

# Introdução a Sistemas Operacionais

---

**Conceudista**

Prof. Me. Claudney Sanches Júnior

**Revisão Textual**

Aline de Fátima Camargo da Silva

# Sumário

---

<b>Objetivos da Unidade .....</b>	<b>3</b>
<b>Contextualização .....</b>	<b>4</b>
<b>Introdução .....</b>	<b>5</b>
<b>Definição de SO.....</b>	<b>6</b>
<b>História do Sistema Operacional.....</b>	<b>7</b>
<b>Interação com o SO .....</b>	<b>13</b>
<b>Tipos de Sistema Operacional.....</b>	<b>16</b>
<b>Em Síntese .....</b>	<b>18</b>
<b>Material Complementar .....</b>	<b>19</b>
<b>Atividades de Fixação .....</b>	<b>20</b>
<b>Referências.....</b>	<b>21</b>
<b>Gabarito .....</b>	<b>22</b>

# Objetivos da Unidade

---

- Estudar os conceitos básicos dos SO;
- Conhecer a evolução histórica dos SO, seus tipos de interação e classificação;
- Entender o relacionamento entre *hardware* e *software* permitindo que consiga se adaptar aos principais SOs facilmente, bem como utilizá-los.

Atenção, estudante! Aqui, reforçamos o acesso ao conteúdo *on-line* para que você assista à videoaula. Será muito importante para o entendimento do conteúdo.

Este arquivo PDF contém o mesmo conteúdo visto *on-line*. Sua disponibilização é para consulta *off-line* e possibilidade de impressão. No entanto, recomendamos que acesse o conteúdo *on-line* para melhor aproveitamento.

## VOCÊ SABE RESPONDER?

Como os Sistemas Operacionais melhor desempenham a relação entre *hardware* e *software*?

# Contextualização



**Figura 1 – Programador**

**Fonte:** Vida de Programador

**#ParaTodosVerem:** a imagem mostra uma história em quadrinho em quatro partes. No primeiro quadrinho há o título “Vida de Programador”, com fundo preto. No segundo, um homem, um programador, em frente a um monitor, ao telefone, com o texto “Não consigo acessar o sistema...está lento e travando”, dito pela pessoa do outro lado da linha. No terceiro, o programador diz “Por favor, me informe o navegador e o sistema operacional”. No quarto quadrinho, a pessoa do outro lado da linha responde “XP 7 e internet Firefox”; então, o programador cai para trás. Fim da descrição.

É comum, entre Empresas de desenvolvimento, haver dois ambientes de desenvolvimento, um de teste e outro de produção.

A maneira mais fácil de você criar um ambiente de teste é instalando um Sistema Operacional em uma máquina virtual.

Você deverá escolher entre a *VirtualBox*, a *VMWare Player* e a *Microsoft Virtual PC*, todas gratuitas e que não vão comprometer o SO residente em seu computador.

Após instalar a máquina virtual, instale uma distribuição do SO que preferir. O *Linux Ubuntu*, o *Debian* ou outra distribuição podem ser instalados facilmente com interface gráfica, ou você poderá escolher a versão do *Windows* que preferir.

Como aluno(a), você poderá pegar um *serial* original pelo convênio MSDNAA.

Crie um documento *Word* com as telas dos passos que você deu e as telas dos testes.

## Introdução

Nesta unidade, o objetivo será estudar os conceitos básicos, teóricos e práticos dos Sistemas Operacionais (SO) e conhecer suas classificações. Você será apresentando(a) a história do funcionamento, a terminologia utilizada e, sempre que possível, o motivo da evolução dos SOs.

Ao se deparar com a Disciplina, é comum que muitos imaginem que vão aprender determinada distribuição de um SO, por exemplo, aprender *Linux*, mas não é esse o objetivo da Disciplina, e, sim, propiciar aos discentes uma visão detalhada sobre os recursos contidos, independentemente da distribuição ou do fabricante em qualquer SO.

Ao final, espera-se que você seja capaz de entender o relacionamento entre *hardware* e *software*, permitindo que consiga se adaptar e utilizar os principais SOs facilmente, pois estará familiarizado(a) com as principais ideias de funcionamento existentes.

Como os SOs são grandes e complexos, seu estudo o(a) ajudará a se tornar um melhor programador por dar ênfase à sua organização interna e a sua estrutura.

Cada seção, aqui apresentada, tem um grau de dificuldade a ser superado na sua aprendizagem; é como uma escada: você deverá subir um degrau de cada vez. Se você decidir pular, poderá sentir dificuldades e cair, sendo necessário voltar e consultar alguma coisa que não viu.

# Definição de SO

O surgimento dos SO permitiu à computação uma mudança drástica no cenário de desenvolvimento de *software*. O SO deixou a tarefa do programador mais rápida. Ele passou a gastar menos tempo de desenvolvimento por possibilitar que se concentrasse apenas na lógica do problema, sem ter a necessidade de conhecer o *hardware* e os comandos de baixo nível dos diferentes dispositivos ligados a um computador.

O SO é uma camada entre *hardware* e aplicação que fornece a utilização de maior racionalidade, portabilidade e dedicação a problemas de alto nível ou abstratos.

A Figura 2 apresenta uma visão de qual local o SO se enquadra entre o *hardware* e as aplicações ou processos.



**Figura 2 – Apresentação SO em camadas**

**Fonte:** Elaborado pelo autor

**#ParaTodosVerem:** a figura mostra um diagrama formado por quatro formas retangulares, dispostas uma acima da outra, nas cores roxo, verde, cinza e vermelha, de baixo para cima. Nessa mesma ordem, em cada forma retangular há os termos "Hardware", "Sistema Operacional", "Aplicação" e "Usuário". Fim da descrição.

Alguns autores definem o SO como um programa ou conjunto de programas inter-relacionados, cuja finalidade é agir como intermediário entre usuário e o *hardware* ou como um *software* gerente do *hardware*. Entretanto, essa definição é bastante simples, sendo ele mais bem definido como um gerenciador de recursos em virtude de outras funções de que se encarrega. Os recursos podem ser memória principal e secundária com seus arquivos, periféricos, tais como unidade de CD-ROM e impressoras, entre outras.

Entre as principais funções do SO, pode-se listar: apresentar ao usuário uma máquina mais flexível; permitir o uso eficiente e controlado dos componentes de *hardware*; permitir o uso compartilhado, eficaz, protegido e seguro dos vários componentes de *hardware* e de *software* por diversos usuários e prover mecanismos de Gerenciamento de Processos, como criação, escalonamento, controle de concorrência, proteção e destruição.

Além disso, cabe ao SO esconder ou tornar transparente aos aplicativos os detalhes do *hardware*, cabendo apenas ao SO conhecer e negociar com ele.

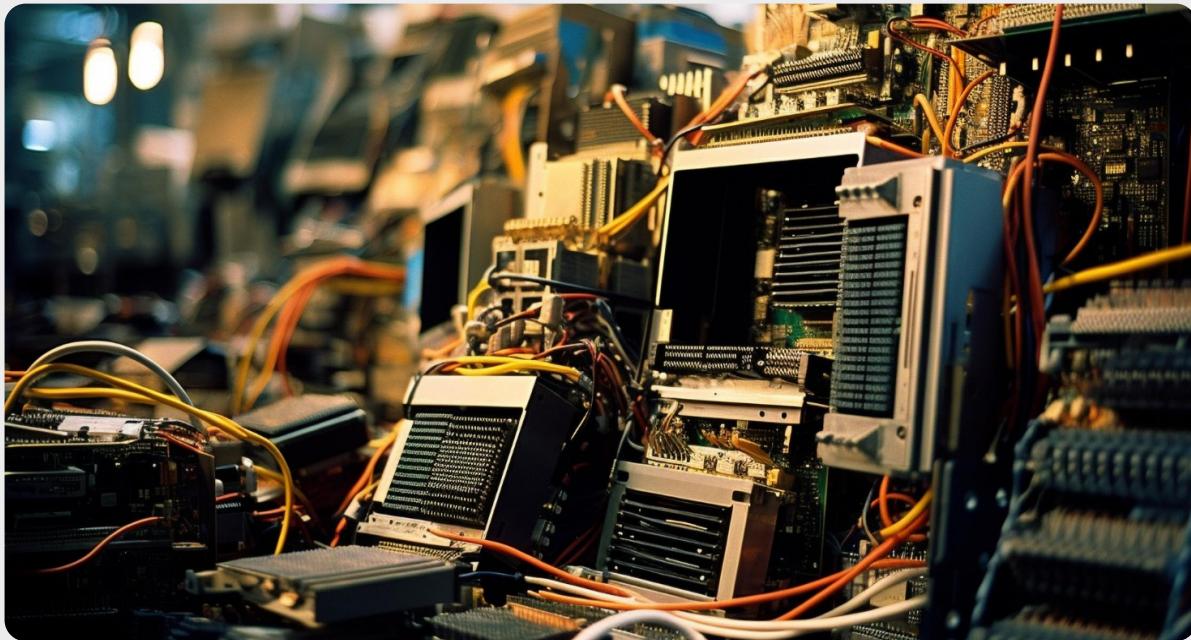
Os serviços que um SO pode oferecer são muitos, mas os principais são:

- Meios para criação de Programas;
- Meios para execução de Programas como mecanismo para carregar na memória, ler e escrever dados (i/o) e inicialização de arquivos;
- Acesso i/o e arquivos negociando as especificidades e formatos dos arquivos;
- Proteção e acesso a recursos e dados;
- Resolver conflitos e contenção de recursos;
- Detectar erros e responder aos erros;
- Fornecer estatísticas dos recursos;
- Monitorar performance.

## História do Sistema Operacional

Os primeiros computadores (1945-1955) não tinham SO e sua programação e operação eram feitas diretamente em Linguagem de Máquina. Os programadores controlavam o computador por meio de chaves, fios e luzes.

Nos primeiros SO (1955-1965), a interação entre ser humano e computador era feita por meio de periféricos de baixa velocidade, ou seja, o Sistema lia cartões perfurados para a entrada de dados, como ilustrado no *link* a seguir, e utilizava impressora para a saída de dados.



**Figura 3 – A Orquestra da Tecnologia da Informação**

**Fonte:** Freepik

**#ParaTodosVerem:** a imagem mostra uma foto de um maquinário antigo de sistemas operacionais, com fios, placas perfuradas que se conectam.

Os programas com as tarefas ou, simplesmente, *job* ou *task*, em inglês, eram agrupados fisicamente e processados sequencialmente, executando uma tarefa após a outra, sem interrupção. Esse tipo de Sistemas é conhecido como processamento *batch* ou em lote, em português, que tem como base um programa monitor, usado para enfileirar as tarefas.

O Programa Monitor estava sempre na memória principal do computador, disponível para execução, mas, após passar o controle para o Programa do Usuário, só executava novamente quando houvesse necessidade, por erro ou fim do Programa do Usuário.

Uma característica que se buscou nesses SOs foi criar uma área de memória protegida a que os programas dos usuários não tivessem acesso, nos quais estaria residindo o monitor.

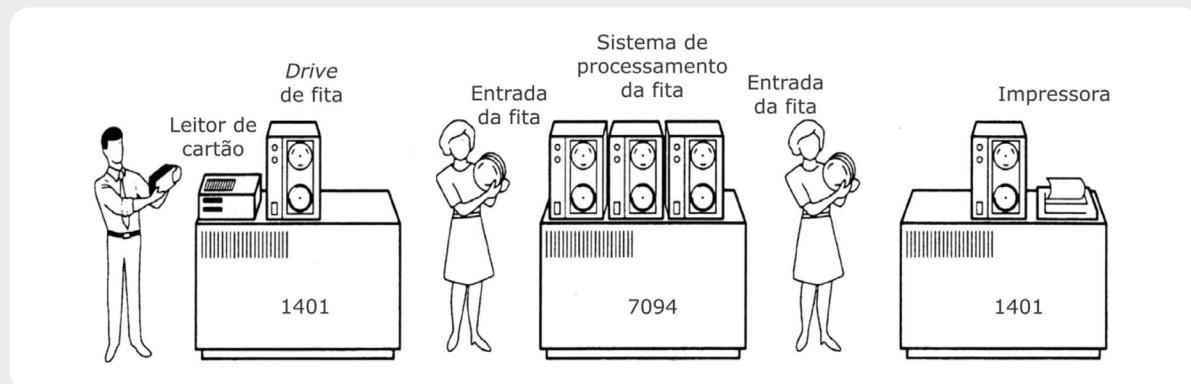
### Saiba Mais



O Sistema *batch* apresentou, em sua evolução, um Sistema de interrupção para que um determinado *job* não monopolizasse o Sistema. Assim, para cada *job* é atribuída uma fatia de tempo e se este usasse toda a fatia de tempo, o Programa Monitor era chamado.

Como mecanismo de segurança, atribui ao Programa Monitor o privilégio de ter algumas chamadas exclusivas; se algum Programa de Usuário tentasse chamá-las, essas chamadas acarretariam uma interrupção.

Em um Sistema do tipo *batch*, procura-se maximizar o número de tarefas processadas por unidade de tempo e minimizar o tempo médio de espera para a execução das tarefas, conforme mostra o link a seguir.



**Figura 4 – Processamento Batch**

**Fonte:** Reprodução

**#ParaTodosVerem:** ilustração de um processo *Batch*. Há três bancadas, primeira com uma leitora e um Drive de Fita e um homem ao lado; a segunda tem três fitas, uma de entrada, de sistema e de saída; a terceira tem um Drive de Fita e um Imp. Ao lado das duas últimas mesas, há uma mulher.

Devido aos tempos significativos de espera de leitura e impressão, o processador acabava ficando a maior parte do tempo ocioso. Lembre-se de que essas máquinas eram muito caras e existiam poucas; assim, era um desperdício mantê-las em espera pela entrada ou saída de dados e programas.

Uma solução para melhor utilizar o processador foi reduzir o tempo de espera de leitura e escrita utilizando uma técnica de *spooling* (1957) (*Simultaneous Peripheral Operation On Line*).

A técnica consiste em ler os dados previamente e gravá-los agrupados em fitas e discos, que são muito mais rápidos, ficando prontos para serem usados quando solicitados pela tarefa.

No início, esse processo era manual, mas se notava que poderia ser automatizado (1960). Os primeiros SOs eram únicos, desenvolvidos para cada computador específico, isso se dava sob encomenda, pois cada máquina tinha sua arquitetura e linguagem proprietária.

Outra solução consistiu na introdução entrada e saída em paralelo com o processamento (1959). Essa técnica permite a existência de mais de um programa na memória principal, aumentando o desempenho do processador. Assim, toda vez que um

programa encerra a saída de dados, outro, que foi previamente carregado na memória, pode ser alocado, evitando o tempo de espera de carga de novos programas.

O surgimento dessa técnica dá início ao SO multiprogramado ou *multiprogramming*, pois, se na memória pode existir mais de um programa, a CPU pode chavear (*switch*) de um programa para o outro, sempre que um programa esteja esperando uma operação de entrada e saída, e, isso é conhecido como multitarefa ou *multitasking*.

O problema observado com essa solução foi que a interação com usuários ficou prejudicada devido à necessidade de esperar pelo processamento completo das demais tarefas do programa que estava rodando, reduzindo a produtividade dos usuários.

### Saiba Mais



Com o objetivo de solucionar o problema de espera do usuário, surgiram os sistemas de *time-sharing* (1965), ou tempo repartido, que utilizam multiprogramação, dividindo o tempo de processamento da CPU entre os processos ativos e os múltiplos usuários. Cada um recebe uma fatia de tempo ou *time-slice* para executar.

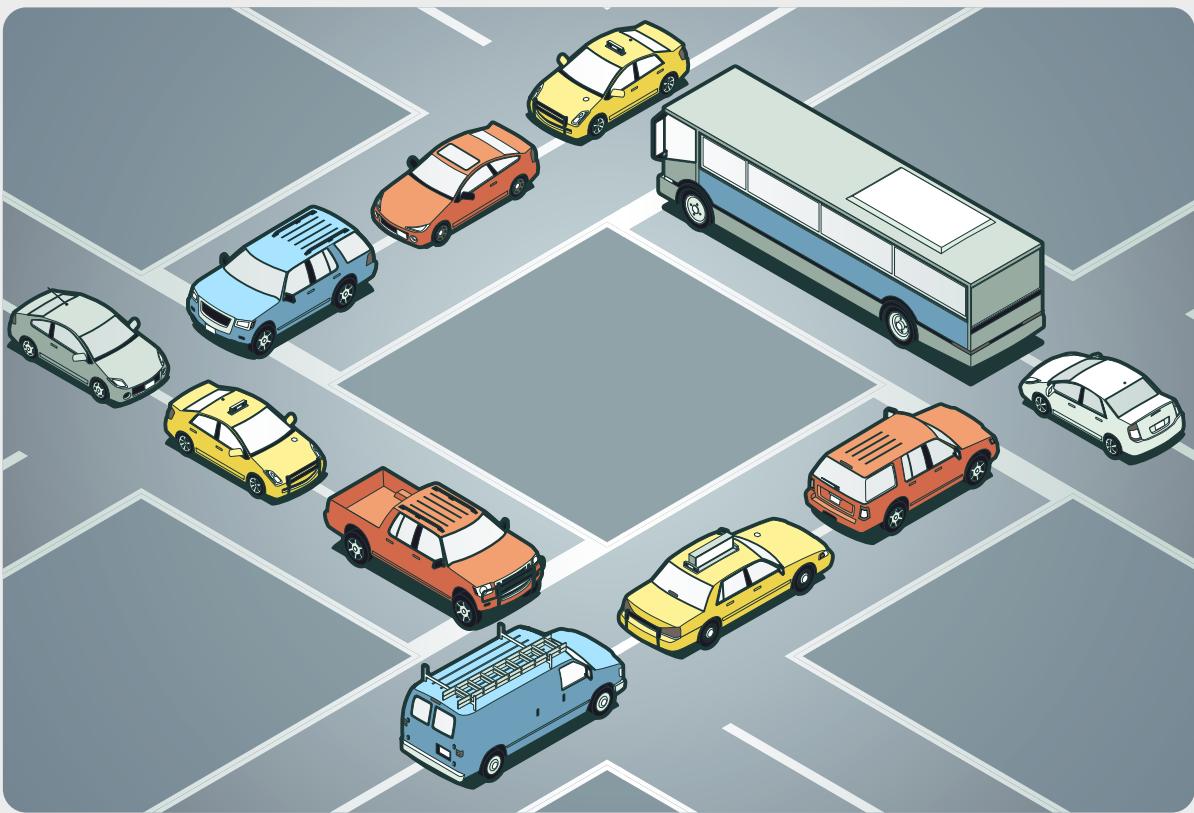
Em um sistema de tempo repartido, procura-se dar um tempo de resposta por comando dentro de limites aceitáveis. O atendimento eficiente com pequenas frações de tempo fornece aos usuários a ilusão de que o Sistema está dedicado unicamente à sua tarefa.

### Trocando Ideias



Para entender os Sistemas Multiusuários, imagine que, conhecendo o tempo lento da reação humana, um usuário típico necessitava de 2 segundos de um minuto do processador de um computador típico na época, o que permitia que um computador atendesse até 30 usuários sem atrasos. Esse era o cálculo que os fabricantes faziam para saber quantos usuários poderiam acessar o sistema.

Os SOs multiusuários tinham como desafio proteger os arquivos de vários usuários, evitando a sincronização imprópria, ou seja, que um programa esperando I/O fique em espera enquanto outro recebe sua solicitação de dados; que os programas acessem a região de dados um do outro, o que recebe o nome de exclusão mútua falida; e que os programas fiquem em *deadlock*, ou seja, um fique esperando o outro, como se dois caminhões estivessem querendo atravessar uma ponte de uma via, conforme ilustra a Figura 5.



**Figura 5 – Deadlock** – Ninguém pode continuar

**Fonte:** Acervo do conteudista

**#ParaTodosVerem:** a imagem mostra uma ilustração de um ônibus e carros dispostos em 4 fileiras, de modo que formam um quadrado, as cores dos veículos são amarelo, vermelho, azul e cinza. Fim da descrição.

Um dos primeiros SOs de propósito geral foi o CTSS (1961), descrito como um Sistema de compartilhamento experimental em computação interativa. Em seguida, o CTSS, o MIT, os laboratórios Bell da AT&T e a General Electric iniciaram o desenvolvimento do Multics (1964-2000), com o objetivo de suportar centenas de usuários. O Multics serviu como base para o estudo e o desenvolvimento de outros SOs.

Um dos desenvolvedores do Multics, Ken Thompson, começou a reescrever um SO em um conceito menos ambicioso, criando o Unics (em 1969) que, mais tarde, passou a se chamar Unix (1973). Alguns SOs derivados do Unix são: IRIX, HP-UX, AIX, Tru64, SCO, família BSD (FreeBSD, NetBSD, OpenBSD, Solaris etc.), Linux e até o Mac OS X.

Na década de 1970, começaram a aparecer os computadores pessoais e, em 1980, William (Bill) Gates e seu colega de faculdade, Paul Allen, fundadores da Microsoft, compram o sistema QDOS de Tim Paterson, batizando-o de DOS (*Disk Operating System*). O DOS vendeu muitas cópias, e evoluiu se tornando o SO padrão para os computadores pessoais chamado de Windows.

Durante esse período, surgiu a necessidade de existir outra classe de Sistemas chamados de tempo real ou *real-time*, caracterizados pelo suporte de aplicações críticas, como o controle de tráfego aéreo, usinas nucleares, centrais telefônicas em que o tempo de resposta deve sempre estar entre limites rígidos predefinidos. O tempo de resposta em SO desse tipo é chamado de prazo e a perda de um prazo, ou seja, o não cumprimento de uma tarefa dentro do prazo, é caracterizada como falha do Sistema.

Outra característica de SO de tempo real é sua interação com o meio ao redor e a sua previsibilidade. Um Sistema pode ser considerado previsível quando se pode antecipar seu comportamento independente de falhas, sobrecargas e variações de *hardware*.

### **BATCH**

Processamento em lote, usado para enfileirar tarefas;

### **SPOOLING**

Processo de transferência de dados, colocando-os em uma área de trabalho temporária em que outro programa poderá acessá-lo;

### **TIME-SHARING**

Tempo repartido que utiliza multiprogramação dividindo o tempo de processamento da CPU entre os processos ativos;

### **TIME-SLICE**

Uma fatia de tempo do *time-sharing*;

## REAL-TIME

Sistema com tempo de resposta predefinidos.

Avanços mais recentes foram a estruturação de Tecnologias para redes locais ou *Ethernet* e o uso de Protocolos como *TCP/IP*, o aumento do poder de processamento dos computadores proporcionando, assim, a pesquisa e o desenvolvimento de SO distribuídos e SO de rede.

A ideia é ter um conjunto de computadores independentes que apresente ao usuário como se fosse um único Sistema consistente, permitindo ao usuário o acesso de recursos compartilhados como *hardware*, *software* e dados.

Assim, teríamos o poder de diversos computadores interligados em rede.

# Interação com o SO

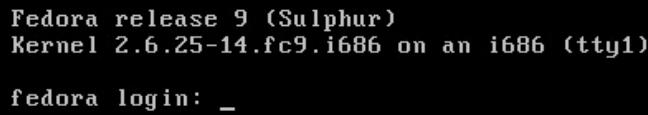
A maioria dos SOs permite que o usuário interaja de maneira direta, mas é comum encontrar maneiras de interagir por intermédio de quatro tipos de interface.

A interface de terminal permite que o usuário envie comandos pertencentes a uma Linguagem de Comunicação especial chamada de Linguagem de Comando (*Command Line Interface*) a qual funciona, exclusivamente, com teclado e mouse. A partir de um *prompt* ou uma tela de acesso simples, os comandos são digitados e interpretados pelo *shell*, um interpretador de comandos.

Ela costuma ser usada por usuários avançados, pois permite a realização de atividades específicas, tais como o gerenciamento remoto, pois utiliza poucos recursos de *hardware* e requer que o usuário conheça os comandos, os parâmetros e a sintaxe da Linguagem.

A Figura 6 exibe um console de comandos em *Linux*, o qual também é chamado simplesmente de *shell* no SO *Unix*. Tem o objetivo de capturar a entrada do usuário e interpretar, enviando os comandos ao núcleo do SO, e imprimir o resultado do processamento na tela.

Quase todos os consoles de comandos dos SOs modernos podem ser usados de forma interativa e no modo batch ou em lote. Na forma interativa, o usuário digita um comando após outro, enquanto no modo *batch*, cria-se um arquivo com uma sequência de comandos que podem ser reutilizados quantas vezes for necessário.



```
Fedora release 9 (Sulphur)
Kernel 2.6.25-14.fc9.i686 on an i686 (tty1)

fedora login: _
```

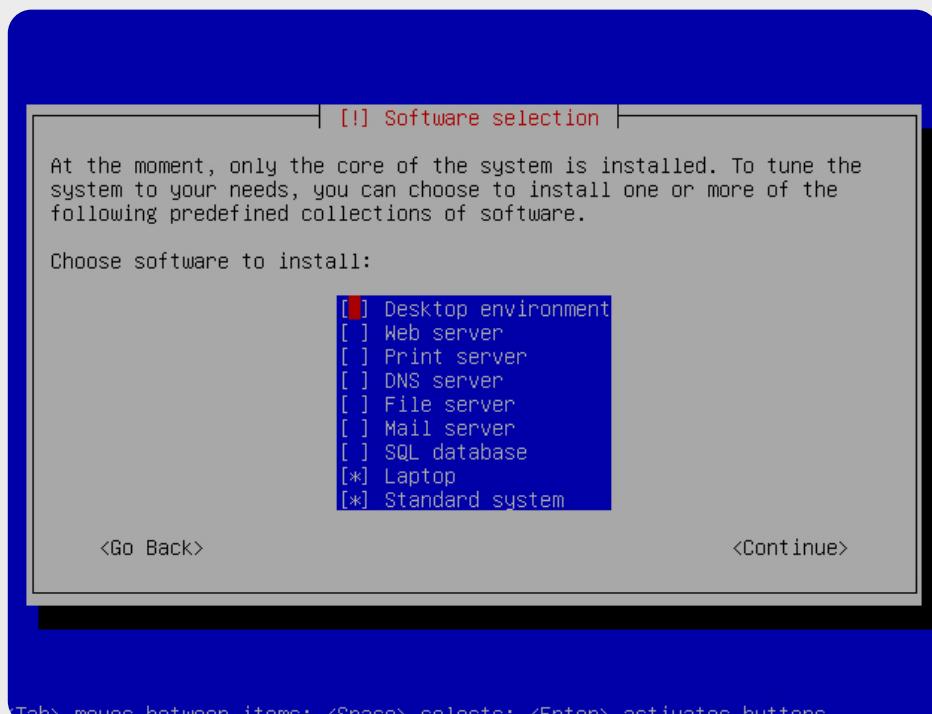
**Figura 6 – Console de comandos ou *Shell***

**Fonte:** Elaborada pelo condeudista

**#ParaTodosVerem:** a imagem mostra um *print* de um console de comando. O fundo da tela é preto e há alguns códigos em branco: Fedora release 9 (Sulphur); Kernel 2.6.25-14.fc9.1686 on an 1686 (tty1); Fedora login: \_. Fim da descrição.

A interface textual é baseada em texto, menus, janelas e botões. É uma evolução da interface de terminal, que foi difundida em aplicações baseadas no SO MS-DOS; porém, seu uso, atualmente, tornou-se raro, sendo comum encontrar-lá no processo de instalação do SO, como apresentado na Figura 5.

Também se pode encontrar a interface textual em Sistemas desenvolvidos entre a década de 1980 e o início da década de 1990.



**Figura 7 – Console textual apresentado durante a instalação**

**Fonte:** Elaborada pelo conteudista

**#ParaTodosVerem:** a imagem mostra um print de um console textual. O fundo é azul, havendo uma caixa de diálogo cinza. Dentro da caixa de diálogo há um texto de instrução e mais um quadro azul, com algumas opções listadas. O texto de introdução é “*At the moment, only the core of the system is installed. To tune the system to your needs, you can choose to install one or more of the following predefined collections of software*”. Fim da descrição.

A interface gráfica, chamada de *Graphic User Interface (GUI)*, além de exibir menus, janelas e botões, apresenta elementos gráficos, como ícones, figuras e imagens. Em SOs que rodam em computadores pessoais, a mais conhecida é o *WIMP*, que consiste de janelas, ícones, menus e ponteiros.

O usuário interage usando um ponteiro. Movendo o mouse é possível controlar a posição do cursor e apresentar as informações em janelas.

Também, é possível interagir com o teclado, teclas de atalhos, toque em dispositivos sensíveis ao toque ou *touchscreen* e, recentemente, com dispositivos de reconhecimento de gestos e expressões faciais. O usuário é capaz de selecionar símbolos e manipulá-los de forma a obter resultados práticos.

A Figura 8 demonstra as *GUI*: SO *Windows Blue*, OS X *Mavericks*, SO *Linux Debian Wheezy* e SO *Linx Ubuntu 13.04*.



**Figura 8 – SO Windows Blue, OS X Mavericks, SO Linux Debian Wheezy e SO Linux Ubuntu 13.04**

**Fonte:** Elaborada pelo conteudista

**#ParaTodosVerem:** a imagem exibe uma montagem feita com as telas de 4 sistemas operacionais. Estão dispostas em dois pares, um embaixo do outro. Cada tela apresenta um diferente layout de ícones e caixas de diálogo. Fim da descrição.

A mais nova interface é a voz ou *Voice User Interface (VUI)*, na qual o usuário interage com o SO mediante os comandos sonoros; tem recentemente encontrado aceitação em SO Mobile como smartphones e tablets.

## Tipos de Sistema Operacional

Há várias maneiras de classificar os SOs, portanto, você deve aprender a diferenciar a terminologia para poder classificar corretamente um SO. Quanto à carga de trabalho, ou *workload*, o SO pode ser *serial*, isto é, ter todos os recursos dedicados a um único programa até o seu término ou ser *concorrente*, no qual os recursos são dinamicamente reassociados entre uma coleção de programas ativos e em diferentes estágios de execução.

Outra classificação é referente ao compartilhamento do *hardware*, que podem ser monoprogramados ou multiprogramados. O primeiro, permite apenas um programa ativo em um dado período de tempo, o qual permanece na memória até o seu término. O segundo, mantém mais de um programa simultaneamente na memória principal a fim de possibilitar o compartilhamento efetivo do tempo da CPU e dos demais recursos.

Os SOs podem ser classificados quanto ao tempo de resposta, se será em *batch* ou interativo. Nos SOs em *batch*, os *jobs* são submetidos em ordem sequencial de execução e, enquanto ocorre o processamento, o usuário fica sem interação, até o término da execução dos *jobs*.

Os SO interativos possibilitam o diálogo com o usuário, podendo ser projetado como sistemas monousuários ou multiusuários, usando conceitos de multiprogramação e *time-sharing*.

Os sistemas monousuários só permitem que um usuário possa interagir com o Sistema, como foi o caso do SO MS-DOS. O SO é dito multiusuário quando provê atendimento a mais de um usuário, como ocorre com o *Unix* e o *Linux*.

A Multiprogramação é a capacidade de o Sistema permitir que mais de um programa esteja presente na memória principal em contraste à monoprogramação em que apenas um programa reside na memória principal.

Quanto à estrutura interna do SO, pode ser monolítica ou de núcleo (*kernel*). Os SOs monolíticos, também conhecidos como monobloco, são os mais primitivos, e consiste de um conjunto de rotinas que executam sobre o *hardware* como se fosse um único programa residindo na memória principal protegida e os Programas dos Usuários são vistos pelo Sistema como sub-rotinas invocadas pelo SO quando ele não executa nenhuma das funções do Sistema.

Os SOs com núcleo ou *micro-kernel* ou, ainda, cliente-servidor, incorporam somente as funções de baixo nível, ou seja, as mais vitais, dando, assim, a base para ser construído o resto do SO. A maioria desses Sistemas é construída como coleções de processos concorrentes.

O *micro-kernel* deve fornecer os serviços de alocação de CPU e de comunicação dos processos sendo que outras funções, como sistemas de servidor de diretórios e arquivos e gerenciamento de memória, são executadas no espaço do usuário. Assim como os serviços e as aplicações, os programas dos usuários são os clientes.

Ainda, o SO pode ser em camadas, nas quais as funções do núcleo irão executar em camadas distintas, de acordo com seu nível de privilégio, como ocorreu no SO *Multics*.

# Em Síntese

---

Novos elementos estão em pauta, como máquinas multiprocessadas, com redes de alta velocidade, processadores rápidos e grandes memórias.

Pode também existir um *software* que pode fornecer uma abstração do *hardware* para vários SOs recebendo o nome de Monitor de Máquinas Virtuais (VM), como ocorre no VMware.

# Material Complementar

---



## Leituras

**O que são máquinas virtuais?**

<https://bit.ly/49dGFTX>

**VM Ware, Virtual Box ou Virtual PC: qual é o melhor programa para virtualização?**

<https://bit.ly/3oxzhvN>

# Atividades de Fixação

---

## 1 – Qual das seguintes afirmações é verdadeira sobre sistemas operacionais?

- a. Os sistemas operacionais são exclusivos para computadores *desktop* e não são usados em servidores.
- b. Um sistema operacional não é necessário para que um computador funcione.
- c. Os sistemas operacionais são responsáveis por gerenciar *hardware*, recursos e fornecer serviços aos aplicativos.
- d. Os sistemas operacionais não têm controle sobre o acesso a arquivos e pastas em um computador.
- e. Um único sistema operacional é compatível com todos os tipos de *hardware* e *software*.

## 2 – Qual é a principal diferença entre “tarefa” e “trabalho” em um contexto de sistemas operacionais?

- a. “Task” e “job” são termos intercambiáveis e têm o mesmo significado.
- a. “Task” se refere a programas em execução em segundo plano, enquanto “job” é um sinônimo de aplicativo.
- a. “Task” representa unidades de trabalho individuais dentro de um “job” ou processo maior.
- a. “Task” é usado exclusivamente em sistemas Unix, enquanto “job” é usado em sistemas Windows.
- a. “Job” é uma tarefa realizada apenas pelo sistema operacional, enquanto “task” é uma coleção de tarefas relacionadas.

Atenção, estudante! Veja o gabarito desta atividade de fixação no fim deste conteúdo.

# Referências

---

DEITEL, H. M. **Sistemas Operacionais**. 3. ed. São Paulo: Pearson Prentice Hall, 2005.

TANENBAUM, A. S. **Sistemas Operacionais Modernos**. 3. ed. São Paulo: Pearson Prentice Hall, 2009.

# Gabarito

---

## Questão 1

**c) Os sistemas operacionais são responsáveis por gerenciar *hardware*, recursos e fornecer serviços aos aplicativos.**

**Justificativa:** os sistemas operacionais desempenham um papel fundamental na operação de computadores e dispositivos. Eles gerenciam recursos de *hardware*, fornecem serviços para aplicativos e permitem a interação do usuário com o sistema. Portanto, a resposta correta destaca a função essencial dos sistemas operacionais.

## Questão 2

**c) “Task” representa unidades de trabalho individuais dentro de um “job” ou processo maior.**

**Justificativa:** em sistemas operacionais, “task” e “job” são termos relacionados, mas têm significados bastante diferentes. Uma “task” representa uma unidade de trabalho individual dentro de um “job” ou processo maior. Por exemplo, um “job” pode ser um programa ou processo em execução, enquanto as “tasks” seriam as tarefas individuais que esse programa realiza.