Fundamentos de softwares de computadores

Prof. Guilherme Dutra Gonzaga Jaime

Descrição

Conceitos básicos e primordiais sobre a execução de softwares. Apresentação das formas como instruções escritas por programadores para serem executadas por CPUs.

Propósito

Compreender a lógica dos softwares, que são parte de nosso mundo e permanecerão assim por muito tempo.

Objetivos

Módulo 1

Conceitos básicos de softwares

Reconhecer conceitos básicos sobre softwares.

Módulo 2

Softwares funcionais

Identificar softwares funcionais, como sistemas operacionais e o firmware.

Módulo 3

Tipos de linguagens de programação

Diferenciar os dois principais tipos de linguagens de programação.

Introdução

Para que os computadores atuais possam ter alguma utilidade para nós, usuários, é necessário que eles recebam instruções das ações que devem executar, por exemplo, para que você assista a um vídeo, ouça um podcast ou leia este texto.

Essas instruções são passadas aos computadores por meio dos softwares, que podemos chamar por agora de programas de computador. São esses programas, com inúmeras finalidades, que determinarão as ações

que podem ser desenvolvidas pelo computador.

De uma maneira simples, os programas são formados por instruções que são executadas por um componente chamado processador. O resultado do processamento das instruções ou de um conjunto de instruções determinará a finalidade do programa.

Estamos falando de computadores, mas isso não ocorre apenas neles. Quaisquer dispositivos eletrônicos podem ter softwares para executar uma ação específica, como controlar o funcionamento de uma geladeira, ou de propósito mais amplo, como o computador que você carrega no seu bolso, o seu smartphone.

Introdução

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.

1 - Conceitos básicos de softwares

Ao final deste módulo, você será capaz de reconhecer conceitos básicos sobre softwares.

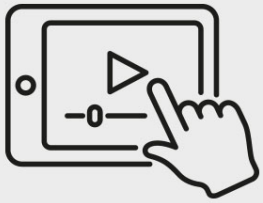
Conhecendo o software

Aqui descreveremos os conceitos básicos sobre software de computadores e responderemos, de forma introdutória, às seguintes perguntas:

- O que é software?
- O que é código de computador?
- Como os softwares rodam (são executados) em um computador?

O que é software?

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Sempre que você baixa um programa ou vê um computador ligado, está presenciando algum software sendo executado.

Até mesmo no passado, telégrafos, telefones, aparelhos de fax, todos já possuíam um software, um conjunto de ações programadas que usavam a estrutura física e lhes davam materialidade: voz, letras e imagens. Cada máquina exercia exatamente a sua função, e repare: estamos indo além da função mecânica.

Exemplo

Imagine uma máquina de costura.

Ela faz sempre a mesma operação, um movimento mecânico repetitivo. Porém, quem atua para que ela execute funções a partir do domínio de técnica é o sujeito; a máquina mecânica é estruturada, e a mente humana é quem faz a função.

Quando um sujeito decidiu que era possível programar a máquina para fazer além da função repetida, para trançar uma estrutura, sucedendo e alcançando a partir disso objetivos claros e novos, criou-se o software. A máquina é capaz de ser programada para cumprir um conjunto de funções definidas e desenhadas pelo sujeito.

O computador é filho da II Guerra Mundial, da necessidade de novos desenvolvimentos, de realizar processamento de informações e comunicação mais eficientes, cruzando dados e armazenamento.

> **Não basta construir a máquina; é necessário executar, definir o que se deseja, programar e reprogramar.**

Softwares para computadores passaram a ser uma demanda, uma construção e sofisticação contínuas.

Sempre que um software é criado, imediatamente passamos a pensar em como melhorá-lo, transformá-lo em algo mais eficiente. Pessoas vivem disso e recriam isso.

Nunca mais dominaremos perfeitamente todos os softwares que desejamos; afinal, sempre que aprendermos sobre um, novas mudanças estarão chegando.

Já jogou videogame? Esses aparelhos mudaram e continuam mudando; atualmente, possuem gráficos e programações cada vez mais complexos.

Para entender tudo isso, é necessário compreender o que é, afinal, um software para computadores.

Afinal, o que é software?

Software é o conjunto de funções executadas na programação de um computador.

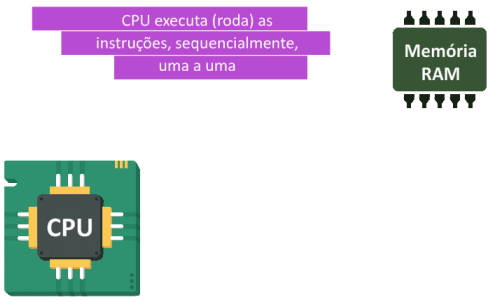
Embora não seja uma entidade física, ganha materialização em sua construção virtual, independentemente da rede.

É a manifestação de uma imagem e dos desdobramentos a partir da construção de códigos e execuções.

Vamos entender melhor esse processo:

> **A CPU (*Central Processing Unit*), também chamada de processador, é uma espécie de cérebro. É ela quem realmente executa o que chamamos de "instruções de código de máquina".**

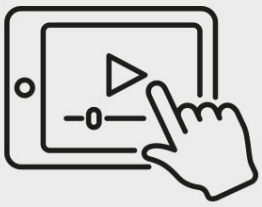
Veja a imagem a seguir:



Execução de instruções pelo computador.

Representação e armazenamento das instruções em hardware

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Essas instruções são extremamente simples e constituídas apenas dos números 0 (zero) e 1 (um), conhecido como código binário.

Podemos pensar no conjunto de instruções de código de máquina que uma CPU é capaz de executar como sendo um idioma de **baixo nível**, ou "código de máquina" (também conhecido como "código nativo"). O idioma do código da máquina está intimamente conectado ao projeto do hardware da CPU, ou seja, não é algo que possa ser mudado à vontade.

Cada família de CPUs compatíveis (por exemplo, a popular família Intel x86 usada em computadores pessoais) possui seu próprio código de máquina específico, que não é compatível com o código de máquina de outras famílias de CPUs.

Vamos a um exemplo:

Extremamente simples

Por exemplo, em uma instrução de código de máquina, é possível adicionar dois números; já em outra pode-se comparar dois números para ver qual deles é maior.

Hardware

Hardware é o termo usado para se referir à parte física da coisa, o equipamento.

Não é compatível com o código de máquina de outras famílias

Por exemplo: os programas de PCs não rodam em smartphones, pois os idiomas de baixo nível dessas CPUs são **incompatíveis**.

Exemplo

Na linguagem JavaScript, a instrução (linha de código) para definir um pixel (ponto) da tela com nível de vermelho para o nível 255 seria:

pixel.setRed(255).

Essa instrução é muito mais complexa do que uma instrução de código de máquina individual que CPUs são capazes de executar. Então, nós a chamamos de uma instrução de alto nível.

> **Instruções de alto nível são facilmente compreensíveis pelos humanos, mas os computadores não são capazes de executá-las.**

O que ocorre, então, é que, antes de ser executada, a instrução de alto nível será expandida em uma sequência – talvez cinco ou dez instruções de baixo nível (código de máquina) –, de modo que, quando essas cinco ou dez instruções forem executadas, uma após a outra, o resultado final terá o efeito de definir o valor vermelho do pixel para 255, conforme designado pela instrução de alto nível escrita pelo programador. Esse processo está resumido na imagem a seguir.

Compilador traduz linhas de código escritas pelo programador em instruções de código de máquina compreensíveis à CPU.

Você pode estar se perguntando:

> ### Como corrigir um erro de programação cometido pelo programador?

Se você quiser adicionar um recurso para corrigir um bug no Firefox (por exemplo), a forma real de fazer isso é voltar ao **código-fonte original** e realizar os ajustes necessários, alterando as instruções.

Em seguida, é necessário executar o **compilador** novamente para compilar (traduzir de linguagem C++ para linguagem de máquina) uma nova versão do Firefox que incluirá os ajustes realizados no código-fonte.

Software de código aberto

É importante construirmos uma noção do que significa software de código aberto.

Trata-se de uma forma de distribuição de software em que o programa compilado é fornecido, mas também há acesso ao código-fonte original do programa.

Geralmente, o código-fonte é acompanhado de uma licença que diz algo como:

“Aqui está o código-fonte, se você quiser criar sua própria versão realizando as alterações que desejar, fique à vontade.”

Atenção!

Trata-se de uma forma de distribuição de software, normalmente gratuita, em que o programa compilado é fornecido, mas também há acesso ao código-fonte original do programa.

Mas qual a diferença entre softwares de código aberto e softwares de código fechado?

Código aberto

Se forem necessários ajustes, correções e incrementações que você queira fazer com o programa – ou se o fornecedor original não existe mais –, você tem total liberdade para agir.

Você pode realizar as modificações, ou talvez prefira contratar alguém para produzir sua própria versão personalizada.

×

Código fechado

Se você precisa de algum recurso diferente ou adicional, ou se há um bug que precisa ser corrigido, você realmente depende do fornecedor, pois somente ele detém o código-fonte.

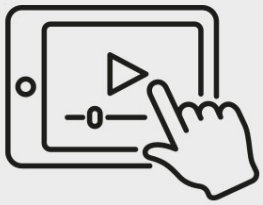
Então, só o fornecedor é capaz de realizar ajustes e correções.

Em geral, existem tipos diferentes de termos de licença para software de código aberto, mas, na maioria das vezes, eles exigem que, se você fizer alterações no código-fonte e adicionar algum recurso, você deve oferecer essas alterações à comunidade. Assim, da mesma maneira que você se beneficiou dos outros ao obter o programa gratuito e seu código-fonte, eles podem se beneficiar do seu trabalho.

Softwares de código aberto X softwares de código fechado

No vídeo a seguir, apresentamos reflexões sobre os impactos dos softwares de código aberto e fechado na sociedade.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Até aqui, comentamos sobre uma categoria de linguagens de programação em que compiladores são usados para traduzir o código-fonte, de forma a obter um arquivo executável que pode ser distribuído aos usuários. Agora, vamos nos concentrar na seguinte questão do próximo assunto.

O que é um programa?

Observe o lado direito desta imagem e perceba que a CPU está executando uma sequência de instruções presentes na memória RAM. Então, um programa/aplicativo como, por exemplo, o navegador de Internet Firefox, ou o editor de textos Microsoft Word, nada mais é do que uma enorme sequência dessas instruções simples de código de máquina.

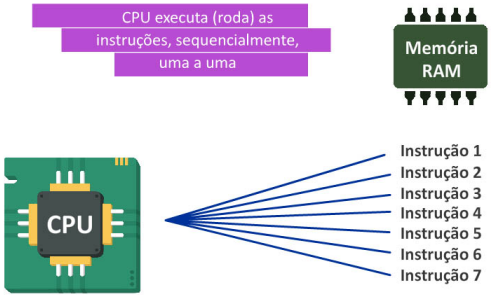
Assim, quando o Firefox está sendo executado no seu computador, isso significa que, em algum lugar na RAM, há um bloco dessas instruções, e a CPU as executa uma após a outra sequencialmente.

Tudo o que você pode ver o Firefox fazendo – como piscar o cursor, conectar-se via rede a uma URL fornecida pelo usuário, desenhar imagens na tela, obter páginas da Web, entre outros – ocorre graças à CPU, que roda as instruções de forma tão inacreditavelmente rápida que você interage com o Firefox de forma fluida e natural.

Portanto, as instruções individuais são realmente triviais, certo?

De que forma isso leva o cursor a piscar no ponto onde podemos digitar algo?

Imagem



Resposta

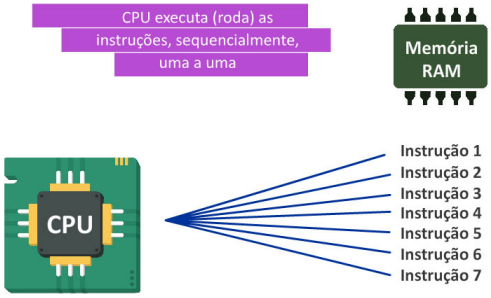
Uma forma bastante simples de pensar sobre isso é usar uma analogia: o relacionamento entre areia e escultura.

Cada instrução em código de máquina seria um grão de areia, que é sem sentido e parecido com todos os outros. Porém, se você juntar grande quantidade de maneira certa, pode construir uma estrutura complexa, conforme sua imaginação e seu objetivo.

Em linhas gerais, é assim que programas como o Firefox, Chrome, Microsoft Word, os jogos digitais, ou qualquer outro aplicativo são construídos.

Se olharmos à esquerda da **imagem**, veremos a CPU trabalhando para executar as instruções de código de máquina disponíveis na RAM.

Imagem



Comentário

Para executar as instruções de código de máquina, as instruções usam um método chamado ciclo busca-execução (Fetch Execute Cycle).

No ciclo busca-execução, a CPU iniciará buscando a instrução 1, carregando-a para dentro de si (CPU), e a executará.

Por exemplo, a CPU adicionará os dois números. Após executar a instrução 1, a CPU simplesmente desce na lista e executa novamente o ciclo busca-execução para a instrução 2. Em seguida, faz o mesmo para a instrução três, e assim por diante. As instruções são executadas uma após a outra, sequencialmente.

Saiba mais

Quando dizemos que uma CPU opera a 4 GHz (gigahertz), ou 4 bilhões de operações por segundo, estamos nos referindo exatamente a essas pequenas instruções.

Existe uma enorme variedade de tipos de instruções, mas há dois tipos específicos que vale a pena comentarmos, mesmo em um curso introdutório. Veja a seguir:

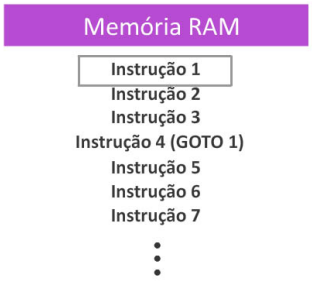
Instrução para alterar a ordem de execução de instruções

Normalmente, a CPU apenas desce a lista e faz a execução das instruções na ordem em que elas estão armazenadas na memória.

Conforme ilustrado na imagem a seguir, digamos que a instrução quatro diga: “Pule para trás e comece a executar novamente na instrução 1”.

Nesse caso, a CPU executaria as instruções na ordem: 1, 2, 3, 4 e, em seguida, em vez de seguir para a instrução 5, daria um pulo para trás e iria para (**goto**) a instrução 1, executando-a novamente, seguida das instruções 2 e 3.

É assim que as estruturas de repetição de instruções (loops) são implementadas pela CPU.



Instrução 4 solicitando que a CPU, em vez de seguir para a instrução 5, volte a executar a instrução 1.

Você sabe o que é **goto**?

A goto é uma instrução na linguagem de informática e significa: vá para a linha indicada e siga executando o código a partir desse local.

Instrução que testa alguma condição

Se a condição for verdadeira, diz, por exemplo, que ela avance para a instrução 5.

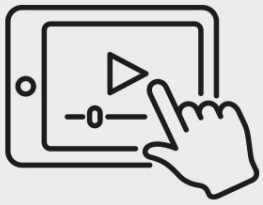
Há uma instrução que olhará para alguma condição; assim, se a condição for verdadeira, é como dizer à CPU: “Vá para (goto) a linha indicada”. Caso seja falsa, a CPU vai para (goto) outra linha.

Então, organizando as instruções, você pode obter o efeito de uma estrutura condicional, conhecida como **declaração if**.

A seguir, você fará uma descoberta a respeito do programa.

Como o programa vai parar na memória RAM?

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Ao serem instalados, programas são colocados em um dispositivo de armazenamento persistente, como HD, SSD ou pendrive.

Vamos entender com o exemplo do Firefox:

Exemplo

O programa é basicamente um arquivo chamado **Firefox.exe**, que possui muitos bytes. Na maioria dos casos, esses bytes são apenas as instruções que compõem o programa, além de alguns ícones e fotos.

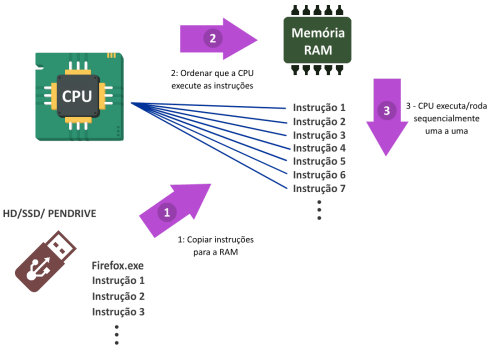
irefox.exe

".exe" trata-se de uma convenção de nomenclatura do Windows usada para nomear o arquivo que é um programa. Isso ajuda a manter as coisas mais claras.

Outros sistemas operacionais podem gerar, por exemplo, um arquivo chamado Firefox, sem extensão, que é o equivalente do Firefox.exe no Windows.

O que acontece quando você clica duas vezes no arquivo Firefox.exe, ou no atalho que aponta para esse arquivo?

Basicamente, são os três passos indicados na imagem a seguir:



O que é carregar/iniciar um programa?

Veja a descrição de cada um desses passos:

Passo 1

Cópia dos bytes (ou seja, instruções) que compõem o arquivo do dispositivo de armazenamento persistente (HD, por exemplo) para uma área desocupada da memória RAM. Essa cópia é comumente chamada de **carregar** (*load*) o programa.

Passo 2

Após o carregamento (cópia do HD para a RAM), a CPU já é capaz de realizar o ciclo busca-execução para rodar o programa Firefox. Então, o passo dois consiste em apenas dizer à CPU: “Aqui está a instrução 1 de um programa; você deve iniciar, a partir dessa instrução, o ciclo busca-execução para rodar o programa”.

Passo 3

A CPU começa o processo de rodar/executar as instruções de forma incrivelmente rápida. Pronto, nosso Firefox está rodando, e o usuário já consegue navegar pela Internet.

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Depois que uma CPU executa uma instrução, o que, geralmente, faz a seguir?

A Apaga as instruções da RAM.

B Executa a instrução anterior na sequência de instruções.

C Executa a próxima instrução na sequência de instruções.

D Grava a instrução no disco.

E Apaga as instruções no disco.

Parabéns! A alternativa C está correta.

Dada a lista de instruções de um programa carregadas na RAM, a CPU as executa uma a uma, sequencialmente, na ordem em que aparecem.

Questão 2

Em código de máquina, para que serve a instrução goto?

A Alterar a ordem de execução de instruções pela CPU, como, por exemplo, para implementar loops de repetição e estruturas condicionais.

B Ordenar ao sistema operacional que finalize a execução de um programa.

C Ordenar ao sistema operacional que a ordem de execução de instruções pela CPU, por exemplo, seja iniciada pela execução de loops de repetição e estruturas condicionais.

D Ordenar à CPU que pause a execução de instruções para aguardar que algum evento definido ocorra.

E Ordenar à CPU que seja lido uma posição da memória RAM indicada pela linha de comando goto.

Parabéns! A alternativa A está correta.

Conforme vimos, a instrução goto é usada para alterar a ordem de execução de instruções. Dois importantes exemplos da utilidade dessa instrução de código de máquina são implementar loops de repetição e implementar estruturas condicionais.

2 - Softwares funcionais

Ao final deste módulo, você será capaz de diferenciar softwares funcionais, como sistemas operacionais e o firmware.

Sistema Operacional

Definição

Comumente, quando se estuda conceitos básicos de software e como ocorre a execução de instruções pela CPU, surgem questionamentos como:

- Quem lida com o duplo clique usado para ordenar que um programa seja executado?
- Quem garante que um novo programa carregado na RAM irá para uma região da memória que esteja realmente ociosa, sem que haja sobreposição de outros programas em execução?
- Quem ordena o carregamento de um programa do HD para a RAM?

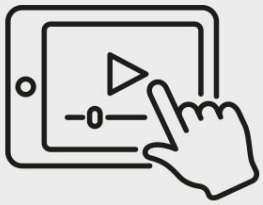
- Ao final da execução de um programa, quem realizada as operações, como, por exemplo, liberar a região da RAM onde o programa esteve durante sua execução?

A resposta às perguntas descritas anteriormente é a seguinte: o **Sistema Operacional (SO)**.

> **O SO é um conjunto de tipos de programas administrativos e de supervisão que organizam todo o sistema. O sistema operacional está para os computadores como o governo está para uma nação.**

O sistema operacional e seu papel em sistema computacional

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



O SO de um computador é como um primeiro programa de supervisão que começa a ser executado quando o computador é inicializado (“inicializa”). Ele desempenha um papel administrativo e contábil invisível nos bastidores.

Quando um desktop, laptop ou smartphone é iniciado, o SO, normalmente, organiza as coisas e inicia um programa “explorador de arquivos” que exibe janelas e menus, entre outros, que mostram ao usuário quais sistemas de arquivos estão disponíveis. Isso permite que ele navegue e opere seu equipamento, solicitando que programas/aplicativos sejam iniciados ou finalizados, interagindo com esses programas/aplicativos.

Em resumo, conforme ilustrado na imagem a seguir, o SO é responsável por intermediar as solicitações que os usuários fazem a programas/aplicações, gerenciando como tudo isso deve ser demandado do hardware.

O sistema operacional e seu papel em sistema computacional.

O sistema operacional mantém as coisas organizadas em segundo plano para que vários programas possam ser executados ao mesmo tempo, o que é conhecido como “multitarefa”. Ele fornece a cada programa sua própria área de memória, de modo que cada programa acessa apenas seus próprios recursos, tentando limitar, por motivos de segurança, o que um programa incorreto ou mal-intencionado pode fazer.

Manter os programas separados é conhecido como “área restrita”. Isso é importante para que cada programa funcione independentemente, sem interferir em outros programas ou no sistema como um todo. Da mesma forma, cada programa tem algum acesso à tela por meio de uma janela, mas essa área de saída é separada da saída de outros programas.

Vamos ver dois exemplos?

Arquivo Word.exe

Um arquivo .exe é essencialmente apenas um arquivo de instruções de código de máquina. Quando você clica duas vezes no programa, está ordenando que o SO “inicie” o programa, executando as etapas de limpeza de alocação de uma área de memória na RAM para o programa, carregando a primeira seção do código de máquina do programa nessa memória e, finalmente, direcionando a CPU para começar a executar esse código.

Câmera digital

Uma câmera digital também é um pequeno computador. Quando é iniciado, ele não executa um programa de gerenciamento de arquivos. Em vez disso, depois que a limpeza básica for configurada, a câmera poderá executar um único programa que desenha os menus, entre outros, na tela da câmera e responde a cliques nos botões da câmera, e assim por diante.

Boot e Reboot

O sistema operacional é, antes de qualquer programa do usuário, a primeira coisa a ser executada quando seu computador é ligado.

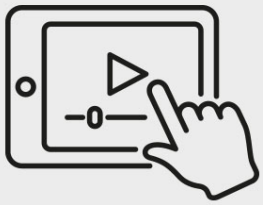
Somente após o carregamento do SO, o usuário pode selecionar os programas que deseja rodar, clicando duas vezes nos ícones correspondentes na área de trabalho.

Você pode estar se perguntando:

> **Qual é o programa que cuida da inicialização do SO logo que o computador é ligado?**

Processo de boot em um computador

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Há um programa especial, e muito pequeno, denominado **firmware**, que é gravado pela fábrica no hardware.

O **firmware** é responsável por detectar quando um computador estava desligado e acaba de ser ligado, e realiza alguns procedimentos iniciais de teste de hardware, para então, basicamente, procurar um dispositivo de armazenamento persistente que contenha um SO instalado.

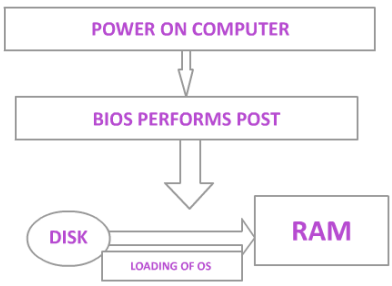
Depois disso, é possível, então, iniciar seu carregamento (cópia das instruções que compõem o SO do disco para a RAM e indicação para a CPU de qual é a primeira instrução do SO a ser executada).

Comentário

O termo **firmware** é usado, geralmente, para se referir especificamente ao firmware de inicialização, que controla um computador desde o momento em que é ligado até o sistema operacional principal assumir o controle.

A principal função do firmware de inicialização é inicializar o hardware e, em seguida, inicializar (carregar e executar) o sistema operacional principal.

Veja, a seguir, o esquema de um computador pessoal. Nesse tipo de computador, o firmware de inicialização é chamado, geralmente, de BIOS (*Basic Input/Output System*), ou sistema básico de entrada e saída.



Processo de boot em um computador.

Este processo é comumente denominado **inicializar** (boot ou boot up).

E quando ocorre o **Reboot**?

Resposta

O reboot ocorre quando ordenamos ao SO que o computador seja reiniciado, ou seja, o próprio SO cuida das tarefas necessárias para a finalização e o desligamento do computador. Imediatamente, o computador é religado, e o processo de boot recomeça.

Versões de sistemas operacionais

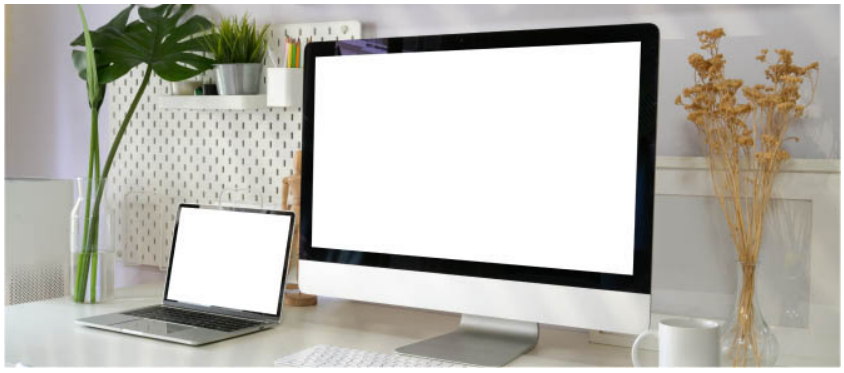
Como já mencionado, a primeira coisa a ser executada quando seu computador é ligado é o SO. Antes mesmo de qualquer outro programa do usuário. Agora que conhecemos conceitualmente os sistemas operacionais, que tal darmos uma olhada em alguns exemplos de SO e onde são aplicados?

Computador pessoal (PC)

Microsoft Windows: proprietário, pago. Usado em cerca de 80% dos desktops/laptops.

Linux: aberto, gratuito. Usado em cerca de 80% dos servidores em ambientes de computação em nuvem. Em desktops e laptops, é usado principalmente por entusiastas e projetos de inclusão social.

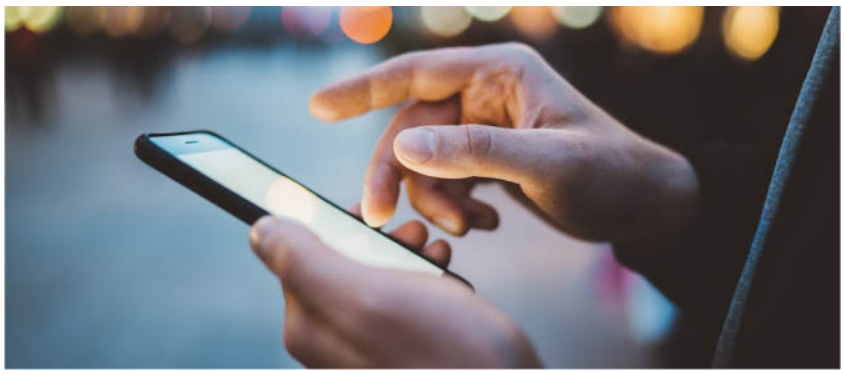
Mac OS X: proprietário, pago, específico para computadores Apple. Usado por cerca de 5% dos desktops/laptops.



Smartphone

IOS: proprietário, específico para Apple Iphone.

Android: aberto, usado pelos demais fabricantes (Samsung, Motorola, LG, Sony etc.). Projeto baseado no Linux.



Depois do que estudamos até aqui, podemos fazer algumas perguntas:

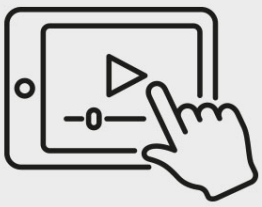
Somente os sistemas operacionais e os firmwares são exemplos de softwares funcionais?

O que define um software como funcional? Qual a sua importância no âmbito do pensamento computacional?

O Software e a sua função

Neste vídeo, você verá o que é um software e qual é a sua função.

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Suponha que você clique duas vezes no Firefox.exe em um computador para executá-lo. Qual das seguintes opções descreve melhor o que acontece?

A O sistema operacional copia as instruções do Firefox do disco para a RAM e, em seguida, a CPU executa as instruções na RAM.

B As instruções para o Firefox são traduzidas para JavaScript e depois executadas pelo navegador.

C O sistema operacional copia as instruções do Firefox para a RAM e, em seguida, a RAM executa as instruções.

D O hardware, de maneira autônoma, copia as instruções do Firefox do disco para a RAM e, em seguida, a CPU executa as instruções na RAM.

E O sistema operacional conecta o dispositivo à Internet e abre a página de busca padrão.

Parabéns! A alternativa A está correta.

Conforme estudamos, o sistema operacional é o nome dado ao conjunto de programas administrativos e de supervisão que intermedeiam a interação humano-computador. Entre as funções do SO, está receber o comando do usuário para iniciar programas (exemplo: duplo clique) e realizar as tarefas administrativas necessárias para que o programa seja executado. Em específico, quando o usuário dá um duplo clique em um arquivo de programa, o SO é quem comanda o carregamento do programa, ou seja, a cópia do conteúdo do arquivo de programa do disco para a RAM. Depois, ele indica à CPU onde, na RAM, está a primeira instrução do programa a ser executado.

Questão 2

Firmwares são componentes fundamentais de computadores, pois:

A são responsáveis por definir qual é o conjunto de instruções suportado por determinada CPU.

B são usados para capturar interações do usuário e fornecê-las ao sistema operacional.

C são responsáveis por dar início ao processo de boot, para carregamento do sistema operacional.

D são usados para traduzir códigos de computador, que são escritos por programadores em código de máquina.

E são responsáveis por guardar as informações na memória RAM, de acordo com a instrução da CPU.

Parabéns! A alternativa C está correta.

O firmware consiste em programas instalados semipermanentemente na memória, usando vários tipos de chips ROM programáveis, como PROMS, EPROMs, EEPROMs e chips flash. O firmware não é volátil e permanece na memória depois que você desliga o sistema.

3 - Tipos de linguagens de programação

Ao final deste módulo, você será capaz de diferenciar os tipos de linguagens de programação.

Conceito

Em sistemas computacionais contemporâneos, é extremamente raro escrever códigos de máquina manualmente. Isso ocorre porque eles são compostos por um número enorme de instruções muito simples; assim, fica difícil para os humanos fazerem esse processo.

Em vez disso, um programador escreve o código (instruções) em uma linguagem de computador de "alto nível", com recursos mais úteis e poderosos do que as operações simples encontradas no código da máquina.

Exemplos de estruturas de alto nível são:

- Estruturas de repetição (loops), em que o programador ordena que um conjunto de instruções seja executado repetidamente;

- A função print(), que imprime algo na tela;
- A estrutura condicional (if), em que o programador solicita que alguma condição seja testada e, caso passe no teste, algumas instruções sejam executadas.

lto nível

O termo “alto nível” designa a ideia de que as instruções são mais próximas de nossa língua falada, se comparadas ao código de máquina, que é de “baixo nível”.

Nenhum desses recursos de alto nível está diretamente presente no código da máquina de **“baixo nível”**. Eles são adicionados por linguagens de programação, como JavaScript, Java, Python, C, C++, entre outras.

Tipos de linguagem de programação

A classificação de linguagens de programação é mais detalhada e abrangente do que a apresentada aqui, mas vamos nos limitar, por questões de simplicidade, a agrupá-las nestas categorias.

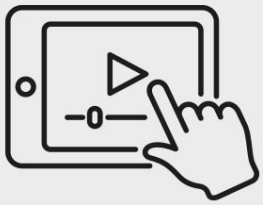
O funcionamento de uma linguagem de programação é definida por seu tipo. Veja, a seguir, quais são.

Linguagem Compilada

São aquelas em que o processo de tradução (compilação) é feito com antecedência e o código é necessariamente executável.

Linguagem Compilada

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



> “
>
> Quando se utiliza uma linguagem compilada, é necessário executar um programa para traduzir os arquivos-fonte, legíveis em linguagem de alto nível, em código executável. As linguagens compiladas têm a vantagem de produzir código de alta performance, o qual está ajustado para o funcionamento em um tipo específico de processador ou arquitetura de processador. Aplicativos compilados, chamados de código binário, só podem rodar no tipo de computador para o qual foram compilados, uma vez que esses aplicativos consistem, na realidade, em instruções em linguagem de máquina, entendidas e executadas pelo microprocessador.
>
> **(INDRUSIAK, 1996, p. 4)**

Veja um exemplo de linguagem compilada:

Na primeira linha do código a seguir, escrito na linguagem C++, o texto/string “Bom dia” está sendo atribuído à variável “a” e, na segunda linha, a variável “b” recebe o conteúdo da variável “a” (ex.: “Bom dia”), e a exclamação é adicionada ao final da frase.

a = **“Bom dia”**;

b = **a + “!”**;

Então, o programador escreve o que é chamado de **código-fonte** na linguagem de programação que escolher. Como vimos, humanos preferem linguagens de alto nível pois são mais fáceis e intuitivas.

Como a CPU consegue executar (rodar) instruções escritas em linguagem de alto nível se sabemos que a CPU só executa código de máquina?

Uma das estratégias utilizadas é usar o **compilador**.

Como já vimos, trata-se de um software de propósito muito específico: olhar para o código-fonte escrito pelo programador e traduzi-lo, para criar um grande corpo de código de máquina compatível com a CPU em que o programador deseja rodar o programa.

Exemplo

Talvez haja uma parte do código-fonte onde exista uma instrução **if** (estrutura condicional), mas não uma instrução específica em um código de máquina para uma instrução **if**.

No entanto, talvez haja uma sequência de cinco instruções de código de máquina que, na verdade, chegam ao mesmo resultado de uma instrução **if**. Portanto, o compilador faz esse tipo de expansão.

Vamos usar o Firefox como exemplo novamente:

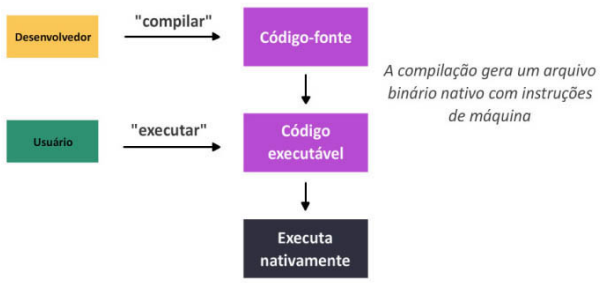
Esse navegador é escrito em C++. Assim, para criar uma nova versão do Firefox, após realizar os ajustes desejados em seu código-fonte, alguém executa o compilador C++, que lê o grande corpo de código-fonte em linguagem de alto nível que constitui o Firefox e produz, essencialmente, o arquivo Firefox.exe. Esse arquivo é a saída do compilador e contém as instruções de código de máquina obtidas mediante a tradução do código-fonte escrito pelos desenvolvedores do Firefox.

A etapa de compilação pode ser feita uma vez e bem antes da execução do programa (por exemplo, produza o Firefox.exe na sede da entidade que o desenvolve, a Mozilla, e depois distribua o Firefox.exe para que usuários de PC com sistema operacional Windows possam usá-lo).

Atenção!

A compilação só precisa ser feita pelo desenvolvedor/programador uma vez.

Conforme ilustrado pela imagem a seguir, o desenvolvedor/programador, que escreveu o código-fonte, realiza a compilação e cria o arquivo executável (exemplo: Firefox.exe), e pode simplesmente enviá-lo para que outras pessoas consigam rodá-lo em seus computadores, contanto que sejam compatíveis com o código de máquina gerado. Os usuários finais não precisam do código-fonte nem do compilador.



Processo de disponibilização de um programa escrito em linguagem compilada.

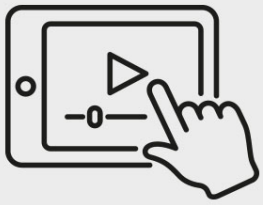
Preste atenção:

> **O processo não funciona ao contrário.**

Ou seja: não é possível, a partir das instruções em código de máquina do Firefox.exe, realizar a tradução reversa e obter o código-fonte em linguagem de alto nível originalmente escrita pelo(s) programador(es). É até possível obter uma versão imperfeita do código-fonte original, mas ela ficará bem distante do ideal.

Linguagem dinâmica ou interpretada

Para assistir a um vídeo sobre o assunto, acesse a versão online deste conteúdo.



A linguagem dinâmica se diferencia por se pautar no tempo de execução, cruzando os dados com os protocolos por meio de bibliotecas, criando “metaobjetos”, quer dizer, bibliotecas complexas de combinação e execução. Java, JavaScripts e Python são exemplos de linguagens de programação dinâmicas/interpretadas.

Uma forma de compreender essa categoria é pensar que, em vez do compilador, é usado outro software de propósito especial denominado interpretador.

Trata-se de um programa que lê código-fonte escrito em uma linguagem como Java, JavaScript, Python, entre outras, e o “executa/roda”.

Exemplo

Talvez o melhor exemplo para linguagem interpretada seja o JavaScript:

Um interpretador para a linguagem JavaScript vem embutido em navegadores de Internet, como o Firefox, o Chrome, o Microsoft Internet Explorer ou o Microsoft Edge.

Quando um navegador se depara com um website que contenha algum código JavaScript embutido, ele pode usar seu interpretador para executar esse código. Portanto, de forma bastante resumida, a maneira como um intérprete funciona é a seguinte: ele executa uma linha de código por vez.

Supondo que o exemplo a seguir seja um código-fonte contendo duas instruções escritas em JavaScript, quando o interpretador JavaScript do navegador fosse executar isso, ele olharia a primeira linha e a executaria.

```
//Código Javascript

a = 1;

b = a + 1;
```

Portanto, neste exemplo, o interpretador diria: “Acho que preciso de um nome de variável ´a´, e preciso colocar o valor ‘1’ nela”. Então, após executar essa linha, ele seguiria em frente e interpretaria/executaria a próxima linha, e assim por diante.

Podcast

Agora que conhecemos as duas linguagens, aperte o play para ouvir o professor Rodrigo Dias comparando-as.

Para ouvir o *áudio*, acesse a versão online deste conteúdo.

Tendências

Falando de modo geral, a tendência para a programação de computadores caminha para o uso de linguagens dinâmicas/interpretadas. Isso ocorre porque há um consenso de que é bastante atraente poder programar de forma mais simples e eficiente, mesmo sabendo que o programa final rodará mais lentamente na CPU.

Isso talvez seja um pouco contraintuitivo, mas podemos refletir sobre essa questão da seguinte forma:

Qual o recurso mais escasso em programação de computadores?

Geralmente, a resposta é: o programador!

Em todo o mundo, o mercado de trabalho da área da computação sofre, cada vez mais, com a falta de profissionais qualificados. Portanto, usar menos horas de trabalho dos programadores é um atrativo muito importante.

O fato de que o programa rodará mais lentamente na CPU é considerado menos relevante, até porque, conforme a lei de Moore, as CPUs estão cada vez mais baratas e o poder de processamento delas cresce continuamente.

Então, se pensarmos em qual será a tendência para os próximos anos, também há um consenso, no mercado, de que o programador continuará a ser considerado um recurso cada vez mais escasso se comparado ao poder de processamento das CPUs.

Para finalizar, vale a pena comentar sobre uma coisa chamada **JIT** (*Just in Time Compiler*). O objetivo dos JITs é tentar obter o melhor dos dois mundos: linguagens compiladas e linguagens interpretadas.

A ideia é ter os benefícios da maior simplicidade de desenvolvimento de programas usando linguagens dinâmica e, ao mesmo tempo, obter um programa que rode mais rapidamente na CPU. Dessa forma, o JIT é responsável por ler parte do código-fonte e tentar compilá-lo rapidamente, antes de executá-lo. O mais interessante é que isso funciona muito bem!

Comentário

Navegadores modernos de Internet, como o Firefox, o Chrome, o Microsoft Internet Explorer e o Microsoft Edge, agora embutem JITs para código JavaScript. Assim, na verdade, quando você está executando o código JavaScript dentro do navegador, o JIT examina trechos do código dinâmico (Javascript) que estão sendo executados com muita frequência e compila o código nativo desses trechos em tempo real.

O intérprete não é usado para casos simples, mas para seções importantes do código dinâmico (como o interior de uma estrutura de repetição), e o JIT cria um bloco de código de máquina na memória.

O código da máquina é executado para essa seção do código dinâmico, oferecendo desempenho semelhante a linguagens compiladas, como C e C++, e é descartado quando o programa é encerrado.

Note que, mesmo com o uso de JITs, linguagens interpretadas possuem desempenho inferior ao de linguagens compiladas com C e C++.

Além do JavaScript, a linguagem Java também usa a tecnologia JIT extensivamente. O enorme ganho de desempenho dos navegadores de Internet nos últimos anos deve-se, em grande parte, à implementação da tecnologia JIT para JavaScript.

Falta pouco para atingir seus objetivos.

Vamos praticar alguns conceitos?

Questão 1

Qual das opções a seguir descreve melhor o que um compilador C++ faz?

A Traduz o código da máquina em código JavaScript.

B Traduz o código-fonte C++ em código JavaScript.

C Traduz o código de máquina em código C++.

D Traduz o código-fonte C++ em código de máquina.

E Interpreta o código C++ e envia à CPU.

Parabéns! A alternativa D está correta.

Já sabemos que CPUs apenas são capazes de executar instruções de baixo nível, ou código de máquina. Entretanto, como instruções de baixo nível são de difícil compreensão para humanos, foram desenvolvidas linguagens de programação de alto nível que admitem instruções de mais fácil compreensão. Nós, humanos, preferimos escrever software em linguagens de alto nível, como, por exemplo, C++. Então, para que a CPU seja capaz de executar as instruções, precisam ser traduzidas da linguagem de alto nível em que foi escrita para a linguagem de máquina. O software que realiza essa tradução é denominado compilador. Portanto, um compilador C++ é responsável por traduzir códigos escritos em C++ para o código de máquina da CPU.

Questão 2

O código de máquina da CPU possui apenas instruções simples e de baixo nível. Uma linguagem de computador (como JavaScript) adiciona recursos de alto nível, como o loop que usamos. Qual das alternativas a seguir é uma instrução de baixo nível?

A Passe por todos os pixels da imagem.

B Se X < 5, então, execute a instrução a = 1.

C Adicione dois números.

D Salve determinada informação no arquivo.

E Carregue o próximo vídeo.

Parabéns! A alternativa C está correta.

Conforme estudamos, CPUs só são capazes de executar instruções muito simplificadas, denominadas instruções de baixo nível, ou código de máquina. Entre as opções listadas, a única que contém uma

instrução de baixo nível é a opção “Adicione dois números”. As demais instruções são de alto nível e, para serem executadas pela CPU, deverão ser traduzidas/desmembradas em um conjunto, em geral de 4 a 6, de instruções de código de máquina.

Considerações finais

Neste conteúdo, você mergulhou na representação do que é um software para computadores, reconhecendo alguns softwares funcionais que fazem parte do seu cotidiano. No fim, aproximou-se da linguagem de programação, podendo, com isso, perceber como a linguagem e sua dinâmica são atualizadas continuamente.

Podcast

Ouça agora um bate-papo sobre fundamentos de softwares de computadores.

Para ouvir o *áudio*, acesse a versão online deste conteúdo.

Explore +

Pesquise e leia o texto *Mercado de TI pode apresentar déficit de 290 mil profissionais em 2024*, publicado em 2019 no site itforum.com.br.

Procure na Internet o artigo *Proposta de integração da engenharia de software nas estratégias empresariais*, de Adalberto Reis e Ivanir Costa, que apresenta uma reflexão sobre essa importante área do pensamento computacional.

Pesquise também o livro *Autonomia, liberdade e software livre*, de Doriedson de Almeida e Nicia Riccio, que traz importantes reflexões a partir do tema “código aberto”.

Consulte o artigo *A eficácia do software de educação ambiental utilizado no ensino à distância*, de Franklin Porto Jr., que apresenta a importante articulação de três temas extremamente atuais: pensamento computacional, meio ambiente e educação a distância.

Referências

CARVALHO, A.; LORENA, A. **Introdução à Computação:** hardware, software e dados. 1. ed. Rio de Janeiro: LTC, 2017.

DALE, N.; LEWIS J. **Ciência da Computação.** 4. ed. Rio de Janeiro: LTC, 2011.

FEDELI, R. D.; POLLONI, E. G. F.; PERES, F. E. **Introdução à Ciência da Computação.** 2. ed. São Paulo: Cengage, 2010.

FLANAGEN, D. **Javascript:** o guia definitivo. 6. ed. Porto Alegre: Bookman, 2013.

GLENN, J. **Ciência da Computação:** uma visão abrangente. 11. ed. Porto Alegre: Bookman, 2013.

INDRUSIAK, L. S. **Linguagem Java.** Consultado em meio eletrônico em: 28 abr. 2020.

TORETTA, L. **Market Share Statistics for Internet Technologies.** netmarketshare.com. Consultado em: 27 abr. 2020.

Material para download

Clique no botão abaixo para fazer o download do conteúdo completo em formato PDF.

Download material

O que você achou do conteúdo?

☆ ☆ ☆ ☆ ☆

Relatar problema