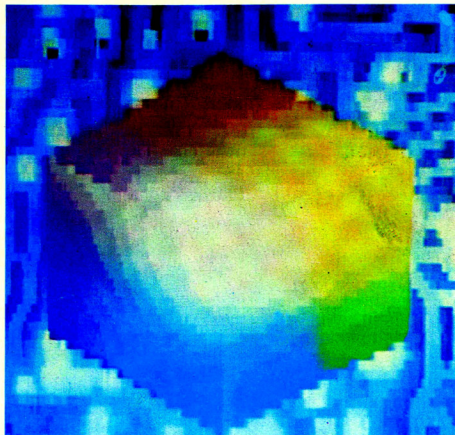


BIBLIOTECA BÁSICA

INFORMÁTICA

**SISTEMAS OPERACIONAIS
E SOFTWARE DE BASE**

10



BIBLIOTECA BÁSICA

INFORMÁTICA

**SISTEMAS OPERACIONAIS
E SOFTWARE DE BASE**

10

Diretor editor:

M.A.Nieto

Coordenação e supervisão técnica:

Eng.º Sergio Rocha Paggioli

Tradução:

Ideli Novo

Projeto:

Rainer K.E. Ladewig

Diretor de arte:

Duilio Sarto F.º

Studio editorial:

Auro Pereira da Silva (chefe), Susana

M.Amaral Couto (revisão), Luiz Carlos

Siqueira Lago (prod. gráfica), Antonio

Carlos Martins, Rubens Tadeu Benedito

Fotocomposição, fotolito:

Omnicolor Gráfica e Propaganda Ltda - Rua Dr. Virgílio de Carvalho Pinto, 619
Pinheiros - CEP 05415 - São Paulo

Impressão

Editora Antártica S.A. - Av. Ramon Freire, 6920 (Pajaritos) - Santiago - Chile

© Antonio M. Ferrer Abello

© Edições Ingelek S.A.

© 1986 para a língua portuguesa Ed. Século Futuro Ltda. - Rua Belisário Pena, 821
Penha - R.J. Fone: 290-6273 - CEP 21020

A editora Século Futuro mantém todos os direitos reservados sobre esta publicação. Fica proibido assim, sua reprodução total ou parcial por qualquer sistema sem prévia autorização do Editor.

ÍNDICE

PREFÁCIO

5 Prefácio

CAPÍTULO I

9 Sistemas operacionais: nascimento, história e modelos

CAPÍTULO II

33 A memória central: uma organização complexa

CAPÍTULO III

51 Sistemas de armazenamento de dados: ordem e integridade

CAPÍTULO IV

75 Software de base: mais além do sistema operacional

PREFÁCIO

Antes de decidir-nos a escrever sobre os sistemas operacionais para computadores domésticos e pessoais de maneira que fosse compreensível para pessoas não experts, pensamos bastante, já que o tema é, sem dúvida alguma, um dos mais complexos no campo da Informática, tratado unicamente em faculdades de Informática e Engenharia em cursos altamente especializados. Por outro lado, a grande difusão dos computadores obrigou as indústrias a criar alguns sistemas operacionais que se adaptassem às características de suas máquinas e às exigências de alguns usuários que não são profissionais de grandes centros de cálculos, mas estudantes e aficionados, comerciantes e pessoas de todo tipo. Assim as características dos sistemas operacionais mudaram em relação àquelas outras dos grandes equipamentos. Foi sendo difundida a idéia de que estes aparelhos tão sofisticados necessitam ser dia a dia mais versáteis, isto é, fáceis de operar por parte de um número cada vez maior de pessoas.

De todas as maneiras, iremos ver ao longo deste livro como os sistemas operacionais dos computadores pessoais repetem no tempo o desenvolvimento experimentado por seus irmãos maiores. De fato, dado que os progressos tecnológicos referentes aos circuitos integrados estão reduzindo cada vez mais a diferença entre "micro" e "mini" computador, assim os sistemas operacionais assumem no campo do computador pessoal os êxitos obtidos no grandes sistemas, transvasando aos "pequenos" muitas de tais ex-

periências. Neste aspecto aparece o Unix, sistema operacional que em aparência começa a predominar, especialmente na classe dos microcomputadores profissionais. Este e outros fatos induzem a considerar não utópica a suposição que, em um futuro não muito distante, as diferenças entre ambas as categorias serão reduzidas até desaparecer.

Este livro, repetimos novamente, não está destinado aos estudantes universitários da Informática, mas àquelas pessoas que desejam aproximar-se a este tema sem serem obrigadas a ler toneladas de livros em inglês, e sem renunciar ao mesmo tempo a um tratamento rigoroso.

A propósito do inglês, este tem sido outro dos problemas que estabelecemos, já que a linguagem utilizada em Informática está repleta de vocabulários americanos, freqüentemente impossíveis de traduzir por tratar-se de neologismos, inclusive em seu próprio idioma de origem. Assistimos assim a difusão de uma estranha língua com palavras do tipo "digitalizar" (de digit). Por outro lado, está claro que o idioma normalmente utilizado em Informática é o inglês, e não sabemos até que ponto seria interessante traduzir cada palavra. O único país do mundo onde tentaram fazer uma operação deste tipo (indubitavelmente de muito mérito), foi a França, porém com resultados não tão satisfatórios como os esperados. De fato, tornou-se uma linguagem pouco prática, já que aquelas pessoas interessadas no campo dos computadores foram obrigadas a estudar dois idiomas: o inglês técnico, para poder seguir os desenvolvimentos e avanços no tema que os interessa, sempre em publicações em inglês, e o "francês-informatizado" para entender-se com os compatriotas.

Nós temos tentado seguir uma via intermediária, utilizando aquelas palavras em português que já estão integradas dentro do vocabulário comum, e os vocábulos ingleses quando for impossível ou fora de lugar a tradução (por exemplo, como bit, byte, etc.).

Temos traduzido alguns termos quando são indistintamente utilizados em inglês e em português.

Vamos tratar agora a questão da organização do livro, cuja matéria, como já dissemos, é ampla e complexa. Nossa intensão não é a de apresentar os problemas de Dijkstra e Brinch Hansen (importantes "cérebros" na matéria) sobre processamentos concorrentes e multiprogramação, mas facilitar o trabalho daquelas pessoas que se aproximam pela primeira vez a estes temas no mundinho dos computadores pessoais.

No primeiro capítulo veremos qual foi a história dos computadores e dos sistemas operacionais desde seu nascimento até ho-

je, para, uma vez definidas as características que estes assumiram nos computadores pessoais, examinar os aspectos fundamentais dos sistemas operacionais mais difundidos.

O segundo capítulo apresenta aquilo que hoje em dia é um dos aspectos de maior interesse: a questão da memória, fazendo também referência à organização adotada pelo IBM-PC.

Na terceira parte serão tratados os dispositivos para o armazenamento de grandes quantidades de dados: fitas e discos. Examinamos dos mesmos suas características quanto ao desenho, e os aspectos de gestão que afetam o interesse ao sistema operacional.

No último capítulo nos referimos àqueles programas que, apesar de não fazerem parte do sistema operacional, constituem, digamos assim, sua natural expansão formando com ele o chamado "software de base", parte essencial no funcionamento e uso eficaz do computador. Estamos falando dos Editores, Assembleadores, Intérpretes, Compiladores e outros que cruzam a fronteira, entrando na área propriamente de aplicações...

Aquelas pessoas que, uma vez concluída a leitura, queiram saber algo mais desta matéria, aconselhamos consultar à bibliografia que incluímos ou que se dirijam a alguma biblioteca técnica.

CAPÍTULO I

SISTEMAS OPERACIONAIS: NASCIMENTO, HISTÓRIA E MODELOS



o sistema operacional, é um instrumento indispensável para fazer do computador um objeto vivo e, sobretudo, útil para nós. De fato, sob este nome são agrupados todos aqueles programas que permitem aos usuários a utilização desse embrulho de cabos e circuitos, que de outra forma seriam difíceis de controlar. Formalmente, um sistema operacional é definido com um conjunto de procedimentos, manuais e automáticos, que permitem a um grupo de usuários compartilhar uma instalação de computador eficaz.

Se for uma instalação particular que não vai ser dividida, utilizada somente por um único usuário é ampliada a definição anterior dizendo que os sistemas operacionais possibilitam ao usuário a realização de determinadas operações (por exemplo, as de entrada/saída) de forma cômoda e eficaz.

Poucas pessoas sabem que, na origem da história dos computadores, isto é, há uns quarenta anos, os sistemas operacionais não existiam; assim, a introdução de um programa para ser executado se convertia em um incrível esforço que somente podia ser levado a cabo por muito poucos experts. O tempo requerido para colocar um programa naqueles mastodontes de escasso cérebro superava em muito o de execução, de modo que era pouco proveitosa a utilização dos computadores para a resolução de problemas práticos.

Como é fácil supor, o nascimento dos sistemas operacionais

e seu posterior desenvolvimento foram decisivos para conseguir o nível de difusão, cada vez maior, dos computadores. A tendência mais generalizada em nossos dias é a de sistemas operacionais “amigáveis” (“friendly”), isto é, orientados para uma compreensão e manejo imediatos por parte do usuário, sem necessidade de grandes esforços mentais.

Acreditamos que a melhor forma de aproximação a este tema é a de conhecer primeiro algo de sua história em relação com o desenvolvimento dos computadores, desde os anos quarenta até nossos dias. De fato, existe uma clara tendência por parte dos computadores pessoais de seguir os passos de seus irmãos maiores até nos mínimos detalhes. Assim os sistemas operacionais dos computadores pessoais percorreram até o dia de hoje aproximadamente o mesmo caminho que os grandes computadores entre os anos quarenta e sessenta. A evolução destes em anos sucessivos nos dirá, portanto, com muita probabilidade, qual será o desenvolvimento que será verificado no campo da microinformática nos próximos anos.

Um pouco de história

A evolução dos sistemas operacionais segue muito de perto ao dos computadores. Estes foram desenvolvendo-se segundo o avanço da tecnologia, desde os primeiros processadores a válvulas até os atuais, baseados em circuitos em grandíssima escala de integração (VLSI, Very Large Scale of Integration). A história dos sistemas operacionais está catalogada por gerações (aproximadamente as mesmas dos computadores): desde a geração zero, nos anos quarenta, quando existiam os processadores, mas ainda nos sistemas operacionais dignos deste nome, à quarta geração, mais sofisticada, utilizada hoje em dia nos sistemas mais avançados.

Antes da década dos cinquenta os sistemas operacionais estavam ancorados na pré-história: os processadores estavam constituídos por enormes caixas de válvulas que apresentavam em seus painéis uma série de interruptores. Por meio destes o operador introduzia no computador uma série de “zeros” e “uns”; iniciava o processamento e examinava, por meio de pequenas lâmpadas, o conteúdo da memória para obter os resultados. Como é fácil de supor, se tratava de um sistema longo, complexo e que fazia muito dificultoso o manejo da máquina.

No começo dos anos cinquenta surgiram os primeiros, rudimentares mas já úteis, sistemas operacionais. Seu objetivo era reduzir os tempos mortos necessários entre a execução de um pro-

gra e outro, já que até aquele momento a maior parte do tempo era perdido nas fases intermediárias, durante as quais o programa era introduzido no sistema, e ao final do processamento, quando havia que extrair os resultados. O programa era “carregado” então por meio de um leitor de cartões, ocupando já nesta fase a memória completa. Ao final do processamento o operador devia carregar outro programa para proceder ao exame de memória, remover fitas, cartões e dados impressos. Somente nesse momento o computador podia aceitar outro programa, repetindo desde o começo as diferentes fases (Fig.1).

Para reduzir estes tempos mortos foram criados os sistemas operacionais por lotes (ou em “batch”), nos quais os programas eram tratados por grupos (lotes) em vez de individualidade. Assim surgem os chamados BATCH, pequenos computadores auxiliares que eram encarregados de ler os cartões dos distintos programas e de transferir seu conteúdo a uma fita magnética. A função do sistema operacional consistia em carregar em memória um programa da fita e executá-lo; ao finalizar era realizado o salto a uma direção de memória desde onde reassumia o controle do sistema operacional, que carregava o programa seguinte e o executava. Desta maneira o tempo entre um tabalho e o outro era nitidamente inferior (Fig.2).

Logo foi imposto outro conceito fundamental, que continua sendo válido em nossos dias: a definição dos dispositivos de entrada e saída por meio de nomes lógicos. Assim já não era necessário conhecer com todo detalhe estes subprocessamentos, mas que genericamente podia ser feita referência a um “leitor de cartões” para a entrada ou a uma “impressora” para a saída. Agora era o sistema operacional que se preocupava de fazer a função de “Interface” entre os programas e o exterior; graças aos IOCS (Input/Output Control System), grupos de programas para gestionar os peritéricos em todos seus detalhes. Hoje em dia este tipo de programas tem sido desenvolvido e aperfeiçoado, permitindo ao usuá-

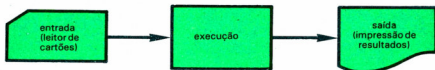


Figura 1 — Processo que permitiam os primeiros sistemas operacionais.

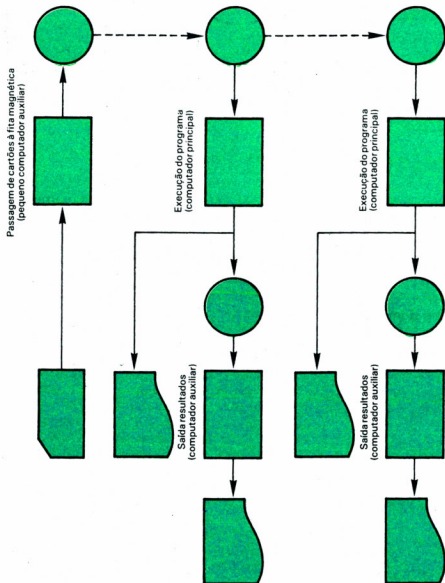


Figura 2 — Sistema operacional por lotes (em batch).

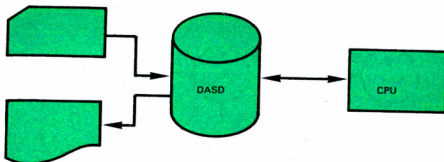


Figura 3 — Situação dos DASD no sistema.

rio toda uma série de operações sofisticadas, independentemente de como as realizem os periféricos. Surgem os DASD dispositivos de armazenamento de acesso direto (Direct Access Storage Devices) (Fig.3).

Na década dos sessenta, com a segunda geração, assistimos ao nascimento dos sistemas de tempo repartido: se trata do início da multiprogramação e do sistema multiprocessador. Vejamos o que significam estes conceitos que, se bem que levam o mesmo resultado externo, são baseados sobre pilares completamente diferentes.

A sincronização entre o processador central e os dispositivos periféricos foi resolvido mediante as interrupções. Uma interrupção é um sinal de sincronização enviado por um dispositivo periférico a um registro acessível para o processador central.

Este registro é examinado pelo processador central depois da execução de cada instrução. Quando seu conteúdo assinala uma interrupção, o processador central suspende a execução do programa em curso e dá início a de outro ("atende a interrupção"). Esta técnica torna possível o funcionamento concorrente de um processador central e seus dispositivos periféricos. A técnica de programação usada para manejar o funcionamento concorrente do computador é denominada MULTIPROGRAMAÇÃO.

Com a multiprogramação são encontrados presentes na memória do computador vários programas de uma vez. O processador, único, comuta seu funcionamento - sua atenção - continuamente de um programa a outro. Ao ser os tempos de comutação muito pequenos em relação com os de reação dos seres humanos, os usuários não percebem uma diferença sensível no funcionamento.

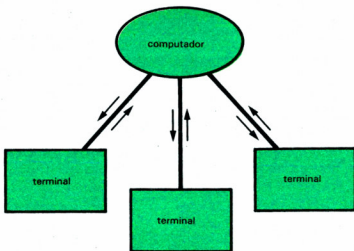


Figura 4 — Com os sistemas operacionais de timesharing o computador observa todos os terminais. Se algum envia uma pergunta, a processa e devolve os resultados ao mesmo terminal, retornando à situação de “escuta”.

O procedimento de tempo repartido, de todos os modos, torna **mais lento os préstimos**, ainda que é muito difícil que o usuário se dê conta disto.

Nos sistemas multiprocessadores, ao contrário, são encontrados vários processadores simultaneamente no mesmo computador: desta forma os **préstimos ficam aumentados extraordinariamente**.

No mesmo período vem à luz outro conceito fundamental da filosofia multiusuário: **a divisão de tempo**, mais conhecida como “timesharing”. Por fim os usuários podiam estabelecer um diálogo com o computador utilizando um terminal dotado de um teclado, por meio do qual era possível enviar mensagens, e outro periférico por onde receber as respostas. **Assim foi possível um funcionamento interativo**, no qual o usuário fazia uma petição, o computador a aceitava, a executava e enviava a resposta ao terminal do usuário (Fig.4).

Este funcionamento permitiu desenvolvimentos formidáveis no campo da Informática. Basta pensar no processamento de elaboração e verificação de um programa. Antes do advento destes

sistemas operacionais o usuário devia perfurar um bom número de cartões (um para cada instrução), introduzi-las em seguida no leitor de cartões e, quando chegava o momento (após algumas horas), retirava uma listagem onde estavam assinalados os erros. Assim, antes de ter uma versão definitiva do programa e conseguir que funcionasse transcorria muito tempo. Com os sistemas operacionais Timesharing, ao contrário, cada usuário podia escrever, compilar e executar os programas de uma maneira muito mais cômoda e rápida.

Mas, como sempre, também está a outra face da moeda: os novos sistemas operacionais, para poder oferecer estes serviços tão úteis, além de ter custado grandes esforços sua implementa-

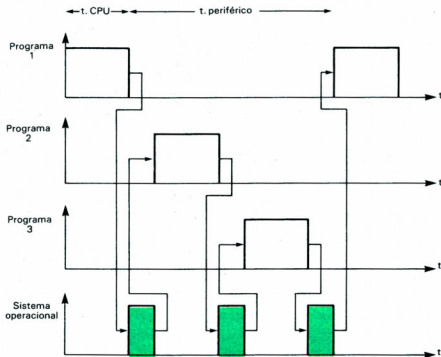


Figura 5 — Sucessão temporal de três programas em multiprogramação. Durante o tempo de atividade do periférico podem trabalhar sucessivamente (em "overlapping") outros sistemas: a linha descontinua indica o tempo de ação do sistema operacional, necessário para salvar os parâmetros de um programa, antes de passar a outro.

ção, requeriam muitíssima memória para poder funcionar. De fato, como é fácil compreender, a multiprogramação requer uma grande quantidade de funções, todo um software que consome muita memória.

Esta técnica nasceu ao comprovar-se que a unidade central do computador permanecia inativa durante a fase de entrada/saída de dados, ao ser esta operação muito lenta em comparação com a velocidade do processador central. Se pensou então que durante estas atividades (que podiam ser realizadas com dispositivos autocontrolados situados nos periféricos) a atenção do computador podia ser desviada para outro programa para economizar tempo, renovando a execução do programa original ao término de sua fase de entrada/saída (Fig. 5). Naturalmente, pode acontecer que o 2.º programa inicie também uma atividade de entrada/saída, de modo que o sistema operacional deve ser encarregado então de ceder o controle a um terceiro programa. Isto pode dar lugar a uma verdadeira cadeia de atividades de cálculo e do Input/Output, baseada na economia do tempo.

A operação de comutação da unidade central de um programa a outro é muito delicada, já que deve conseguir salvar toda a informação em posse do processador, incluindo o conteúdo dos registros e as direções em que está trabalhando, de forma que possa ser recuperado depois no momento de restaurar as condições de funcionamento.

Está claro que quando o número de programas “simultaneamente” em processamento (isto é, em atividade comutada) se faz considerável, a quantidade de trabalho que o sistema operacional deve efetuar para salvar e restabelecer os contextos das distintas atividades adquire grandes proporções. Se trata de um limite para este tipo de organização: de fato, é necessário não superar nunca uma certa carga para o sistema, pois de outra forma perde completamente sua eficácia, chegando a uma situação na qual todo o tempo-máquina é consumido em executar operações de mudança de contexto.

IBM 360 e Unix

Entre os anos sessenta e setenta foram realizados os maiores esforços na melhoria dos sistemas operacionais. Os produtos mais importantes são o Sistema Operacional da família IBM 360 e o Unix. Totalmente diferentes quanto a delineamentos e objetivos: o primeiro dominou o palco dos grandes computadores até há pouco tempo, e o segundo está conseguindo-o agora.

A terceira geração de computadores tinha como fim o de estar destinada a um campo de aplicações o mais extenso possível. A firma IBM pensava realizar, com a família 360, um sistema capaz de satisfazer as exigências mais dispares do usuário. **Se queria oferecer um computador que o usuário pudesse utilizar durante um grande número de anos, ainda com exigências diferentes das originais, sem ver-se obrigado a mudar por isto a estrutura completa.**

Esta forma de pensar foi muito importante na história dos computadores e condicionou sua evolução até nossos dias. Além disso, para convencer aos compradores em posse de outros computadores a passar aos IBM, o sistema operacional do 360 era capaz, por meio de simuladores e emuladores, de executar diretamente programas realizados em outros computadores.

O nascimento deste sistema foi como uma bomba no mercado internacional. As firmas competidoras tiveram que tomar prontas medidas, cada uma de um modo diferente. A RCA, como outras, pretendeu copiar o sistema operacional de 360 para vendê-lo a um preço inferior. Esta tentativa foi um fracasso tão clamoroso que provocou a retirada da RCA do mundo da Informática com um déficit superior aos 500 milhões de dólares.

Outras companhias, como a General Electric e a Burroughs, trataram de realizar um sistema operacional não compatível com o 360, porém mais eficiente e potente. Este projeto também acabou sendo demasiadamente ambicioso, pelo que a General Electric teve, por sua vez, que abandonar o mundo dos computadores.

Firmas como Control Data, Univac e Honeywell, ao contrário, se limitaram a desenvolver sistemas mais simples e similares aos mais antigos da IBM, chegando a convencer àqueles usuários que não se decidiam a passar à categoria dos 360.

Por outro lado, as esperanças da IBM de realizar um único sistema operacional para toda a série 360 ficaram também frustrados devido a grande quantidade de necessidades diferentes dos usuários. Foram assim realizados quatro sistemas operacionais, de acordo com as distintas dimensões dos computadores:

- DOS/360 para os menores,
- OS/MFT e OS/MVT para médios/grandes,
- CP-67/CMS para os mais potentes.

Possuíam como característica comum a de oferecer aos usuários um número notável de programas de suporte que permitiam uma utilização mais simples e proveitosa do computador.

Poucos anos depois do nascimento da série 360 saía à luz um sistema operacional baseado em uma filosofia totalmente diferente: O Unix. Concebido por um grupo de pesquisadores da Bell Laboratory que haviam participado no estudo de um sistema operacional destinado a competir com o 360, o Multics, o Unix foi projetado com o fim de oferecer um sistema operacional econômico no qual a gestão de programas em disco fosse independente do dispositivo. O Unix opunha assim simplicidade e eficácia à complexidade e enormidade do 360 IBM.

Desenvolvido em um ambiente universitário, possui características muito mais avançadas com respeito às da maior parte dos sistemas operacionais. Sua vantagem mais interessante sob um ponto de vista industrial é a de ter sido desenhado de forma que possa ser adaptável às mais diversas arquiteturas, as quais somente exige que estejam dotadas aos menos de 256 Kbytes de memória e de um microprocessador de 16 ou mais bits. Este fato permite supor que o Unix será um dos sistemas operacionais mais difundidos a nível de minis e médios computadores, além de ser o candidato com maiores possibilidades para a próxima geração de computadores pessoais.

A atual geração de sistemas operacionais está claramente influenciada pelo fenômeno mais importante dos últimos anos: a difusão dos microprocessadores. Graças a estes dispositivos, aqueles computadores enormes que até poucos anos custavam milhões, hoje em dia podem ser fabricados por poucas dezenas de milhares.

Inclusive a CPU do IBM 360 foi reduzida a um circuito integrado com dimensões de poucos milímetros quadrados. Deste modo, os sistemas operacionais tiveram que adaptar-se ao fato de que, com preços decididamente razoáveis, pudessem ser obtidos hoje práticos impensáveis em outro tempo. De fato, já existem disponíveis redes de computadores onde o sistema operacional deve ser encarregado da gestão atenta e inteligente de uma enorme quantidade de dispositivos (computadores, periféricos e unidades de memória).

Uma das novidades mais interessantes dos sistemas operacionais modernos é a técnica de gestão de Memória Virtual, graças à qual o usuário pode utilizar programas de dimensões bastante superiores à quantidade de memória disponível. Este "milagre" é devido a que, a seu devido tempo, são carregadas no computador as "páginas" de software e de dados necessários nesse instante, tomadas das memórias de massa externas. De tudo isto é encarregado o sistema operacional, sem que o usuário deva possuir ne-

nhum tipo de noção acerca de como está constituído internamente o computador. Isto é, por fim está sendo produzido aquele que era o propósito dos estudiosos de sistemas operacionais desde suas origens: conseguir que o usuário não tenha porque ser um expert de eletrônica ou informática, mas tão somente o que seu nome indica, um usuário. Desta maneira, os modernos computadores são apresentados com uma série de "menús" (telas com possíveis opções) que permitem escolher as funções que deseja executar, e será o serviço sistema operacional quem se preocupará de manejar todos os dispositivos de hard e soft que a operação requiera.

Os sistemas operacionais dos computadores pessoais

Graças aos avanços conseguidos com o desenvolvimento da tecnologia e a lei do mercado, os sistemas operacionais dos computadores pessoais estão colocando-se rapidamente ao nível de seus irmãos maiores. Desde as primeiras placas como de jogo, onde uma série de interruptores acendiam alguns LEDs vermelhos e verdes, passamos a placas de microprocessadores muito mais elaboradas.

Até há pouco, nenhum computador pessoal estava dotado de sistema operacional, mas somente de um "Monitor". E não se trata da tela de visualização, mas de um programa, escrito a propósito para cada placa de microprocessador, que é encontrado armazenado em uma EPROM (Erasable Programmable Read Only Memory, memória de somente leitura programável e apagável) e **que cumpre somente algumas operações, das mais elementares, de um sistema operacional**, permitindo a utilização de um terminal de vídeo, um teclado, uma impressora, um cassete de fitas magnéticas e, às vezes, uma unidade de disco. De fato, transportar um sistema operacional desde um grande computador ficava demasiadamente oneroso para os primeiros microprocessadores, já que os sistemas operacionais requerem um grande quantidade de memória e, ao menos, uma unidade de disco, elementos estes que comportam um aumento considerável no preço do equipamento. O problema foi inicialmente resolvido por meio das memórias EPROM, já que estes circuitos integrados permitem complementar e conservar permanentemente em memória um certo número de programas de utilidade a preços razoáveis.

As funções destes "monitores" eram limitadas somente às essenciais para o funcionamento do computador, isto é: escrita e leitura de memória e registros em forma hexadecimal, possibilidade de fazer executar e parar os programas, manejo de tela, tecla-

do e impressora, e gravação e leitura de dados e programas a partir de fitas de cassete ou discos flexíveis.

Nos encontramos, portanto, na primeira geração dos sistemas operacionais. Um primeiro passo importante foi a realização de intérpretes da linguagem BASIC, por meios dos quais podem ser transmitidos comandos ao computador. Deste modo, o usuário não deve preocupar-se de gerenciar pessoalmente os canais de comunicação, programando as portas de entrada/saída registro por registro. É suficiente uma instrução "PRINT", por exemplo, para que o intérprete BASIC se ocupe de todas as operações de controle necessárias.

Mas a verdadeira revolução foi produzida quando, com o desenvolvimento tecnológico, foi possível comercializar unidades de disco e memórias a preços competitivos. Assim foram criados os primeiros sistemas operacionais desenhados especificamente para os computadores pessoais. Hoje em dia, por fim, o usuário dispõe de meios muito eficazes para aproveitar ao máximo, e com o grau de comodidade que deseja seu pequeno computador.

No início, como é lógico, cada firma construtora tratava de realizar um sistema operacional próprio, de tal modo que fosse incompatível com qualquer outro computador. Isto fazia com que fosse impossível utilizar um software de aplicação realizado para um computador distinto. Agora então, dado o enorme número de microcomputadores presentes no mercado, isto supunha uma grande desvantagem para os compradores, já que cada aparelho podia executar unicamente os programas escritos especificamente para ele. A situação constituía também um problema para as pequenas empresas fabricantes, que não conseguiam desenvolver grandes pacotes de programas, pois era tarefa demasiadamente grande para suas estruturas. A única saída viável era a de estabelecer um standard, comum para todos os equipamentos, para a realização do software.

Na segunda metade dos anos setenta começou a ser difundido na América um sistema operacional desenhado especialmente para microprocessadores de 8 bits: o CP/M. Não é que seja precisamente o melhor sistema operacional sob muitos pontos de vista: o modo de gerenciar os discos faz mais lento o acesso, as mensagens de erros são indecifráveis e não é possível nenhuma técnica refinada em questão de gestão de arquivos. Mas possui uma grande vantagem: foi realizada de tal maneira que pode ser adaptado com facilidade a todos os computadores que utilizam um microcomputador com código Intel 8080, Z-80 ou compatíveis, sem problemas de intercâmbio de memória, periféricos e discos. Deste mo-

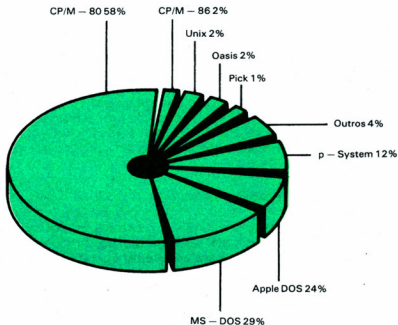


Figura 6 — Distribuição no mercado dos sistemas operacionais mais utilizados (dados de 1983).

do foi feito realidade aquilo que havia sido sempre um sonho para a indústria do computador pessoal: transportar o software de um equipamento a outro.

A Digital Research, sociedade criadora do CP/M, realizou toda uma gama de versões deste sistema operacional, que são adaptados a quase todos os computadores pessoais em circulação, apropriando-se nada menos que de 58% do mercado mundial (Fig. 6). Além disso, após comprar a licença, qualquer firma pode adotar o CP/M em seus computadores somente seguindo as instruções proporcionadas pelo fabricante.

Enquanto isso, os gigantes da indústria dos computadores pessoais não poderiam ficar de braços cruzados. Assim realizaram ou encarregaram seus próprios sistemas operacionais: o **DOS da firma Apple** e o **MS-DOS da Microsoft** (o **PC-DOS**) para **IBM**. Ambos foram criados especificamente para determinados equipamentos, obtendo um bom resultado. De todas as maneiras também adotaram, como sistema operacional adicional, o CP/M: a quantidade de software disponível para ele (sobretudo jogos) é

tal, que tanto Apple como IBM, para não perder um elevado número de possíveis compradores, tiveram que aceitá-lo.

O CP/M é um standard, de fato, para os computadores de 8 bits, também tem muita difusão o CP/M 86 para os de 16 bits. No entanto, na área dos processadores de 16 bits a empresa IBM está impondo como standard o MS-DOS: como vemos, de alguma maneira está sendo repetida, em pequena escala, a história da série 360...

Anatomia de um sistema operacional

Após ter visto o nascimento e desenvolvimento dos sistemas operacionais tanto em computadores grandes como pequenos, vejamos agora sua organização. Evidentemente, não se encontra dentro de nossos objetivos descrever neste volume o correspondente aos grandes computadores, naturalmente mais complexos.

Observando a Figura 7 pode chegar a ter certa idéia da complexidade das funções de um sistema operacional. Quando comparamos um computador pessoal nos encontramos frente a toda uma série de dispositivos de cujo interior sabemos pouco ou nada. Existe à nossa disposição um teclado, uma unidade de vídeo, uma impressora, uma unidade de discos e o computador em si, composto por sua vez de uma unidade central (ou CPU), a memória interna e vários elementos de entrada/saída.

O sistema operacional deve encarregar-se de organizar todos estes dispositivos de maneira que para nós possam continuar sendo, em seu interior, totalmente desconhecidos. Imagine a quantidade de problemas que nos seriam delineados se cada vez que necessitássemos utilizá-los tivéssemos que ocupar-nos deles! Vamos colocar, tão somente como exemplo, o funcionamento de um terminal de vídeo. Cada vez que pressionarmos uma tecla e aparece na tela o carácter correspondente, o sistema operacional desenvolve toda uma complexa série de operações elementares: dispor uma direção de entradas de dados, codificar o carácter teclado, após ter eliminado os múltiplos pressionamentos (devidos a rebatimentos mecânicos das teclas), memorizá-lo em uma área interna, denominada "buffer", convertê-lo em uma forma visualizável e enviá-lo à tela. Como pode comprovar, se trata de uma série de operações demasiadamente complicadas e longas para que estejam todas as vezes a cargo do usuário...

Mas o trabalho do sistema operacional não termina aqui. De fato, uma vez que o usuário completou o comando, este deve ser estudado: um programa do sistema, denominado "analisador sin-



tático" (em inglês "parser"), deve verificar se a sintaxe do comando é correta ou não e decidir, portanto, se deve fazer referência a sua memória interna, aos discos ou a outros dispositivos periféricos. Nos próximos capítulos veremos quais são os grandes problemas que auxiliam a utilização destes recursos.

23

trap" (de carregamento inicial) que, após ligado, toma do disco o sistema operacional, o carrega em memória e o transpassa o controle. Com esta operação é evitado elegantemente impor uma escolha definitiva do sistema operacional ao computador: bastará mudar a versão que é encontrada no disco para dotar ao computador de um sistema operacional renovado (inclusive, às vezes, distinto).

O CP/M, por exemplo, foi modificado e colocado em dia várias vezes ao longo dos anos sem que os usuários tivessem que abrir para nada o computador; tão somente é necessário mudar o disquete e deixar que o bootstrap se preocupe de carregar a nova versão.

Este método também é utilizado nos computadores maiores: no VAX 11/780, por exemplo, é possível usar o VMS ou o Unix indiferentemente, pois é suficiente fornecer um ou outro sistema no momento do booting (inicialização; é a execução do bootstrap).

Em certos computadores domésticos e portáteis, ao contrário, o sistema operacional está gravado diretamente em ROM: deste modo é evitada a necessidade do disco, mas, por contra, fica vinculado ao sistema operacional escolhido.

Uma vez carregado em memória, o sistema operacional oferece ao usuário a possibilidade de escolher entre uma variada gama de comandos e serviços. Os comandos formam, por regra geral, parte do sistema e, portanto, ficam todos carregados em RAM no momento do bootstrap. Sua execução será assim muito rápida, bastando somente realizar o salto a uma direção de memória determinada. Os comandos permitem levar o cabo operações como o exame da memória do computador, a cópia de arquivos (e de programas), o exame dos arquivos residentes em disco, ou então fornecer informação acerca do comprimento de um arquivo, da quantidade de memória disponível ou do espaço ocupado no disco.

Outros elementos mais sofisticados são os chamados "serviços" (ou utilidades - "utilities") fornecidos pelo sistema operacional, como editores, compiladores, Assembladores, editores e processadores de texto e tratamento de gráficos. Estes serviços residem normalmente em memória, dada a considerável quantidade de espaço que ocupam, mas que são transferidas unicamente quando são solicitados pelo usuário. De todas as maneiras falaremos com mais detalhe destes programas no capítulo 3.

Panorâmica dos sistemas operacionais presentes no mercado

Agora que tratamos em grandes passadas de seu funciona-

mento, vamos ver quais os sistemas operacionais mais difundidos hoje em dia no mundo dos computadores pessoais. Na Figura 6 é mostrada uma visão do mercado atual, mas deve-se ter em conta que se encontra em constante e rapidíssima evolução; vários dos que aparecem podem dispor de mais de um sistema operacional.

CP/M 80, o "bus do software"

É o mais difundido dos sistemas operacionais para equipamentos baseados em microprocessadores de 8 bits. É modular, independente do hardware e pensado para sistemas monousuário. Comercializado por Gary Kildall no início dos anos setenta com a versão 1.4 ao preço de 70 dólares, foi desenvolvido ao longo destes últimos anos com várias versões, dentre as quais a 2.2. foi a mais difundida.

Realizado por Digital Research, seu êxito firmou-se principalmente em dois fatores: sua reconfigurabilidade e a enorme quantidade de programas disponíveis no mercado. Desenvolvido originariamente para os microprocessadores Intel 8080 e 8085 possuía o código executável compatível com o Zilog-Z-80. Em versões sucessivas foi adaptado completamente para o Z 80 que, ao ser o processador mais comercializado, exerceu a missão de veículo de arraste do CP/M.

O CP/M pode ser entendido dividindo-se em quatro partes principais:

- BIOS: sistema básico de entrada/saída,
- BPOS: sistema operacional, básico de disco,
- CCP: processador de comandos de console,
- TPA: área de programa transitório.

O fator que mais o ajudou a impor-se foi seu módulo BIOS, isto é, aquela parte que permite adaptar o CP/M às características (memória, periférico e discos) de qualquer computador que utilize o Z 80. Tudo isto sem esquecer que, segundo a opinião da maioria, o CP/M é um sistema operacional fácil de aprender por parte dos recém-iniciados, com um péssimo diagnóstico de erros e um método de acesso aos discos bastante lento. Mas, ainda que pareça curioso, é precisamente esta escassa otimização do sistema que permitiu uma notável adaptação às características de diferentes computadores. Com o CP/M podem ser gerenciados programas de um comprimento até 8 Mbytes cada um, que utilizem até 16 discos, com a possibilidade de proteger os programas individualmente e de repartir um disco em 16 áreas diferenciadas.

A última versão do CP/M é a 3.0, denominada vulgarmente CP/M Plus. Oferece uma organização dos dados em árvore (veja, mais adiante, o que comentamos no Unix), uma gestão da memória de acordo com as grandes quantidades desta, disponíveis hoje em dia e sofisticações típicas dos sistemas operacionais modernos.

Além disso, foi introduzido no mercado o CP/M Personal, totalmente residente em ROM, que permite implantá-lo em computadores domésticos. Isto permitiu também aos sistemas portáteis, dispor de um verdadeiro sistema operacional.

Apple DOS

Foi desenvolvido especialmente para o Apple II, e graças a enorme difusão que este teve no mercado, obteve grande êxito. Possui um grande ponto fraco, **é utilizável somente neste computador.**

Trata-se de um sistema operacional estudado a propósito para aproveitar as características do Apple, porque ocupa pouca memória e deixa à disposição do usuário a maior quantidade possível de RAM. Existe no mercado uma quantidade verdadeiramente notável de software de aplicação. Assim, pode ser encontrado um número incrível de jogos (o que sem dúvida alguma contribuiu de forma determinada para a difusão deste computador no mercado americano), mas também grande abundância de programas de aplicação desde os de administração de armazéns até outros mais sofisticados que fizeram possível também a utilização do Apple em aplicações profissionais.

O futuro deste sistema operacional está claramente ligado às intensões de sua empresa, que parece mostrar-se decidida a continuar utilizando este produto para suas futuras aplicações na área dos 8 bits.

MS-DOS, o standard da IBM

Desenvolvido pela firma americana Microsoft, é conhecido também como PC-DOS, já que foi o sistema operacional escolhido pela IBM para seu Personal Computer (IBM-PC). Tem a sua disposição uma enorme quantidade de programas de aplicação, sobretudo no campo das aplicações profissionais. **Foi escrito para os microprocessadores de 16 bits 8086 e 8088 da Intel,** (o segundo tem um bus de dados externo de 8 bits, mas internamente trabalha também com 16), de modo que funciona unicamente em equipamentos que adotam este tipo de CPU.

Foi iniciada uma veloz corrida entre inúmeras casas fabricantes para realizar computadores de 16 bits baseados nas CPU 8086/8088; são os chamados "compatíveis" (IBM-PC, claro). O MS-DOS repete assim o fenômeno CP/M na área dos 16 bits. De todas as formas, se trata de um sistema operacional avançado em relação aos seus predecessores de 8 bits, já que **está dotado de uma estrutura em árvore para catalogar os programas que são encontrados em disco**: esta é precisamente uma das tendências dos sistemas operacionais da última geração, como o Unix.

Sob o ponto de vista sistemático, o MS-DOS possui um grande mérito pela possibilidade de acrescentar com facilidade novas unidades de disco sem penosas operações. **Sob o ponto de vista do usuário, ao contrário, a maior vantagem é a abundância de software disponível**. Ao ter-se imposto como o mais vendido entre os computadores pessoais, o IBM-PC induziu com rapidez a todas as empresas de software do mundo a comprometer-se ao máximo na redação de programas para este sistema, superando atualmente inclusive aos disponíveis para o CP/M.

Outra vantagem do MS-DOS é a do **tratamento das mensagens de erro** que, enquanto no CP/M ficava demasiadamente reduzida e escura, é, ao contrário, **extensa e clara** neste sistema.

Empurrados pelo êxito obtido, estão sendo produzidos hoje distintas **versões deste sistema**, com melhores e novas prestações. **Talvez a mais interessante seja a Windows**, da firma Microsoft, **que permite a execução de vários programas a um tempo**. Cada programa se "apropria" de uma parte da tela, de maneira que o usuário pode observar seu funcionamento em paralelo. No entanto, não é possível contudo, em rigor, falar de um verdadeiro sistema multi-área, como o Unix ou CP/M concorrente, já que os programas, para poder funcionar no modo que é definido como "concorrente", devem residir todos juntos em memória, o que supõe um forte limite pela quantidade de memória disponível.

Nos sistemas operacionais mais avançados, para remediar tal inconveniente é utilizado o disco de uma maneira muito intensa, tratando de fazer residir cada vez na memória central somente pequenas partes dos programas em execução, transportando-os do disco à RAM quando for necessário e apagando-os quando já não façam mais falta.

A família CP/M de 16 bits: muitos filhos, mas nem todos afortunados

Com o advento dos microsistemas de 16 bits, o CP/M não podia, ficar atrás. Assim foi reescrito em uma versão para o micro-

processador de 16 bits 8086 na Intel, chamado CP/M-86, que permite usá-lo, por exemplo, no IBM-PC, que o adotou como alternativa a seu MS-DOS.

Sua filosofia é a mesma que a do CP/M 80 (porque ainda permanecem esses pequenos vícios de indecifrábilidade para o usuário), mas foram incluídas certas melhorias que pretendem fazer do CP/M-86 um dos vitoriosos na guerra desencadeada entre os sistemas operacionais para os micros de 16 bits, ainda que até agora teve pouco êxito devido à supremacia no mercado do MS-DOS.

Mas a característica mais interessante dos esforços da Digital Research é o desenvolvimento de uma família CP/M de 16 bits: de fato, existe disponível uma variedade multiusuário, denominada MP/M-86, uma versão multiárea-monousuário, o CP/M concorrente e o CP/M-68K para computadores baseados no microprocessador de 32 bits (e bus de dados externo de 16) 68000 Motorola.

O MP/M-86 (disponível também para 8 bits com o nome de MP/M-2) é um dos mais conhecidos entre os ainda raros multiusuários.

Dispõe de uma gestão da memória a participação fixa e oferece também uma forma simples, mas eficaz, na blocagem dos registros.

O produto mais interessante de toda a gama é, com toda segurança, o CP/M concorrente. Este sistema permite a um só usuário ter vários programas funcionando ao mesmo tempo em um único computador. Pode, por exemplo, estar correndo um programa que realiza o balanço mensal, enquanto o usuário se dedica a escrever uma carta na tela e em uma impressora está saindo a listagem ou os resultados de outro programa. Este sistema operacional, que requer ao menos 256 Kbytes de memória e um abundante suporte de memória de massa (normalmente um disco rígido), foi realizado "ex professo" para o IBM-PC, no qual funciona com regularidade, ainda que agora é encontrado também disponível para outros computadores que utilizam o 8086.

A versão com olhar mais para o futuro de todos os CP/M é, sem dúvida alguma, o CP/M-68K. Se trata, de fato, do primeiro de toda a gama que está escrito em linguagem C (veja mais adiante o parágrafo sobre o Unix); isto faz que, sempre que tenha à sua disposição os compiladores apropriados, possa ser utilizado em computadores com microprocessadores diferentes. A primeira versão foi escrita para tratar de marginalizar ao recente Unix, que funciona precisamente e com preferência, em computadores baseados no CPU 68000 da Motorola. Mas esta ânsia é muito difícil que chegue a cumprir-se: o Unix chegou com força e, com as características

e prêmios que oferece, é muito superior.

O grande Unix

Considerado por todo o mundo como o sistema operacional dos anos oitenta, apareceu em 1984 no mundo dos computadores pessoais após o acordo firmado entre IBM e Bell Labs para uma futura implementação no IBM-PC.

É um sistema que **permite o funcionamento tanto multiárea como multiterminal**, facilitando assim a presença de vários usuários conectados em um mesmo instante ao mesmo computador. O sistema do êxito deste sistema operacional, que obrigou a rivais como o CP/M e o MS-DOS a inspirar-se nele para manter suas vendas, reside, por um lado, em ter sido escrito em C (o qual faz muito facilmente transportável) e, por outro, em sua filosofia, altamente inovadora.

Podem ser distinguidos três níveis: envolvente, núcleo e File System.

O **envolvente** (em inglês "Shell"), **atua como interface entre o sistema e o usuário**. Para seu manejo, o usuário pode utilizar um tipo de linguagem a muito alto nível, o "Command Language", que permite um uso bastante eficiente. De fato, este modelo de linguagem possui uma forma particularmente coloquial que permite ser utilizado inclusive por pessoas não experts na arte da programação. Além disso, existe a possibilidade de escolher e personalizar este aspecto: cada firma que adota o Unix pode remodelar o Shell; inclusive a empresa mãe permite ao usuário a possibilidade de escolher entre dois.

O **núcleo**, ao contrário, **é a parte do sistema operacional que gestiona diretamente o hardware do computador**; se corresponde na prática com o BIOS do CP/M. Permanece em memória durante todo o tempo em que o Unix está ativo, controlando todas as funções de acesso ao disco, tratamento da memória e operações de entrada/saída.

O **File System** (sistema de arquivos) **é a parte mais original**. Está organizado com uma estrutura em árvore (Fig. 8) que permite, por exemplo, a existência de vários programas com um mesmo nome, ainda que para ser alcançados deveremos recorrer distintos caminhos (pathway).

O usuário Unix tem à sua disposição três possíveis tipos de arquivos: comuns, diretórios ("directory") e especiais.

Os arquivos-diretório são listas de outros arquivos ou de outros diretórios e se comportam como os comuns, mas não podem

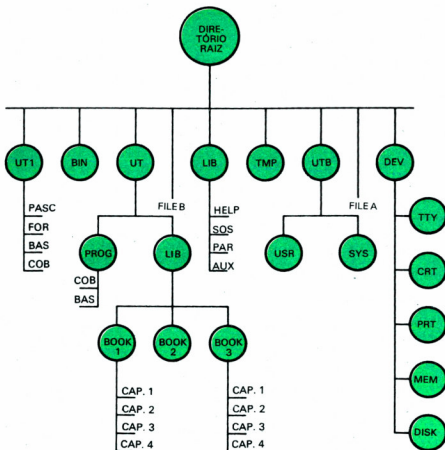


Figura 8 — Organização em árvore do Unix. A linha ondulada indica o percurso, desde a raiz aos sub-diretórios, necessário para alcançar um arquivo determinado.

ser escritos por outros programas; somente podem ser manejados pelo sistema. Os diretórios podem ser vários e estão todos dependentes do Diretório Raiz, que é um arquivo que contém os nomes de todos os Diretórios. Recorrendo às sucessivas cadeias de Diretórios se pode chegar a cada um dos arquivos.

Antecipando algo sobre o que insistiremos a seu devido tempo, o arquivo CAP.2, incluído no diretório BOOK, que forma por sua vez parte do LIB, que, por último, é encontrado na lista UT, é

reconhecido como o nome composto *UT/LIB/BOOK/CAP.2*. Na Figura 8 está remarcado com linha o percurso que, desde a raiz, leva ao arquivo.

Os arquivos especiais são conceitualmente mais difíceis de entender, por não ser uma coleção de dados ou informações como os demais. De fato, são utilizados para poder aceder aos canais de entrada/saída de uma maneira fácil e eficiente. Cada mecanismo de entrada/saída tem ao menos um arquivo especial, cujo nome indica o tipo de dispositivo que gestiona. Um programa, ou um usuário, que desejam utilizar, por exemplo, uma impressora, não necessitam saber nada acerca deste dispositivo: bastará que façam referência ao arquivo "impressora" normalmente para que o sistema operacional gestione a operação.

Outros sistemas operacionais

Existem alguns sistemas operacionais que no Brasil ainda não são muito conhecidos, mas que nos EUA têm certa difusão, apesar de não serem em absoluto comparáveis aos sistemas tratados até agora.

Destes, o mais extenso é o "p-System", desenvolvido pela Universidade de San Diego, na Califórnia (a mesma do UCSD Pascal). Se trata de um sistema para um só usuário, cuja principal característica é a adaptabilidade a todos os computadores. Além disso, programas escritos com o p-System para microprocessadores de 8 bits, podem ser facilmente transferidos a um p-System de 16 bits. Ao ter sido realizado em um ambiente universitário está muito difundido na América como instrumento didático, mas naturalmente são escassos os programas de aplicação de tipo profissional. Este fato, acrescentando à falta de compiladores para linguagens menos científicas que o Pascal e o FORTRAN, faz incerto seu futuro.

Tem sido, ao contrário, muito bem aceito no mundo comercial o sistema operacional Oasis. Apto para computadores de 8 a 16 bits, mono e multiusuário, é muito fácil de utilizar e oferece bons préstimos tanto sob o ponto de vista da segurança como da organização dos arquivos. Por desgracia, devido a uma série de motivos, a versão de 16 bits entrou no mercado demasiadamente tarde em relação ao Unix e agora fica muito difícil cobrir a diferença que foi criada.

Totalmente orientado para o mundo comercial está o BOS. Este é um sistema multiusuário, desenvolvido na Inglaterra com o fim de suportar uma linguagem, o Micro Cobol, com algumas características decididamente de gestão. Se trata de um sistema que

funciona na atualidade em vários computadores, permitindo a gestão dos arquivos ISAM (Indexed Sequential Access Method). Ao ter como única linguagem o COBOL não permite assegurar a possibilidade de um grande desenvolvimento no futuro.

Concluimos nossa descrição com um sistema operacional muito interessante: o **Pick**. Desenvolvido no princípio para os micro-computadores foi agora readaptado para os aparelhos fundamentados em microprocessadores em **versões tanto monousuário como multiusuário. Oferece préstimos de grande relevância, como a organização da memória virtual, e a capacidade (intrínseca) para gestionar bases de dados; é muito veloz e fácil de utilizar.** Na atualidade está disponível somente para alguns poucos computadores baseados no microprocessador 68000 da Motorola, mas já existe na América uma versão para o IBM-PC.

CAPÍTULO II

A MEMÓRIA CENTRAL: UMA ORGANIZAÇÃO COMPLEXA

Generalidades

A organização da memória principal, interna, central ou primária (em contraposição à secundária ou de massa, da qual falaremos no capítulo 3), que de todas estas formas pode ser chamada é um dos aspectos mais importantes no mundo do computador. Precisamente dentre as funções mais árduas para um sistema operacional destaca-se a gestão da memória interna, a tal ponto que podemos afirmar que, junto com a dos discos, tem sido o fator de maior influência na hora de projetar e realizar os distintos sistemas operacionais.

A memória central é aquela parte do computador na qual deve ser encontrado obrigatoriamente um programa para poder ser executado. Sem ela (e, conseqüentemente o software necessário em cada ocasião) não seria possível nenhum tipo de funcionamento. Além disso, quanto mais memória temos à nossa disposição, mais amplo será o uso que poderemos dar ao computador, já que nos permitirá escrever um maior número de linhas de programa e executar programas mais complexos, que requerem mais quantidade de memória para tratar os dados.

Nos computadores pessoais mais econômicos, nos quais a memória principal não é precisamente abundante, o sistema operacional proporciona mecanismos para utilizar a memória secundária, por exemplo, uma fita magnética ou um disco flexível, como expansão da primária.

Quando o usuário esgotou a quantidade de memória disponível e não pode armazenar o programa que está teclando, o sistema operacional é encarregado de gravar na memória auxiliar a informação presente na memória principal, deixando-a assim disponível para aceitar novas informações. Este sistema, ainda que simples, torna tremendamente lento o funcionamento, sobretudo se a memória principal disponível é pequena. Por outro lado, ao ser este dispositivo bastante custoso, o problema da quantidade de memória disponível é importante inclusive nos grandes computadores, onde existe nada menos que milhões de bytes. Por causa desta situação tem sido desenvolvidos métodos que permitem digamos assim, “simular memória” quando não existe. São as denominadas **técnicas de memória “virtual”**, que **permitem uma utilização bastante proveitosa da memória disponível**. No momento somente estão à disposição dos grandes computadores, mas serão com toda probabilidade utilizadas pelos computadores pessoais dos próximos anos, para permitir um melhor aproveitamento dos mesmos.

Organização da memória

Ao longo da história dos computadores o problema da memória interna tem sido tratado sempre com apreensão, já que pode incidir muito no preço do equipamento. Como é fácil de supor, a quantidade que deveria ser incorporada era avaliada cuidadosamente tratando de instalar a menor possível, para não exceder no custo, mas de forma que fosse suficiente para armazenar os programas do sistema operacional e os do usuário. Ultimamente, graças ao grande desenvolvimento da tecnologia, os preços baixaram um pouco, mas o problema, ainda que menos grave, continua existindo.

Em anos anteriores foram colocados à venda computadores pessoais com uma quantidade de memória apenas suficiente para conter o mais simples dentre os jogos “marcianinhos”. Deste modo o comprador se encontrava com a desagradável surpresa que, ao tratar de realizar um programa um pouco complicado, este não cabia no computador...

Por outro lado, o problema de utilizar a menor quantidade possível de memória sem diminuir os préstimos do sistema tem sido um objetivo que, desde sempre, atraiu aos estudiosos dos sistemas operacionais, que chegaram a desenvolver soluções verdadeiramente engenhosas. Graças a eles, como já dissemos, é possível hoje em dia fazer funcionar, nos grandes computadores, pro-

gramas maiores que as dimensões reais da memória.

Qual é a estratégia que o sistema operacional deve utilizar? Será um modo de trabalho para um só usuário, ou para várias pessoas trabalhando simultaneamente?

No primeiro caso será possível escolher entre uma forma de ubicação simples ou múltipla, de forma que seja permitido um funcionamento de tipo "multitask" (isto é, vários programas de um mesmo usuário funcionando ao mesmo tempo). No segundo caso terá que escolher a filosofia para a gestão multiusuário 'timesharing', multiprogramação com divisão de memória ou de petição de página.

Vamos examinar com mais detalhe o funcionamento da memória de um computador. Como já vimos, existe dois tipos de memória: a primária, ou central, e, a secundária, chamada também periférica ou de massa. Desta última trataremos no terceiro capítulo do livro. A memória central está geralmente constituída por circuitos de somente leitura (ROM, PROM, etc.), e RAM, para leitura e escrita.

Os primeiros, de somente leitura, são freqüentemente, de um tipo intermediário, como as EPROM (isto é, que podem também ser reescritas por meio de aparelhos específicos, bastante simples e econômicos); são circuitos integrados que, programados pelo fabricante no momento da construção do computador, não podem ser modificados por nenhum programa (ainda que, em casos como o antes citado das EPROM, sim pelo usuário). São, em geral, utilizados para conter programas de "bootstrap", pequenas partes ou, inclusive, todo o sistema operacional (ou um intérprete BASIC) em aparelhos que não possuem discos.

A vantagem destes dispositivos, além da de não ver voláteis como a RAM, é que, ao não poder ser reescritos desde o programa, garantem uma proteção segura contra acessos não desejados. Além disso, não necessitam transferências de informação dos discos, o que faz mais veloz o funcionamento dos programas contidos neles.

O inconveniente é que, ao não poder manipular seu conteúdo, conservam seu campo de direcionabilidade de forma definitiva, enquanto que às vezes fica útil reorganizar a disponibilidade de memória em base às exigências do momento.

As memórias de tipo RAM são empregadas, ao contrário para conter dados e programas durante o funcionamento destes, já que nesta fase é necessário que ambos se encontrem na memória central, pois a CPU não pode aceder ao disco cada vez que deve executar uma instrução.

A memória periférica (discos e fitas) é, pois, empregada somente para conservar as informações que voltarão a ser utilizadas.

Quando é necessário executar um programa que reside em disco, se transfere primeiro à memória central; o processador o executa e (por regra geral) o resultado é novamente transportado ao disco (em outros casos os resultados são levados somente à impressora e/ou tela). Quando o computador é desligado, o conteúdo da RAM é destruído, enquanto que o armazenado em disco fica gravado de forma permanente.

Nos computadores mais modernos é utilizada também uma memória denominada “cachê”, que possui algumas características parecidas com as da RAM, mas com um tempo de acesso muito mais breve. Estes dispositivos são muito caros, por este fato são utilizados com moderação. Cada vez que um programa é executado, o sistema operacional transfere a parte em execução (cujas dimensões dependem da quantidade de memória “cachê” disponível) a estes circuitos ultra-velozes, onde o funcionamento requer tempos muito mais curtos.

Naturalmente, a quantidade desta memória deve ser tal que evite que o tempo empregado pelo sistema operacional para transferir o programa desde a RAM normal à veloz seja maior que o economizado graças à “cachê”, pois em outro caso o sistema teria efeitos negativos no tempo de execução.

Para fazer ainda mais veloz a execução de um programa foram estudadas estratégias de busca (“fetch”) mais refinadas. O “fetch” é aquela fase na qual o processador vai buscar na memória o código da instrução que deve executar. Até a pouco tempo, pensava-se que, ao não ser possível conhecer de antemão o código que o processador necessitaria em seguida, não havia outro remédio senão realizar isto conforme o processador fosse requerendo.

Hoje em dia, no entanto, estão sendo desenvolvidas estratégias cada vez mais sofisticadas que permitem, com certa segurança, prever onde o programa deverá ir procurar a seguinte instrução, economizando assim uma grande quantidade de tempo. De fato, o pré-fetch é feito com antecipação, no preciso momento em que a CPU executa a instrução correspondente. Queremos fazer notar que este método, parcialmente adotado na atualidade em algumas CPUs de 8 e 16 bits, parece destinado a afirmar-se em um futuro próximo. Se trata de técnicas muito atuais, inclusive para os melhores computadores pessoais.

Como é gerenciada a RAM

Os computadores pessoais, como os primeiros que foram criados, utilizam uma forma de gerenciar a memória do tipo "simples usuário com distribuição contínua". Isto é, que somente um usuário de cada vez pode utilizar o computador e todos os recursos do equipamento CPU, memória, discos, periféricos) estão totalmente à sua disposição e, enquanto não são utilizados, permanecem à espera de ordens.

Os programas são dispostos na memória central segundo o esquema da Figura 1: na cabeça o sistema operacional com todos seus procedimentos, em seguida o programa do usuário e, por último, a área de memória não necessária, que permanece inutilizada. Deste modo se o usuário deseja executar um programa maior que a quantidade de memória disponível, o sistema operacional o comunicará que isto é impossível.

A única solução sofisticada que este tipo de organização apre-

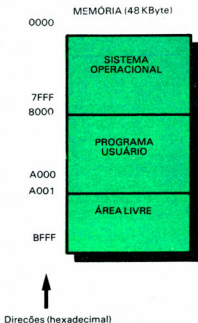


Figura 1 — Distribuição simples contígua: típica partição funcional da memória.

senta é o uso dos "overlay": o programa do usuário é subdividido em rotinas, utilizadas freqüentemente, e seções, que são utilizadas uma só vez; na memória central são carregadas somente aquelas partes do programa cujo uso é requerido com mais freqüência. As outras, ao contrário, são carregadas e executadas uma por uma, com maior emprego de tempo, certamente, mas com a vantagem de poder trabalhar sobre porções de memória mais reduzidas. Está claro que o esforço de subdividir um programa é elevado, de modo que se o sistema operacional não é capaz de efetuá-lo automaticamente, será preferível recorrer a esta técnica somente em caso de absoluta necessidade.

O sistema de proteção fica muito mais fácil, pois tendo um só usuário e um só programa em funcionamento, o único problema é impedir que o programa do usuário possa ser escrito acidentalmente sobre as posições de RAM reservadas ao sistema operacional. Para evitar isto, um registro específico é encarregado de controlar todos os acessos executados pelo usuário. Assim, quando existe uma tentativa de penetrar na zona de memória reservada ao sistema operacional, é colocado em marcha um mecanismo que interrompe o programa e assinala a infração.

Reduzir os tempos mortos

Nos anos cinquenta foram feitos estudos estatísticos acerca do funcionamento de um programa, e perceberam que aproximadamente 60-70% do tempo de execução estava na realidade dedicado a operações de entrada/saída, nas quais o processador não era utilizado. Assim, desde então tratou-se de aproveitar ao máximo estes tempos mortos para melhorar os préstimos do computador, dando origem a toda uma série de técnicas de gestão da memória que se comprometiam a não deixar nunca ociosa a CPU.

Esta inquietude deu lugar à multiprogramação: ao separar completamente as fases de processamento das de controle dos periféricos é possível aproveitar muito melhor o processador. Para fazer isto é necessário realizar a gestão de entrada/saída de uma forma "inteligente", de tal modo que seja possível o manejo dos periféricos de uma forma totalmente desvinculada do processador central. Assim será possível, por exemplo, efetuar a impressão de um programa, ao mesmo tempo que está sendo executado outro.

De fato, já existe no mercado alguns computadores pessoais dotados de dois processadores: um que é a CPU propriamente dita e outro que é encarregado do controle dos periféricos. Nos computadores maiores são complementados alguns "controladores de canal" que são ocupados de governar, de forma autônoma, os dis-

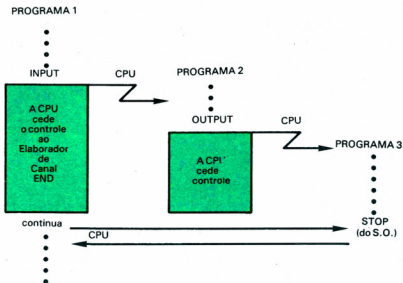


Figura 2 — Multiprogramação: passagem da atenção da CPU de um programa a outro nas fases de entrada/saída.

positivos externos. **Surge então o problema de como otimizar o uso da memória nestas condições.**

Observando a Figura 2 pode ser visto como, quando o programa 1 libera a atenção da CPU para executar operações de entrada/saída, o sistema operacional indica a execução do programa 2. Poderia ocorrer que este tivesse que fazer também operações de I/O antes de que o programa 1 tivesse terminado as suas. Neste caso, o sistema operacional poderá decidir a execução de um terceiro programa, e assim todas as vezes que seja necessário até que o primeiro esteja disponível para voltar a utilizar a CPU.

Naturalmente, para passar de um programa a outro é necessário salvar cada vez as informações relativas a que deixamos no instante em que foi suspensa sua execução e carregar as do novo cuja execução retomamos. Este procedimento torna um pouco de tempo, chamado tempo de serviço, da CPU. Enquanto o "tempo de serviço" que estes procedimentos de intercambio comportam sejam similares aos tempos mortos do processador, o assunto anda bem. Mas quando estes tempos não são suficientes se assiste a um progressivo deterioramento dos préstimos, que pode chegar a pro-

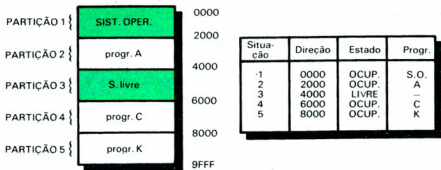


Figura 3 — Distribuição em partições fixas: o esbanjamento de memória pode ser muito grande.

vocar inclusive um “System Overhead” (desbordamento do sistema), durante o qual o equipamento gastará todo seu tempo para executar programas de serviço em lugar dos programas de usuário.

Após comprovar que este tipo de gestão comporta uma utilização melhor do computador, devemos considerar de que maneira repartir os programas na memória. Multiprogramação significa, naturalmente, manter vários programas a um mesmo tempo na memória central: se trata de fazê-lo com a maior eficácia possível. O método mais simples é o de subdividir a memória em “divisões” e situar um programa em cada divisão segundo o esquema da Figura 3. Em uma tabela guardada na memória é refletida a numeração das divisões, as dimensões de cada uma, suas direções e se estão livres ou ocupadas. Sua atribuição pode ser levada a cabo de forma estática ou dinâmica.

No primeiro caso (refletido na Figura 3) as divisões são fixas, porque é produzido um notável esbanjamento de memória, já que somente em contadas ocasiões os programas possuem as dimensões que haviam sido calculadas e, portanto, abundante espaço permanecerá inutilizado.

Na atribuição dinâmica as áreas são atribuídas com base nas dimensões do programa que foi encarregado. Desta maneira a memória é melhor aproveitada, mas também é produzido um fenômeno negativo: à força de subdividir a memória ocorre que algumas pequenas zonas não serão aproveitadas, dando lugar à chamada “fragmentação”. Para organizar o melhor possível a distri-

buição dos programas na memória ou, para dizer em linguagem técnica, situar e resituar as divisões, tem sido desenvolvidos engenhosos algoritmos: os mais famosos são chamados "first fit" e "best fit"

No "first fit" existe uma lista das zonas livres disposta segundo a ordem crescente de suas direções de partida, que não examina uma a uma até encontrar a primeira que satisfaça as exigências do espaço do novo programa do usuário. Deste modo permanece livre o fundo da memória para satisfazer possíveis exigências de grandes trabalhos.

O "best fit", ao contrário, utiliza uma tabela com as dimensões dos espaços livres, ordenadas por ordem crescente de seu tamanho. Isto permite a utilização da área que seja melhor adaptada ao programa que deve ser guardado, sem ter que procurar ao longo de toda a tabela.

Um estudo realizado pela IBM utilizando um IBM 360/65 deu os seguintes resultados: com programas de médias dimensões (128 Kbytes), a porcentagem média de tempo de espera do processador com uma distribuição simples contínua era de 65%; com um sistema de multiprogramação com quatro programas em funcionamento, esta porcentagem era reduzida aproximadamente a 10%.

Vantagens da memória virtual

Um método para tratar de resolver o problema da fragmentação é o de compactar periodicamente as zonas usadas da memória. A compactação é um método de resituação dinâmica que consiste em levar todos os segmentos utilizados a um extremo da memória, combinando todos os espaços livres no outro extremo. Na Figura 4, em (a), é vista a memória ocupada por três processamentos (A, B, e C) antes da compactação, e em (b) observamos como fica a distribuição de memória após realizar esta operação.

Isto conduz, no entanto, a uma série de problemas posteriores, como a modificação de todas as instruções que fazem referência a uma direção concreta da memória e de todas as estruturas de dados que fazem uso de ponteiros a direções. Isto é, para poder "compactar", o programa deve ser "resituável": as direções especificadas pelo programa na definição das instruções deverão poder ser transformadas em direções físicas diferentes em cada deslocamento dos programas efetuados pela compactação.

Esta é a técnica básica necessária para utilizar as chamadas "memórias Virtuais". Nos sistemas que fazem uso deste tipo de organização, a direção de uma posição de memória usada nas ins-

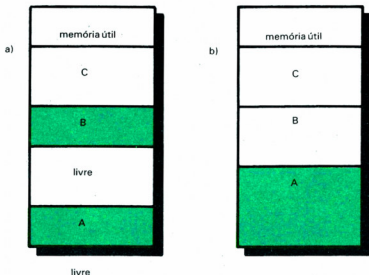


Figura 4 — Atribuição e disponibilidade de memória antes (a) e depois (b) de um processamento de compactação.

truções de um programa, codificado e pronto para a execução, pode deferir da direção (física) efetiva na qual o sistema operacional escreverá seu conteúdo. Este fato provoca uma **distinção entre a direção escrita pelo programador, que chamaremos “direção virtual”, e a que será na realidade utilizada durante a execução, que chamaremos “direção física”**.

O modo de atuar não é nada mais simples, pois deverão ser mudadas tanto as direções das distintas instruções, como os operandos que fazem referência à memória (escrita e leitura, saltos, chamadas). Ainda mais difíceis de resolver são os direcionamentos indiretos, isto é, aqueles nos quais a direção efetiva está por sua vez contida em uma célula de memória: neste caso, a operação de resituação deverá ser aplicada duas vezes.

Uma possível solução vem descrita na Figura 5. Nela é feito uso de um registro base para a resituação. Cada operação de direcionamento é executada somando à direção obtida pela instrução o conteúdo do registro base. Assim, no exemplo da figura, o programador deseja carregar no registro A o byte contido na direção “virtual” 10000. O sistema operacional somará a esta direção o conteúdo do registro base, neste caso igual a 30000. O dado então será lido na direção 40000, onde foi resituado.

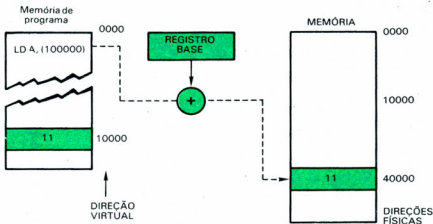


Figura 5 — A distinção entre direções virtuais e físicas permite a resituação da memória. O registro-base converte a direção virtual em física.

A vantagem deste método é poder gestionar de um modo mais flexível a memória, eliminando a fragmentação da qual falamos anteriormente. O tempo requerido para efetuar as operações de resituação e o preço dos registros especiais de resituação diminuem as vantagens deste método.

Organização em páginas

Uma primeira melhora é produzida com a organização em páginas. Com este método o espaço de memória utilizado pelo programa é dividido em "páginas" de comprimento (pequena) pré-fixado. A memória física, ao contrário, é subdividida em blocos, cujas dimensões são idênticas às das páginas. Deste modo é possível colocar as páginas do programa em blocos de memória fisicamente não contínuos.

Este aperfeiçoamento permite aproveitar melhor as zonas livres de memória colocando, por exemplo, somente algumas das páginas de um programa. A correspondência entre páginas e blocos é realizada em base a um conjunto de tabelas, chamadas PMT (Page Map Table, tabelas de mapa de páginas). Naturalmente é o sistema operacional o que é encarregado de manipular e colocar em dia estas PMT.

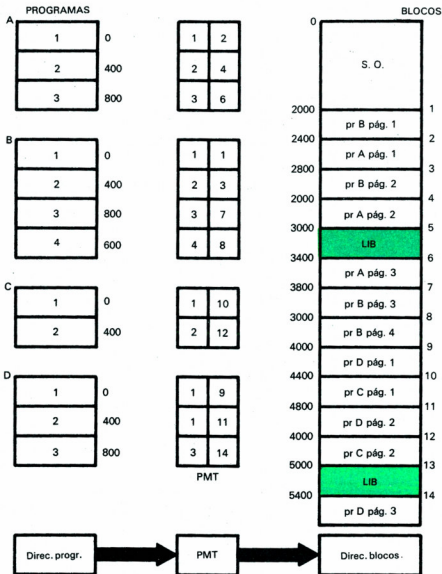


Figura 6 — Gestão da memória por páginas. As páginas que compõem cada programa são colocadas na memória em blocos não contínuos. O sistema operacional é encarregado de situar as páginas nos blocos segundo vão ficando livres. Para isto se serve da PMT (Page Map Table). Por exemplo, as páginas 1, 2, 3, 4 do programa encontram-se colocadas, respectivamente, nos blocos 1, 3, 7, 8 da memória física.

Na Figura 6 pode ser visto um exemplo, com quatro programas colocados a um mesmo tempo na memória. O programa B foi subdividido em quatro páginas que, por meio da PMT, são posicionadas em quatro blocos da memória física.

A gestão desta técnica é bastante complexa e requer um grande trabalho por parte do computador. As conversões páginas-blocos são efetuadas por registros muito velozes, para acelerar assim o tempo de execução, mas por estas características ficam muito caros.

O desenvolvimento deste método leva a um dos mais utilizados hoje em dia na gestão da memória nos grandes sistemas multiusuários: a "paginação".

A característica comum a todos os sistemas tratados até agora era que permitiam carregar em memória os trabalhos de vários usuários ao mesmo tempo, até a saturação da memória física disponível. Uma vez esgotada, não podia ser colocado em funcionamento um novo trabalho até liberado a memória o suficiente para poder carregá-lo.

Ao observar a execução de um programa subdividido em páginas fica de manifesto que não é necessário que todas as páginas sejam carregadas simultaneamente na memória para assegurar seu funcionamento, já que as rotinas de um programa não são utilizadas todas seguidas. Algo parecido ocorre com os arquivos de dados, em geral muito extensos: somente uma parte é utilizada cada vez durante as distintas fases da execução. De modo que é possível realizar um sistema operacional que permita gestionar e aproveitar uma "memória virtual" cujas dimensões sejam bem maiores que as (internas) físicas disponíveis.

Este tipo de organização implica, naturalmente, incrementar a complexidade dos sistema operacional, que deverá decidir quais páginas terá que manter na memória e quais não e, além disso, as operações que será necessário levar a cabo quando um programa requerer o acesso a uma página que não se encontra na memória nesse momento. Neste último caso, se não ficar espaço disponível terá que tirar algumas páginas para fazer lugar às outras, necessárias; o problema fundamenta-se em saber quais tirar, já que se retirar algumas páginas não adequadas, que o programa vai solicitar depois, todo o processamento anterior terá de ser repetido, com a perda de tempo que isto acarreta.

Para determinar quais páginas podem ser prescindidas se recorre a teorias estatísticas que operam segundo uma filosofia do tipo "probabilístico" sobre a futura utilização das páginas, baseando-se na história passada do programa. Deste modo, os al-

goritmos do tipo "FIFO" (First Input First Output, primeiro a entrar, primeiro a sair) eliminam a página que foi carregada na memória há mais tempo, enquanto que os do tipo "LRU" (Least Recently Used, usada há mais tempo) substituem a qual não é feita referências desde há mais tempo. Ambos os métodos possuem vantagens e defeitos, por isso não é possível prever se um é melhor que o outro.

O principal problema da paginação é que, dada a quantidade de funções necessárias, a sobrecarga do sistema operacional é notável, pelo qual torna-se mais lenta a velocidade do computador. Este e outros modos de funcionamento de parecida complexidade podem, chegando ao limite, conduzir ao fenómeno denominado "thrashing" (perda do controle) no qual é consumido quase todo o tempo em executar os programas de gestão, permitindo ao usuário dispor somente de 1% do tempo disponível. Em essência, o thrashing consiste em que o processador solicita páginas a maior velocidade da qual podem ser fornecidas. Chega um momento em que o processador tem que ser detido ("processador ocioso"), pois todos os processamentos estão esperando para que seja produzida uma transferência de página.

Outra técnica de gestão da memória é a segmentação. É conceitualmente parecida com a paginação, mas utiliza em lugar de páginas "segmentos" de dimensões variáveis, que são definidos pelo usuário no interior da instrução como parte da direção. Se utilizarmos a segmentação, toda direção será composta de um identificador de segmento e um deslocamento. Assim, uma instrução Assembler do tipo:

LD RO, < 2 > 1000 H

significará: carrega no registro RO o conteúdo da célula de memória 1000 (hexadecimal) pertencente ao segmento 2. Naturalmente, o sistema operacional procederá a operar uma conversão segmento-bloco para colocar o programa segundo suas exigências. O interessante deste tipo de organização reside em que os microprocessadores de 16 bits permitem implementar esta técnica com facilidade. Parece lógico, portanto, supor que os futuros desenvolvimentos dos computadores pessoais estarão orientados segundo esta filosofia.

Um caso Típico: A memória do IBM-PC

A memória central do IBM-PC está subdividida, como todas,

DIREÇÃO		FUNÇÃO	
DECIMAL	HEXADEC.		
0 . 48 K	00000 . 0C000	16 + 64K Byte RAM System Board	Proporcionada com o computador
64 K . 624 K	10000 . 9C000	até 448 K instaláveis pelo usuário	Para usos gerais
640K	A0000	Reservada	
656K ... 688K	A4000 ... AC000		Para visualizações e usos particulares
704K	B0000	VIDEO B/N	
720K	B4000		
736K	B8000	VIDEO Color	
752K 768K	BC000 C0000		
944K	EC000	192KB - expansão de memória	Aplicações especiais
960K	F0000	Reservada	
978K . 1008K	F4000 . FC000	48KB - ROM do Sistema Operacional	Proporcionados com o computador

Figura 7 — Distribuição da memória no IBM-PC.

em ROM e RAM. A ROM, programada pela empresa, contém aquela parte do sistema operacional que não pode ser modificada. Para efetuar qualquer mudança terá que substituir o circuito integrado por outro.

Os programas mais importantes contidos na ROM são as rotinas de gestão dos dispositivos de entrada/saída e o intérprete BASIC para fitas. Na versão base deste computador a ROM ocupa 40 Kbytes, situados ao fundo da área de memória direccionável. Existe também disponíveis suportes para outros 8 Kbytes se o usuário quer utilizá-los para armazenar programas de uma forma permanente.

Para carregar os programas que normalmente residem em disco, como o Advanced Basic Interpreter, e os do usuário, tem disponíveis além de 64 Kbytes de memória RAM. Na System Board (placa do sistema) existe também espaço para outros 192 Kbytes de RAM (até 256), que o usuário pode incorporar. Para necessidades maiores podem ser acrescentadas outras placas que permitem um máximo de até 640 Kbytes.

O espaço de memória está organizado sobre 1 Mega-byte (1.048.575 bytes), segundo o gráfico da Figura 7, partindo da direção 00000 até a direção FFFFF (hexadecimal). Para ter acesso a ela podem ser criados blocos, chamados segmentos, de um tamanho de 64 Kbytes cada um, que podem iniciar em qualquer ponto da memória, com a única condição de que sua direção de partida seja divisível por 16. Isto equivale a que na numeração hexadecimal a última cifra da direita de qualquer direção inicial de segmento seja sempre 0.

A parte baixa da RAM, desde a direção 0 em diante, é utilizada para carregar o sistema operacional DOS e o Advanced Basic desde o disco, além dos programas que o usuário deseja utilizar. Portanto, os caracteres escritos mediante o programa Editor terão que ser introduzidos nesta zona de memória.

Uma centena de Kbytes no centro da memória estão reservados para a gestão da tela, em branco e preto, a cores e inclusive gráfica, além de para outros cartões que é possível introduzir no computador.

No fundo, ao contrário, estão colocados os 48 Kbytes de ROM reservados para o sistema operacional. O modo de direccionamento utilizado é do tipo "segmentado", isto é, cada célula de memória está localizada por um par de números, como vimos anteriormente: um que identifica o segmento e o outro a posição relativa da informação em seu interior (offset). Deste modo, a forma de direccionar uma posição de memória já não é unívoca. De fato, vamos supor que temos que ler o conteúdo da célula situada na direção física 28519: se o segmento ao qual estamos nos referindo parte da direção 0, o offset será precisamente o 28519. Se, ao contrário, começa na localização 15000, a direção à qual teremos que

nos referir será a 13519 (tudo isto em decimal).

Para ter acesso a segmentos externos ao utilizado normalmente pelo intérprete BASIC é necessário utilizar a instrução DEF SEG. De fato, ao especificar um valor compreendido entre 0 e 65535, este é multiplicado por 16 e é convertido na direção inicial do segmento em uso desde esse ponto do programa em diante. Assim, a instrução:

```
10 DEF SEG = HC000
```

fixará o início do segmento na direção $C000_H \times 16_D = C0000_H$

As instruções sucessivas farão referência a este número para calcular as direções.

O BASIC utilizado no IBM-PC permite também ter acesso diretamente às células de memória por meio de uma série de instruções que poderão fazer referência a uma direção compreendida no segmento ativado pela DEF SEG ou a uma direção simbólica, expressa por um nome. Para ler um byte da memória é utilizada a instrução PEEK (n), onde "n" expressa direção compreendida entre 0 e 65535. Assim, ao colocar, por exemplo,

```
10 CELULA = PEEK (&H5000)
```

o programa lerá o conteúdo da direção 5000_H e o colocará na variável CELULA.

Ao contrário, para escrever é utilizada a instrução POKE n,m que introduz o valor "m" (um byte) na direção de memória "n". A instrução

```
POKE 1000,10
```

escreverá o número 10_D na célula de memória de direção 1000_D no segmento em curso.

Para transferir um bloco de memória ao disco (ou fita) ou ler um programa do disco (ou fita) e colocá-lo em uma posição determinada de memória existem as instruções BSAVE e BLOAD. Com a instrução

```
BSAVE "COPIA", &H1000, &H100
```

o sistema operacional colherá os 100_H (em hexadecimal = 256 em decimal) bytes que partem da direção 1000_H do segmento ativo e os gravará no disco, atribuindo-lhes o nome de COPIA. Por contra

BLOAD "COPIA", & H100

lerá do disco o conteúdo do programa COPIA colocando-o na memória central, no interior do segmento corrente, a partir da posição relativa 100_H.

No caso desta não estar definida, o programa será carregado a partir do ponto indicado por uma variável do programa situada na memória. A instrução VARPTR proporciona a direção, no interior do segmento ativo, da variável solicitada. É oportuno recordar também que o espaço de memória ocupado por esta variável dependerá de seu tipo: um número inteiro ocupará dois bytes, enquanto que um carácter ocupará um só byte.

CAPÍTULO III

SISTEMAS DE ARMAZENAMENTO DE DADOS: ORDEM E INTEGRIDADE



Os programas em execução em um computador trocam informações com os usuários através do teclado, terminal de vídeo, impressora, leitores de cartões, etc. Estes programas terão, portanto, que armazenar a informação recebida de uma maneira estável e permanente. Com este fim terá que utilizar um dispositivo, acessível no menor tempo possível, que possa armazenar uma quantidade considerável de dados e que seja acessível para todos os usuários do sistema, mas de forma que os dados não possam misturar-se uns com os outros. *Os dispositivos de armazenamento massivo utilizados na atualidade são os discos, rígidos e flexíveis, e as fitas magnéticas.*

Neste capítulo examinaremos as características físicas de discos e fitas, os modelos existentes e sua organização. Trataremos do conceito de arquivo, a maneira na qual são estruturados os dados que tem que armazenar em disco e a parte do sistema operacional que é encarregado da gestão dos arquivos.

Memória de massa

Sob o nome de memória de massa, ou memória secundária, estão englobadas as estruturas que permitem armazenar grandes quantidades de dados de uma forma permanente, isto é, discos e

fitas magnéticas. Podemos fazer distinções entre estes dois dispositivos com base em características como preço, quantidade de informação armazenável, velocidade e tipo de acesso. Estes parâmetros vão unidos às suas características e, por desgracia, frequentemente, se encontram em conflito. Assim não é possível ter um suporte velocíssimo, capaz de armazenar enormes quantidades de dados a um preço irrisório. Nos computadores pessoais (como nos micros domésticos atuais) havia até pouco tempo como única memória auxiliar o cassete de fitas: lento, pouco fiável, com escassa capacidade de armazenamento, mas de baixo custo. Naturalmente, este último era o parâmetro predominante. Agora, ao contrário, são utilizados discos flexíveis e rígidos de dimensões reduzidas, realizados expressamente para um mercado em forte expansão.

Fitas magnéticas

A memória de massa mais econômica é, indubitavelmente, a fita magnética. Fazem parte desta categoria tanto as fitas comuns utilizadas nos computadores pessoais como as grandes de 2400 pés usadas nos grandes centros de cálculo.

A capacidade de armazenamento de uma fita magnética é medida por:

- **comprimento físico da fita**, que costuma ser expresso em pés,
- **densidade de gravação**, que representa a quantidade de caracteres que podem ser armazenados na unidade de comprimento.
- Os códigos mais frequentemente utilizados são: ASCII e EBCDIC.

A conexão entre o computador e o cassete é realizada por meio de um dispositivo de interface que é encarregado de transformar a informação disponível no computador em formato binário em outra de tipo analógico válida para realizar a gravação (Fig. 1). O sinal analógico pode estar modulado em amplitude ou em frequência (Fig. 2).

Na modulação por amplitude são utilizadas técnicas de tipo PAM (Pulse Amplitude Modulation) que fazem corresponder ao valor 1 do bit um sinal de maior amplitude que a associada para o valor 0. Esta técnica permite uma complementação muito econômica, mas oferece também menos garantias, já que os computadores pessoais que adaptam esta solução apresentam, por regra

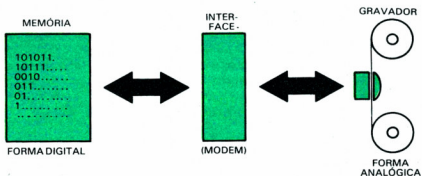


Figura 1 — Transferência de informação do computador ao cassete.

geral, problemas ligados à intensidade sonora de gravação, ruídos e outros problemas deste tipo.

Uma melhora notável é obtida ao empregar técnicas de modulação de frequência, tipo FSK (Frequency Shift Key). Utilizam uma correspondência bit-frequência, fazendo corresponder uma frequência diferente a cada valor do bit. Mas este sistema tampouco se livra de problemas, ressintindo de fenômenos como o Wow (flutuação de baixa frequência) e o Flutter (oscilação), típicos dos cassetes de fitas. De todos os modos, o grau de confiança nestes aparelhos está, na realidade, em função do grau de complexidade da estrutura previamente escolhida e, portanto, do preço.

As vantagens destes dispositivos ao compará-los com os discos são, sobretudo, duas: a econômica e a grande quantidade de informação que é possível armazenar. Por isto precisamente é que as fitas são utilizadas, por regra geral, para o "back up" (cópia de segurança) em computadores médios e grandes. Isto é, são empregadas para armazenar, ao final da jornada ou da semana, todos os programas e dados residentes em disco para criar um arquivo de segurança.

O principal inconveniente do armazenamento em fita reside na escassa velocidade que apresenta este sistema. Isto é devido principalmente à largura da banda da fita e o dispositivo de conversão dos dados (que não permitem sobrepassar uma certa velocidade na leitura/escrita) e, sobretudo, ao fato de que a fita é um dispositivo de acesso (físico) seqüencial.

tará pensar em uma fita musical corrente: se quisermos, por exemplo, escutar a sexta música, teremos que avançar a fita até este ponto, passando pelas outras músicas anteriores ainda que não nos

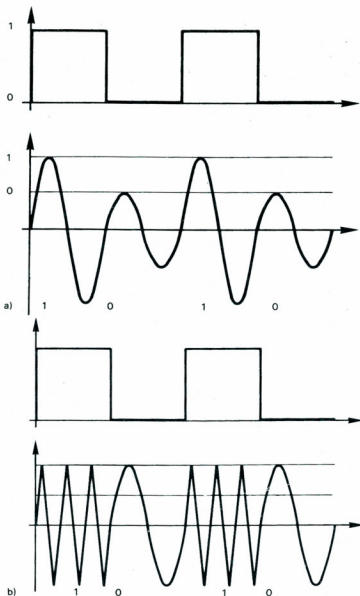


Figura 2 — Técnicas de modulação. a) modulação em amplitude (mais sensível aos sons); b) modulação em frequência do tipo FSK: os valores 1 e 0 do bit estão representados por sinais (analogicos) de distintas frequências.

interessem. O mesmo ocorre com os computadores: para ter acesso ao enésimo arquivo terá que perder tempo fazendo correr a fita até o ponto desejado. Em fitas grandes este tempo desaproveitado pode ser enorme (inclusive de muitos minutos).

Para solucionar ao menos em parte este problema foram estudados mecanismos de procura mais velozes. Voltando ao exemplo anterior, se sabemos exatamente onde se encontra a música que escolhemos, não teremos que escutar toda a fita até encontrá-la, iremos com a máxima velocidade possível ao número de volta determinado previamente.

Deste modo, se ao começo da fita colocamos um índice de todos os arquivos que contém e que nos dê o número de volta no qual é encontrado, o tempo de acesso será notavelmente reduzido. Tem que tomar a precaução, cada vez que é acrescentado um arquivo na fita, de colocar em dia o índice que se encontra no começo. Isto implica que terá que deixar um espaço livre, no qual o sistema operacional será encarregado de gravar e, em cada momento, atualizar o índice dos arquivos, cujo número estará, portanto, limitado pela quantidade de espaço que deixamos ao começo da fita. Ainda que apesar destes métodos as fitas não poderão competir nunca em velocidade com dispositivos de acesso direto como são os discos, melhoramos sua eficiência.

Discos rígidos e flexíveis

Voltando a utilizar a analogia musical, consideremos desta vez um disco. Como é sabido, para ouvir uma parte determinada do mesmo basta situar a cabeça leitora no sulco, adequado; efetuamos um "acesso direto" à música escolhida. O disco de um computador funciona da mesma maneira, somente varia seu aspecto. É **fabricado sobre um suporte duro ou flexível e o sulco**, visível no disco musical, é aqui invisível, já que **se trata de um traçado magnético, chamado "pista"**. A pista, por sua vez, está dividida em "setores", com capacidade normalmente para 128 bytes.

Os discos flexíveis, disquetes ou floppy disks (Fig. 3) são os mais utilizados nos Computadores Pessoais e micros por sua **considerável velocidade de acesso, fiabilidade e preço**. Construído sobre um suporte de plástico flexível recoberto por um material magnético, o disquete é dividido, por sua vez, em várias classes, em base a diferentes parâmetros.

Em relação com as dimensões temos hoje em dia três classes:

- **o clássico floppy, com diâmetro de 8 polegadas;**
- **o minifloppy, cujo diâmetro é de 5,25 polegadas;**

- e o microfloppy, com um diâmetro de 3,5 polegadas.

Existe além dos disquetes de tamanho não standard, alguns de dimensões muito reduzidas e outros com um sistema de gravação das pistas em espiral em vez de com círculos concêntricos, que estão destinados a aplicações particulares.

Houve um tempo em que a diferença nas dimensões implicava uma maior ou menor quantidade de informação armazenada. Na atualidade, graças aos desenvolvimentos tecnológicos, tudo isto foi modificado: de fato, ao aumentar a densidade de gravação de dados, um minifloppy pode conter até um milhão de bytes. Vamos proceder agora a outras classificações.

A primeira é uma **relação à superfície de gravação: são distinguidas em simples ou dupla (por um ou por ambos os lados)**. Esta, mais que uma distinção entre os discos em si, se refere às **unidades de disco** (isto é, as unidades de I/O), já que são estas precisamente as que podem ou não ler os dois lados do disco. Qualquer disquete pode ser utilizado por ambos os lados, pois estão as duas magnetizadas; como muito, será preciso fazer as perfurações utilizadas pela unidade de disco para controlar a proteção do suporte.

Quanto à densidade de gravação, tanto as informações contidas na pista como o número de pistas presentes no disco podem ser mais compactados. A densidade de gravação da pista simples pode ser dupla ou quádrupla, enquanto que o número de pistas de um disco pode ser duplicado.

A última distinção é realizada **com relação ao modo de identificar os setores: podem ser "soft" ou "hard sectored"**. Os "soft sectored", que são a quase totalidade dos discos utilizados nos computadores pessoais, têm um buraco que corresponde com o começo das pistas: um dispositivo fotoelétrico é encarregado de identificar o primeiro setor da pista, enquanto que os seguintes serão identificados por um mecanismo "software" que conhece o número e situação dos setores que tem na pista. Os "hard sectored", ao contrário, possuem um orifício para cada setor, motivo pelo qual a identificação é puramente mecânica.

A fiabilidade que oferecem estes dispositivos é hoje considerável, apesar de que, sobretudo em relação com os disquetes de elevada densidade de gravação, os discos rígidos são preferíveis para aplicações nas quais se requer grande confiança acerca de sua segurança.

Os discos rígidos, normalmente de tecnologia Winchester, são na atualidade o meio mais veloz, capaz e fiável disponível como

memória de massa pelos grandes computadores.

Graças ao desenvolvimento da tecnologia estamos assistindo à realização de discos rígidos com dimensões como as de mini e microflopys, com características iguais às de suas irmãs maiores, mas a um preço nitidamente inferior.

A produção destes dispositivos está, no momento, orientada para o mundo dos computadores profissionais e pessoais mais avançados, que requerem um grande suporte de memória de massa.

Vejam agora as características dos discos rígidos. Podem ser divididos em removíveis e fixos, e estes, por sua vez, em simples e múltiplos. Construídos sobre suportes metálicos (daí a denominação de "rígidos") possuem dimensões iguais ou, mais frequentemente, superiores às dos disquetes (14 polegadas). A distinção entre discos removíveis e fixos indica que os primeiros podem ser extraídos da unidade de disco (como os floppy), por este fato com uma só unidade de disco podem ser manejados vários discos. Os sistemas de discos rígidos utilizados pelos grandes computadores são formados por pilhas de discos rígidos com várias cabeças.

Organização de uma memória de massa

Como vimos no item precedente, um disco está dividido em pistas e setores (Fig. 3). Uma pista é uma linha de circunferência que, ao girar o disco a grande velocidade, passa rapidamente de baixo da cabeça, podendo esta ler ou escrever sobre a pista.

Os discos standard estão, em geral, organizados com 48 pistas por polegada nos discos de densidade simples, e com 96 pistas por polegada nos de dupla densidade. Isto é traduzido em uma média de 40 pistas disponíveis para um minifloppy de densidade simples e 80 para um de dupla densidade, em cada face. A pista, por sua vez, está dividida em setores, cujas dimensões variam, mas em geral, vão desde uma capacidade mínima de 128 bytes a uma máxima de 4096 bytes.

Nos miniflopys de densidade simples cada pista está dividida em 16 setores de 128 bytes cada uma; teremos, portanto, 2 kbytes de dados por pista. Em um disquete de 8 polegadas, como o Shugart SA 800, podem ser armazenados 5,2 Kbytes por pista, para um total de 400 Kbytes, por lado. De todas as maneiras, ao utilizar um minifloppy com dupla densidade e duplo lado podem ser alcançados os 370 Kbytes de um IBM-PC, por exemplo, comparáveis aos 400 Kbytes de floppy disk comum de 8 polegadas.

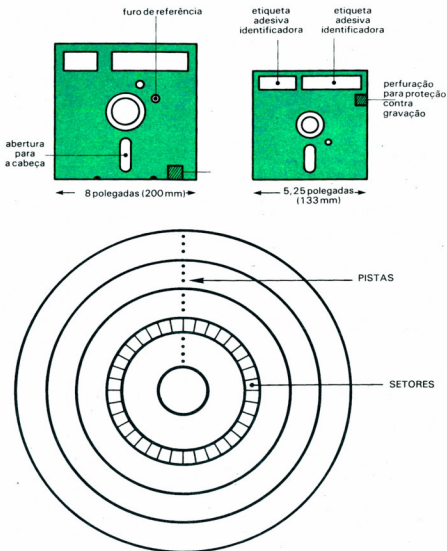


Figura 3 — Disquetes (Floppy disk) de 8 polegadas e minifloppy de 5 1/4 polegadas. Subdivisão em pistas e setores.

Estas cifras aumentam consideravelmente ao tratar-se de um disco rígido: um Starfire de 10 Mbytes possui 306 pistas, com 32 setores por pista.

Os discos novos são vendidos virgens ou, como dizemos em nossa gíria, “não formatados”. Isto é devido ao fato de que muitas firmas organizam seus próprios discos de uma maneira totalmente diferente, porque um gravado sob o sistema operacional CP/M não poderá ser lido com o sistema operacional MS-DOS, e vice-versa. A operação de formatar é aquela com a qual o sistema operacional inicializa um disco.

O comando “Format” é encarregado (nos discos soft-sectored) de dividir o disco em pistas e setores, gravando as informações necessárias para o movimento da cabeça na fase de procura do dado, além da criação do diretório.

A distribuição inicial dos setores no interior da pista comporta algum problema na organização: não convém ordená-los de forma seqüencial, contínuos. Vejamos, de fato, o que ocorre durante o acesso dos dois setores consecutivos. Após ter lido o primeiro setor o sistema operacional é disposto a repetir as mesmas operações para ler o segundo, mas enquanto isso o disco continua girando (com uma velocidade que pode ser da ordem de 3600 rpm), porque, para aceder ao seguinte setor, a cabeça deverá esperar que

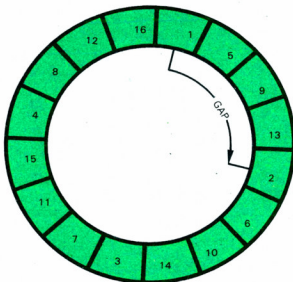


Figura 4 — Técnica de interposição dos setores nas pistas do disco: o gap dá ao sistema operacional tempo para dispor-se a ler o seguinte setor.

tenha sido completado o giro. O tempo que é perdido nesta operação pode chegar aos 40 e 100 milissegundos, segundo a velocidade do disco utilizado.

A maior parte dos sistemas operacionais não utilizam um sistema de ordenação dos setores de tipo seqüencial, mas uma técnica chamada de "interposição", na qual os setores estão colocados segundo a seqüência da Figura 4. Deste modo, na hora de acessar ao setor sucessivo, o sistema operacional terá tempo de reagir, sem ter que esperar ao cumprimento do giro.

O principal problema da gestão dos discos fundamenta-se na maneira de reservar espaço para os arquivos de dados e os programas, organizados em blocos denominados registros ("record") que são, em certo modo, o equivalente das páginas em gestão da memória central. É preciso otimizar o espaço disponível e reorganizar o disco ao ser apagado ou gravado um programa. Poderia ocorrer que o espaço recuperado não correspondesse com o necessário para colocar um novo programa. Neste caso, é óbvio que não se deverá utilizar uma colocação seqüencial, isto é, a gravação de um registro após outro, em ordem numérica para cada arquivo, mas um sistema mais sofisticado.

Uma das técnicas mais utilizadas é a dos ponteiros adiante-atrás (Fig. 5). Cada registro possui duas informações suplementares, uma situada na cabeça e outra na cauda, que dão, respectivamente, a posição (pista-setor) do registro precedente e a do seguinte. Assim, o sistema operacional não tem problemas na hora de reconstruir, quando tiver que carregar na memória, um programa fragmentado em inumeráveis registros separadamente por todo

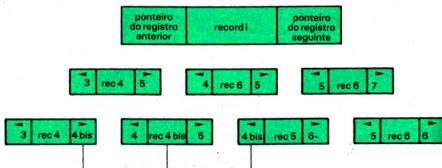


Figura 5 — Registro com ponteiros adiante-atrás: a) organização; b) exemplo de inserção de um novo registro.

o disco.

Outro método consiste em confeccionar uma lista com todos os registros em que está subdividido o arquivo, de maneira que o sistema operacional possa aceder diretamente ao interior do programa; mas esta vantagem coadjuva os inconvenientes do espaço ocupado pela tabela e o tempo necessário para realizar a conversão tabela-registro.

Seja qual for o método usado, o fato de apagar um programa, determina o nascimento de áreas vazias, de dimensões variáveis, que o sistema operacional deve reorganizar para armazenar novos arquivos. Esta operação de limpeza é denominada "garbage collection" (literalmente "recolhida de lixo") e pode ser feita à noite, quando o computador está mais livre, ou então quando o sistema operacional não encontra de imediato suficientes setores para colocar o arquivo necessário.

Na realidade, esta função é bastante complexa e nem sempre dá resultados desejados, porque geralmente se costuma preferir reescrever o disco após tê-lo reformatado.

Outra técnica muito simples e rápida para informar acerca do estado dos setores (livres ou ocupados) é a utilização de um "bit map" (na Figura 12 é observado um). Se trata de um mapa dos setores do disco onde cada bit corresponde a um setor: o primeiro bit, ao primeiro setor, e o último bit, ao último setor. O estado do bit corresponde ao estado do setor: se o bit está em 1, o setor está livre, o contrário se está a 0.

A última operação efetuada pelo Format consiste em reservar certo número de setores do disco para um índice geral, denominado Catálogo ou, mais comumente, Diretório, que contém todas as informações relativas aos arquivos que residem no disco.

Vejamos agora brevemente de que maneira está organizado o formato de uma fita. Por suas características físicas a fita permite somente acessos sequenciais, porque os arquivos estarão organizados simplesmente segundo blocos contínuos com, em todo caso, um encabeçamento especial para distinguir blocos de dados de blocos de programas. Infelizmente, cada firma adotou um método diferente para organizar as fitas (encabeçamento dos blocos, comprimento, etc.), motivo pelo qual é praticamente impossível ler com nosso computador fitas gravadas por um fabricante diferente.

Arquivos: esses desconhecidos

Com o nome genérico de arquivo entendemos hoje toda se

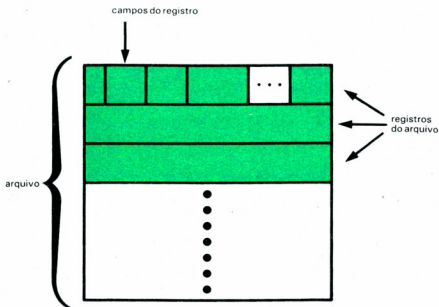


Figura 6 — Estrutura de um arquivo.

qüência de informações gravadas em um suporte físico. Esta sequência é utilizada para agrupar blocos de dados, programas de usuário ou de sistema, isto é, que recolhe um grupo de informações homogêneas, como podem ser o conjunto das instruções de um programa, código executável, uma base de dados para um arquivo, etc.

Como vimos anteriormente, a unidade menor de informação à qual podemos fazer referência em um disco é o setor: seu formato, escolhido no momento de inicializar o disco, pode ser de 128 bytes ou um múltiplo deste. Os arquivos são subdivididos em registros e os registros em campos (Fig. 6). Esta é a mínima unidade de informação a que pode fazer referência o usuário quando quiser aceder a um arquivo. O número de bytes que compõem um registro pode algumas vezes ser especificado pelo usuário segundo suas necessidades. Posteriormente, o programa de gestão do disco terá que efetuar a transformação registro-setor.

No interior do disco o arquivo está catalogado em um índice que contém uma série de informações suplementares denominadas atributos. Estes atributos costumam ser:

1. proprietário do arquivo;
2. dimensões;
3. posição no disco;
4. direção na memória;
5. data da criação;
6. data de utilização mais recente;
7. tipo;
8. proteções;
9. formato do registro.

Em seguida detalhamos o significado destes atributos:

1. O proprietário (ou criador) do arquivo indica, nos sistemas multiusuário, o possuidor ou a origem do arquivo.

2. Este parâmetro é apresentado ao usuário em forma de número de registros, já que esta é a unidade de informação que ele vê. Mas deve-se ter em conta também outro parâmetro: o formato do registro que, ao ser definível pelo usuário, pode variar de um arquivo a outro. Somente se considerar a união destes dados poderemos saber as dimensões reais de um arquivo.

3. A posição no disco indica a pista e o setor no qual começa realmente o arquivo no disco.

4. A direção na memória é, ao contrário, um dado definido pelo usuário, que serve para dar uma direção absoluta aos procedimentos executáveis (chamados correntemente programas). Em geral, decide em qual zona da memória carregar o arquivo retirado do disco, mas o usuário pode atribuir ao arquivo uma locação de memória absoluta na qual o programa será carregado e executado.

5. A data de criação (em certos sistemas é indicada até a hora) informa acerca de quando foi criado o arquivo. Este dado é útil quando são utilizados nomes parecidos para revisões de arquivos e já não nos lembramos de qual versão se trata.

6. A data de utilização mais recente indica, ao contrário, quando foi a última vez que foi modificado ou consultado o arquivo em questão. A utilidade desta informação será apreciada ao fazer sucessivas modificações ao mesmo arquivo. Os sistemas operacionais mais evoluídos são encarregados de colocar em dia automaticamente a versão mais recente do arquivo, criando outro novo (com um sufixo numérico que indica a cronologia) cada vez que se tem acesso a ele em uma fase de "editing" (edição).

7. O tipo é um parâmetro que especifica o formato do arquivo, isto é, a forma na qual foi estruturado. Os tipos podem ser:

- fonte (.SOU),
- de texto (.TXT),
- código objeto (.OBJ),
- listagem do programa (.LIS),
- ou expressam também a linguagem na qual foi escrito o programa (.PAS,.FOR,.BAS,.ASM, etc.).

8. Com a evolução dos computadores e a crescente importância dos dados neles guardados e dos programas, foi necessário desenvolver formas de proteção para os arquivos que impeçam que possam ser lidos, apagados, reescritos e inclusive o conhecimento de sua existência. É possível proteger um arquivo de forma que outro usuário que tenha a possibilidade de aceder ao mesmo disco não possa manipular ou investigar as informações nele contidas. Nos computadores maiores, utilizados por vários usuários ao mesmo tempo, foram adotadas técnicas de chaves de acesso, subdividindo aos usuários em quatro grupos:

- proprietário, quem criou é o proprietário do arquivo;
- grupo, usuários que compartilham com o proprietário certo número de recursos do arquivo;
- mundo, todos os demais;
- sistema, o System Manager, que deve poder aceder a todos os arquivos.

Em certos sistemas monousuários e nos computadores pessoais também podem ser protegidos o arquivo e todo o disco da escrita. Isto serve para defender os arquivos de nossas próprias distrações.

O File System

Um dos componentes mais importantes do sistema operacional é o **programa que gerencia os arquivos**, denominado File System. **Suas principais funções são:**

- **gestão do acesso:** proporciona o acesso aos dados contidos no arquivo em base e sua organização;
- **gestão dos arquivos:** organiza o armazenamento dos arquivos em disco, as referências, controla o uso dos arquivos comuns e garante sua segurança;
- **atribuição do espaço em disco:** é encarregado de ceder o espaço disponível no disco aos arquivos;
- **verificação (check) dos arquivos:** é uma das funções mais delicadas, pois é a que verifica se os dados contidos nos arquivos são corretos, encarregando-se de assinalar se alguma anomalia deixou fiabilidade à informação contida em um deter-

minado setor do disco.

O conteúdo desta parte do sistema operacional, que por si é complexa em um computador pessoal com um só disquete, é tremendo quando se trata de gerenciar um grande centro de cálculo. Basta pensar que aqui existe dezenas (ou centenas) de usuários, cada um dos quais possui centenas de arquivos. Existe nestes casos, em geral, várias dezenas de discos disponíveis e certas ordens de organização hierárquica que regulam a possibilidade de acesso dos usuários aos recursos que compartilham.

O File System tem, portanto, que classificar e atender um número enorme de petições de acesso a discos, garantindo, ao mesmo tempo, a integridade das informações e o acesso em um tempo tal que o usuário não se dê conta de sua existência.

Na Figura 7 podemos ver uma estrutura típica em árvore, sobre a qual o File System organiza aos usuários e seus arquivos correspondentes. O índice raiz corresponde ao índice geral do disco; os índices usuários, por último, as "folhas" terminais são os arquivos dos usuários, contidos em seus diretórios.

Estruturas deste tipo e ainda mais complexas são atualmente de uso comum nos grandes sistemas multiusuários, como o Unix, no qual a enorme ramificação de usuários e arquivos faz necessária uma estrutura particularmente complexa que permita uma administração eficiente do sistema.

O File System deve ser capaz de executar certo número de operações, como:

- criar, modificar e apagar um arquivo;
- permitir que, em um sistema multiusuário, um arquivo possa

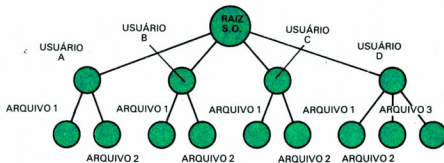


Figura 7 — Estrutura em árvore dos arquivos. Permite uma classificação melhor e oferece a possibilidade de dar o mesmo nome a arquivos que pertencem a subdiretórios diferentes.

- ser utilizado por vários usuários; **modelo relacional**
- intercambiar informações entre diferentes arquivos;
- possibilidade de reclamar um arquivo, em sistemas de vários discos, sem ter que especificar em qual dispositivo físico se encontra;
- visualizar a informação contida nos arquivos independentemente da estrutura que tenham adotado ao ser armazenadas em disco (registro, setores, etc.), de forma que o usuário possa aceder a elas com maior comodidade possível.
- criar um sistema de "back-up" automático que permita ao usuário, se por qualquer motivo perde o arquivo sobre o qual está trabalhando, recuperar ao menos parte de seu trabalho. Esta função deve ser realizada automaticamente em cada sessão de edição: quando o usuário acede a um arquivo que já existe, se faz uma cópia do arquivo que permanecerá de reserva no disco.

Poderia acontecer que, por exigências do programa, fosse necessário aceder a alguns dados do arquivo contido em disco sem ter que por isto executar uma fase de edição. Pois bem, **o File System proporciona ao usuário uma série de alternativas ao sistema, com as quais é possível fazer várias operações nos arquivos:**

- CREATE: cria um novo arquivo, encarregando-se de incluir seu nome no índice do disco e de reservar-lhe um espaço;
- DELETE: destrói um arquivo já existente, eliminando seu nome do índice e deixando livre o espaço reservado para ele;
- OPEN: prepara um arquivo para o acesso por parte de um usuário;
- CLOSE: conclui o acesso ao arquivo, impedindo qualquer outro acesso até o seguinte OPEN;
- RENAME: muda o nome do arquivo, atuando a nível do índice do disco;
- COPY: duplica o arquivo, mudando o nome se for desejado;
- APPEND: permite a união de dois ou mais arquivos em um só;
- SET PROPERTIES: permite a modificação das características que são encontradas no índice do disco;
- READ/WRITE: permitem ler ou escrever, respectivamente, os setores do arquivo em disco diretamente.

Um dos acidentes que ocorrem com maior frequência, sobretudo com os floppy disks, é que se perde a conexão entre um se-

tor e outro, o que determina que o usuário já não pode ler o arquivo. Para estes casos existem alguns programas, denominados Disk Repair, que inspecionam todos os setores do disco, eliminando os alterados e reestabelecendo a conexão entre os que estão em bom estado, o que permite ao usuário recuperar a maior parte (esse ao menos é o propósito) da informação.

Bases de dados: hierarquias e relações entre os dados

O desenvolvimento tecnológico e a demanda cada vez maior fez possível que os dispositivos de memória de massa, como os discos rígidos e flexíveis, adquirissem preços exequíveis. Isto empurrou aos usuários a criar arquivos de dados maiores, armazenados permanentemente em disco. As aplicações comerciais e industriais que requerem grandes arquivos para a gestão de contabilidade, tratamento de pessoal, etc., fizeram necessária a criação de uma estrutura de dados muito ampla e complexa que ofereça garantias e segurança. Assim nasceram as Bases de Dados (ou Data Base), que acrescentam a estas vantagens outras muito interessantes.

Uma organização empresarial possui muitas aplicações que utilizam os mesmos dados; por exemplo: um programa de gestão de salário e outro de organização do pessoal para a repartição do trabalho farão uso de uma base de dados comum que contenha os empregados da empresa. Se estes programas trabalhassem de forma totalmente independentes haveria um esbanjamento inútil de memória, já que os dados estariam duplicados, portanto será muito melhor organizar uma só lista de empregados a qual possam ter acesso ambos os programas.

Já vimos uma das peculiaridades que uma base de dados oferece: a possibilidade de ser compartilhada por vários processamentos. Mas, naturalmente, surge em paralelo o problema da segurança das informações feitas "públicas". É necessário organizar a estrutura com distintos privilégios de acesso, de maneira que cada usuário esteja autorizado somente a certas operações. Existirá assim quem poderá, por exemplo, modificar a importância de um salário, quem tão somente poderá ler o mesmo ou quem, talvez, poderá ler somente o de certas pessoas. Por outro lado, o fato de compartilhar a informação também tem suas vantagens: ao ter que trabalhar sobre a mesma base de dados se correrá menos o risco de que vários processamentos utilizem valores diferentes para uma mesma informação.

Outra característica importante das bases de dados é que sua estrutura deve ser independente do formato do dado armazenado.

do. Graças a isto, de fato, é possível adaptar a estrutura a diferentes exigências sem ter que modificar todo o sistema.

Para facilitar o uso de uma base de dados é proporcionada também uma linguagem específica, por meio da qual pode manipular-se a base de dados com comandos diretos ou em forma programada, mas sem ter que recorrer, como antes, a programas escritos em BASIC ou COBOL. Esta linguagem é freqüentemente potente e cômoda para o usuário. Se, por exemplo, o usuário de um sistema bancário quer saber quantos clientes depositam quantidades superiores a 50.000 cruzados em um mês nos últimos oito meses, não tem que fazer uma solicitação para cada tributo, mas terá a possibilidade de especificar todas as suas petições com um só comando.

Isto nos leva a um dos aspectos mais importantes de uma base de dados: os diferentes modelos do sistema segundo o tipo de chaves de acesso desejadas.

Imaginemos, por exemplo, que o encarregado de uma loja de ferragens (informatizada, claro) está interessado no artigo "chave de parafusos estrela". É possível que deseje saber quantas chaves estrela estão no armazém ou quantas dentre todas pertencem a uma determinada firma. Também poderia querer saber que quantidade de empresas têm chaves estrela em seus catálogos, ou então, diretamente, se a casa X tem uma chave estrela determinada. Como pode ser visto, inclusive em um caso em aparência tão simples,

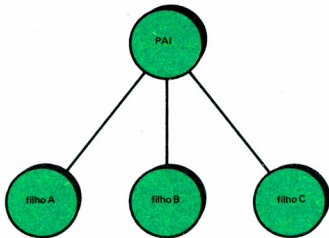


Figura 8 — Base de dados de tipo hierárquico de forma estruturada e simples, não é ainda suficiente para expressar relações complexas muitos-em-muitos.

existem diferentes solicitações e, portanto, distintos métodos de acesso.

Hoje em dia os modelos estudados e utilizados são três: hierárquico, de rede, relacional.

O **modelo hierárquico** (Fig. 8) apresenta uma estrutura em árvore onde as relações entre os termos estão reguladas por uma hierarquia pai-filho: um pai pode ter vários filhos, mas um filho não pode ter vários pais. Este modelo é relativamente fácil de complementar e de utilizar: as relações entre os dados são claras e bastante simples de seguir. No entanto, é demasiadamente rígido e por isso está começando a cair em desuso.

Para o caso em que tenha que expressar relações nas quais cada filho possua vários pais (se fala também de relação muitos-em-muitos, em contraposição à de um-em-muitos do modelo hierárquico), nasceu o **modelo de rede** (Fig. 9). É muito mais flexível que o precedente, mas tem também seu lado negativo devido a que, quando o número de conexões cresce, a rede se faz tão complexa que fica muito difícil decifrá-la.

Foi desenvolvido assim um modelo que supera estes inconvenientes e que é mais semelhante ao modo no qual representa-

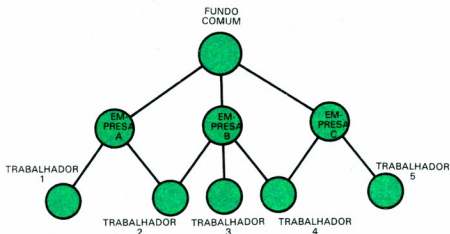


Figura 9 — Base de dados com organização de rede: o sistema reticular faz possível ter em conta todas as relações, mas é demasiadamente confuso e intrincado e, em casos complexos, é quase impossível de decifrar.

Relação: ferramentas

NÚMERO	EMPRESA	MATERIAL	PREÇO	QUANTIDADE	CIDADE
0153	BETA	AÇO	1000	10	LONDRES
10679	BETA	VANÁDIO	2000	10	LONDRES
80753	USAG	FERRO	500	5	ROMA
43799	USAG	AÇO	1200	15	ROMA
00152	BETA	CROMO	3000	12	LONDRES
35421	KRUPP	AÇO	1500	5	VALENCIA

Figura 10 — Base de dados de tipo relacional. É semelhante aos esquemas mentais do usuário e permite, ao mesmo tempo, uma gestão mais veloz dos acessos.

mos as estruturas de dados: o **modelo relacional** (Fig.10). Este sistema está composto por um conjunto de tabelas (linhas e colunas) que expressam as relações entre entidades diferentes; na figura mencionada aparece uma relação do tipo "ferramentas" na base de dados de uma loja de ferragens.

Sua função é clara: a cada tipo de ferramenta do armazém é atribuído um número de ordem, uma firma fabricante, um material de fabricação, um preço e a quantidade disponível. Somente

Relação: EMPRESA-CIDADE

EMPRESA	CIDADE
BETA	LONDRES
USAG	ROMA
KRUPP	VALENCIA

Figura 11 — Exemplo de projeção das relações. A potência do modelo relacional firma-se em que permite extrair de uma (ou várias) relações da base de dados visões diferentes, segundo as chaves de acesso e os mais variados critérios.

o primeiro campo de cada linha é único (de forma que seja fácil individualizá-las), todos os demais podem ser iguais.

O número de colunas indica o grau de relação. Ao mudar pequenos subconjuntos das distintas relações em jogo em uma base de dados relacional obtemos várias projeções possíveis, como a da Figura 11, que facilita a nova relação Empresa-Endereço.

Este modelo é compreensível com rapidez por parte do usuário e facilmente modificável, permitindo todo tipo de combinações. Além disso, os tempos de procura são muito menores, ao não necessitar nenhuma modificação de complicados ponteiros.

Começa-se a pensar, inclusive a nível de computadores pessoais, implementar algum tipo de Bases de Dados no próprio sistema operacional, com o fim de permitir ao usuário gestionar automaticamente suas Bases de Dados sem ter que recorrer a complexos programas adicionais. Por outro lado, existe no mercado inúmeros e variados pacotes de software para a gestão de bases de dados pessoais, organizadas inclusive segundo o modelo relacional (dBase II e III).

A gestão de arquivos no CP/M

Após ter visto os princípios relacionados com a gestão dos arquivos em um sistema de memória de massa, trataremos agora de como são executadas estas funções por um dos sistemas operacionais mais difundidos hoje em dia entre os computadores pessoais: o CP/M. *A gestão de arquivos no CP/M é efetuada pelos programas CCP e BDOS.*

Ao ter sido desenhado este sistema operacional com o fim de que pudesse ser adaptado a computadores com as características mais dispare, com a só condição de que a CPU seja uma Z80, o tipo disquete utilizado varia desde o de 8 polegadas até o de 5 1/4, desde densidade simples e dupla, desde hard até soft sectorado.

Para que fique simples, aqui nos ocuparemos da versão 1.4 do CP/M que utiliza um floppy disk do tipo IBM 3740, de 8 polegadas, densidade simples, soft sectorado. O disco está formado por 77 pistas, e cada pista por 26 setores. O formato dos setores é de 128 bytes e a correspondência entre setores registros é de 1:1. *A operação de formato organiza o disco da seguinte maneira: as duas primeiras pistas contêm informações do sistema e um carregador de bootstrap; na terceira pista, 16 setores estão reservados para o índice dos arquivos, enquanto que as restantes pistas estão disponíveis para o usuário. A máxima dimensão permitida de um arquivo é de 256 Kbytes.*

2. ^a cifra 1. ^a cifra	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	-	-	-	-	-	-	-	-	-	-	-	-	-

Figura 12 — Bit Map do CP/M, uma maneira bastante simples de conhecer o estado dos setores.

O CP/M utiliza para encadear os setores pertencentes a um mesmo arquivo **uma lista e setores, chamada File Control Block**. Cada uma destas listas, com um comprimento de 32 bytes, direciona 16 Kbytes. Estão normalmente contidas em disco, no interior do diretório, mas quando é utilizado o arquivo são carregadas em memória para permitir assim um acesso mais veloz.

Quando falamos da organização dos setores no disco vimos que era conveniente dispor os setores de modo contínuo, já que havia que esperar a que se completasse uma revolução inteira do disco entre um e outro acesso. Para evitar isto **o CP/M utiliza uma separação física (gap) de 6 setores entre um setor "lógico" e outro**.

O sistema utilizado para conhecer o estado dos setores é o BIT MAP. Consiste em um grupo de 8 setores contínuos (logicamente). Enquanto um dos setores do grupo está livre, o bit correspondente é mantido a 0. Quando todos foram atribuídos, o bit que corresponde é colocado a 1. Na Figura 12 aparece um possível BIT MAP do CP/M: até o bit 90 todos os setores estão ocupados, e a partir dele, todos estão livres.

Vejamos agora como funciona o **acesso do usuário a um arquivo**. Quando o usuário proporciona este nome no File Control Block, (FCB) do disco; se existe, encontrará a informação relativa a sua colocação no disco (pista-setor) e a transferirá à memória, a um segundo FCB criado pelo CCP.

Cada vez que é escrito algo no arquivo (para o que é necessária uma modificação do disco), o conteúdo do FCB é atualizado. Ao concluir a sessão o FCB presente em memória será copiado sobre o que se encontra em disco, atualizando assim todas as modificações efetuadas.

CAPÍTULO IV

SOFTWARE DE BASE: MAIS ALÉM DO SISTEMA OPERACIONAL

Suportes do sistema: com eles tudo fica mais fácil



Como já vimos no primeiro capítulo, o usuário de um computador não deve preocupar-se hoje em dia, como os de antigamente, de acionar uma série de interruptores para programar seu computador ou proporcionar-lhe um pacote de cartões mediante um leitor para executar um programa. Com o modo de funcionamento interativo, o usuário, por meio de um terminal de vídeo, pode escrever um programa, corrigir os erros, executá-lo, solicitar ao computador mais informação e inclusive utilizar programas incluídos previamente no sistema operacional para resolver certos problemas.

Na atualidade o usuário não tem por que saber como gestiona o computador, a memória ou os discos, ou como recebe os dados. O que interessa é utilizar o sistema da maneira mais simples possível para obter resultados. Isto é, **o computador deve ser um meio e não um fim**: se um usuário tem que resolver um problema, não deve ter que passar todo o tempo dispondo o computador para que funcione segundo suas necessidades, mas deve poder utilizá-lo imediatamente e sem muitas dificuldades. Esta é a finalidade do software de base (ou software do sistema), **que compreende o sistema operacional e outros muitos programas de suporte.**

Para este propósito, um **software de base moderno deve es-**

tar estruturado de maneira que ofereça aos usuários ao menos dois tipos de facilidades:

- suportes para o programa em execução,
- suportes para o usuário.

Os primeiros são encarregados de proporcionar ao usuário todos os meios de que se necessita durante a execução: memória, acessos a discos, entrada e saída de dados, impressão, etc. Enquanto que os segundos devem facilitar a realização de qualquer outro desejo do usuário. O programa deverá ser carregado em memória sem que o usuário tenha por que buscar uma área livre e colocá-lo dentro. Quando um dado tiver que ser inserido em um disco será o sistema operacional quem se preocupará de organizar todas as operações: busca do nome do programa no índice do disco, direcionamento da pista-setor correspondente (veja capítulo 3), localização em memória de uma área livre para a transferência, ocupação da área e posterior transferência.

Vemos que, inclusive neste caso, **existe toda uma série de operações que tornaria muito complexo o trabalho do programador se não existissem os programas de serviço, suporte ou assistência.** Outro, dos suportes essenciais que o software de base deve proporcionar é o controle de segurança. Poderia acontecer que o usuário, intencionalmente ou não, quisesse aceder a programas ou áreas proibidas, o que produziria problemas ou danos mais ou menos sérios. Para evitar isto, o software de base é ocupado de controlar toda operação efetuada pelos usuários ou por seus programas.

Com relação aos suportes proporcionados ao usuário, está sendo imposta ultimamente a filosofia de que sejam "amigáveis", isto é, serviços, fáceis de entender e práticos nas fases de criação, verificação e funcionamento do programa. Como estas assistências dependem da empresa que produz ou vende o computador, não é possível estabelecer um produto standard. Veremos em seguida os de uso mais comum.

Os Editores

Para compor um programa ou um texto, o usuário deve ter à sua disposição um instrumento capaz de aceitar os caracteres escritos e memorizá-los em seguida em disco.

Nos inícios da programação (isto é, há poucos anos...) o pro-

gramador tinha que teclar em uma máquina perfuradora um pacote de cartões com todas as instruções e introduzi-lo posteriormente no dispositivo leitor. Passado um tempo, que podia ser inclusive o dia seguinte, o computador assinalava os erros cometidos e, com toda probabilidade, o programador tinha que repetir tudo novamente. Este método, além de ser terrivelmente pesado, reduzia a possibilidade de manejar os computadores a alguns poucos experts privilegiados de roupão branco.

Hoje em dia, por sorte **todos os computadores possuem um programa que permite ao usuário escrever um texto, revisá-lo durante a composição, corrigir os erros e, por último, armazená-lo.** Estes programas, denominados "Editores", são encontrados disponíveis em várias versões e estão em constante evolução para que um número cada vez maior de usuários possam utilizá-los.

Vejamos agora as características destes programas, nascidos como auxílio ao usuário para permitir-lhe escrever textos com mais facilidade e flexibilidade, e que estão agora cada vez mais orientados para a edição de textos (Text Editor), isto é, para a composição de textos destinados à sua impressão.

Os Editores são divididos em duas grandes categorias: orientados para a linha e orientados para o carácter. Destes últimos derivaram posteriormente os pacotes comerciais conhecidos pelo nome de Word Processor (literalmente processador de palavras, ainda que costumam ser conhecidos como processador ou tratamento de textos, ainda com a ambiguidade que isto acarreta). Para entender esta distinção, vamos explicar como funcionam na prática estes programas.

Um Editor, uma vez "chamado" pelo usuário, por regra geral com o comando EDIT, **se ocupa de reservar uma zona de memória** destinada a recolher os caracteres escritos, **pergunta como desejamos que o programa seja chamado, reserva no disco um arquivo para este fim, deposita os caracteres na zona de memória reservada expressamente e, uma vez concluída a sessão de edição, é encarregado de armazená-lo todo em disco.**

Na realidade, **além** destas, que são as principais funções, o programa **deve desenvolver, em estreita relação com o sistema operacional, toda uma série de operações diferentes. As principais são:**

- uma vez iniciada a edição se ocupa de comprovar se o programa que o usuário quer "editar" é novo ou já existe; no segundo caso deverá ser encarregado de fazer uma cópia de segurança antes de permitir que o usuário escreva em cima;

- na hora de aceder a um programa partilhado são criados problemas de segurança: nem todos os usuários podem, por exemplo, escrever nos programas alheios. Existem regras de proteção que disciplinam os acessos, e o Editor, por meio do sistema operacional, não permite o acesso aos trabalhos protegidos;

- em um sistema multiusuário poderia acontecer que dois usuários quisessem ter acesso simultaneamente a um mesmo arquivo. Também neste caso o Editor impedirá o segundo acesso: seria uma boa "brincadeira" que enquanto alguém estivesse escrevendo um programa, outra pessoa fosse mudando as instruções...

A distinção que assinalamos anteriormente sobre os editores orientados para a linha ou para o carácter firma-se na forma de apresentar um texto e, portanto, no modo de operar do usuário.

Os editores orientados para linha permitem escrever, apagar e corrigir caracteres no interior da linha em questão. Após ter pressionado a tecla de "enter" ou "carriage return", a linha escrita torna-se definitiva, ainda que continue aparecendo na tela.

Para fazer uma correção é necessário dispor o modo de funcionamento "comandos", situar-se na linha que se deseja corrigir especificando seu número, expressar com um comando os caracteres que deseja substituir e efetuar a mudança. Como pode ser visto, se trata de um procedimento muito rebuscado e que faz perder muito tempo na hora de examinar o texto para encontrar e corrigir os erros. Estes tipos de Editores ficam particularmente abandonados, sobretudo quando se tem que compor textos, introduzir novos fragmentos, fazer correções ou desenvolver as diversas operações que são normais em tipografia.

Nasceram assim os Editores orientados ao carácter, chamados também "screen editor" (editor de tela), que permitem, por meio de quatro flechas (acima, abaixo, direita, esquerda) ou de um "paddle (alavanca manipuladora), mover à vontade o cursor ao longo do texto e efetuar todas as operações necessárias. Está claro que é muito fácil para o usuário, sobretudo se dispõe de um "paddle", seguir o texto escrito na tela e fazer correções: é como ler a página de um jornal servindo-se do dedo conforme se vai lendo.

Um dos métodos para compor textos atualmente no mercado é o do novo super personal da Apple: o Macintosh. Seu Editor é também gráfico: permite colocar no meio do texto certas figuras que o operador desenhou na tela por meio do "paddle" (que neste caso é o conhecido "mouse"). Além disso, tratando sempre

de fazer o computador o mais funcional possível, os comandos disponíveis estão visualizados graficamente nas margens da tela: desta maneira, até um usuário que desconheça a terminologia usada é capaz de executar as operações desejadas.

Temos que fazer uma distinção entre os "pseudo-editores" BASIC, disponíveis nos primeiros computadores pessoais e micro domésticos mais econômicos, e os verdadeiros editores, que são encontrados juntos com os sistemas operacionais dignos de tal nome e já implementados em quase todos os computadores pessoais (como o CP/M e similares). Vamos examinar o IBM-PC que possui ambas possibilidades: um modo de funcionamento BASIC, no qual podem ser escritos programas em linguagem BASIC e utilizar suas funções para gestionar o computador, e o editor EDLIN, que permite escrever programas em qualquer linguagem.

O Editor BASIC está praticamente funcionando sempre e considera "programa" tudo aquilo que vê na memória. Para iniciar um novo trabalho é necessário utilizar o comando NEW, que destrói o antigo programa presente na memória e prepara o Editor para aceitar outros novos. Este Editor está orientado para a linha, pelo qual memoriza uma linha cada vez. Tem que ser colocado um número ao início de cada linha, pois de outro modo o BASIC interpreta a instrução que foi escrita como um comando que tem de ser executado de forma direta e o leva a cabo após pressionar a tecla <enter>. Em modo indireto, ao contrário, o BASIC aceita e memoriza todas as instruções, precedidas pelo número de ordem escrito. No interior de cada linha podemos efetuar todas as modificações que desejamos por meio das flechas de deslocamento horizontal

Se queremos inserir uma nova palavra no meio das que já existem podemos utilizar o modo INSERT. Com o comando INS o computador permite introduzir um caracter novo, deslocando os outros para a direita. Se o que queremos é modificar linhas de programas já existentes, teremos primeiro que visualizá-las na tela e operar em seguida as mudanças.

Para visualizar um programa ou parte dele existem duas possibilidades, materializadas nos comandos LIST ou EDIT. Por meio destes comandos podemos visualizar uma ou várias linhas. Utilizando as flechas de controle do cursor nos posicionamos sobre os caracteres que queremos mudar e escrevemos em cima os corretos (ou acrescentamos outros com a tecla INS). Pressionando <enter> a linha será modificada também na memória. Para apagar uma linha bastará ativar o comando DELETE, especificando o número da linha ou linhas que se quer mudar.

Uma vez terminada a edição, o programa poderá ser guardado em fita ou disco por meio do comando SAVE. Desta maneira será possível conservá-lo de forma permanente e voltar a utilizá-lo carregando-o na memória com o comando LOAD. Uma vez que o programa é encontrado na memória será procedida a execução com o comando RUN.

Editor de BASIC é, como vemos, muito simples e essencial, e somente permite a edição de programas em BASIC. Para outras exigências temos à nossa disposição o **Editor EDLIN**.

Este, quanto a seu uso, é bastante parecido ao anterior, mas permite editar textos com formatos mais gerais. Também neste caso cada linha está caracterizada por um número, mas agora não faz parte do programa: a numeração é realizada pelo Editor e somente serve para fazer referência às linhas sem ambigüidades. O número de linha está compreendido entre 1 e 65529.

O EDLIN reside em disco e opera sobre programas contidos em disco. Se o arquivo que vamos editar já está criado carrega-o na memória desde o disco e, ao acabar, assinala o fim do arquivo de entrada; se não existia nos indica com a mensagem "New File" (novo arquivo).

O Editor é apresentado em modo "comandos" ao usuário com um "*". Para escrever o texto tem que pressionar um I: é situado assim em INSERT mode e tudo o que introduzimos será memorizado. O Editor, por sua parte, escreve na tela o número correspondente à linha que o usuário está editando: como já dissemos, este número não tem nenhum significado, na realidade, no contexto do programa.

Com o comando CTRL BREAK termina o INSERT mode e EDLIN volta a apresentar o carácter "*", indicando o funcionamento em modo "comandos". Nesta situação o usuário pode executar uma série de manipulações no texto:

- busca de uma seqüência (SEARCH): o Editor procura uma seqüência de caracteres determinada proporcionada pelo usuário ao longo do texto, parando na primeira linha onde a encontra. Isto é muito útil quando, ao voltar a tratar um programa editado anteriormente, se quer aceder a certo ponto sem ter que voltar a ler tudo;
- visualização do texto (LIST): permite ler na tela as linhas do programa compostas anteriormente sem modificá-las;
- substituição de um grupo de caracteres (REPLACE): permite a substituição de um grupo de caracteres por outros, ope-

rando sobre todo o texto. Se, por exemplo, temos que mudar em todo um programa o número 1000 pelo 999, com este comando é economizada a busca e correção todas as vezes que apareça o número 1000;

- comprovação de um linha: pressionando o número de uma linha,
- o Editor apresentará seu conteúdo na tela permitindo ao usuário sua modificação;
- fim da edição sem atualizar em disco as mudanças realizadas (QUIT);
- fim da edição (END): com este comando, o Editor salva em disco o conteúdo da memória.

Este Editor é, portanto, muito mais potente e versátil que o precedente.

Existem além desses outros Editores que pertencem às categorias dos "Screen Editor", mais conhecidos com o nome de WORD PROCESSOR (processadores de textos, chamados também sistemas de vídeo-escrita). Estes permitem compor textos para fins tipográficos e proporcionam na tela a imagem da página, com a possibilidade de efetuar correções e inserções, deslocando-se para cima ou para baixo.

Nos computadores pessoais melhores já existem produtos deste tipo, mais elásticos e potentes ainda que os Editores utilizados nos grandes computadores. Citamos entre estes o Word Star, programa que permite compor tanto textos como programas.

Tradutores e intérpretes

Agora que conhecemos os meios utilizados para escrever os programas, vejamos o que nos oferece o computador para verificá-los e executá-los.

Um computador funciona por meio de uma unidade central que executa uma série de operações definidas por seqüências de "0" e "1" (códigos binários). Escrever programas expressos em forma binária (ou hexadecimal) é uma operação das mais complexas e longas, de modo que recorreu-se a algum meio para simplificar o trabalho dos programadores.

Assim nasceram as linguagens Assembler (ou Assembly) e os programas das mesmas, que são os que transformam os símbolos mnemotécnicos (ou mnemônicos) que constituem as instruções em linguagem Assembler a código máquina.

Os Assembler possuem quatro vantagens de interesse em re-

lação aos códigos binários ou hexadecimais da linguagem máquina:

- estão compostos de símbolos mnemotécnicos e não por códigos numéricos da linguagem máquina;
- as direções são simbólicas e não absolutas;
- são muito mais fácil de ler;
- fica mais fácil incluir dados no programa.

Estas linguagens imitam bastante fielmente a estrutura do computador e, graças ao uso de símbolos mnemotécnicos para as instruções, permitem escrever e ler com maior facilidade os programas. O programa Assembler é encarregado mais tarde de traduzir para a linguagem máquina as instruções escritas pelo programador, obtendo alguns códigos compreensíveis para a unidade central.

A outra grande vantagem de não utilizar a linguagem máquina reside em que o programador não terá que preocupar-se das direções às quais faz referência na confecção do programa, pois será mais tarde o Assembler (e em seguida o "Linker") quem ordenará e calculará convenientemente estas direções.

O nascimento dos Assembler trouxe associado o dos **disassembler**. Estes programas, como seu próprio nome sugere, cumprem a função inversa, isto é, passam o código executável de um computador à linguagem Assembler. Fica fácil adivinhar a utilização que pode ser dada a estes programas: se os aplicamos ao código de um programa executável (por exemplo, um jogo ou uma parte do sistema operacional) por si só indecifrável ao ser uma longa sucessão de números, nos encontramos frente ao mesmo programa, mas em forma simbólica, ficando assim muito mais compreensível. Desta maneira, um bom programador será capaz de entender como funciona o programa e inclusive, se o desejar, de adaptá-lo às suas necessidades. Isto explica porque os programas disassembler não são sempre entregues junto com o sistema operacional...seria a anarquia total!

De todas as maneiras, em muitas ocasiões também as linguagens Assembler são insuficientes para as exigências de programação. Basta como exemplo observar a Figura 1: é mostrada a maneira na qual é implementada uma simples operação de multiplicação, escrita na linguagem Assembler do Z80, isto é, associado à CPU Z80. Como se vê, fica bastante incompreensível e, a menos que se trate de um expert, não fica fácil descobrir que essa longa sucessão de siglas corresponde a uma simples multiplicação entre dois números. Por este motivo nasceram as linguagens de al-

0	MULT ; Multiplicação entre dois ; inteiros de 16 bit ; sem sinal ; Multiplicação em HL ; Multiplicando em DE ; Resultado em HL
1	LD B, 16
2	LD C,D
3	LD A, E
4	EX DE,HL
5	LD HL,0
6 MLOOP:	SRL C
7	RR A
8	JR NC,NOADD-\$
9	ADD HL,DE
10 NOAD:	EX DE,HL
11	ADD HL,HL
12	EX DE,HL
13	DJNZ MLOOP-\$
14	RET
15	END

Figura 1 — Subrotina de multiplicação para a CPU Z80.

to nível, que expressam um modo de programar o computador mais compreensível para o homem e o mais simples portanto.

As linguagens de alto nível apresentam as seguintes vantagens:

- requerem **menos tempo para aprendê-las**. Isto determina que cada dia os novos programadores são habituados mais rapidamente ao ambiente de trabalho e que começam a produzir mais velozmente. Por outro lado, os usuários de computadores pessoais aprendem com rapidez a fazer seus próprios programas;
- **mais facilidade para eliminar erros** De fato, não é necessário recorrer aos estranhos truques que a linguagem Assembler requer, nem é tampouco indispensável conhecer a estrutura do computador que está utilizando;
- **maior potência expressiva** que faz possível uma linguagem orientada para o problema e não para a máquina. Assim, se-

quências de operações como as da Figura 1 são recompiladas em uma só instrução, o que dá lugar simultaneamente a que diminuam as probabilidades de erro;

- **documentação mais simples** Um programa escrito em Assembler requer, por regra geral, a presença de um manual que explique seu funcionamento; ao contrário, se o programa está bem escrito em uma linguagem de alto nível, permite que sejam subentendidos muitos de seus mecanismos de funcionamento;
- **relativa independência do computador** Hoje em dia muitos dos programas de alto nível funcionam em todos os computadores disponíveis. Assim, se escrever um programa em Pascal para o IBM-PC terá a possibilidade, com poucas variações formais, de adaptá-lo para o Apple, que utiliza uma CPU totalmente diferente. Este fator é uma grande vantagem para a economia dos programas. Não poderá, ao contrário, fazer o mesmo com um programa escrito em Assembler, já que esta linguagem reproduz fielmente as características da CPU para a qual foi escrito;
- **maior velocidade na hora de confeccionar um programa** Como o número de instruções que se tem de escrever é muito inferior e tem mais estruturação e linearidade no programa, fica óbvio que o tempo empregado é reduzido consideravelmente.

Os programas escritos em uma linguagem de alto nível tem de ser tratados por outros, chamados compiladores, que são encarregados de traduzi-los ao código que a CPU do computador em questão utiliza. A principal falha que apresentam estes compiladores, é que, ao ter que resolver situações de caráter geral, o código produzido por eles não está otimizado. Portanto, um bom programador em Assembler é capaz de resolver um mesmo problema com um programa mais veloz e que ocupa menos espaço na memória. Isto, sem dúvida, é certo: um compilador produz um código executável que ocupa, em média, de duas a três vezes o espaço estritamente necessário. Mas devemos tomar em consideração também dois fatores importantes:

- quando um programa é muito vasto e complexo fica muito difícil que inclusive um bom programador em Assembler consiga dominar a situação e otimize o programa;
- para programas muito amplos o custo, tanto de pessoal como de tempo empregado, fica tão alto na versão Assembler

que é preferível a desvantagem oferecida pelo compilador.

Para programas verdadeiramente complicados, como pode ser o caso de um sistema operacional, hoje em dia se prefere escrever todo em uma linguagem de alto nível para poder prová-lo em um tempo razoável. A experiência indica que tão somente 20% do programa é crítico sob o ponto de vista da velocidade de execução. Uma vez localizados estes pontos críticos o que se faz é reescrever somente estas partes em Assembler, deixando o resto escrito em linguagem de alto nível.

Os compiladores disponíveis na atualidade para os computadores pessoais são capazes de traduzir quase todas as linguagens utilizadas nos grandes computadores. Isto foi conseguido graças à enorme difusão do sistema operacional CP/M primeiro e o MS-DOS depois, que possuem quase todas as linguagens mais extensas.

As mais conhecidas são:

- FORTRAN: é a linguagem de alto nível de grande utilização mais antiga;
- COBOL: é a mais utilizada no campo da gestão (caracterizada por um excesso de palavreado tal que os programas escritos nesta linguagem batem todos os recordes em comprimento...);
- PASCAL: a primeira linguagem inspirada nos princípios da programação estruturada moderna é, com toda segurança, uma das mais compreensíveis e autodocumentadas entre as existentes hoje em dia;
- C: verdadeira linguagem para a programação estruturada. É "amistosa", flexível, muito eficiente, potente e portátil (intercambiável). Permite aproximar-se com simplicidade, quando necessitamos, ao Assembler. Nela está escrito grande parte do UNIX, assim como compiladores e intérpretes de muitas outras linguagens (inclusive a animação das cenas de "O retorno de Jedi" foi programada em C).

Existem outras linguagens que começam a ser difundidas entre os computadores pessoais. Um dos exemplos mais interessantes é a LISP. Está disponível para o Apple e em sistema operacional CP/M e é o primeiro escalão para as aplicações de Inteligência Artificial.

Tratamento à parte merece o BASIC, que é, com toda segurança, entre as linguagens de alto nível, a mais fácil de aprender:

não é casualidade que esteja implementada em todos os computadores pessoais.

Mais possui também desvantagens: a falta de estruturação lógica que a caracteriza a faz incômoda sob o ponto de vista de tratamento dos dados e da compreensão dos programas. De fato, na criação de programas muito complexos a utilização freqüente de saltos incondicionais (GOTO) e a impossibilidade de usar uma estrutura adequada torna freqüentemente, difícil de decifrá-los. Para programas simples e imediatos continua sendo, sem dúvida alguma, a linguagem mais eficaz.

Uma característica típica do BASIC (utilizada às vezes por outras linguagens de alto nível) é a de não ser compilada mais interpretada. **Um intérprete é um programa que, residindo na memória, traduz diretamente as instruções da linguagem em código máquina segundo vai executando o programa.**

O intérprete, ao ter que efetuar cada vez que executa uma instrução sua tradução (enquanto que os compiladores o fazem de uma vez por todas) é, por regra geral, mais lento que um compilador, mas apresenta indubitáveis vantagens. Em primeiro lugar, é de uso muito imediato: o usuário nem se dá conta da tradução nem está ocupado nelas nas vezes das complexas operações de compilação ou outras tarefas que este requer freqüentemente (linker, loader, etc.). Além disso, ao não produzir código executável do programa, não necessita amplas áreas de memória para armazená-lo. Por este motivo são preferidas as linguagens interpretadas, como o BASIC, para implementações em computadores pessoais que, em suas versões mais econômicas, não tem precisamente uma abundante memória.

Ao estar muitas vezes memorizado sobre uma EPROM, além disso, o BASIC interpretado não necessita discos nem fitas, mas somente memória suficiente para armazenar o programa editado pelo usuário. Como não é produzido código-máquina, toda a memória está destinada unicamente a conter o texto teclado pelo usuário e os valores das variáveis do programa.

Carregadores (loaders) e instrumentos de comprovação

Como vimos anteriormente, os programas escritos pelo usuário são traduzidos em linguagem máquina por Assembler ou compiladores. Ao contrário, o "carregador" (loader) é um programa complexo que toma textos em código de máquina (ou, com mais propriedade, código objeto) e os prepara para ser executados no computador.

- As funções do carregador são fundamentalmente quatro:
- reservar espaço de memória para o programa (recolocação);
 - conectar entre si todos os módulos de programas diferentes (linking);
 - voltar a calcular todas as direções não absolutas, isto é, aquelas que dependem da colocação do programa (recolocação);
 - carregar o código de execução e os dados em memória.

Na atualidade, estas funções, tanto nos grandes computadores como nos pessoais, são feitas em duas fases: a fase de enlace (link) e a de execução.

Vamos examinar, a modo de exemplo, de que maneira funciona neste aspecto o CP/M, que opera de um modo muito parecido ao de outros muitos sistemas operacionais.

Quando o operador escreve programas em Assembler ou em linguagens de alto nível não se preocupa em atribuir direções absolutas, mas se limita a colocar algumas etiquetas (label) às quais fará referência nos saltos, nas chamadas subrotinas ou no emprego das tabelas de dados, isto é, utiliza algumas referências "relativas". Além disso, é comum dividir o programa em vários módulos ou utilizar partes do programa de biblioteca (comentaremos isto em um próximo item), pelo qual será necessário depois fazer as oportunas conexões entre as distintas partes do programa.

O que se ocupa de efetuar estas conexões é o **programa enlaçador (linker)**. Ao especificar no comando LINK todos os nomes dos programas, já compilados ou "assemblados", que se quer utilizar, estes são unidos e é produzido um código, executável diretamente pela máquina. É o mesmo código objeto do qual falamos anteriormente, mas nele já haviam sido efetuadas as conexões entre os distintos programas, e as etiquetas que compiladores e "assembladores" haviam deixado, digamos assim, no ar são "materializadas".

O linker é ocupado, portanto, de organizar as direções do programa para a execução. Normalmente, o sistema operacional CP/M carrega o programa na direção 103 hexadecimal, mas também é possível especificar uma direção diferente para o programa em si (código executável) e para os dados.

Por meio de opções /P: direção e /D: dados o CP/M permite ao usuário que fixe a direção a partir da qual quer que o programa seja carregado. Mas terá que prestar muita atenção ao especificar estas direções para eliminar coisas que já existem (e ainda servem) ou carregar o programa em uma zona de memória que não existe... De fato, o linker não exerce controles de prote-

ção e continua com o processamento apesar de que tenham sido cometidos erros deste tipo.

Outra opção útil oferecida pelo CP/M é a lista de todas as referências globais dos programas: ao especificar /M o linker dará as direções de todos os símbolos que declaramos globais (como tabelas de dados, pontos iniciais de subrotinas e outros parecidos). Este fator é muito útil na fase de verificação do funcionamento de um programa, que trataremos mais adiante. Uma vez obtido o código executável se dá o comando de execução (/G no CP/M) e o sistema operacional é encarregado de carregar o programa em memória e de executá-lo.

Aqueles que já tentaram escrever um programa sabem que na primeira execução, e quase com total segurança..., não funcionará. A maneira de sanar este pesado e inevitável inconveniente é a de repassar com toda paciência o programa, tratando de descobrir o que é que falha.

Se os programas são muito longos e complexos, este trabalho é particularmente difícil. Alguns sistemas operacionais proporcionam um programa chamado **debugger** (corretor, depurador). **Graças a ele é possível seguir o programa passo a passo**, executando uma instrução cada vez, o que facilita a localização do ponto de funcionamento anômalo.

Nos primeiros sistemas operacionais e computadores pessoais mais simples, o único meio de que se dispunha era examinar o conteúdo da memória; o usuário podia bloquear a execução do programa (por meio, por exemplo, de uma tecla de interrupção) e, em seguida, visualizava o conteúdo da memória do computador (fazia um "dump") para tratar de descobrir o ocorrido. Com isto se compreende a grande utilidade da opção vista do linker, que permite saber as direções dos símbolos do programa. De fato, se utilizamos algumas variáveis (isto é, células de memória cujo conteúdo varia durante a execução do programa) vendo seus valores será possível ter uma idéia acerca do funcionamento do programa. Um exemplo pode ser uma instrução que escreve um "1" lógico em uma célula ao início de certa parte do programa, ou diferentes valores segundo o percurso efetuado em um salto condicional. Graças à execução passo a passo, o usuário será capaz de comprovar o percurso efetuado pelo programa.

Outro programa de suporte é o "traçador" (tracer); com ele é simulada a execução das instruções, visualizando sobre a marcha o conteúdo dos registros do processador. Como na realidade tal simulação é freqüentemente mais lenta que a execução do programa, em muitos casos é oferecido um cartão adicional que exer-

ce a função “tracer” com um processador dedicado. Este método é bastante interessante, já que o usuário, ao poder seguir simultaneamente os dados da tela e seu programa, distinguirá no instante uma possível anomalia. De todos os modos, os mais modernos sistemas operacionais oferecem alguns **depuradores ou debuggers interativos** com os quais o usuário pode manipular seu programa, pará-lo e colocá-lo em andamento de novo, examinar estruturas de dados em memória e os registros da CPU e modificar seu conteúdo. Se conhecer a direção da instrução na qual se supõe que existe algum mal funcionamento, pode introduzir (ao finalizar algum passo crítico) um “ponto de ruptura” (break point), isto é, um ponto no qual o programa cessa sua execução e cede o controle ao operador, que poderá realizar todas as operações necessárias na memória e nos registros da CPU.

Um elemento que é muito útil mas que, no momento, está somente disponível para grandes computadores, é o debugger **simbólico** que executa todas as funções vistas até o momento, mas operando diretamente sobre os símbolos e não sobre as direções. Isto torna-se indispensável se desejamos comprovar de uma forma rápida e eficaz o funcionamento dos programas escritos em uma linguagem de alto nível, na qual, naturalmente, tudo é simbólico e, em consequência, é impossível introduzir “break points”, já que cada instrução é, em geral, traduzida mediante várias palavras da linguagem máquina.

Queremos fazer notar que as linguagens interpretadas consentem uma discreta depuração (debugging) com muita facilidade. Por exemplo, em BASIC, ao executar um programa é obtida uma mensagem típica (SYNTAX ERROR) cada vez que o intérprete tropeça com uma linha sintaticamente incorreta, parando nessa linha. Assim é possível corrigi-la no momento e fazer partir novamente o programa (com o comando CONT). Em relação com os erros de natureza lógica é boa tática intercalar, durante a fase comprovação, linhas de STOP que contenham ao menos algumas instruções para a impressão dos dados críticos (quando o programa para será possível revisar tudo o que desejarmos e continuar novamente com o comando CONT). Tudo isto sem ter que comprar potentes mas custosos debuggers e sem arriscar-nos a ficar loucos com novas complicações, aborrecidas e complexas. Somente terá que prestar atenção para evitar que alguma linha errônea não seja descoberta, o que pode acontecer se pertence a um ciclo que é executado somente em contadas ocasiões (os compiladores verificam todas as instruções em cada intervenção).

DDT, o depurador do CP/M

Vamos ver agora um dos depuradores mais utilizados nos computadores pessoais: o DDT (Dynamic Debugging Tool), proporcionado pelo CP/M (existe outro parecido para o MS-DOS).

Com o comando DDT <nome do programa> é carregado em memória o programa que tem que depurar e é colocado sob o controle do DDT, que se apresenta com "...". **Com ele é possível escolher entre dois modos de funcionamento: o direto e o passo a passo (trace).**

No modo direto são especificados um ou mais pontos de parada (break point) e, por meio do comando GO, é iniciada a execução do programa. Este será detido quando encontrar o break point; então poderemos examinar o estado dos registros e modificar seu conteúdo com o comando "X" (eXamine). Também pode ser comprovado o conteúdo da memória com o comando D (Display), que a apresenta em linhas de 16 bytes cada vez. Outro comando útil é o MOVE: desloca um bloco, do comprimento que especificarmos, de uma direção a outra. Ao contrário, com o comando FILL podemos escrever uma constante de um byte em toda a área de memória; é muito útil, por exemplo, para colocar a zero uma tabela.

O modo passo a passo (trace) faz proceder a execução do programa instrução por instrução, visualizando a cada passo o estado dos registros da CPU, ainda que, naturalmente, também é possível executar certo número de instruções de uma maneira automática e controlar o estado dos registros depois.

Outro comando que é muito útil é o "L" (List), já que, permite desensamblar até doze linhas de programa. Além disso, visualiza os mnemotécnicos das instruções Assembler, permitindo ao programador conhecer qual o significado têm esses números que ocupam as células da memória que estão sendo examinadas.

Se o usuário quer introduzir outro código no programa pode, por meio do comando "A" (Assembler), incorporar instruções já codificadas em linguagem máquina, na direção desejada. Posteriormente poderá comprovar a exatidão da correção com o comando L.

Como vimos, este instrumento de comprovação é muito útil e versátil. Por desgraça, o DDT também tem um inconveniente, que é o de ter sido escrito para o microprocessador 8080 Intel e, portanto somente funciona com o código deste (ou com outro compatível). É óbvio que, se mudar de processador, fará falta outro debugger, como o citado anteriormente do MS-DOS.

Bibliotecas de programas

Existem alguns suportes oferecidos pelo sistema operacional que tratam de evitar que muitos usuários percam tempo inutilmente. De fato, muitos de nós nos esforçamos em escrever programas que façam determinadas funções, que são de uso muito geral. Para evitar isso existem certos pacotes (package) de software que permitem que o usuário utilize suas funções simplesmente com uma chamada desde seu programa. Seu conjunto é o que se denomina "biblioteca" de programas.

Existem disponíveis, em geral, muitíssimas funções matemáticas para as linguagens Assembler que, por si só, não as possuem: multiplicação, divisão, raiz quadrada, elevação à potência, etc. Também são disponíveis pequenos programas que permitem utilizar muitas das funções de uso interno do sistema operacional: acesso direto a disco, manejo de I/O...

Naturalmente, as linguagens de alto nível dispõem, por norma geral, das funções aritméticas elementares, mas às vezes, sobretudo para usos científicos, são requeridas funções analíticas que podem ser muito complexas. Assim, existem funções trigonométricas (seno, cosseno, etc.), logarítmicas, para o cálculo matricial, cálculo vetorial, resolução de sistemas de equações de várias incógnitas, funções estatísticas das mais raras, etc.

Uma das funções mais solicitadas para cálculos científicos e financeiros é o uso de operações com ponto flutuante. Existem pacotes de aritmética com cálculos de ponto flutuante muito eficazes, mas bastante lentos. Por este motivo costuma ser incorporado um módulo hardware, especializado no processamento direto destas funções freqüentemente por meio de um co-processador específico que é incorporado à CPU (este é o caso do co-processador Intel 8087, opcional no PC IBM). E tudo isto sem ter necessidade de programas de biblioteca que simulem seu funcionamento: a economia de tempo é considerável.

Um campo no qual o usuário de computadores pessoais começa a encontrar as bibliotecas de serviço muito úteis é o dos gráficos. Já existem disponíveis programas que desenharam curvas, retas, figuras e outras funções sem que o usuário tenha que fazer nenhum tipo de cálculo, somente deve indicar o que deseja.

Outro setor muito solicitado é o do Word Processing, ou tratamento (processamento ou composição) de textos. É muito útil poder dispor de um computador para compor cartas, artigos e livros. Os textos são compostos como se de programas comuns se

tratasse por meio do teclado; o processador de textos é encarregado em seguida de enquadrar o escrito, organizando as margens direita esquerda, superior e inferior, o espaço entre as linhas, o número de caracteres por linha... Os mais sofisticados chegam a gerar inclusive o índice de todos os parágrafos, o índice analítico das palavras, a lista das figuras e tabelas, em resumo, a estrutura completa de um livro.

Conclusão

Além do visto, fica ainda por comentar o vasto mundo dos pacotes de aplicações, tanto para a automatização e escritórios e gestão de pessoal (desde as tabelas eletrônicas às bases de dados pessoais), para comunicação, assim como os (tão na moda) pacotes integrados (como o célebre Lotus 1-2-3), etc., todos eles muito mais difundidos nos computadores pessoais que nos de grandes dimensões.

O fato de nos fixarmos nos processadores de texto se deve, em primeiro lugar, a motivos de espaço e, em segundo, que estes temas serão tratados com maior amplitude em posteriores volumes da coleção. Talvez a escolha tenha dependido do fato de que o Word Processador é descendente dos editores, presentes originalmente nos sistemas operacionais mais modernos dos computadores, tanto grandes como médios. Existe, portanto, uma continuidade entre a macro e a microinformática. Este foi o fator que nos orientou a tratar, em um livro de divulgação, as peculiaridades de vários sistemas operacionais, colocando em evidência suas diferentes características. Algumas destas não são encontradas presente nos computadores pessoais, mas tem que ter em mente que a próxima geração de computadores pessoais está adotando CPUs cada vez mais potentes e com espaços de memória muito mais amplos, o que faz possível a comercialização de muitas destas sofisticações também em microcomputadores. Assim mesmo é importante destacar que:

- algumas destas funções não têm grande utilidade prática, ao menos ao nível da microinformática individual;
- a exigência de uma aproximação cada vez mais "amigável" (user friendly) fortemente para realizar interfaces destas características amigáveis muito mais em computadores pessoais que nos grandes sistemas; a nova geração de computadores pessoais, com tecnologia avançada, acentua esta positiva tendência (enquanto que o próprio Unix apresenta fre-

qüentemente uma semântica de comandos muito menos simples e clara que a que está em uso em computadores pessoais).

Esperamos que com este livro tenhamos esclarecido um pouco o que são e para que servem esses grandes desconhecidos que são os sistemas operacionais e todos os programas de suporte, contribuindo assim também para tornar mais claro o entendimento e o diálogo com os computadores.



uitas vezes poderemos observar que, na hora de falar de um computador, não aparece por nenhum lado a menção do sistema operacional. Isto supõe que não tem nenhuma importância ou então é uma falha ou esquecimento do vendedor.

Realmente nem todo o mundo sabe o que é e para que serve um sistema operacional e, contudo, sua influência é tão grande que pode fazer com que programas existentes no mercado (de jogos ou de aplicações) não possam ser executados em seu computador por dispor de um sistema operacional distinto ao que necessita o programa para funcionar corretamente.

O mesmo poderia ser dito de elementos como o editor de textos, compilador, carregador, enlaçador.... programas de suporte que, junto ao sistema operacional e as linguagens de programação, constituem o software de base ou do sistema.