

A História, Conceito e Estrutura dos Sistemas Operacionais

Marcia Helena Pereira

A História, Conceito e Estrutura dos Sistemas Operacionais

Introdução

Boas-vindas, caro(a) estudante. Você sabia que o sistema operacional é o software principal que faz o controle de todo o hardware e outros softwares em um computador? Neste conteúdo, você conhecerá um pouco sobre os conceitos e histórias do SO e entenderá que eles têm como missão gerenciar os dispositivos de entrada e saída.

Além disso, conhecerá sobre alguns serviços do sistema operacional, suas chamadas, estruturas de sistemas, o que são processos em execução de um programa, quando realiza as ações especificadas, o que é escalonamento e quais são as operações sobre ele. Vamos viajar nesse mundo da tecnologia? Contamos com você. Bons estudos!

Objetivos da aprendizagem

- Apresentar a história dos sistemas operacionais.
- Discutir sobre a importância dos sistemas operacionais.
- Entender as principais características do surgimento dos sistemas operacionais.
- Compreender os serviços do sistema operacional.
- Compreender as chamadas de sistema, tipos de chamadas.
- Apresentar a estrutura de sistemas (monolítico, camada, microkernel).
- Entender o conceito de processo e escalonamento de processos.
- Discutir sobre as operações sobre processos.
- Apresentar os algoritmos de escalonamento.

História e conceito dos Sistemas Operacionais

Segundo Wazlawick (2016), a evolução dos sistemas operacionais começa na pré-história e passa pela invenção dos primeiros mecanismos de contagem, especialmente nas civilizações babilônica, fenícia, grega e romana, além de influências dos matemáticos árabes e indianos, com a criação do conceito de algoritmo, da criptografia e do zero como dígito. Também pode-se afirmar que a evolução dos sistemas foi importante para a evolução futura da computação. As Idades Antiga e Média testemunharam a invenção de mecanismos de computação analógica, ou seja, mecanismos que não calculam diretamente com números, mas com grandezas físicas.



Atenção

A matemática era usada para transações comerciais, a maioria dos problemas podiam ser resolvidos com poucas operações aritméticas simples. Com o crescimento das áreas de astronomia, engenharia e economia e a necessidade cada vez maior para se obter tabelas complexas e números, foi preciso criar algo para a realização de cálculos rápidos, evitando exaustão. O conceito de “computador” surgiu nessa época e era uma profissão de calcular mecanicamente.

Algumas formas primitivas de representação de informação, de acordo com Wazlawick (2016, p. 8), são o ábaco (c. 2700 a.C.), o relógio mecânico (724), a calculadora de Leonardo da Vinci (1502) e a régua de cálculo (1622).

Na história, a partir da contagem dos dedos, somos levados a eras seguintes com a construção e aperfeiçoamento das primeiras calculadoras mecânicas. Vejamos, de acordo com Wazlawick (2016, p. 8): Relógio Calculador de Schickard – 1623; Pascalina – 1642; Leviatã de Thomas Hobbes – 1651; Método Llullístico de Athanasius Kircher – 1669; calculadora de Samuel Morland – 1673; sistema binário – 1679; e contador mecânico de Leibniz – 1694.



Curiosidade

Com a intenção de agilizar a realização de cálculos matemáticos e de tornar os processos que são repetitivos um pouco mais automatizados, o matemático francês Blaise Pascal (1623-1662) inventou uma máquina de somar para auxiliar seu pai no processo de arrecadação de impostos, em 1654.

Charles Babbage (1792–1871) foi o inventor do primeiro computador programável do mundo. Inspirado no francês Joseph Marie Jacquard, criador do tear mecânico com uma leitora automática de cartões, Babbage, idealizou uma máquina de tecer números e fizesse cálculos controlada por cartões. Aprimorando suas pesquisas, em 1822, apresentou o projeto da máquina diferencial, capaz de resolver equações polinomiais, possibilitando a construção de tabelas de logaritmos. Babbage, contou com a ajuda de Ada Augusta King, na tradução de documentos do francês para o inglês e também na projeção de programas para a máquina que ele estava idealizando, porém, a máquina nunca chegou a existir. A linguagem de programação Ada tem esse nome como uma forma de retribuição ao trabalho da pesquisadora.



Figura 1 - Charles Babbage e Ada Lovelace

Fonte: Wikimedia Commons (2021).

#PraCegoVer: duas fotografias em preto e branco, à esquerda Charles Babbage e à direita Ada Lovelace.

Pesquisas iniciais com dispositivos elétricos, como os relês, permitiram, entre 1880 e 1930, que as primeiras máquinas somadoras ou contadoras com base em eletromecânica fossem construídas pelo estatístico Hermann Hollerith (1860-1929) que foi um dos fundadores da IBM, e sua máquina fez o primeiro senso automático americano em 1910, reduzindo tempo.

O potencial das mulheres na área de TI, também merecem destaque, Margaret Hamilton, foi uma das mulheres programadoras do programa Apollo e desenvolveu um software responsável pelo sucesso da missão de levar o homem à Lua.



Saiba mais

Assim como Margaret, Ada Lovelace e tantas outras mulheres fizeram grandes contribuições na área. GNIPPER, Patrícia.

Mulheres históricas: Margaret Hamilton criou o programa de voo da Apollo 11. Disponível em: <https://canaltech.com.br/internet/mulheres-historicas-margaret-hamilton-criou-o-programa-de-voo-da-apollo-11-78811/>. Acesso em: 19 abr. 2021.

No século XIX, ocorreram avanços nas ciências da Lógica e Matemática, devido, principalmente, aos pesquisadores George Boole, Gottlob Frege, entre outros, que lançaram os fundamentos teóricos para a Ciência da Computação a nascer no início do século XX com trabalhos como o de Alan Turing, segundo Wazlawick, (2016, p. 61).

Alan Turing é considerado por muitos um dos pais da computação, pois sua contribuição foi de grande relevância na época da Segunda Guerra Mundial. Essa guerra, ocorrida entre 1939 e 1945, deu impulso aos governos – especialmente o americano e o britânico –, para a construção de máquinas automáticas que decifrassem códigos inimigos e calculassem trajetórias de mísseis. Assim, foram criados decifradores como a Bomba de Turing, os Colossus e o calculador de trajetórias ENIAC (que possuía 17.468 válvulas, pesava 30 toneladas, tinha 180 m² de área construída, com velocidade de 100 kHz e 200 bits de memória RAM), que só ficou pronto depois da guerra, destaca Wazlawick (2016).



Curiosidade

Universal Automatic Computer (UNIVAC 1) foi um computador construído em 1951 nos EUA, projetado por J. Presper Eckert e John Mauchly. Para programá-lo, foi necessário ajustar cerca de 6.000 chaves e conectar vários cabos a um painel. A entrada e a saída de informações eram realizadas com o auxílio de uma fita metálica. Geralmente, esses equipamentos eram acompanhados de um dispositivo impressor chamado UNIPRINTER que, sozinho, consumia 14.000 W! Ao todo, foram vendidas 46 unidades do Modelo I do UNIVAC.

Após todo o entendimento sobre o contexto histórico da computação, podemos dizer que sistema operacional é um programa que realiza a intermediação da interface entre o usuário e o hardware do computador, ou seja, é o responsável pela comunicação entre a máquina e o usuário.

O ritmo crescente da informatização já aproximou o momento em que praticamente todos os membros da sociedade estão em contato com os computadores. O sistema operacional é quem faz a interface entre o hardware de um computador e uma pessoa, e a tarefa dos criadores do SO é tornar essa interface o mais amigável possível.



Figura 2 - Usuário, hardware, softwares

Fonte: Pixabay (2021).

#PraCegoVer: ilustração com seis ícones mostrando usuário, máquina, sistema operacional e periféricos.

O sistema operacional fornece comunicação entre o usuário, programas e dispositivos de hardware, garantido o funcionamento conjunto de todos os dispositivos do computador e fornece ao usuário acesso aos seus recursos. O sistema operacional inclui drivers de dispositivo, programas que controlam a operação de dispositivos e coordenam a troca de informações com outros dispositivos.

Para Silberschatz (2017, p.4)

Um sistema operacional é semelhante a um governo. Os componentes de um sistema de computação são seu hardware, software e dados. O sistema operacional fornece o meio para o uso adequado desses recursos na operação do sistema de computador. Como um governo, o sistema operacional não executa nenhuma função útil por si mesma. Simplesmente fornece um ambiente no qual outros programas podem realizar tarefas úteis.



Saiba mais

Acompanhe o vídeo a seguir, que apresenta informações gerais sobre a história dos SOs. "História dos Sistemas Operacionais". Disponível em: <https://youtu.be/9rC9GiX1Io>. Acesso em: 19 abr. 2021.

A tarefa principal do sistema operacional é garantir a execução de qualquer trabalho, controlando todos os dispositivos conectados ao computador, fornecendo acesso a eles para outros programas.

Para Machado (2013, p. 5):

Nos primeiros computadores, a programação era realizada em linguagem de máquina, em painéis através de fios, exigindo, um grande conhecimento da arquitetura do hardware. Isso era uma grande dificuldade para os programadores da época. O surgimento do sistema operacional minimizou esse problema, tornando a interação entre usuário e computador mais simples, confiável e eficiente.

A evolução dos SOs está diretamente relacionada ao desenvolvimento dos computadores em si. Considere a evolução dos sistemas operacionais da primeira

geração dos computadores, até as gerações atuais de computadores. Vejamos o quadro a seguir.

Evolução dos sistemas operacionais		
1ª geração 1945–1955	Válvulas e painéis de conectores	Eram utilizadas válvulas com programação em linguagem de máquina, dificultando o processo. Os dados eram armazenados em cartões perfurados e, depois, em fita magnética.
2ª geração 1955–1965	Transistores e sistemas de lote (batch)	<p>Nessa geração ocorreu a substituição da válvula pelo transistor. O transistor era muito menor do que as válvulas, tinha menor consumo de energia, gerava menos calor e era um pouco mais rápidos. Era um sistema de processamento em lote, ou <i>batch</i>, no qual os cartões eram expostos a uma leitora e gravados em uma fita de entrada (A). A fita era lida pelo computador, que executava um a um os programas e os resultados do processamento eram gravados em uma fita de saída (B). No final, a fita de saída era lida e impressa (C).</p> <p style="text-align: center;">Processamento batch</p> <pre> graph TD subgraph A [] Job1[Job 1] Job2[Job 2] Jobn[Job n] Job1 --> Job2 Job2 --> Jobn Job1 -- "Cartões perfurados" --> FitaEntrada((Fita de entrada)) end subgraph B [] FitaEntrada --> Processamento[Processamento] Processamento --> FitaSaida((Fita de saída)) end subgraph C [] FitaSaida --> Relatorio1[Relatório 1] FitaSaida --> Relatorio2[Relatório 2] FitaSaida --> RelatorioN[Relatório n] Relatorio1 --> Relatorio2 Relatorio2 --> RelatorioN end </pre> <p>Fonte: MACHADO (2013, p. 9). #PraCegoVer: figura dividida em três partes: (a): ilustração de três cartões perfurados, seta para a direita, retângulo escrito Processamento, seta para a direita, ilustração de uma fita de entrada. No meio da imagem, item (b): ilustração de uma fita de entrada, seta para a direita, retângulo escrito Processamento, seta para a direita, ilustração de uma fita de saída. Na parte de baixo da imagem, item (c): ilustração de uma fita de saída, seta para a direita, três retângulos ondulados na base, chamados relatórios.</p>

Evolução dos sistemas operacionais

3ª geração 1965-1980	Circuitos integrados (CIs) e multiprogramação	Nessa geração foram usados os circuitos integrados, conhecidos como microchips, iniciando-se as linguagens de alto nível. É marcada pela invenção de várias linguagens de programação importantes, como COBOL e BASIC. Surgiram as redes de computadores, em especial a invenção da comutação de pacotes e a construção da ARPANET. A implantação dos CIs possibilitou a redução de custos, a diminuição do tamanho do equipamento e o aumento da capacidade de processamento. Houve avanços em termos de SOs, principalmente aqueles relacionados aos conceitos de multiprogramação, multiprocessamento <i>time-sharing</i> e memória virtual.
4ª geração 1977-1999	Computadores pessoais	Essa geração corresponde ao surgimento dos processadores, unidade central de processamento (UCP). Foram criados os SO, como MS-DOS, UNIX, Apple's Macintosh e novas linguagens de programação, como as orientadas a objeto. Os computadores tornaram-se mais rápidos, passaram a ter menor custo, ser menores em espaço físico e ter mobilidade. Surgiram microprocessadores.
5ª geração Dias atuais	Computação móvel	A quinta geração é marcada pela inteligência artificial, conectividade e miniaturização. Nos dias atuais, os computadores estão conectados a outros dispositivos, como os da nossa residência, e há troca de informações com outros usuários por meio da inteligência artificial e da conectividade.

Tabela 1

FONTE: Elaborado pela autora (2021).

#PraCegoVer: quadro com três colunas e cinco linhas sobre as cinco gerações dos SOs.

Apartir da quarta geração, devido ao amplo uso de redes de computadores e ferramentas de processamento on-line, os usuários passaram a ter acesso a computadores distribuídos geograficamente, com base nos quais, cada vez mais, novos PCs são criados, os quais podem ser usados tanto de forma autônoma quanto como terminais de sistemas de computação mais poderosos.

Grande parte dos computadores atuais, sejam eles formados por dispositivos móveis, desktops, servidores etc., possibilitam que os usuários consigam operar a máquina por meio de aplicativos.

Aplicativos

Programas desenvolvidos com a função de auxiliar o usuário a desempenhar tarefas específicas. Quase todas as tarefas que um usuário desempenha em um computador, seja ele de grande ou de pequeno porte, móvel ou fixo, são realizadas com o auxílio de aplicativos. Um aplicativo precisa de um SO para fazer a interface com os recursos disponíveis no hardware.

Atualmente, na quinta geração, verificamos a consolidação das redes sem fio. Os SOs estão presentes em diversos dispositivos móveis, oferecendo todos os seus serviços aos usuários.

Os *smartphones* tornaram-se onipresentes e novas interfaces entre usuário e hardware são oferecidas. Tais recursos tornam a comunicação com o computador mais inteligente, simples e eficiente.



Figura 3 - Computação móvel

Fonte: Plataforma Deduca (2021).

#PraCegoVer: fotografia colorida mostra notebook, celular e tablet.

A computação móvel começou com o SO Symbian, escolhido pelos fabricantes Samsung, Sony, Nokia e Ericsson. Entretanto, os SOs do Blackberry e IOS da Apple, rapidamente começaram a ganhar mercado. Posteriormente, os SOs tornaram-se mais proativos, incorporando mecanismos automáticos para a detecção e para a recuperação de erros apresentados no sistema.

Em termos de software, várias linguagens foram criadas a partir de Ada até os tempos atuais, tais como as linguagens Turbo Pascal, C, C++, Eiffel, Perl, entre outras. Também foram criadas as linguagens de programação voltadas para a web, tais como Java, PHP, Python, Ruby e JavaScript, entre outras.

Os sistemas operacionais utilizados atualmente, cujas versões vão sendo superadas e adaptadas a cada momento, são: MS DOS; Microsoft Windows; Mac OS; OS / 2; UNIX; e Linux.

Estrutura dos Sistemas Operacionais

Segundo Machado (2013), a classificação de um sistema operacional pode variar de acordo com os recursos de implementação de algoritmos internos para gerenciamento dos principais recursos de um computador (processadores, memória, dispositivos), características dos métodos de design usados, tipos de plataformas de hardware, critérios de eficiência, características de implementação de soluções de rede e muitas outras propriedades.



Figura 4 - Tipos de sistemas operacionais

Fonte: Machado (2013, p. 15).

#PraCegoVer: ilustração mostra hierarquia dos Tipos de sistemas operacionais: sistemas monoprogramáveis / monotarefa, sistemas multiprogramáveis / multitarefa e sistemas com múltiplos processadores.

Existem vários processos de aplicativos que concorrem por recursos, por exemplo, o escalonamento de processos ou agendador de tarefas, é uma atividade organizacional feita pelo escalonador da CPU ou de um sistema distribuído, portanto, existe um controle e alocação de recursos de acordo com o tipo de sistema operacional, mantendo assim, as diferentes políticas de agendamento que podem ser implementadas.

Os tipos de sistemas operacionais podem ser: sistemas monoprogramáveis / monotarefa, sistemas multiprogramáveis / multitarefa e sistemas com múltiplos processadores. A figura abaixo, apresenta a estrutura monoprogramável, o programa

não utiliza todos os recursos do sistema, havendo a subutilização de recursos, segundo Machado (2013, p. 16):

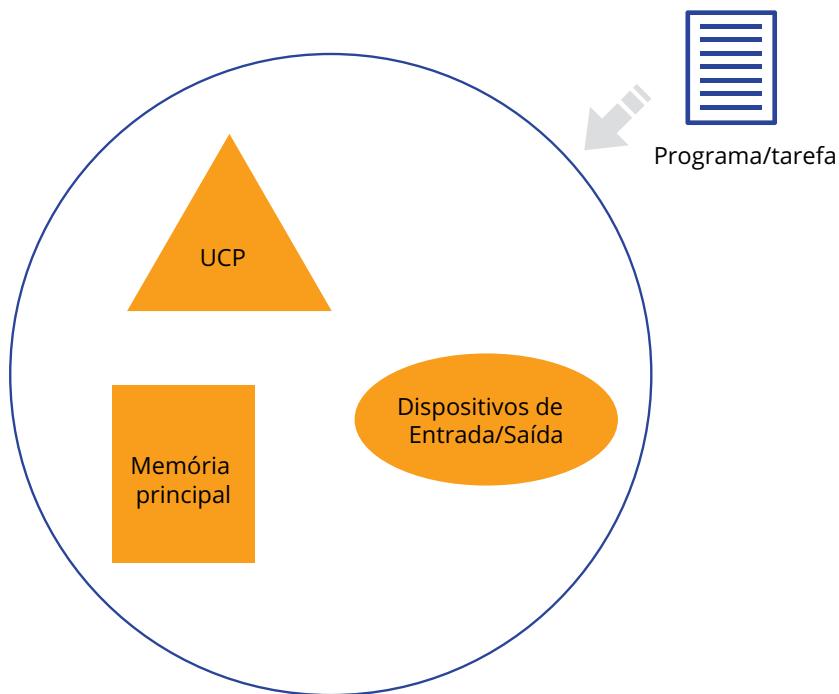


Figura 5 - Sistemas monoprogramáveis / monotarefa

Fonte: Machado (2013, p. 16).

#PraCegoVer: círculo com três formas dentro: um triângulo, no qual está escrito UCP; um retângulo, no qual está escrito Memória principal; e uma forma oval, na qual está escrito Dispositivos de Entrada e Saída. À direita e acima do círculo há um retângulo vertical com linhas sob o qual está escrito programa / tarefa. Ao lado desse retângulo, entre ele e o círculo, há uma seta que aponta para o círculo.

Para Machado (2013, p. 15):

Os **sistemas monoprogramáveis** estão tipicamente relacionados ao surgimento dos primeiros computadores na década de 1960. Posteriormente, com a introdução dos computadores pessoais e estações de trabalho na década de 1970, este tipo de sistema voltou a ser utilizado para atender máquinas que, na época, eram utilizadas por apenas um usuário. Os sistemas monotarefa, como também são chamados, caracterizam-se por permitir que todos os recursos do sistema fiquem exclusivamente dedicados a uma única tarefa.

Nos sistemas multiprogramáveis ou multitarefas, os recursos computacionais são divididos e utilizados pelos usuários e pelas aplicações.

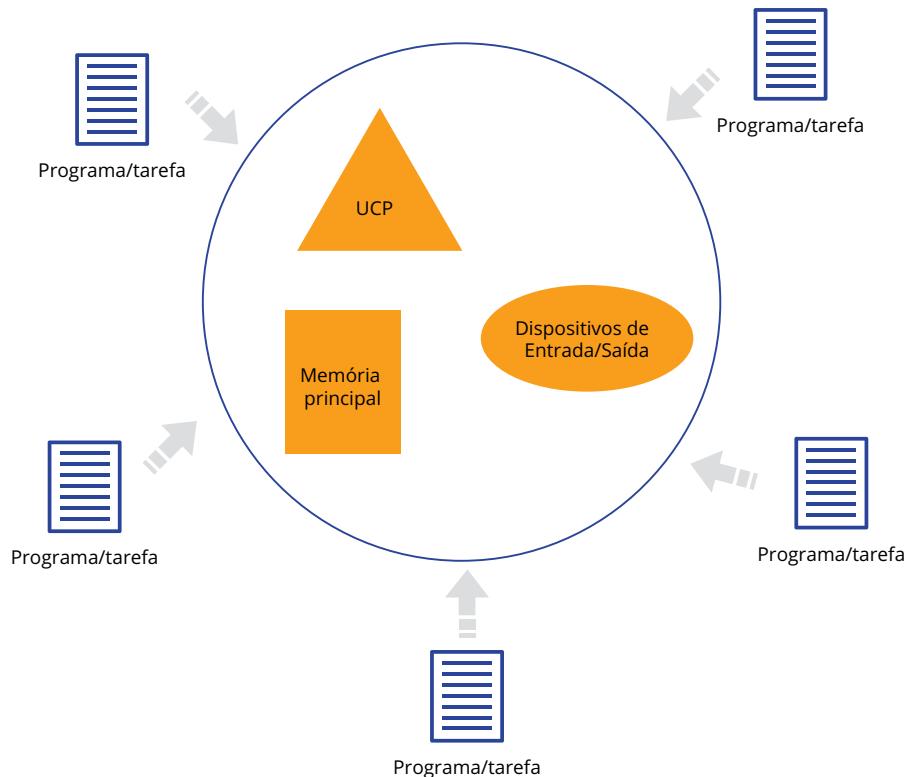


Figura 6 - Sistemas multiprogramáveis/multitarefa

Fonte: Machado (2013, p. 16).

#PraCegoVer: círculo com três formas dentro: um triângulo, no qual está escrito UCP; um retângulo, no qual está escrito Memória principal; e uma forma oval, na qual está escrito Dispositivos de Entrada e Saída. Ao redor do círculo há cinco retângulos verticais com linhas, sob os quais está escrito programa / tarefa. Ao lado de cada um desses retângulos, entre eles e o círculo, há uma seta que aponta para o círculo.

De acordo com Machado (2013, p. 36):

Os sistemas multiprogramáveis surgiram a partir de limitações existentes nos sistemas operacionais monoprogramáveis. Neste tipo de sistema, os recursos computacionais, como processador, memória e dispositivos de E/S, eram utilizados de maneira pouco eficiente, limitando o desempenho destas arquiteturas. Muitos destes recursos de alto custo permaneciam muitas vezes ociosos por longos períodos de tempo. Nos sistemas monoprogramáveis somente um programa pode estar em execução por vez, permanecendo o processador dedicado, exclusivamente, a essa tarefa.

A estrutura da arquitetura de um sistema operacional segue o princípio da separação de interesses, de acordo com Silberschatz (2015). Esse princípio apresenta a estruturação do sistema operacional em partes relativamente independentes, que

fornecem recursos individuais simples, mantendo assim a complexidade do design gerenciável.

Os sistemas operacionais modernos, diferentemente dos primeiros que surgiram, possuem uma estrutura de vários níveis, como o kernel do sistema operacional que trabalha diretamente com o hardware. Segundo Tanenbaum (2003), um kernel é um programa ou uma coleção de programas relacionados, que usa os recursos de hardware de um computador e é o único programa que permanece em execução no computador durante todo o tempo.

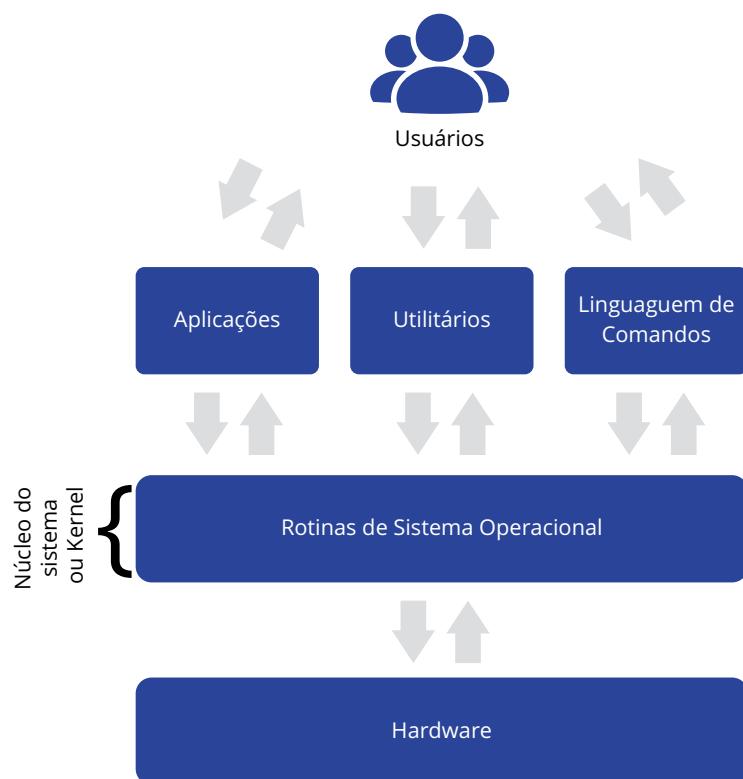


Figura 7 - Estrutura de um sistema de computação

Fonte: Machado (2013, p. 47).

#PraCegoVer: estrutura em camadas de um sistema de computação. Divide-se em: Aplicações, Usuários, Linguagem de comando, Utilitários, núcleo do sistema ou kernel, Rotinas do Sistema Operacional e Hardware.

A classificação dos sistemas operacionais pode ser por tamanho, suporte a recursos específicos, velocidade, acesso à rede e outros. Vejamos os detalhes a seguir.

- **Batch (de lote):** os programas eram colocados em uma fila para a execução. Hoje, a aplicação processa tarefas sem interação direta com o usuário. Portanto, o conjunto de comandos deve ser executado em sequência e sem interferência do usuário, este é o significado do termo *lote*.

- **De rede:** sistema capaz de oferecer aplicações locais, recursos que estejam localizados em outros computadores da rede, por exemplo, impressoras. Além disso, disponibiliza seus recursos de forma mais controlada.



Figura 8 - Sistemas de rede

Fonte: Plataforma Deduca (2021).

#PraCegoVer: círculo ao centro, com dois cilindros divididos em três partes cada um. Ao redor, há oito itens com setas entre eles e o círculo central apontando para ambos. Os itens ao redor são notebooks, CPUs, smartphones etc.

- **Distribuído:** em um sistema operacional distribuído, os recursos de cada máquina estão disponíveis globalmente, de forma transparente aos usuários. Ao lançar uma aplicação, o usuário interage com sua janela, mas não sabe onde ela está executando ou armazenando seus arquivos: o sistema é quem decide, de forma transparente.
- **Multiusuário:** identifica cada recurso dentro do sistema (arquivos, processos, áreas de memória, conexões de rede) e impõe regras de controle de acesso.
- **Desktop:** usuário doméstico e corporativo, ou seja, atividades básicas como edição de textos, tabelas, navegação na web e reprodução de mídias.



Figura 9 - Estrutura de um sistema de computação

Fonte: Plataforma Deduca (2021).

#PraCegoVer: foto de um notebook ao centro, com um tablet à esquerda, um celular à frente e óculos à direita.

- **Servidor:** permite a gestão de grandes quantidades de recursos como: memória, disco e processadores, priorizando limites sobre o uso dos recursos de aplicativos e usuários.

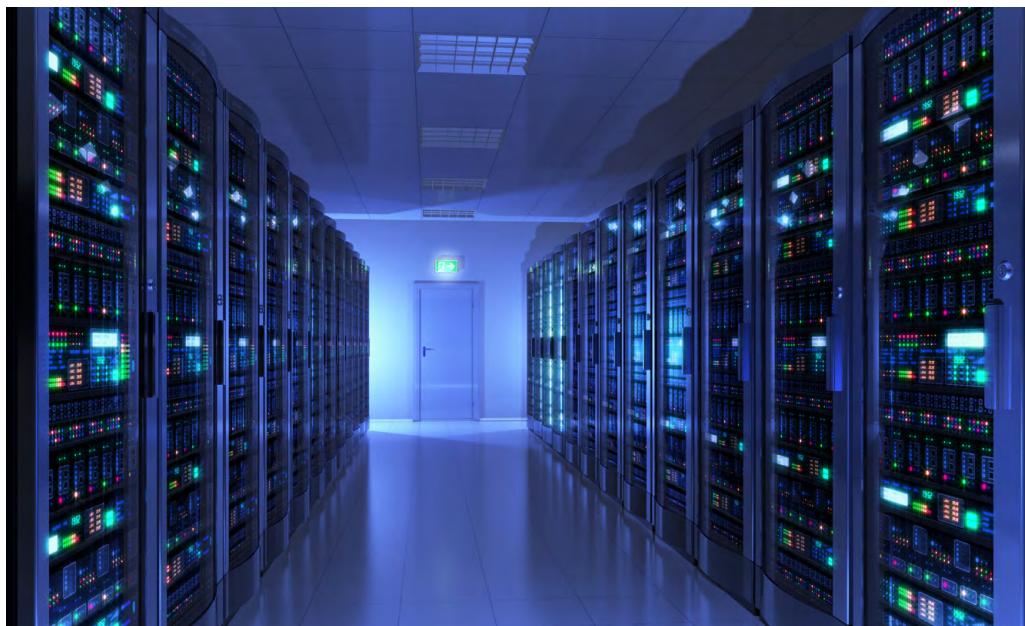


Figura 10 - Servidor

Fonte: Plataforma Deduca (2021).

#PraCegoVer: foto colorida de uma sala com servidores à esquerda e à direita.

- **Embarcado:** conhecido como embutido ou embedded. Opera com poucos recursos de processamento, armazenamento e energia, tais como: telefones celulares, automação industrial, equipamentos eletrônicos de uso doméstico.



Figura 11 - Sistemas embarcados

Fonte: Plataforma Deduca (2021).

#PraCegoVer: foto colorida mostra o painel de um automóvel com tela touch no centro. O motorista segura o volante com a mão esquerda e com a direita mexe na tela touch que está mostrando o GPS.

- **Tempo real:** há duas classificações de tempo real. Uma chamada de *soft real-time systems* (sistema em tempo real suave), que é a perda de prazos e implica na degradação do serviço prestado, por exemplo, gravação de CDs ou reprodução de mídias. Os sistemas de tempo real suave (soft) permitem a possibilidade de atraso em relação a um determinado intervalo, mas cada atraso tem um certo custo E outra é a *hard real-time systems* (sistema em tempo real duro), são usados quando um atraso na resposta do sistema não é permitido em hipótese alguma, em caso de atraso os resultados não são mais necessários para ninguém, e o atraso é considerado uma falha.



Curiosidade

O *Big Data* refere-se ao gerenciamento de uma grande quantidade de dados, enquanto o *Cloud Computing* (Computação em nuvem) é uma estrutura que possibilita o processamento e o armazenamento de dados de forma acessível via internet.

- **Computadores móveis:** SO sob medida capaz de gerenciar um hardware mais compacto em termos de processamento, memórias e dispositivos de entrada e de saída, envolve dispositivos pequenos em tamanho com capacidade de memória e processamento limitadas, com baixo consumo de energia, com curto tempo de inicialização e com conectividade limitada.



Figura 12 - Notebook

Fonte: Plataforma Deduca (2021).

#PraCegoVer: fotografia colorida mostra homem usando notebook. A seu lado há uma pilha com dois ou três livros.

Você já compreendeu que o SO libera os usuários finais e os desenvolvedores de aplicações do contato com os detalhes complicados da manipulação direta com o hardware. Para dar essa facilidade, o SO precisa oferecer uma série de chamadas para realizar as operações no hardware e nas aplicações.

As chamadas são denominadas interfaces de programação de aplicativos (APIs).

API

Uma API fornece uma chamada ao sistema. Por meio dela, o usuário instrui o SO a trabalhar. Assim, um desenvolvedor de aplicações, por exemplo, precisa saber apenas que rotinas do SO ele precisa chamar a fim de executar uma determinada tarefa no hardware. Já para o usuário comum, essas chamadas às APIs são praticamente transparentes.

Para Silberschatz (2015, p. 30):

A primeira entrada de que o programa precisará são os nomes dos dois arquivos: o arquivo de entrada e o arquivo de saída. Uma abordagem é aquela em que o programa solicita os nomes ao usuário. Em um sistema interativo, essa abordagem demandará uma sequência de chamadas de sistema, primeiro para exibir uma mensagem de alerta na tela e, em seguida, para ler a partir do teclado os caracteres que definem os dois arquivos. Em sistemas baseados em mouse e em ícones, é exibido, geralmente, um menu de nomes de arquivos em uma janela. Essa sequência requer muitas chamadas de sistema de I/O. Uma vez que os dois nomes de arquivo tenham sido obtidos, o programa deve abrir o arquivo de entrada e criar o arquivo de saída. Cada uma dessas operações requer outra chamada de sistema. Condições de erro que podem ocorrer, para cada operação, podem requerer chamadas de sistema adicionais. Quando o programa tentar abrir o arquivo de entrada, por exemplo, pode descobrir que não há arquivo com esse nome, ou que o arquivo está protegido contra acesso.

Entre os componentes centrais do SO, temos: escalonador de processos; gerenciador de memória; gerenciador de E/S (entrada e saída); gerenciador de comunicação interprocessos (IPC); e gerenciador de sistemas de arquivos.



Saiba mais

Um *driver* é um programa responsável por possibilitar a comunicação entre o SO e o hardware vinculado a ele. Praticamente todos os dispositivos de armazenamento ou de entrada e de saída de informações possuem um *driver* associado, desde um teclado, um mouse, uma impressora, até um HD, um pen drive, ou qualquer outro dispositivo.

Os SOs de hoje, tendem a ser muito complexos, principalmente em função da infinidade de dispositivos de hardware: de celulares a notebooks, de desktops a servidores.

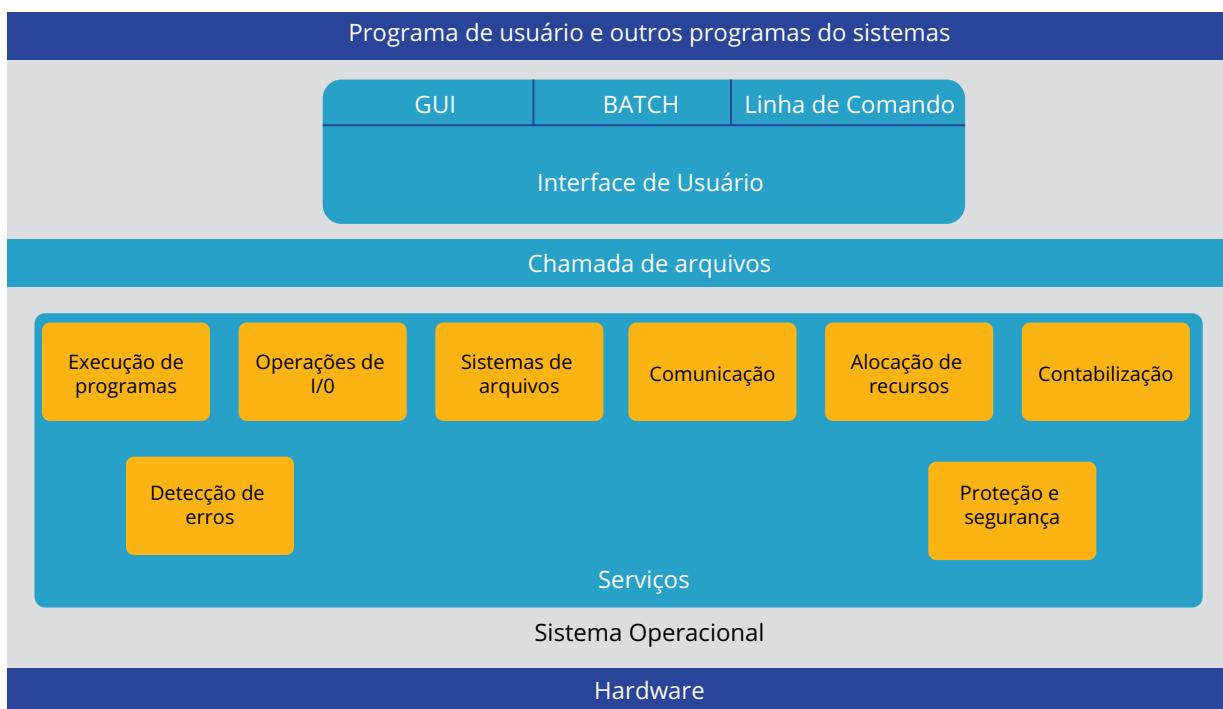


Figura 13 - Visão dos serviços do sistema operacional

Fonte: Silberschatz (2015, p. 86).

#PraCegoVer: retângulos subdivididos em itens. no topo, lê-se Programa de usuário e outros programas do sistema, que se divide em: GUI, BATCH e Linha de comando. Abaixo vem Interface de usuário e logo abaixo Chamadas de sistemas, que engloba: Execução de programas, Operações de I/O, Sistemas de arquivos, Comunicação, Alocação de recursos, Contabilização; Detecção de erros e Proteção e segurança e Serviços. Por fim aparecem o Sistema operacional e o hardware.

Tendo em vista essa diversidade, as diferentes arquiteturas existentes podem nos ajudar a compreender melhor a complexidade dos SOs atuais, responsáveis por comandar os diversos tipos de dispositivos com os quais lidamos todos os dias.

Os primeiros SOs foram desenvolvidos com base no modelo de **arquitetura monolítica**. Essa é uma das mais antigas e comuns arquiteturas de SOs. Nesse tipo de arquitetura, cada componente do SO, também chamado de módulo, está contido no núcleo, podendo comunicar-se diretamente com quaisquer outros componentes. Essa comunicação direta entre os componentes faz desse tipo de SO um ambiente altamente eficiente. Portanto, a construção do núcleo do SO é modular; baseada em módulos, a partir dos quais é gerado um executável.

O desenvolvimento e a manutenção desse tipo de sistema são difíceis. Além disso, em função do agrupamento desses componentes, é complicado isolar um problema que aconteça em algum deles. Geralmente, esse tipo de arquitetura é utilizado em SOs embutidos, principalmente porque esse tipo de sistema se adequa mais a pequenos hardwares. Tais hardwares são mais limitados em termos de componentes e se destinam a funções mais específicas. Ainda assim, se forem feitos complementos, esse tipo de arquitetura pode ser útil em aplicações mais abrangentes.



Saiba mais

Segundo Fowler (2016), o termo arquitetura de microserviços surge nos últimos anos para apresentar uma maneira particular de projetar aplicativos de software como suítes de serviços implementáveis independentemente. Lucio, J. P. D.. ANALISE COMPARATIVA ENTRE ARQUITETURA MONOLÍTICA E DE MICROSERVIÇOS. Departamento de Informatica e Estatística. Universidade Federal de Santa Catarina (UFSC). Disponível em: <https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/182309/ArtigoSBC.pdf?sequence=2&isAllowed=y>. Acesso em: 30/09/2021.

A **arquitetura em camadas** surgiu porque os SOs se tornaram maiores e mais complexos e, nesse caso, os componentes que realizam tarefas semelhantes são agrupados em camadas. Cada uma dessas camadas comunica-se exclusivamente com a camada imediatamente acima ou abaixo dela. Esse tipo de sistema é considerado modular, pois uma implementação em uma camada pode ser modificada sem exigir modificações em outras camadas. Essa modularidade impõe mais consistência e estrutura ao SO.



Figura 14 - Hierarquia

Fonte: Plataforma Deduca (2021).

#PraCegoVer: fotografia colorida mostra mãos e tronco de uma pessoa dentro organizando bloco de forma hierárquica.

Na hierarquia de um sistema em camadas, cada uma delas trata do gerenciamento de alguma parte do hardware. Assim, uma determinada camada fornece e gerencia uma visão abstrata da parte do hardware para camadas superiores. Entretanto, um nível não é obrigado a usar os serviços de um nível inferior.

O THE (computador – Electrologica X8)

Um dos primeiros SOs a adotar a modelagem em camadas O THE (computador – Electrologica X8) foi criado por Edsger Dijkstra e seus alunos. Possuía uma estrutura de seis camadas constituída da seguinte forma:

Camada 0: alocação do processador e pela multiprogramação.

Camada 1: gerenciamento (alocação) de memória para os processos.

Camada 2: comunicação entre operador-processo (interpretador de comandos e o sistema operacional) e pela comunicação entre processos.

Camada 3: gerenciamento de entrada/saída.

Camada 4: programas de usuário.

Camada 5: chamada de operador de sistema.

O problema nesse tipo de disposição em camadas é que uma solicitação do usuário precisa passar por várias camadas até ser atendida. Sistemas como Windows XP e LINUX implementam esse tipo de arquitetura em camadas.

Já a **arquitetura de micronúcleo** fornece um número pequeno de serviços para manter um núcleo pequeno e escalável. Por isso, nesse tipo de arquitetura, a maioria dos componentes é executada fora do núcleo. O gerenciamento de memória de baixo nível, a comunicação e a sincronização entre processos estão entre os serviços oferecidos pelo micronúcleo. Como o micronúcleo não depende de cada um dos componentes para ser executado, mesmo que haja falha individual em um dos componentes, tal lapso não comprometerá o funcionamento do SO. Além de ser extensível, portável e escalável, o sistema com base em micronúcleo também possui alto grau de modularidade, fato que requer um nível maior de comunicação entre os módulos. Essa comunicação expandida pode comprometer o desempenho do sistema.

Gerenciamento de Processos

Os computadores possuem a capacidade de executar várias tarefas ao mesmo tempo. Durante a execução de um programa, o computador pode também fazer a leitura de um pen drive, executar um vídeo ou fazer o download de um arquivo da internet simultaneamente.



Figura 15 - Pen drive

Fonte: Plataforma Deduca (2021).

#PraCegoVer: foto colorida mostra um pen drive azul.

Para Tanenbaum (2003), um processo é apenas um programa em execução acompanhado dos valores atuais do contador de programa, dos registradores e das variáveis.

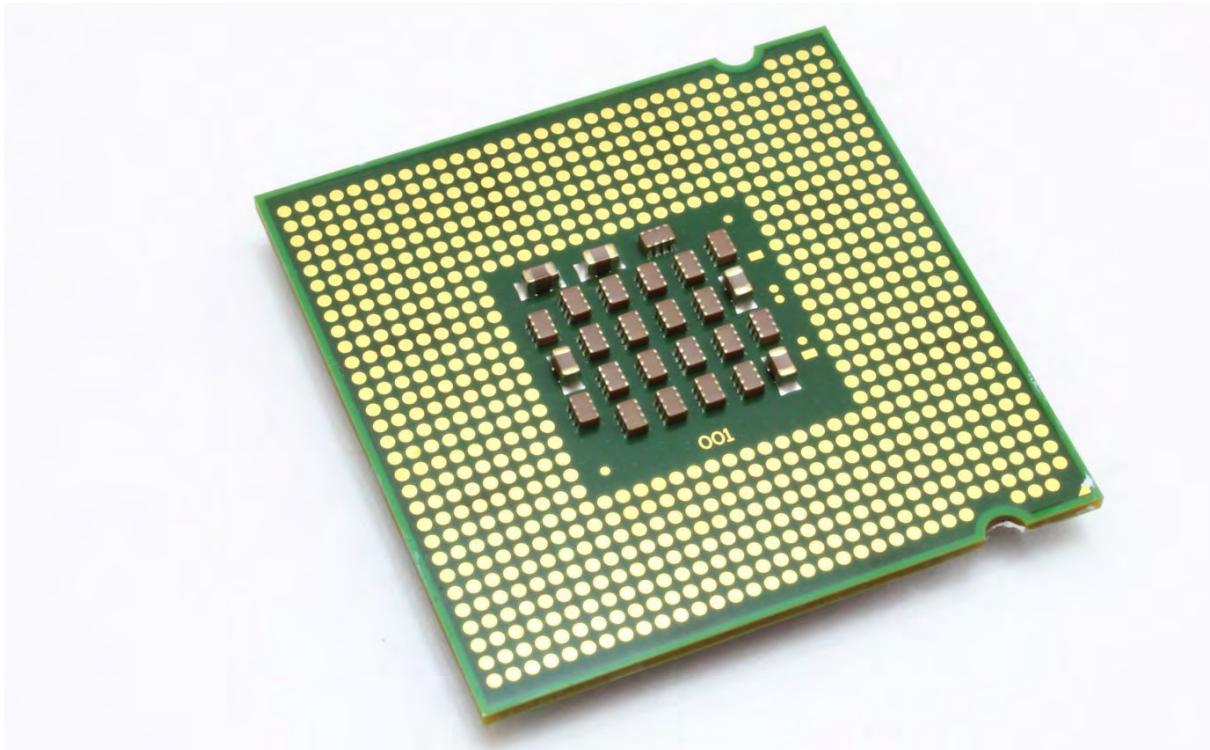


Figura 16 - Processador

Fonte: Plataforma Deduca (2021).

#PraCegoVer: A imagem é um processador, quadrado com pontos de encaixe dentro da placa mãe.

Conceitualmente, cada processo tem sua própria CPU virtual. É claro que, na realidade, a CPU troca, a todo momento, de um processo para outro, mas, para entender o sistema, é muito mais fácil pensar em conjunto de processos executando (pseudo) paralelamente do que tentar controlar o modo como a CPU faz essas alternâncias.

Em um computador multiprogramado, existem vários processos competindo pela CPU ao mesmo tempo. Essa ocorrência acontece quando dois ou mais processos se encontram em estado de prontidão. No caso de somente uma CPU estar disponível, uma escolha deverá ser feita por ela para saber qual processo será executado em seguida. A parte do sistema operacional responsável por essa escolha é denominada escalonador de processos.

Consideremos três níveis de escalonamento: alto, médio e baixo. O **escalonamento de alto nível**, também conhecido como escalonamento de *Jobs*, ou ainda como escalonamento em longo prazo, tem a finalidade de identificar a quais *jobs* o sistema dará permissão para disputarem os recursos de sistema de modo ativo. Pelo fato de poder determinar quais *jobs* serão admitidos pelo sistema, às vezes, esse nível é chamado escalonamento de admissão. Ao serem admitidos e inicializados, os *jobs* vão se tornar processos ou grupos de processos.

Após um *job*, que pode ter um ou mais processos, ter sido admitido pela política de escalonamento de alto nível, os processos permitidos para competir por processadores serão determinados pela política de **escalonamento de nível médio**. Essa política tem o poder de suspender e retomar processos por um determinado período de tempo com a finalidade de conseguir uma operação sem alterações do sistema. A política de **escalonamento de baixo nível** de um sistema tem como objetivo determinar quais dos processos ativos o sistema indicará a um processador.

Objetivos de escalonamento

Para desenvolver uma disciplina de escalonamento, um projetista de sistemas deve levar em consideração vários fatores, como o tipo do sistema e a necessidade do usuário.

Podemos afirmar que as disciplinas de escalonamento se classificam em preemptivas e não preemptivas.

Uma disciplina é **preemptiva** quando o sistema tem o poder de retirar o processador do processo que estiver executando. Nesse caso, o processador poderá executar apenas uma parte do código de um processo e, em seguida, fazer um chaveamento do contexto.

Uma disciplina é **não preemptiva** quando o sistema disponibilizar uma CPU a um processo e não puder mais retirar o processador que foi designado a esse processo. Nesse contexto, podemos afirmar que, cada processo, no caso de ter recebido um processador, executará até concluir ou até devolver espontaneamente seu processador.



Atenção

Escalonamento preemptivo: nos sistemas em que processos de alta qualidade exigem uma resposta em curto tempo, utiliza-se o escalonamento preemptivo.

Nos sistemas **não preemptivos**, no caso dos processos mais curtos, a atividade solicitada pode ficar muito tempo para ser atendida, pois ele deve esperar a conclusão dos processos mais longos.

Uma vez que a qualidade de funcionamento de todo o SO também depende da qualidade do planejamento, faz-se necessário considerar os algoritmos de escalonamentos. De acordo com Tanenbaum (2003), um processo precisa estar em uma dessas situações: pronto, em execução, ou bloqueado. Quando falamos em **execução** significa que um processo em execução é o processo que está realmente usando a CPU naquele momento. Quando um processo está **pronto**, está numa fila a espera para acessar ao CPU e alterar seu estado de Pronto para em execução, mas não pode obter acesso à CPU devido a outro processo que o utiliza. Porque o processo foi bloqueado? Ele não deixa de estar bloqueado até quando está **bloqueado** não pode ser executado até que algum evento externo ocorra.

Objetivos do algoritmo de escalonamento de processos

Os objetivos do escalonamento de processos são: utilização máxima da CPU; alocação razoável da CPU; taxa de transferência máxima; tempo mínimo de retorno; tempo mínimo de espera e tempo mínimo de resposta.

Alguns algoritmos de escalonamento, segundo aponta Silberschatz (2015, p. 139-140), são os seguintes:

- **First Come First Serve (primeiro a chegar primeiro a ser servido) (FCFS)**: algoritmo de escalonamento mais simples, que programa de acordo com os tempos de chegada dos processos.
- **Shortest Job First (trabalho mais curto) (SJF)**: é um algoritmo de escalonamento que atribui a cada processo a duração de seu próximo tempo de execução da CPU. É um algoritmo de escalonamento não preemptivo.
- **Longest Job First (trabalho mais longo primeiro) (LJF)**: é semelhante ao algoritmo de agendamento SJF, mas nesse algoritmo de escalonamento, damos prioridade ao processo com o maior tempo de execução.
- **Shortest Remaining Time First (menor tempo restante primeiro) (SRTF)**: é o modo preemptivo do algoritmo SJF, no qual os trabalhos são agendados de acordo com o menor tempo restante.
- **Longest Remaining Time First (maior tempo restante mais longo) (LRTF)**: é o modo preemptivo do algoritmo LJF, no qual damos prioridade ao processo com o maior tempo de burst restante.
- **Round Robin**: cada processo é atribuído a um tempo fixo (*Time Quantum* (tempo quântico) / *Time Slice* (tempo de fatia)) de forma cíclica.

Conclusão

Ao longo dos anos, as histórias sobre a computação vêm evoluindo de forma exponencial. Neste conteúdo, conhecemos um pouco sobre os sistemas operacionais (SO), a história da computação e o que isso trouxe para os dias atuais. Vimos também que o hardware e os sistemas operacionais passaram por grandes evoluções.

A interação humano-computador se tornou tão simples que todos podem trabalhar com ela. Assim, para a grande maioria dos usuários, um computador não teria utilidade sem ter alguma forma de sistema operacional, ou seja, um futuro de desenvolvimento da computação, recursos e inovações na tecnologia para a humanidade.

Referências

MACHADO, F. B. **Arquitetura de sistemas operacionais**. 5. ed. Rio de Janeiro: LTC, 2013.

Fowler, S. **Product Ready-Microservices**.ed. O'Reilly Media, 2016.

SILBERSCHATZ, A. **Fundamentos de sistemas operacionais**. 8. ed. Rio de Janeiro: John Wiley & Sons, 2015.

TANENBAUM, A. S. **Sistemas operacionais modernos**. 2. ed. São Paulo: Pearson Prentice Hall, 2003.

WAZLAWICK, R. S. **História da computação**. Rio de Janeiro: Elsevier, 2016.