

Sistemas Operacionais



Material Teórico



Introdução a Sistemas Operacionais

Responsável pelo Conteúdo:

Prof. Me. Claudney Sanches Junior

Revisão Textual:

Prof.^a Dr.^a Selma Aparecida Cesarin



- **Introdução;**
- **Definição de SO;**
- **História do SO;**
- **Interação com o SO;**
- **Tipos de SO.**



OBJETIVO DE APRENDIZADO

- Estudar os conceitos básicos dos SO. A unidade apresenta a definição de SO e mostra sua evolução histórica; detalha os tipos de interação e apresenta a classificação dos SOs.
- Entender o relacionamento entre hardware e software permitindo que consiga se adaptar aos principais SOs facilmente, bem como utilizá-los.



Orientações de estudo

Para que o conteúdo desta Disciplina seja bem aproveitado e haja maior aplicabilidade na sua formação acadêmica e atuação profissional, siga algumas recomendações básicas:



Assim:

- ✓ Organize seus estudos de maneira que passem a fazer parte da sua rotina. Por exemplo, você poderá determinar um dia e horário fixos como seu “momento do estudo”;
- ✓ Procure se alimentar e se hidratar quando for estudar; lembre-se de que uma alimentação saudável pode proporcionar melhor aproveitamento do estudo;
- ✓ No material de cada Unidade, há leituras indicadas e, entre elas, artigos científicos, livros, vídeos e sites para aprofundar os conhecimentos adquiridos ao longo da Unidade. Além disso, você também encontrará sugestões de conteúdo extra no item **Material Complementar**, que ampliarão sua interpretação e auxiliarão no pleno entendimento dos temas abordados;
- ✓ Após o contato com o conteúdo proposto, participe dos debates mediados em fóruns de discussão, pois irão auxiliar a verificar o quanto você absorveu de conhecimento, além de propiciar o contato com seus colegas e tutores, o que se apresenta como rico espaço de troca de ideias e de aprendizagem.

Contextualização



Veja a charge a seguir: <https://goo.gl/UEiHmP>

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 uma versão do *Windows* que preferir.

Como aluno(a), você pode 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 apresentado(a) à história do funcionamento, à terminologia utilizada e, sempre que possível, ao motivo da evolução dos SOs.

Ao se deparar com a Disciplina, é comum que muitos imaginem que irão aprender determinada distribuição de um SO, como, por exemplo, aprender *Linux*, mas não é esse o objetivo da Disciplina, e sim, propiciar aos alunos uma visão detalhada sobre os recursos contidos, independente 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 deve subir um degrau de cada vez.

Se você decidir pular, pode sentir dificuldades e cair, sendo necessário voltar e consultar alguma coisa que não viu. Assim, para que possa entender esses conceitos, esta Unidade está organizada da seguinte forma:

- A Seção 2 apresenta a definição de SO;
- A Seção 3 mostra o surgimento e a história;
- A Seção 4 detalha os tipos de interação;
- A Seção 5 apresenta os tipos de SO.

Ao final do estudo e das atividades desta Unidade, você deve ser capaz de:

- Entender e caracterizar os SOs;
- Conhecer o relacionamento entre *hardware* e *software*.

Não deixe de utilizar os fóruns associados à unidade para apresentar e discutir qualquer dificuldade encontrada.

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 permitir 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 à aplicação maior racionalidade, portabilidade e dedicação a problemas de alto nível ou abstratos.

A Figura 1 apresenta uma visão de onde o SO se enquadra entre o *hardware* e as aplicações ou processos.



Figura 1 – Apresentação SO em camadas

Alguns autores definem 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 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 diversos 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 SO

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.



Leitor de Cartão Perfurado da IBM em 1923: <https://goo.gl/ZCHLnV>

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 nestes SOs foi criar uma área de memória protegida a que os programas dos usuários não tivessem acesso, em que estaria residindo o monitor.

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 utilizasse 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.

Num 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.



Processamento Batch: <https://goo.gl/RwkR8m>

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 utilizados quando solicitados pela tarefa.

No início, esse processo era manual, mas se notou que poderia ser automatizado (1960). Os primeiros SOs eram únicos, desenvolvidos para cada computador específico, isso se dava sobre 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 inicio ao SO multiprogramado ou *multiprogramming*, pois, se na memória pode existir mais de um programa, a CPU pode chavar (*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 foi prejudicada devido à necessidade de espera pelo processamento completo das demais tarefas do programa que estava rodando, reduzindo a produtividade dos usuários.

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.

Num 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.

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 accessem 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 2.

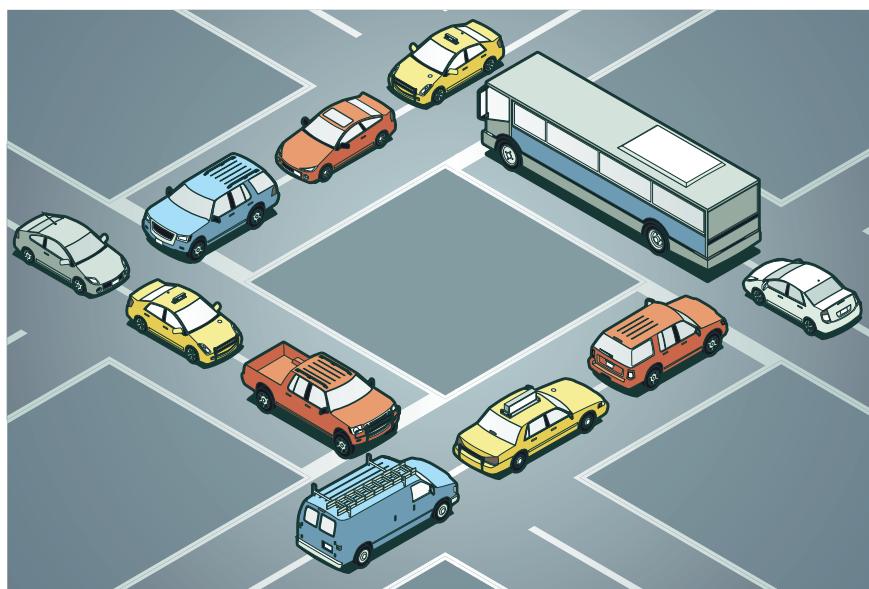


Figura 2 – *Deadlock* – Ninguém pode continuar

Um dos primeiros SOs de propósito geral foi o CTSS (1961), descrito como um Sistema de compartilhamento experimental em Computação interativa.

Após 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. 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 num conceito menos ambicioso, criando o Unics (em 1969) que, mais tarde, passou a se chamar Unix (1973). Alguns SOs derivados do *Unix* são: *IRIXG*, *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 caracterizado 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*.

A
Z

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 meio de quatro tipos de interface.

A interface de terminal permite que o usuário envie comandos pertencentes à uma Linguagem de Comunicação especial chamada de Linguagem de Comando (*Command Line Interface*) que 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.

Costuma ser utilizada 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 3 apresenta um console de comandos em *Linux*, que também é chamada 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.

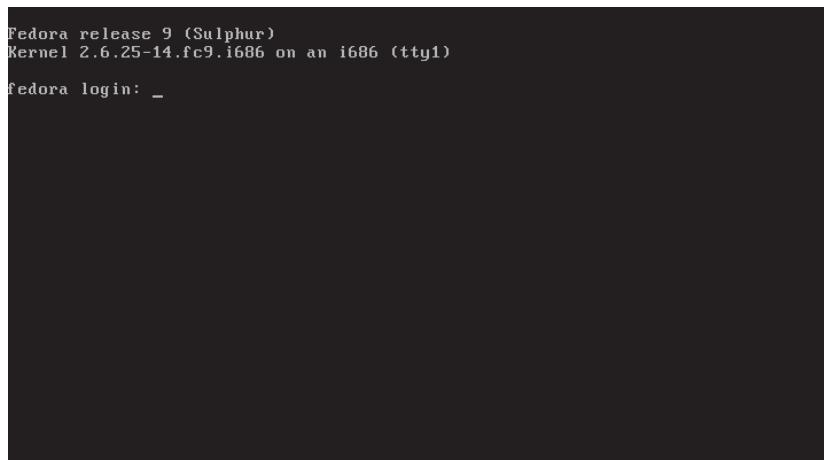


Figura 3 – Console de comandos ou Shell

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 encontrá-la no processo de instalação do SO, como apresentado na Figura 4.

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

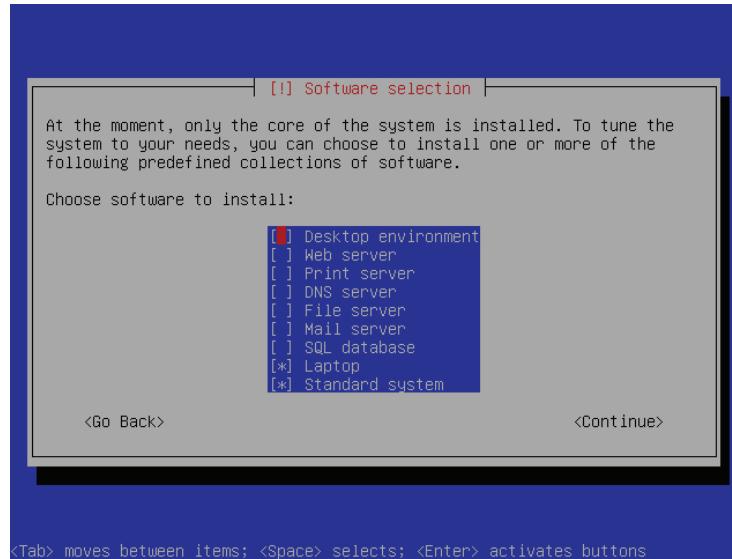


Figura 4 – Console textual apresentado durante a instalação

A interface gráfica, chamada de *GUI* (*Graphic User Interface*), além de apresentar *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 5 apresenta as *GUI*: *SO Windows Blue*, *OS X Mavericks*, *SO Linux Debian Wheezy* e *SO Linux Ubuntu 13.04*.

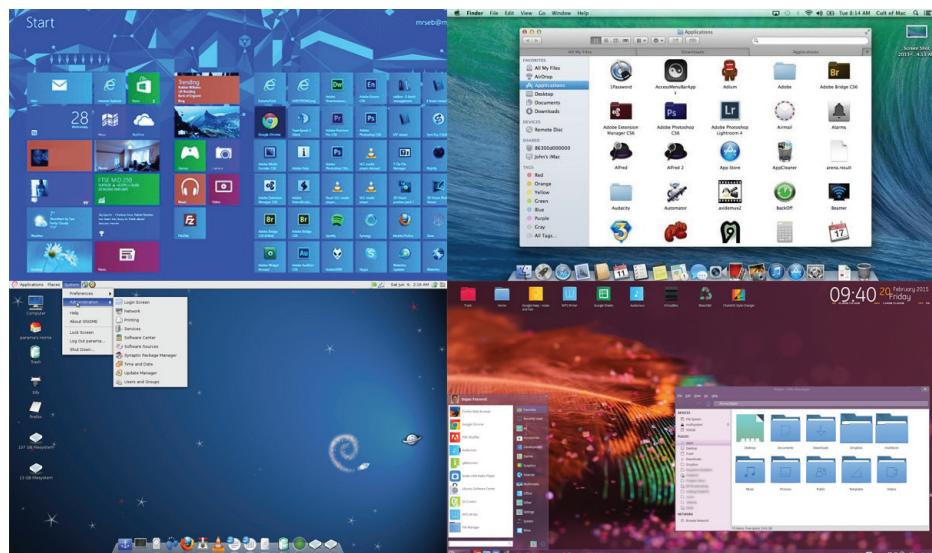


Figura 5 – SO Windows Blue, OS X Mavericks, SO Linux Debian Wheezy e SO Linux Ubuntu 13.04

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

Tipos de SO

Existem diversas maneiras de classificar os SOs, 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, ou seja, 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 para permitir o compartilhamento efetivo do tempo da CPU e dos demais recursos.

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** permitem 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*.

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, 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 está executando 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*.

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

Indicações para saber mais sobre os assuntos abordados nesta Unidade:

Leitura

O que são máquinas virtuais?

<https://goo.gl/nMoLUu>

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

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

Referências

DEITEL, H.M.; **Sistemas Operacionais**; 3º Edição; São Paulo; Pearson Prentice Hall, 2005.

TANENBAUM, A.S.; **Sistemas Operacionais Modernos**; 3º Edição; São Paulo; Pearson Prentice Hall, 2009.



Cruzeiro do Sul Virtual
Educação a Distância

www.cruzeirodosulvirtual.com.br
Campus Liberdade
Rua Galvão Bueno, 868
CEP 01506-000
São Paulo - SP - Brasil
Tel: (55 11) 3385-3000



Cruzeiro do Sul
Educacional