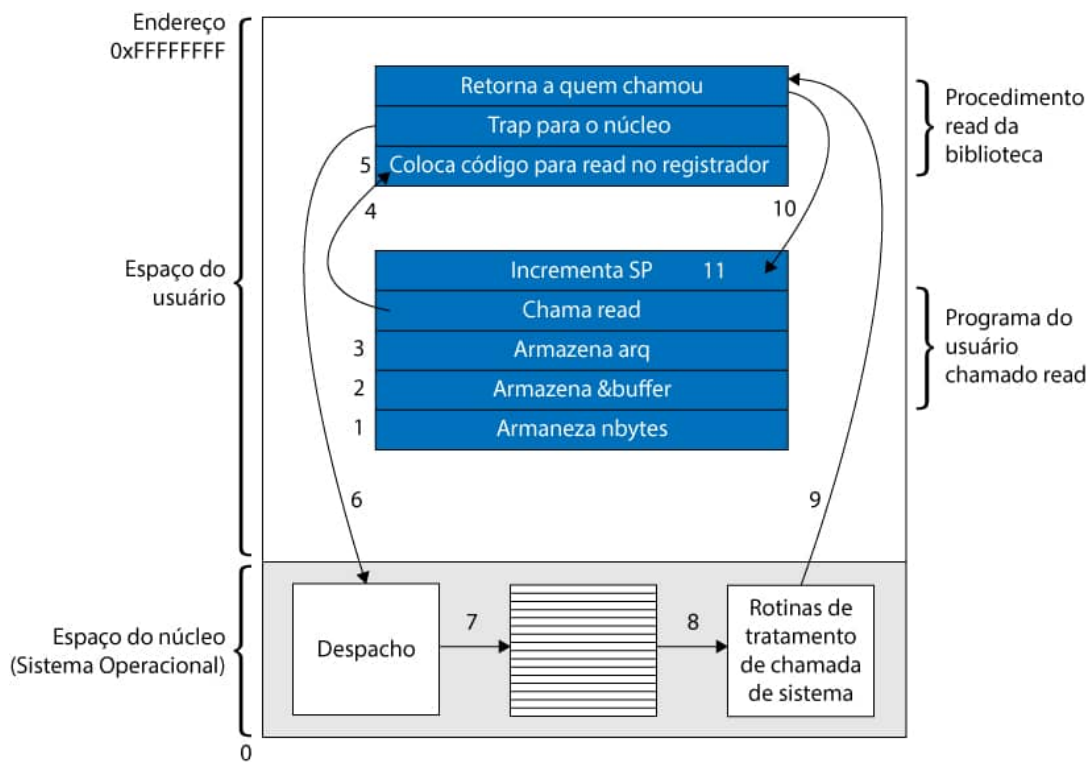


Figura 17 - Passos para a realização da system call "read".

Fonte: Adaptado de Tanenbaum, 2010.

Observe uma breve descrição a seguir:

PASSOS 1 A 3

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

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 S.O., por exemplo, em um registrador (o passo 5).

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.

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.

PASSO 8

É a execução da rotina de tratamento da chamada de sistema.

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).

PASSO 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 (figura 18) consiste em:

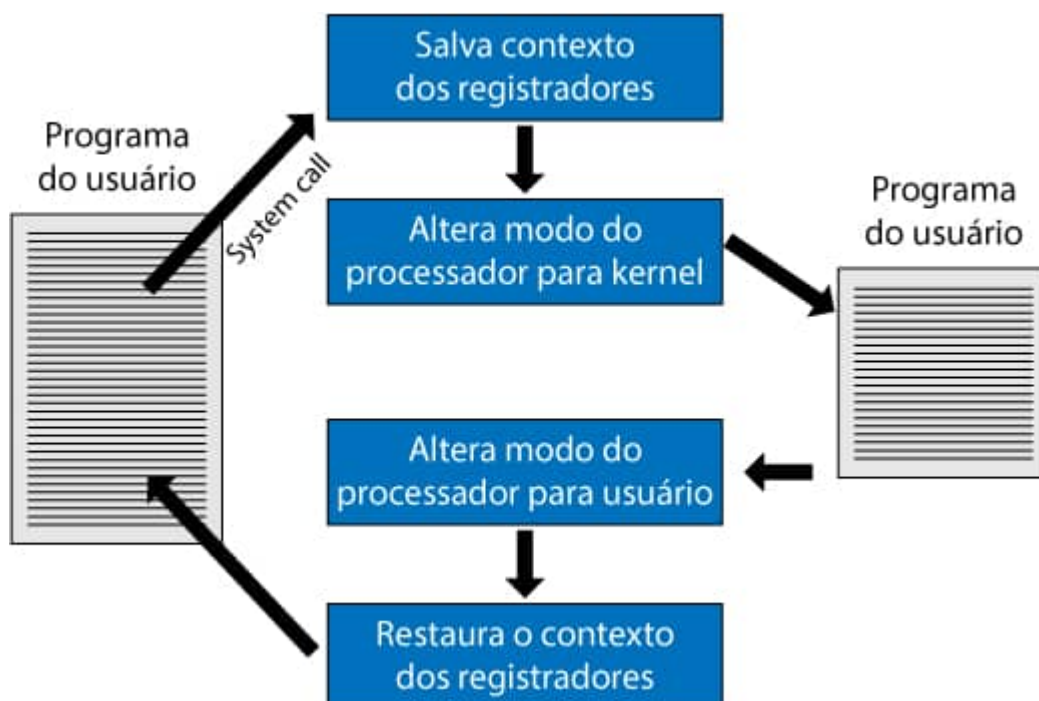
Realização da chamada ao sistema pela aplicação que deseja executar rotina(s).



Processamento da solicitação pelo kernel.



Retorno, para a aplicação solicitante, de uma resposta, com um estado de conclusão, indicando se houve algum erro.



📷 Figura 18 – Chamada a uma rotina de sistema.

Fonte: Adaptado de Machado, 2007.

★ EXEMPLO

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.).

A API Win32 possui diferenças em relação ao POSIX.

Uma delas 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.

A API Win32 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 2 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	Destroi 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

Atenção! Para visualizaçãocompleta da tabela utilize a rolagem horizontal

Tabela 2 – Chamadas da API Win32 que correspondem aproximadamente às chamadas do POSIX. Fonte: Adaptado de Tanenbaum, 2010.

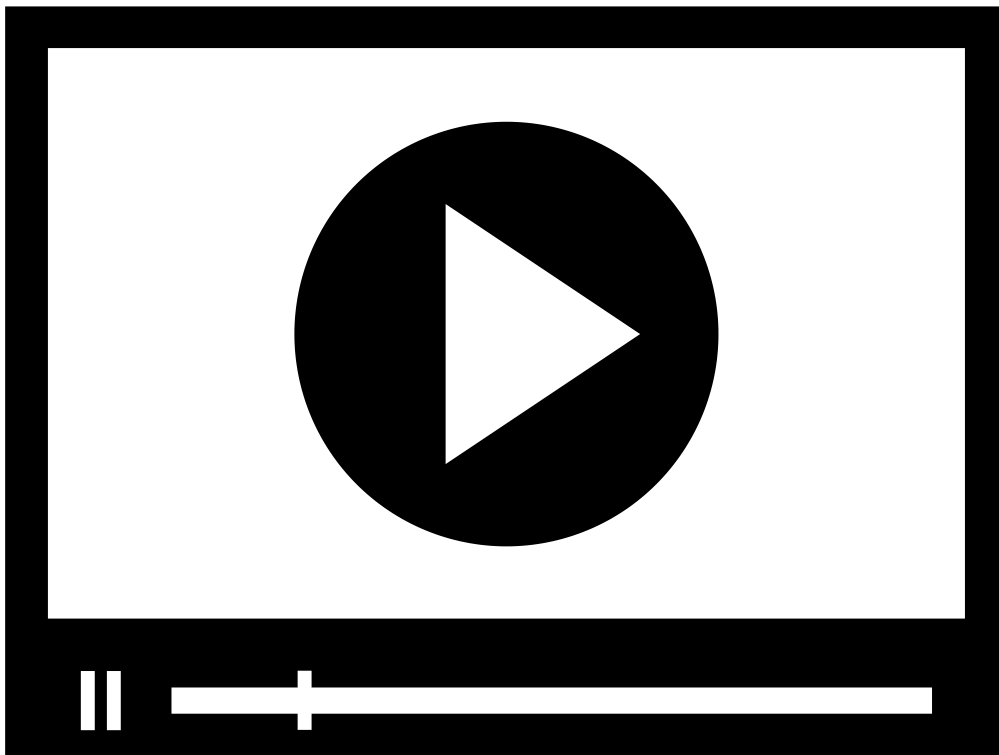
Podemos classificar os tipos de chamadas ao sistema em: Chamadas para o gerenciamento de **processos**, **arquivos** e **diretórios**. A Tabela 3 mostra alguns exemplos desses tipos de chamadas.

<i>Gerenciamento de processos</i>	
Chamada	Descrição
pid=fork()	Cria um processo filho idêntico ao pai
pid=waitpid(pid, &statloc, options)	Espera que um processo filho seja concluído
<i>Gerenciamento de arquivos</i>	
Chamada	Descrição
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
Gerenciamento do sistema de diretório e arquivo	
Chamada	Descrição
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
Diversas	
Chamada	Descrição
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

Atenção! Para visualizaçãocompleta da tabela utilize a rolagem horizontal

Tabela 3 - Exemplos de tipos de chamadas ao sistema. Fonte: Adaptado de Tanenbaum, 2010.



Agora, confira maiores detalhes sobre operação das system calls e detalhes do sistema POSIX e dos sistemas Win32:



MODOS DE ACESSO

As instruções executadas pelas aplicações podem ser dos tipos:

PRIVILEGIADAS

Incluem as instruções que de algum modo podem comprometer o sistema.

NÃO PRIVILEGIADAS

São as instruções que não oferecem riscos ao sistema.

As instruções podem acessar o sistema de dois modos:

MODO KERNEL OU SUPERVISOR

Uma aplicação possui o modo de acesso supervisor quando é constituída de instruções privilegiadas. A aplicação 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.

❓ VOCÊ SABIA

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.

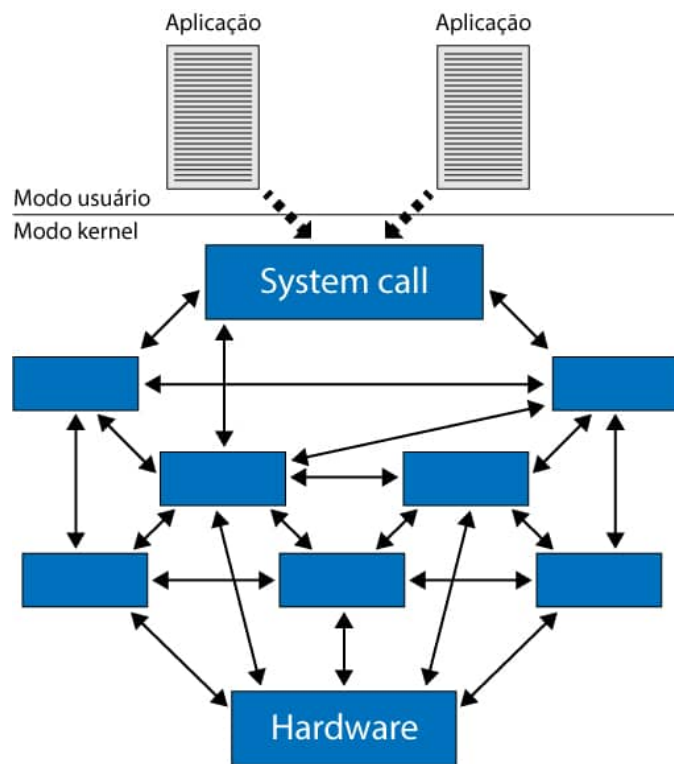
ARQUITETURAS DE KERNEL

Existem variações quanto à concepção da estrutura do núcleo do sistema operacional. As principais arquiteturas dos sistemas operacionais são:

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. Uma ilustração se encontra na figura 19. 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.



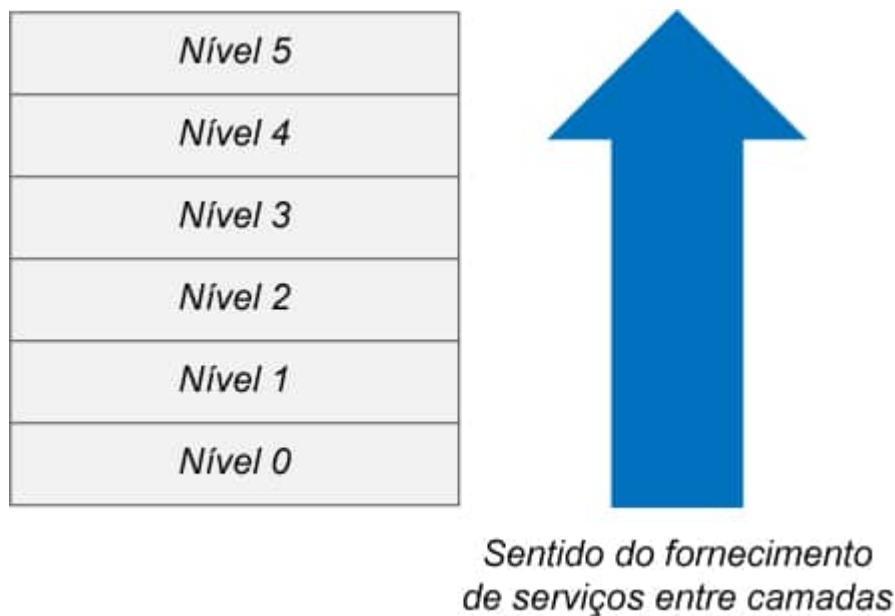
📷 Figura 19 – Arquitetura monolítica.

Fonte: Adaptado de Machado, 2007.

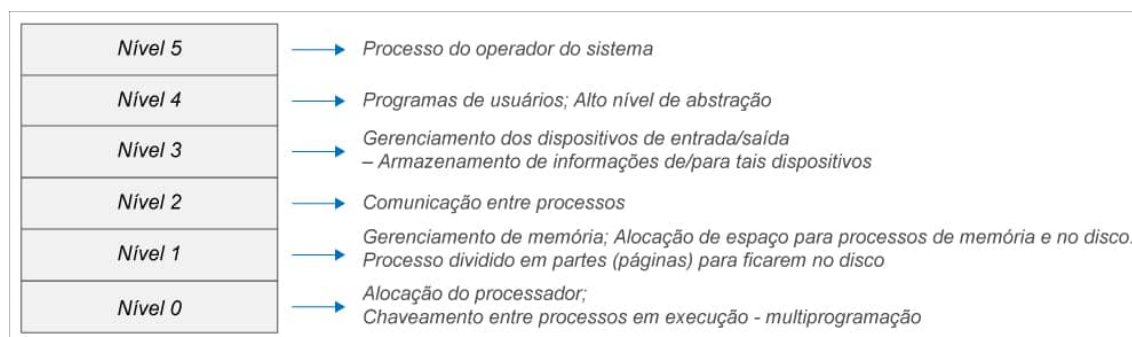
ARQUITETURA DE CAMADAS

Nessa arquitetura, o sistema é dividido em várias camadas, que oferecem um conjunto de funções. Cada camada oferece suas funções para a camada imediatamente superior. 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, usam duas camadas.



📷 Figura 20 – Arquitetura de camadas.



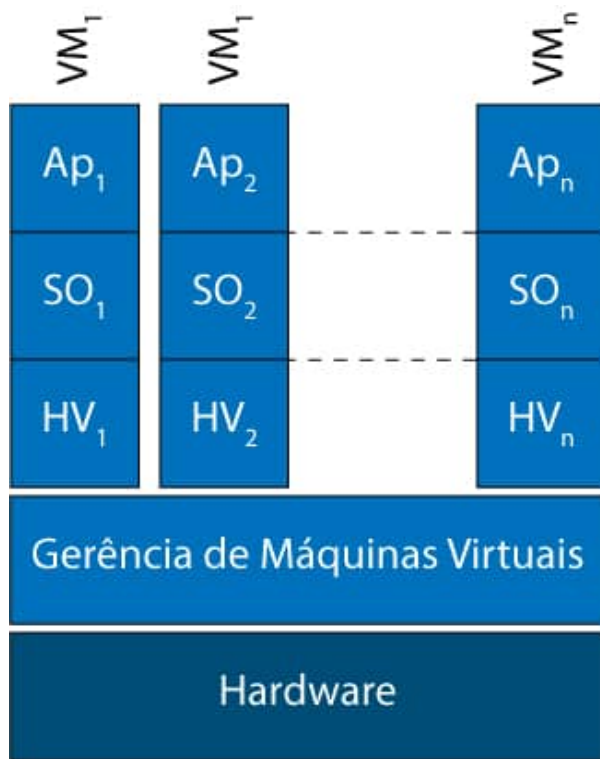
📷 Figura 21 – Arquitetura de camadas.

Fonte: Adaptado de Tanenbaum, 2010.

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.



📷 Figura 22 – Máquina virtual.

Fonte: Adaptado de Machado, 2007.

💬 COMENTÁRIO

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.

Quando os clientes alugam uma máquina virtual, uma mesma máquina física suporta muitas máquinas virtuais simultaneamente, logo poderão executar os sistemas operacionais e softwares que quiserem por uma parte do custo de um servidor dedicado. Pode-se usar a virtualização também para executar dois ou mais sistemas operacionais ao mesmo tempo.

Para executar o software de máquina virtual em um computador, a CPU deve ser virtualizável.

❓ VOCÊ SABIA

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.

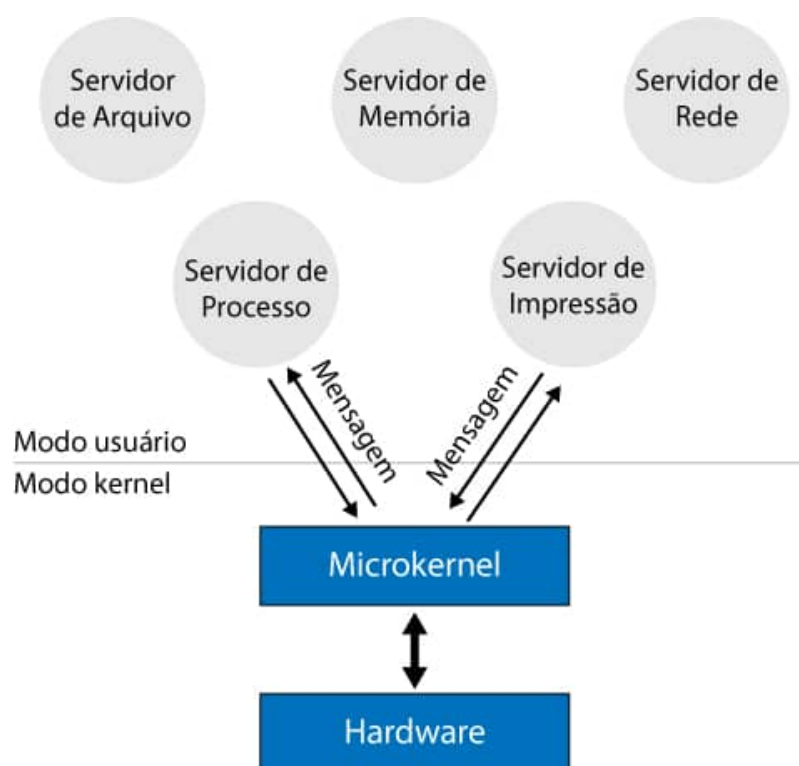
Essa situação deixou de existir ao substituir os hypervisors de tipo 1, executados diretamente no hardware, pelos hypervisors de tipo 2, executados como programas aplicativos na camada superior do sistema operacional, conhecida como sistema operacional hospedeiro. O hypervisor é o monitor de máquina 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**.

É difícil de implementar na prática, sendo que usualmente é 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 que 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.

★ EXEMPLO

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.

VERIFICANDO O APRENDIZADO

MÓDULO 4

- ⦿ Analisar a arquitetura, instalação do Linux e comandos básicos

CONCEITOS

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).

SAIBA MAIS

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, ou ainda 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.



 Figura 24 – Logotipo do Ubuntu.

Fonte: wikimedia.

Cada versão do Ubuntu recebe uma numeração e um codinome, que seguem a lógica mostrada na tabela 4.

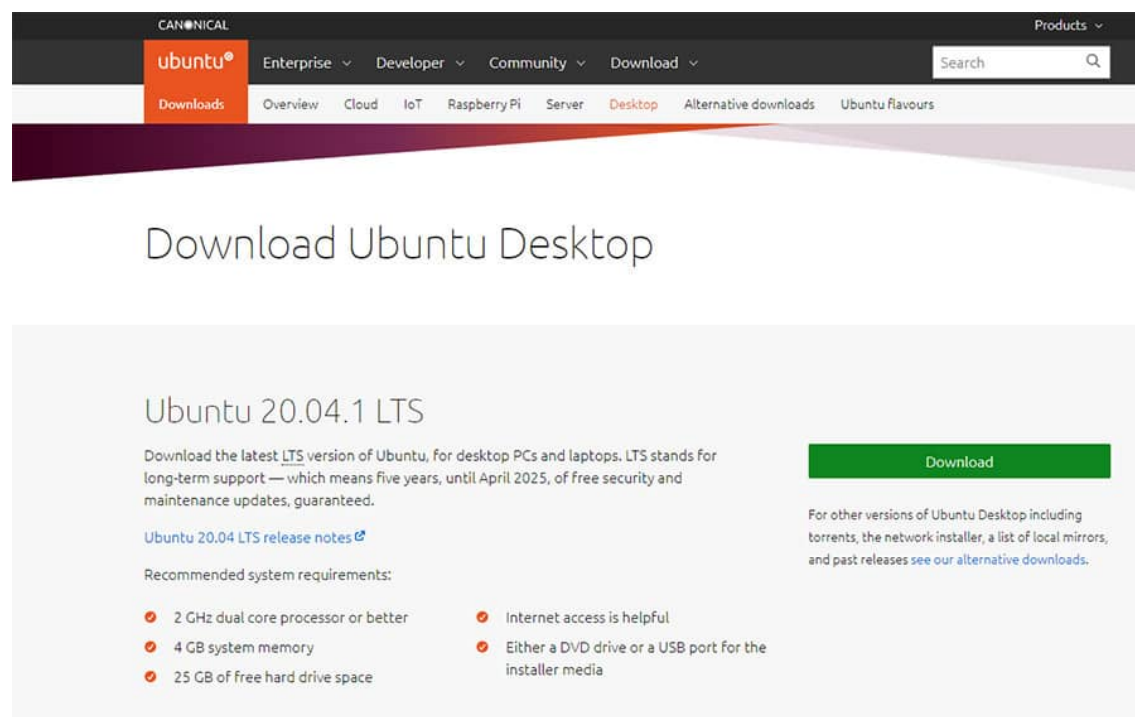
Denominação da versão	20.	04	LTS	Focal Fossa
Significado	Ano (2020)	Mês (04 – abril)	Long-Term Support (Suporte de longo prazo - 5 anos de suporte (atualizações))	Codinome

Atenção! Para visualizaçãocompleta da tabela utilize a rolagem horizontal

Tabela 4 – Exemplo de numeração e um codinome do Ubuntu.

Fonte: O Autor.

Faça o download do arquivo de imagem “iso” correspondente à versão atual do Ubuntu acessando a página do Ubuntu, conforme mostrado na figura 25.



📷 Figura 25 – Página para download do Ubuntu.

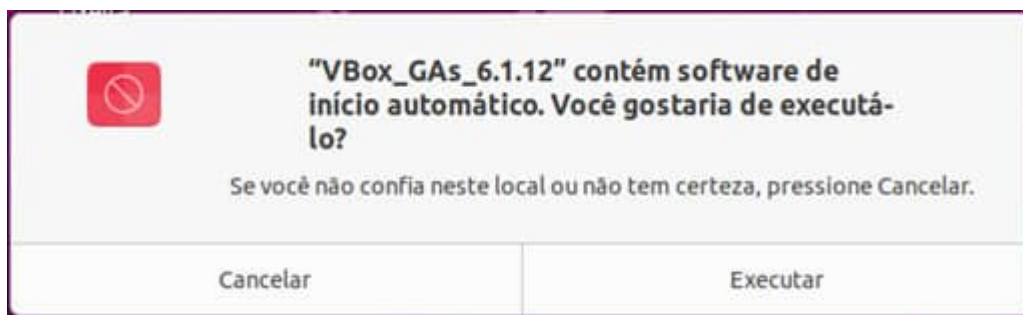
Fonte: O Autor.



Observe no vídeo a seguir, a instalação do Linux:

💡 DICA

Você poderá melhorar a utilização do S.O. 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.



📷 Figura 26 – Adicionais para convidado do VirtualBox.

Fonte: O Autor.

ESTRUTURA DE DIRETÓRIOS NO LINUX

O Linux possui uma estrutura única de pastas, que partem de uma única raiz, como ilustrado na figura 27.

Todos os dispositivos, internos ou externos, fixos ou removíveis, são montados em algum ponto abaixo da raiz.

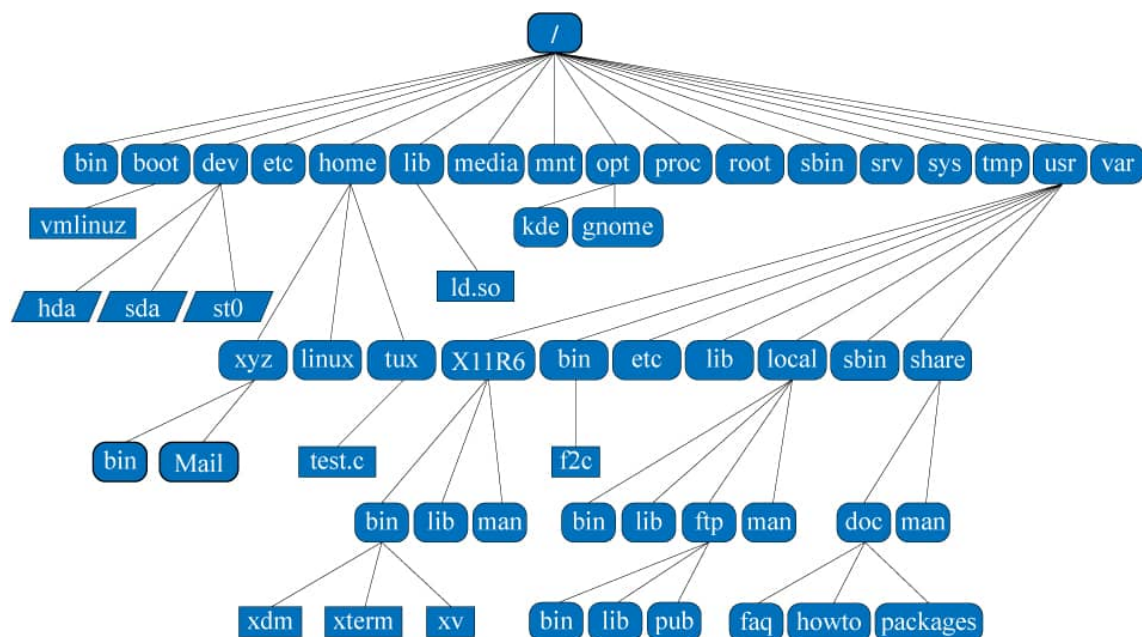


Figura 27 – Estrutura de pastas do Linux.

📢 ATENÇÃO

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. Por exemplo, "texto", "Texto" e "TEXT0" são três arquivos distintos. Quando for escrever um caminho de diretórios, use o separador "/". Exemplo: /tmp/Texto.txt.

As principais pastas e seus significados resumidos estão mostrados na tabela 5.

/	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

Atenção! Para visualizaçãocompleta da tabela utilize a rolagem horizontal

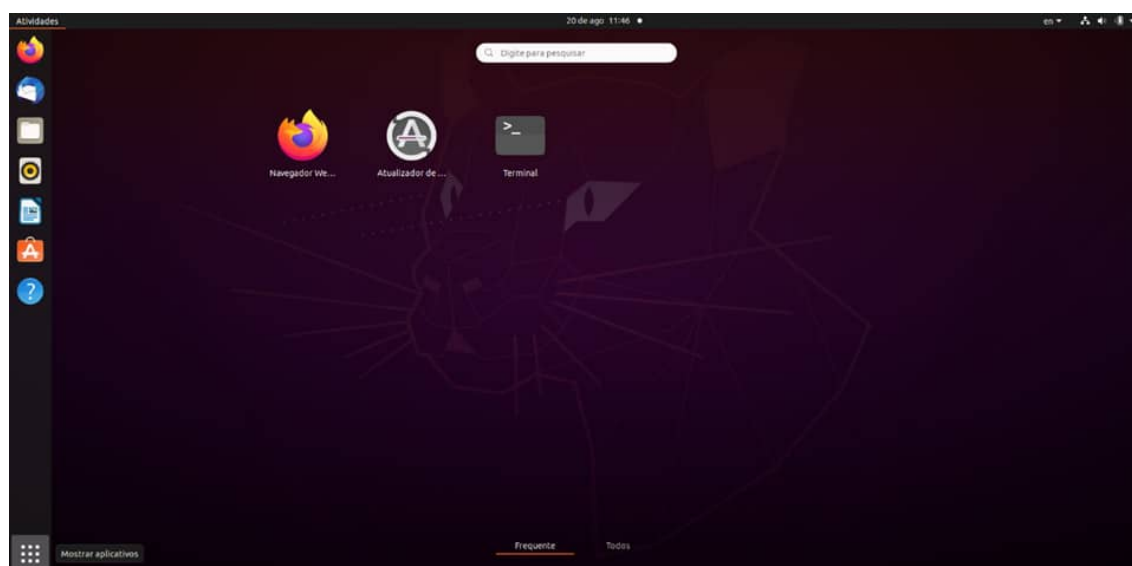
Tabela 5 – Diretórios e seus significados.

Fonte: O Autor.

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.

COMANDOS DO LINUX

A seguir, serão mostrados 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 figuras 28 e 29.



📷 Figura 28 – Abertura do terminal no Ubuntu Desktop. Fonte: O Autor.

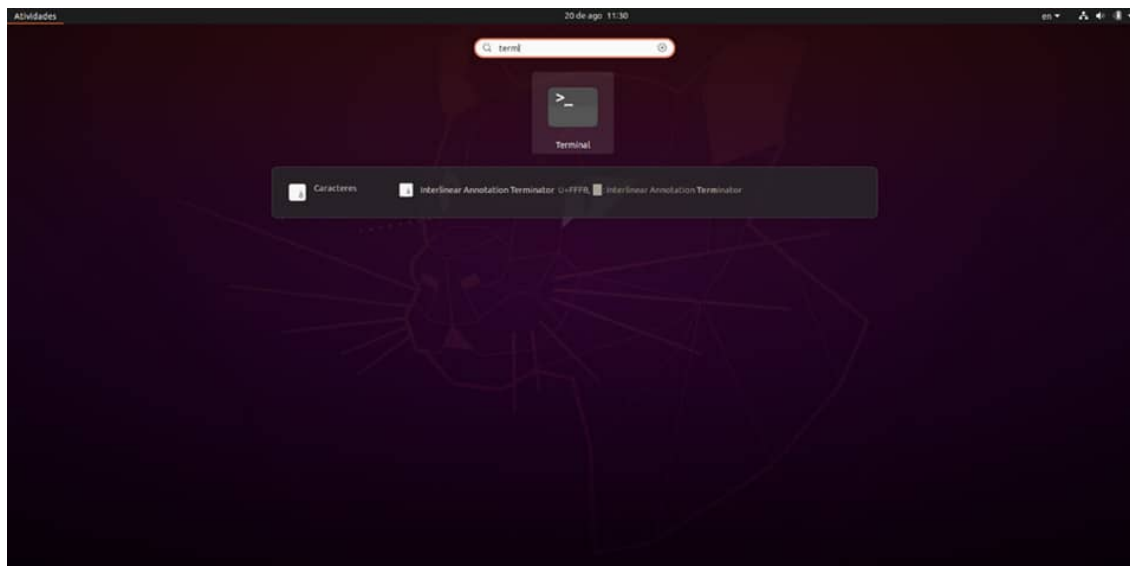


Figura 29 – Abertura do terminal no Ubuntu Desktop. Fonte: O Autor.

O **terminal** (ou prompt de comando) será aberto, conforme a figura 30. 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")



Figura 30 – Terminal.

Fonte: O Autor.

SAIBA MAIS

Veja uma lista com os **Comandos básicos para uso via terminal**.

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.

Executar os seguintes comandos:

```
./configure
```

```
make
```

```
make install
```

VERIFICANDO O APRENDIZADO

CONCLUSÃO

CONSIDERAÇÕES FINAIS

O sistema operacional surgiu com o objetivo de tornar o uso do computador mais fácil e conveniente para o usuário.

É muito importante o entendimento da ideia de abstração em sistemas de computação. O sistema operacional é um conjunto de rotinas que tem o objetivo de abstrair as camadas

inferiores e o hardware, no modelo de máquina de camadas. Essas rotinas em geral constituem o núcleo do sistema operacional, embora algumas rotinas possam ser executadas no espaço do usuário, conforme varia a estrutura do núcleo do sistema operacional.

Existem (e existiram) diversos sistemas operacionais ao longo da história. Um dos mais conhecidos e bastante utilizado é o Linux, que conta com uma enorme gama de distribuições à disposição dos usuários.



PODCAST



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.

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.

CONTEUDISTA

Fabio Henrique Silva

 **CURRÍCULO LATTES**