

Introdução à Informática

```
procedure SendFile(Socket: TCustomWinSocket; FName: string );
var
  Header: TMsgHeader;
  Fs: TFileStream;
  S: TMemoryStream;
begin
  // se o arquivo NAO existe
  if not FileExists(FName) then
    SendError(Socket, MSG_ERR_FILE_NOT_FOUND)
  else
    try
      // abre o arquivo
      Fs := TFileStream.Create(FName, (fmOpenRead or fmShareDenyWrite));
      Fs.Position := 0;
      // prepara o cabeçalho
      with Header do begin
        Length := MSG_ERR_FILE_NOT_FOUND;
        MsgSize := Fs.Size;
      end;
      // cria o "pacote"
      S := TMemoryStream.Create;
      // escreve o cabeçalho
      S.Write(Header, SizeOf(Header));
      // anexa o conteúdo do arquivo
      S.CopyFrom(Fs, Fs.Size);
      // send to socket
      S.Position := 0;
      Socket.SendStreamThenDrop(S);
      // **Nao precisa destruir o TMemoryStream, sera destruido automaticamente**
      // mas precisa destruir o TFileStream...
      Fs.Free;
    except
      // em caso de erro...
      SendError(Socket, MSG_ERR_CANNOT_SEND);
    end;
end;
```

Noções de Software

José Luís Carneiro



Salvador
2006

Introdução

Um computador nada mais é do que uma máquina, projetada para minimizar o esforço dos seres humanos na execução de tarefas rotineira. Porém, sem o software ou programa, essa máquina tão sofisticada não serve para nada. Por isso, vamos tentar entender todos os aspectos ligados à formulação dos softwares para computadores e suas características principais. O software, ou programa de um computador é organizado em camadas ou níveis, partindo do hardware propriamente dito até as aplicações do usuário.

Programa de computador é um conjunto de comandos ou instruções que tem por finalidade a solução de um problema ou tarefa.
--

Máquinas virtuais

Como todos sabemos, se ligarmos o nosso computador e não enviarmos a ele alguns comandos ele simplesmente não executará tarefa alguma.

Você já sabe que a eletricidade faz o computador funcionar, que ele tem vários circuitos eletrônicos e que esses mesmos circuitos são os responsáveis por suas funções.

Então, para “conversar” com ele precisamos de alguns ajudantes (intérpretes), que traduzirão os nossos comandos (no formato de palavras) na linguagem que o computador entende de fato (sinais elétricos). Assim funciona o computador, utilizando os programas como nossos intérpretes. Mas há uma distância entre a linguagem “entendida” pela máquina e a linguagem que utilizamos para a nossa comunicação. Imaginemos o computador como um conjunto de camadas ou níveis, tendo cada nível a sua própria linguagem.

Mas, por que usar níveis e linguagens?

Acontece que o computador usa um código específico, formado por bits e bytes, enquanto o ser humano se comunica por meio de palavras. Obviamente, são duas linguagens muito diferentes, e por isso foram necessários vários aperfeiçoamentos para que pudéssemos atingir os atuais níveis de “comunicação” com o computador.

Ainda assim, é muito difícil manter uma conversação direta com a máquina, e por esse motivo existem os níveis, que aperfeiçoam a linguagem já existente em níveis inferiores.

Cada vez que uma linguagem é melhorada, ela fica mais próxima da usada pelo ser humano. Aliás, o objetivo é esse mesmo, tornar a linguagem do computador cada vez mais parecida com a linguagem humana.

Pensar num computador como uma máquina de vários níveis ajuda os projetistas a entenderem melhor os problemas relacionados à construção de um computador. É sempre melhor dividir um grande problema em problemas menores, facilitando assim a sua resolução. Uma **máquina virtual** consiste numa linguagem e nos comandos que ela pode executar em cada nível. Nesse sentido, observe a figura a seguir. Percorreremos todos os seus níveis, do mais baixo (nível 0) ao mais elevado (nível 6).

Nível	Categoria	Exemplos				
6	Aplicações do usuário	Jogos	Processadores de texto	Planilhas eletrônicas	Bancos de dados	Sistemas comerciais
5	Linguagens orientadas à resolução de problemas	Visual Basic, Delphi, C++, Java, Pascal, etc.				
4	Linguagem de montagem	Montadores, compiladores e interpretadores				
3	Sistema operacional	MS-DOS, Windows, Linux, Unix, MacOS, etc.				
2	Máquina convencional	<i>Assembly</i>				
1	Microprogramação	Instruções gravadas em CPUs				
0	Lógica digital	Projetos de <i>hardware</i>				

Nível 0 – Lógica Digital

O nível 0 corresponde ao nível mais primitivo de um computador; o nível de hardware. O que podemos encontrar nesse nível são os circuitos digitais mais simples (*AND*, *OR*, *NOT*, *NAND* e *NOR*) construídos a partir de um componente eletrônico chamado transistor.

Os circuitos integrados (CI's ou chips) contêm vários circuitos lógicos que os projetistas utilizam para a construção de seus computadores. Os fabricantes de tais circuitos buscam sempre torná-los cada vez menores e, ao mesmo tempo, aumentam sua capacidade, dotando-os a cada dia de mais funções.

Resumindo, o nível da lógica digital trabalha apenas com os algarismos 0 e 1 e realiza operações elementares com a informação, armazenando-a e movendo-a conforme as necessidades. Nessa camada ainda não existe o conceito de programa.

Nível 1 – Microprogramação

Nos computadores mais antigos, cada instrução do processador tinha seus próprios circuitos para realizar uma operação. Isso era possível porque a quantidade de instruções disponibilizadas pelo processador não era tão grande e, nesse caso, a implementação ficava mais fácil.

Ao se colocar cada instrução de um computador com o seu próprio circuito, naturalmente a complexidade da construção do processador cresce bastante. Tentando resolver esse problema, alguns estudos levaram à construção de uma camada intermediária entre o nível 0 (lógica digital) e o nível 2 (a máquina convencional), o verdadeiro nível das instruções de máquina, o nível da **microprogramação**.

O nível da microprogramação faz uso dos circuitos eletrônicos (que fazem parte do nível 0) para implementar as instruções de um computador, combinando os circuitos lógicos que o nível da Lógica Digital oferece, a fim de efetuar multiplicações, divisões, transportar os bytes entre os registradores da CPU e a RAM, controlar o tempo que um certo byte estará disponível para efetuar uma operação aritmética, permitir transferências entre as memórias de um computador (RAM, *cache* interna ou externa), etc.

<p>Microprograma é um conjunto de vários pequenos programas que são utilizados para executar as instruções do computador, que serão utilizadas no nível superior.</p>
--

Uma instrução ou comando é executado através de vários passos e uma combinação de passos para executar uma determinada função resulta em um **microprograma**. Nessa máquina virtual existem vários pequenos microprogramas para executar várias funções do processador.

Nível 2 – Máquina Convencional

O nível de máquina convencional lida diretamente com o conjunto de instruções oferecido pelos microprocessadores, (cada fabricante possui o seu próprio projeto de instruções de máquina).

As instruções desse nível operam os registradores do microprocessador, acessam a memória principal, controlam os periféricos, executam operações aritméticas, etc. Assim, os níveis 0, 1 e 2, respectivamente, os da Lógica Digital, da Microprogramação e da Máquina Convencional trabalham o hardware propriamente dito do computador.

Nível 3 – Sistema Operacional

Por volta de 1943, surgiram os primeiros computadores, equipamentos gigantescos que ocupavam várias salas. As pessoas que operavam essas máquinas eram extremamente qualificadas e eram obrigadas a trabalhar diretamente com os circuitos da máquina. Quando acontecia algum problema, como válvulas queimadas, era preciso identificá-las e trocá-las, para que tudo voltasse a funcionar normalmente. Os programadores daquela época tinham que se preocupar com detalhes do funcionamento interno do computador.

Os cartões perfurados surgiram nesse momento, e qualquer trabalho era feito com eles. Os programadores tinham que literalmente perfurar os cartões e colocá-los nos computadores, inserindo manualmente os cartões que interpretariam o seu programa previamente perfurado. Se ocorresse erros, a correção dos mesmos obrigaria a executar novamente toda essa rotina, mesmo que estivesse faltando apenas o ponto final no programa. Em consequência, os programadores perdiam muito tempo nessa tarefa, e o trabalho de programação era de grande complexidade. Chegou-se à conclusão, então, de que estava faltando um agente facilitador, algo que assumisse a tarefa de gerenciar todo esse trabalho, retirando-o da responsabilidade do programador. Esse agente, depois de criado, transformou-se numa nova máquina virtual, presente em todos os computadores modernos: o **sistema operacional**.

O sistema operacional tem como objetivo gerenciar o funcionamento do computador como um todo. Qualquer tipo de programa que quisermos executar em nossa máquina necessitará primeiramente de um sistema operacional instalado, senão nenhum programa poderá ser rodado.

<p>Serviços são comandos enviados do usuário ao sistema operacional para a execução de algumas tarefas.</p>
--

Naturalmente, o sistema operacional precisa “entender” bastante de *hardware* para poder liberar os programadores e usuários dessa tarefa. Ele disponibiliza para os programadores e usuários certas tarefas, comumente chamadas de **serviços do sistema operacional**, ou simplesmente serviços.

Nível 4 – Linguagem de Montagem

Estamos no antepenúltimo nível de Máquinas Virtuais do computador. Até o nível do sistema operacional, os programas e as instruções estão em um formato numérico (hexadecimal, por exemplo) mais familiar para o ser humano, embora ainda mantenham muito da linguagem da máquina, pois não é tão fácil assim transformar sinais elétricos em números - é bom não esquecer que a programação nos níveis precedentes está ainda ligada intimamente ao hardware.

O sistema operacional, apesar de facilitar muito a forma de tratamento do *hardware*, ainda está impregnado de instruções não muito amistosas para o ser humano. E até difícil imaginar, por exemplo, quantos passos têm que ser dados no *hardware* para colocar um caractere ou o ponteiro do mouse na tela, e mais trabalhoso ainda é buscar, no disco rígido, os bytes necessários à execução de um programa. Várias etapas têm que ser cumpridas para que essas operações, aparentemente muito simples, possam ser executadas com sucesso; é desnecessário dizer que nessa fase o sistema operacional é muito exigido.

Após várias pesquisas, chegou-se à conclusão de que seria interessante colocar mais um nível no computador: o nível da **linguagem de montagem** (*Assembly Language*), que poderíamos chamar de **linguagem de máquina**. Nessa linguagem os comandos numéricos existentes no nível de sistema operacional são traduzidos para termos mais fáceis de serem entendidos pelo ser humano, porém ainda muito pouco amigáveis.

Nível 5 – Linguagens orientadas para a resolução de problemas

As linguagens orientadas para a resolução de problemas são as mais utilizadas atualmente. Você já deve ter ouvido falar em Visual Basic, Delphi, C++, Java e Pascal. Existem, além dessas, dezenas de outras linguagens de programação no mercado, cada uma com suas peculiaridades e seus comandos próprios, voltados para situações específicas de trabalho em que uma determinada linguagem pode ser mais útil do que as outras. As pessoas que trabalham com essas linguagens são chamadas de programadores de aplicação, ou simplesmente programadores.

Nível 6 – Aplicações do Usuário

O último nível das nossas máquinas virtuais corresponde às aplicações geradas pelos programadores e utilizadas pelos usuários dos computadores, que nem precisam ter conhecimentos profundos de informática para tanto.

Exemplos de programas nesse nível são os utilitários, compactadores e descompactadores, processadores de texto, planilhas, editores gráficos, jogos, enciclopédias eletrônicas e programas destinados ao ensino de assuntos específicos, como idiomas, matemática, biologia, etc.

Outros programas também muito utilizados são aqueles desenvolvidos dentro de uma empresa com o objetivo de controlar suas atividades principais - por exemplo, sistemas de folha de pagamento, controles de estoque ou de vídeo locadoras, automação de caixas de supermercado, mala direta, etc.

Outro tipo de aplicação que encontramos atualmente e que não exige vastos conhecimentos de informática é o uso da Internet. A linguagem dos programas de acesso à rede tornou-se bastante amigável, e cada vez mais pessoas que nem imaginavam um dia usar um computador sentem-se à vontade, agora, utilizando-o a fim de acessar a Internet.

Programa Compilado ou Interpretado

Um programa escrito numa linguagem de programação precisa ser traduzido para o formato binário, executável pelo computador. O processo de tradução pode ser realizado por dois tipos de tradutores: os **compiladores** ou os **interpretadores**. Ambos têm a mesma função, mas diferem quanto ao modo de realizar o processo de tradução.

Compiladores e Interpretadores são programas especiais que lêem o programa escrito pelo programador em linguagem simbólica de alto nível, chamado de **programa-fonte**, e o traduzem para o formato binário, chamado de **programa-objeto**.

O compilador gera um programa final e executável pelo computador chamado de **programa-executável**.

Num programa interpretado, entretanto, cada instrução em linguagem de alto nível é traduzida em instruções binárias apenas no momento da execução do programa, dessa forma evitando que a cada alteração do programa-fonte necessitemos compilar tudo de novo.

O processo de linkagem é basicamente o último processo de geração de um programa executável, realizado pelo linkeditor, cuja função é combinar os programas-objeto e as bibliotecas de funções e comandos da linguagem utilizada, com rotinas em linguagem de máquina e outros utilitários, de forma a gerar um programa executável em nível de SO.

As duas técnicas apresentam vantagens e desvantagens. Muitos preferem os programas compilados, porque eles são mais rápidos (já que a tradução das instruções é feita previamente). Por outro lado, nos programas interpretados, apesar de serem mais lentos, tem-se maior controle sobre a operação da máquina.

Vantagens e Desvantagens

1. Como um compilador gera programas na linguagem do computador, ele precisa conhecer as instruções desse mesmo computador. Devido a isso, se quisermos criar programas para um computador IBM temos que adquirir compiladores para esse tipo de máquina; se desenvolvermos programas para a linha Apple, o compilador tem que ser feito para esse tipo de micro. O mesmo acontece com as outras linhas de computador (por exemplo, computadores RISC).
2. O programa gerado pelo compilador, isto é, o programa-objeto, executado diretamente pelo processador para o qual ele foi feito. Conseqüentemente, de posse do programa-objeto executaremos o programa em qualquer máquina para a qual ele tenha sido compilado (Alpha, PowerPC, Pentium). Já na interpretação, o programa interpretador precisa estar instalado na máquina em que ele executará o programa do usuário, porque o interpretador não cria nenhum programa-objeto.
3. Como o compilador precisa traduzir instruções de alto nível (de difícil entendimento para o computador) para baixo nível (a preferida do computador) e a tradução de uma instrução

desse tipo requer várias instruções de baixo nível, o código gerado pelo compilador (código-objeto) é grande, comparado ao código que o interpretador utiliza (código-fonte).

4. Lembre-se que o interpretador precisa ler, traduzir e executar o código criado pelo programador a cada vez que o programa precisar rodar, o que causa um retardo na execução do programa, já que os comandos têm que ser lidos e interpretados a cada execução, o que não acontece com o compilador, cujo trabalho de ler e traduzir o programa-fonte ocorre somente na hora da geração do código-objeto. Um programa compilado é sempre mais rápido do que um interpretado.
5. Uma linguagem interpretada torna mais fácil a alteração dos programas-fonte, porque eles sempre estarão disponíveis na máquina que o executará, diferentemente de uma linguagem compilada, na qual quaisquer alterações que devam ser feitas serão seguidas da geração do código-objeto, para poder executar os programas.
6. Comercialmente, as linguagens interpretadas são pouco utilizadas, porque os programas-fonte relativos a um sistema ou a uma aplicação teriam que ser instalados no computador do cliente. Nesse caso, o cliente, de posse dos programas-fonte, poderia repassar a terceiros esses programas, os quais seriam personalizados e utilizados em outras situações sem fazer o devido pagamento a quem de fato criou os programas. Imagine uma escola que tenha contratado um analista e um programador para a informatização de suas unidades, e esses profissionais decidissem usar linguagens interpretadas. Se o dono do colégio decidisse “fornecer” os programas-fonte a algum outro dono de escola, os programas seriam personalizados e instalados em outros colégios, que não precisariam contratar os serviços dos profissionais e, principalmente, não os pagariam.

Sistemas Operacionais

Quando você liga o micro, entra em ação um programa conhecido como sistema operacional. Sabemos que o Sistema Operacional é o programa que administra o funcionamento de todos os componentes de *hardware* (equipamentos) e gerencia o trabalho dos *softwares* (programas), sem um sistema operacional, o computador não passa de um conjunto de peças com função alguma.

Quando você utiliza um programa qualquer, é o sistema operacional que verifica se o equipamento está disponível, caso esteja, o sistema envia informações para ele. Isso ocorre, por exemplo, na impressão de um documento que se acaba de digitar. Acionado o comando para imprimir, o sistema operacional verifica se a impressora está ligada, se tem papel, se está ocupada imprimindo outro documento, etc. Somente após essas verificações no equipamento, ele envia os dados do documento para a impressora.

Existem vários sistemas operacionais, produzidos por diferentes empresas; os mais famosos são: MS-DOS, Unix, Linux, Windows (3.11, NT, 2000, 3.11, 95, 98, Me, XP), MacOS, BeOS, etc.

O sistema operacional mais popular, atualmente, é o Windows, por ser um dos primeiros sistemas com interface gráfica. Isto é, que utiliza telas com botões, barras e menus acionados facilmente por meio de um *mouse*, evitando a digitação dos comandos, necessária nos sistemas operacionais que não possuem essa interface.

Existem versões de Windows para diferentes utilizações. O Windows 98 é o sistema operacional preferido para uso doméstico e de pequenas empresas. Já o Windows 2000 é o sistema operacional utilizado pelas médias e grandes empresas, devido à sua maior segurança e capacidade para trabalhar em grandes redes (rede é a conexão entre vários computadores). O Windows XP possui duas versões: *Home* para o usuário doméstico e *Professional* voltada para o usuário corporativo.

Atualmente, o Linux é o sistema operacional que mais cresce em números de usuários, por ser um sistema operacional mais estável e gratuito. Entretanto, sua interface ainda não é muito amigável, o que desagrada a alguns usuários.

Como dito anteriormente um sistema operacional é uma camada de software que controla os aspectos técnicos da operação de um computador. Ele livra o usuário da máquina dos detalhes técnicos de baixo nível da operação de uma máquina e freqüentemente oferece facilidades.

Não existe nenhuma definição universal do que seja um sistema operacional, podemos defini-lo como sendo um software que está instalado na máquina antes de adicionarmos qualquer outra coisa.

Normalmente, o sistema operacional possui alguns elementos chaves:

- Uma camada de *software* para controla o *hardware* do micro, como *drives* de disco, teclado e monitor.
- Um sistema de arquivos que fornece um meio organizar os arquivos logicamente.
- Uma linguagem de comando simples que habilita os usuários a rodar seus próprios programas e manipular seus arquivos de maneira simples.

Os sistemas operacionais podem ser classificados pelo número de tarefas que eles podem realizar simultaneamente e por quantos usuários podem usá-los simultaneamente, ou seja, monousuários ou multiusuários e monotarefas ou multitarefas. Na tabela abaixo, temos alguns sistemas operacionais classificados segundo esse critério.

Sistema Operacional	Usuários	Tarefas	Processadores
MS/PC-DOS	Monousuário	Mono-tarefa	Mono-processado
Windows 3.x	Monousuário	Quase multitarefa	Mono-processado
Macintosh System 7	Monousuário	Quase multitarefa	Mono-processado
Windows 9x	Monousuário	Multitarefa	Mono-processado
Amiga DOS	Monousuário	Multitarefa	Mono-processado
BeOS	Monousuário	Multitarefa	Multi-processado
VMS	Multiusuário	Multitarefa	Mono-processado
Unix	Multiusuário	Multitarefa	Multi-processado
NT	Multiusuário	Multitarefa	Multi-processado
Windows 2000	Multiusuário	Multitarefa	Multi-processado
Windows XP	Multiusuário	Multitarefa	Multi-processado
Linux	Multiusuário	Multitarefa	Multi-processado

Os primeiros sistemas (MS-DOS e Windows 3.1) são monousuários, monotarefas e baseados numa biblioteca contida em uma ROM de funções básicas chamada BIOS (sistema básico de entrada e saída). São essas funções que são responsáveis pela exibição de caracteres na tela, gravações de dados no HD, etc.

Embora todos os sistemas operacionais possam fornecer interrupções e, portanto, simular a aparência de multitarefas em algumas situações, os computadores equipados com esses sistemas operacionais não podem ser pensados como um sistema multitarefa em qualquer sentido.

O Windows 95 trocou o antigo enfoque de co-rotina de quase multitarefa por um contexto verdadeiro de troca, mas somente para um único usuário, sem proteção de memória. Ou seja, se um programa travasse, poderia tornar instável todo o sistema. Dessa forma, apesar da afirmação de que alguns Windows da linha 9x (98, 98SE) são multitarefas, a probabilidade da queda de uma aplicação implicar na queda de todo o sistema é grande.

O sistema MacOS 7 pode ser considerado como monousuário e quase multitarefa, isto significa que é possível rodar várias aplicações simultaneamente – um gerenciador de janelas pode simular a aparência de vários programas rodando simultaneamente, mas cada programa obedecendo a regras específicas para se conseguir a ilusão.

No Windows NT foi adicionado um *kernel* próprio com proteção de memória, baseado no sistema VMS, originalmente escrito para DEC/Vax. Versões recentes do Windows NT e Windows 2000 (uma versão aperfeiçoada do NT com proteção de memória) permitem múltiplos *logins* (sessões de trabalho distintas) também através de um terminal servidor.

O Unix é provavelmente o mais antigo sistema operacional existente hoje. Ele está disponível em várias implementações. Originalmente projetado na AT&T (1969), o Unix desmembrou-se em dois ramos: o BSD (*Berkeley Software Distribution*) o System 5 (AT&T). A versão da BSD foi desenvolvida como um projeto de pesquisa na Universidade de Berkeley, Califórnia. Muitos recursos de transmissão de dados via rede e facilidades para o usuário surgiram destas modificações. Para evitar que as diversas versões do Unix se tornassem incompatíveis entre si, formou-se um comitê de padronização chamado POSIX, composto pela maioria dos desenvolvedores.

Anos depois do surgimento do Unix, um estudante finlandês estava trabalhando com o Minix, um pequeno sistema operacional clone do Unix, criado por Andrew Tannenbaum para fins educacionais. Ele pretendia alterar o código fonte do Minix para criar um “Minix melhor do que o Minix”. Em 1991, após algum tempo de trabalho, Linus Torvalds, publicou, num grupo de discussão (*comp.os.minix*) uma mensagem informando a versão 0.02 do seu sistema: Linux.

Por ser baseado no Unix, o Linux, desde o seu surgimento apresenta diversas características do seu antepassado: multitarefa, multiusuário, multiprocessamento, grande estabilidade, robustez e eficiência. Ele foi distribuído como um sistema operacional *open-source* (código aberto, todos têm acesso ao código-fonte).

Na verdade, o que Linus desenvolveu foi o *kernel* do Linux, o conjunto de rotinas e programas que permitem o funcionamento do sistema operacional. A partir daí, diversas outras pessoas colaboraram com o projeto desenvolvendo (ou adaptando) *drivers*, arquivos de configuração e de ajuda e interfaces. Para que fosse mais amigável para os usuários finais, várias interfaces gráficas foram desenvolvidas para o Linux, como as interfaces WindowMaker, Gnome e KDE.



O conjunto de *kernel*, *drivers*, interfaces gráficas e arquivos de configuração e ajuda é normalmente chamado de distribuição. Enquanto o *kernel* do Linux é gratuito, as distribuições podem ser cobradas, desde que seja possível obtê-las gratuitamente.

Por ter código aberto, o Linux vem ganhando adeptos rapidamente e toda uma cultura vem se desenvolvendo ao seu redor. É cada vez mais comum vermos o desenho de Tux, o simpático pingüim que se tornou a sua mascote.



BIBLIOGRAFIA

ALMEIDA, Marcus. **Fundamentos de Informática**. Rio de Janeiro: Brasport, 2002.

CORNACHIONE Jr., E. B. **Informática Aplicada às Áreas de Contabilidade, Administração e Economia**. São Paulo: Atlas, 1998.

GEHRINGER, Max; LONDON, Jack. **Odisséia Digital**. São Paulo: Abril, 2001.

GIL, Antonio de Loureiro. **Sistemas de Informações Contábil/Financeiros**. São Paulo: Atlas, 1999.

GONICK, Larry. **Introdução Ilustrada à Computação**. São Paulo: Harbra, 1986.

KANAAN, João Carlos. **Informática Global**. São Paulo: Pioneira, 1998.

SOUZA, Hudson C. S. Apostila de Informática I, 2003.