Rodnay ZAKS

O MANUAL DE CP/M

Incluindo MP/M

SYBEX

EDITORA CAMPUS

ZAKS O MANUAL DE Incluindo MP/M

Títulos de Computação da Editora Campus

ADMINISTRAÇÃO DE SISTEMAS DE INFORMAÇÃO – *P. Ein-Dor & E. Segev*

ANÁLISE DE SISTEMAS PARA SISTEMA DE INFORMAÇÃO BASEADO EM COMPUTADOR – *J. C. Wetherbe*

APLICAÇÕES DE MICROPROCESSADORES – *J. A. Kuecken*

APLICALC: UM SOFTWARE EDUCACIONAL, PESSOAL E PROFISSIONAL EM BASIC – *E. A. Meili*

BASIC BÁSICO, 4ª ed. – *J. C. Pereira Filho*

BASIC PARA APLIÇAÇÕES COMERCIAIS – *D. Hergert*

BASIC PARA MICROS PESSOAIS, 2ª ed. – *J. C. Pereira Filho*

BRINCANDO COM O COMPUTADOR – *J. A. Moreira*

COBOL PARA ESTUDANTES, 4ª ed. – *A. Parkin*

COMO LIDAR COM O COMPUTADOR – *H. C. Lucas Jr.*

COMPUTADORES BRASILEIROS – INDÚSTRIA, TECNOLOGIA E DEPENDÊNCIA – *P. B. Tigre*

COMPUTADORES PARA USUÁRIOS – *J. C. Pereira Filho, Coordenador*

Vol. I – Aplicação de Computadores
Vol. II – Equipamentos e Sistemas de Computação
Vol. III – Programas e Programação de Computadores
Vol. IV – Seleção de Sistemas de Computação

A CONSTRUÇÃO DE UM COMPILADOR – *V. W. Setzer & I. S. H. Melo*

CRIANÇA TAMBÉM FAZ PROGRAMAS, 2ª ed. – *J. A. Moreira*

DOCUMENTAÇÃO DE SOFTWARE – *J. D. Lomax*

ESPECIFICAÇÃO DE SISTEMAS – *S. J. Waters*

ESTRUTURAS DE DADOS, 2ª ed. – *P. S. Veloso et al.*

FUNDAMENTOS DE ESTRUTURA DE DADOS – *E. Horowitz & S. Sahni*

FUNDAMENTOS DE PROCESSAMENTO DE DADOS – *W. T. Price*

GERÊNCIA DE BASES DE DADOS PARA MICROCOMPUTADORES – *E. G. Brooner*

GRAFOS E ALGORITMOS COMPUTACIONAIS – *J. L. Szwarcfiter*

GUIA DE LINGUAGENS DE COMPUTADORES – *H. L. Helms Jr.*

GUIA PARA PROGRAMADORES, 2ª ed. – *M. Bohl*

IMPLANTAÇÃO DE MICROS E MINICOMPUTADORES COMERCIAIS – *P. A. Knight*

INTRODUÇÃO À CIÊNCIA DA COMPUTAÇÃO COM WATFIV E FORTRAN, 2ª ed. – *S. E. R. Carvalho*

INTRODUÇÃO À PROGRAMAÇÃO COM PASCAL, 2ª ed. – *S. E. R. Carvalho*

INTRODUÇÃO À PROGRAMAÇÃO DE COMPUTADORES, 3ª ed. – *H. V. R. Corrêa Silva et al.*

INTRODUÇÃO À PROGRAMAÇÃO FORTRAN – *J. C. Pereira Filho*

INTRODUÇÃO À SEGURANÇA DO COMPUTADOR – *M. B. Wood*

INTRODUÇÃO A SISTEMAS DE BANCOS DE DADOS – *C. J. Date*

INTRODUÇÃO AO PROCESSAMENTO DE TEXTOS – *G. L. Simons*

INTRODUÇÃO AO VISICALC, 2ª ed. – *E. A. Garbin*

JCL/SISTEMA 370, 3ª ed. – *G. D. Brown*

LCP – LÓGICA DE CONSTRUÇÃO DE PROGRAMAS, 2ª ed. – *J. D. Warnier*

LCS – LÓGICA DE CONSTRUÇÃO DE SISTEMAS – *J. D. Warnier*

LINGUAGEM DE PROGRAMAÇÃO ALGOL, 2ª ed. – *L. M. Segre*

LINGUAGEM PASCAL – *V. L. Strube de Lima*

MANUAL DE COBOL ESTRUTURADO – *D. D. McCracken*

MICROCOMPUTADORES PARA APLICAÇÕES COMERCIAIS – *W. Barden Jr.*

MICROPROCESSADORES/MICROCOMPUTADORES – *A. J. Khambata*
Vol. I – Arquitetura
Vol. II – Software e Sistemas

1001 APLICAÇÕES PARA O SEU COMPUTADOR PESSOAL – *M. Sawusch*

ORGANIZAÇÃO DE BANCOS DE DADOS, 4ª ed. – *A. L. Furtado e C. S. Santos*

PRINCÍPIOS DE SISTEMAS OPERACIONAIS, 3ª ed. – *C. C. Guimarães*

PROGRAMAÇÃO EM ASSEMBLER E LINGUAGEM DE MÁQUINA – *D. C. Alexander*

PROGRAMAÇÃO SISTEMÁTICA EM PASCAL, 2ª ed. – *N. Wirth*

RPG-II – *J. C. Pereira Filho*

REDES DE COMPUTADORES – ASPECTOS TÉCNICOS E OPERACIONAIS – *D. A. Menascé e D. Schwabe*

O SEU COMPUTADOR PESSOAL – *M. Waite e M. Pardee*

SISTEMAS DE VIDEOCASSETE: TEORIA E MANUTENÇÃO – *G. P. McGinty*

SISTEMAS OPERACIONAIS PARA MICROCOMPUTADORES – *M. Dahmke*

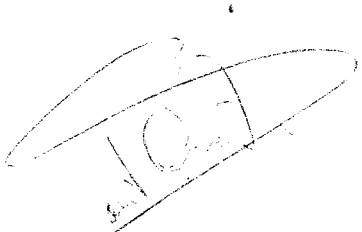
TÉCNICAS E PRÁTICA DE PROGRAMAÇÃO – *A. Chantler*

TECNOLOGIA DA INFORMAÇÃO: UM GUIA PARA EMPRESAS, GERENTES E ADMINISTRADORES – *J. Eaton e J. Smithers*

USANDO CP/M: UM GUIA EM ENSINO PROGRAMADO – *J. N. Fernandez & R. Ashley*

MICROINFORMÁTICA INTERNACIONAL

Uma publicação bimestral sobre Segurança e Prevenção de Fraudes em Computadores, Aplicação de Microprocessadores e Microcomputadores e Microeletrônica.
Assinaturas: 10 ORTN's



Procure nossas publicações nas boas livrarias ou comunique-se diretamente com:

EDITORA CAMPUS LTDA.
Livros Científicos e Técnicos
Rua Barão de Itapagipe, 55
20261 Rio de Janeiro – RJ – Brasil
Telefone: (021) 284 8443

Atendemos também pelo reembolso postal.

Rodnay ZAKS

O MANUAL DE CP/M

Incluindo MP/M

Tradução
Ricardo Reinprecht

ED. CAMPUS LTDA.

Capa
Otavio Studart

Diagramação, composição, paginação e revisão
Editora Campus Ltda.
Rua Barão de Itapagipe 55
Tel.: (021) 284-8443
20261 Rio de Janeiro RJ Brasil
Endereço telegráfico: CAMPUSRIO

ISBN 85-7001-188-1
(Edição original: ISBN 0-89588-048-2 SYBEX, INC. USA.)

Ficha Catalográfica
CIP-Brasil. Catalogação-na-fonte
Sindicato Nacional dos Editores de Livros, RJ.

Z25m

Zaks, Rodnay.
O Manual de CP/M : incluindo MP/M / Rodnay Zaks ; tradução [de] Ricardo Reinprecht. – Rio de Janeiro : Campus, 1984.

Tradução de: The CP/M handbook with MP/M.
Apêndices.
ISBN 85-7001-188-1

1. CP/M (Sistema para microcomputadores) 2. MP/M (Sistema para microcomputadores) 3. Sistemas operacionais (Microcomputadores) I. Título

84-0401 CDD – 001.642

Agradecimentos

Desejo agradecer as contribuições de muitas pessoas que me forneceram ajuda e sugestões valiosas no sentido de aperfeiçoar esta obra, no que se refere à sua abrangência. Tony Bove redigiu muitas das descrições iniciais dos comandos. David Haverty, do Centro de Computação de Berkeley, e Bruce Chichow apresentaram valiosos comentários que auxiliaram o aperfeiçoamento deste manual. Dorothy Kildall, da Digital Research, apoiou consistentemente esse esforço ao fornecer informações recentes a respeito de novos desenvolvimentos. Naturalmente, o autor é o responsável por quaisquer imprecisões que o livro contenha, apesar do esforço feito para eliminá-las. O autor receberá, agradecido, quaisquer sugestões posteriores de usuários do CP/M, que possam servir para o aperfeiçoamento de sua obra.

Sumário

Ilustrações

Prefácio

O propósito deste livro é ensinar a usar o CP/M e seus recursos. Não se pressupõe nenhum conhecimento prévio a respeito de computadores. No entanto, o que constitui um pré-requisito indispensável é que se tenha acesso a um sistema de computadores equipados com CP/M.

O CP/M tornou-se o sistema operacional padrão para microcomputadores. A maioria dos usuários de sistemas baseados em microcomputadores irá, um dia, utilizar o CP/M. Dependendo dos programas de aplicação que executam no computador, irão usar parte ou todos os recursos fornecidos pelo CP/M. Por exemplo, um funcionário encarregado da entrada de dados, que digita os dados em um programa de contas a receber, normalmente só necessitará saber como ativar o programa de contas a receber e solucionar os erros. Por outro lado, um programador com experiência anterior pode querer instalar um novo programa permanente no microcomputador ou executar funções sofisticadas de depuração em arquivos. Este livro foi estruturado para atender a esta ampla variedade de necessidades.

O Capítulo 1 apresenta uma introdução ao CP/M e mostra como ligar o computador e executar todas as operações usuais em arquivos, incluindo a duplicação de disquetes. Depois de ler o Capítulo 1, o leitor aprenderá a operar o seu microcomputador equipado com CP/M e a executar as seguintes funções: criar um arquivo, copiar um arquivo, lidar com disquetes, copiar disquetes, e ainda utilizar vários comandos importantes operando com arquivos. Esse conhecimento será o suficiente para permitir executar, com segurança, os mais conhecidos programas de aplicação. Provavelmente o aprendiz se surpreenderá com o pouco tempo necessário para ficar apto a utilizar o computador com o CP/M.

Depois de aprender os fundamentos básicos do CP/M, é provável que o leitor queira conhecer mais alguma coisa a respeito. O Capítulo 2 faz referência ao CP/M, a ser lido e depois consultado quando se desejar uma informação específica. Apresenta uma descrição geral e compreensiva de todos os comandos CP/M, com exceção do PIP, que é descrito no Capítulo 3. Embora não seja necessário à maioria dos usuários compreender todas as opções disponíveis no CP/M, um conhecimento genérico irá melhorar a eficácia de qualquer usuário do CP/M.

Uma compreensão profunda do programa de transferência de arquivos PIP é indispensável ao usuário com prática de CP/M. O Capítulo 3 descreve o PIP, com todos os detalhes, ensina a intercalar arquivos, a fazer uma listagem de múltiplos arquivos na impressora, e a usar os numerosos recursos adicionais disponíveis.

O Capítulo 4 permite a participação de uma sessão exemplo com o programa editor, 'ED'. O ED é um potente programa de processamento de textos que pode ser usado para criar ou manipular arquivos de texto de acordo com sua conveniência. Embora o ED seja complexo, é relativamente fácil de aprender.

A essa altura do livro, o leitor terá aprendido tudo, detalhadamente, a respeito das capacidades do CP/M, e então poderá estar interessado em saber como opera o CP/M. O Capítulo 5 leva ao interior do CP/M e explica a sua operação interna. Esse conhecimento não é necessário para se usar o CP/M, mas é exigido se houver o desejo de modificá-lo.

O Capítulo 6 utiliza um formato conveniente para resumir todos os comandos e sím-

bolos utilizados pelo CP/M (detalhados no Capítulo 2). O Capítulo ó serve como fonte de consulta essencial para o usuário do CP/M.

O Capítulo 7 apresenta uma importante coleção de "dicas" práticas. Estando o leitor familiarizado com o CP/M, e na fase de utilização freqüente do computador, existem diretrizes importantes que devem ser seguidas. O capítulo 7 oferece recomendações sobre a maneira de lidar com problemas e de evitar os problemas práticos que podem surgir quando se emprega o CP/M. Este capítulo deve ser considerado leitura essencial para todos os usuários.

Finalmente, o Capítulo 8 apresenta uma breve perspectiva histórica do CP/M e do seu futuro.

Muitas tabelas de consulta úteis são apresentadas nos apêndices, as quais devem ser consultadas depois da leitura do livro. Essas tabelas incluem os códigos binários usuais, mensagens de erro, símbolos e comandos oferecidos pelo CP/M, ED e PIP.

O CP/M foi elaborado para facilitar o uso dos microcomputadores. *O Manual CP/M com MP/M* deverá tornar simples o uso do CP/M.

Esta obra abrange o CP/M e suas várias versões, incluindo o CP/M 1.4 e o CP/M 2.2, e o novo sistema operacional de múltiplos usuários denominado MP/M. É também aplicável a sistemas operacionais compatíveis com CP/M, como, por exemplo, o CDOS da Cromemco.

Nota:

Para as operações de entrada ou requisição de dados pelo teclado, foi utilizada a seguinte convenção:

TECLAR – Para comandos de tecla *única* ou *simultânea*
Exemplo: ↵ ou CTRL C

DIGITAR – Para comandos que utilizam mais de uma tecla
Exemplo: MOVCPM ↵

Capítulo 1

Introdução ao CP/M - MP/M

INTRODUÇÃO

O objetivo deste capítulo é ensinar-lhe a executar operações fundamentais no seu microcomputador utilizando o CP/M. Não há necessidade de nenhum conhecimento prévio a respeito de computadores. O leitor irá aprender, inicialmente, o vocabulário e as definições relacionadas com a operação do computador. Em seguida aprenderá a ligar o computador, inserir seu *Disquete do Sistema*, e chamar o CP/M; aprenderá a respeito de *arquivos*, a criá-los, nomeá-los e fazer cópias de um arquivo ou de um disquete completo. Depois aprenderá a usar o teclado, assim como a tela e a impressora para manipular, apresentar ou imprimir o conteúdo de um arquivo. Quando chegar ao fim deste capítulo, terá aprendido a utilizar todos os comandos mais importantes de CP/M.

DEFINIÇÕES BÁSICAS

A Figura 1.1 mostra um sistema típico de computadores. Este sistema inclui o computador, as unidades de discos, a impressora e o terminal de vídeo. Para usar o computador, a pessoa deverá estar sentada diante do terminal e digitar no teclado. As mensagens serão, então, apresentadas na tela do terminal. Mediante o uso da impressora, a pessoa também será capaz de imprimir textos se o desejar. Os programas a serem executados pelo computador estarão armazenados nos disquetes inseridos em uma das unidades de discos.

Neste capítulo o leitor aprenderá, passo a passo, a executar todas as operações necessárias para usar o seu microcomputador.

SISTEMA DE COMPUTADOR

Um sistema de computador possui dois tipos de componentes: *o hardware* (equipamento físico) e *o software* (suporte lógico ou de programação). O hardware refere-se aos componentes físicos de um sistema (pinos, porcas, fios etc.). O software refere-se aos programas e aos arquivos.

Elementos do hardware

Os elementos do hardware de um microcomputador típico (o computador, o teclado e o terminal de vídeo [CRT], a impressora e um ou mais *drives*-unidades de discos) são apresentados na Figura 1.1. Os elementos adicionais do hardware, como fitas magnéticas (gravadores) e outros dispositivos (microfone, leitora de cartões etc.) também podem ser acrescentados a um microcomputador.

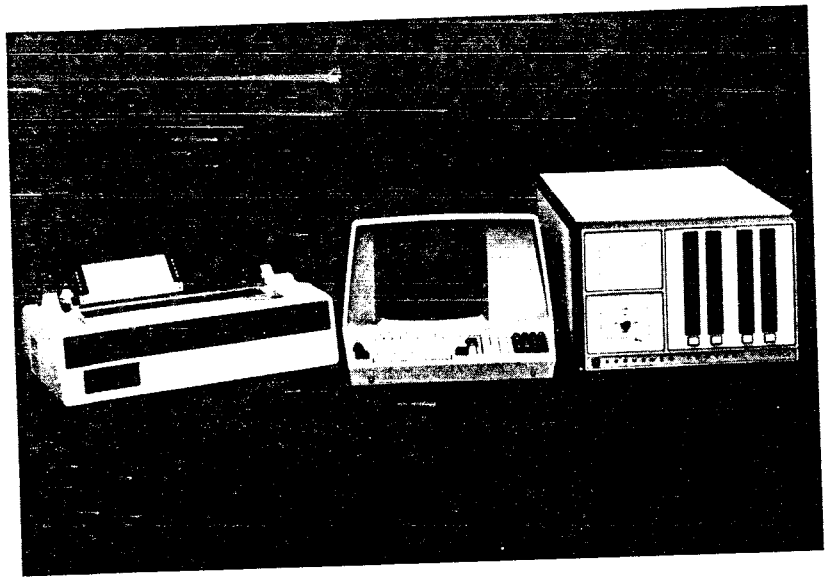


Figura 1.1: *Um microcomputador típico*

O computador

O computador propriamente dito é, caracteristicamente, instalado em uma espécie de gabinete. Considerando que a maioria das aplicações de CP/M exige, geralmente, uma grande quantidade de memória (48K ou 64K) e dois drives de discos, muitos fabricantes instalam o computador e dois drives de discos na mesma caixa. Foi o que apresentamos na Figura 1.1. Com o TRS80, o Exidy, e um mais antigo, o SOL, o microcomputador é instalado na mesma caixa como o teclado.

A função do computador é manipular informações. Sua operação é controlada por *programas* colocados na *memória* do computador. O objetivo da memória do computador é armazenar informações, quer se trate de programas ou de dados. Seu tamanho é medido em palavras (bytes de 8 bits para um microcomputador de 8 bits), em múltiplos 1K, sendo 1K = 1024. Tamanhos típicos são 16K, 32K, 48K e 64K.

Com a tecnologia moderna, a maior parte da memória do computador é volátil e seu conteúdo irá desaparecer quando o computador for desligado. Em outras palavras, toda vez que um programa tem de ser executado, é necessário que seja passado do disco para a memória do computador. Essa operação é executada automaticamente pelo CP/M.

Os discos

Já que a memória do computador (denominada "RAM" para memória de acesso aleatório) é volátil, isto é, não retém as informações quando a energia elétrica deixa de ser conectada, cada computador exige um dispositivo de armazenamento permanente. Discos magnéticos, flexíveis ou rígidos, são utilizados em pequenos computadores para esse objetivo. Todas as informações podem ser preservadas nesse meio, incluindo os programas, arquivos (coleções de textos ou dados) e uma cópia do próprio programa CP/M.

Terminal CRT (vídeo e teclado)

O terminal CRT consiste de uma combinação de um vídeo CRT (uma tela semelhante à de um aparelho televisor) e de um teclado. Representa o meio pelo qual a pessoa pode se comunicar diretamente com o microcomputador. O teclado é empregado pelo usuário para digitar caracteres que são interpretados pelo programa que está sendo executado pelo computador. A tecla CRT apresenta as informações ao usuário. Infelizmente, assim como a memória interna do computador, o CRT é volátil, isto é, a informação é representada visualmente na tela, por um período de tempo determinado, e depois desaparece.

Na maioria dos sistemas comerciais utiliza-se um terminal CRT clássico, que combina um teclado com um vídeo CRT. Nos casos em que o teclado já esteja incorporado no móvel do computador, acrescenta-se um monitor de vídeo separado (ou integrado).

A impressora

A impressora é um dispositivo para *cópias impressas.* O papel da impressora é fornecer impressão de qualquer informação solicitada pelo usuário. A impressora é utilizada para fazer uma listagem de programas e documentos.

Agora que já nos familiarizamos com os componentes de hardware de um sistema, passaremos à definição dos componentes de software.

Os componentes de software

O termo "componentes de software" refere-se ao programa (uma seqüência de instruções) e aos dados. Mais especificamente, um *programa* é uma seqüência de instruções que, uma vez colocadas na memória do computador, irá dirigi-lo para executar as ações específicas. Os *dados* são coleções de caracteres ou números manipulados pelos programas. Os programas e os dados são denominados, logicamente, *arquivos,* depois de o usuário lhes atribuir um nome. Mais adiante o leitor aprenderá a usar uma variedade de programas e a criar ou manipular tipos comuns de arquivos.

O CP/M em si é um programa especial, ou melhor, uma coleção de programas geralmente fornecida em um disquete. Os programas utilizados neste livro serão armazenados em disquetes.

Existem duas classes essenciais de software: software de sistema e o de aplicações. O *software de sistema* é o software geralmente fornecido junto com o microcomputador que é necessário para operá-lo. Inclui o CP/M assim como uma variedade de programas "utilitários", como o PIP e o ED, que serão descritos detalhadamente mais adiante.

O *software de aplicações* é uma coleção de programas que um usuário pode empregar para executar tarefas específicas. Exemplos de software de aplicações incluem um programa de mala direta, um programa de inventário, um programa de contabilidade, ou um programa de processamento de palavras.

Definição do CP/M e MP/M

CP/M é uma abreviatura de *Programa-Controle para Microprocessadores.* MP/M representa *Programa-Controle de Multiprogramação para Microprocessadores.* Tanto o CP/M como o MP/M são *sistemas operacionais.* O objetivo do CP/M ou de qualquer outro sistema operacional é executar comandos do usuário e permitir ao usuário utilizar, de acordo com sua conveniência, todos os recursos de hardware fornecidos pelo computador. Por exemplo, enviará o texto à impressora, lerá e processará informações enviadas pelo tecla-

do, e apresentará informações no CRT (vídeo). Além disso, o sistema operacional CP/M executará tarefas internas, como gerenciar o espaço do disco, ou gerenciar o espaço da memória do computador.

Uma vez instalado na memória do computador, o CP/M se torna uma parte integrante do sistema completo, e muitas vezes nos referimos a ele como "o sistema". (Cabe assinalar que na gíria dos computadores "o sistema" pode também ser usado para descrever o conjunto de componentes de hardware, isto é, o computador, a impressora, o CRT e os drives). Neste texto, quando nos referimos exclusivamente a programas, "o sistema" significa CP/M -- o sistema operacional.

Operação do sistema

A operação do sistema completo se tornará clara à medida que o utilizarmos. A função essencial do sistema operacional CP/M é a de permitir ao usuário utilizar, como lhe convier, os recursos do microcomputador. Logo após o computador ter sido ligado, o sistema operacional é colocado dentro da memória do computador e começa a supervisionar o teclado, na expectativa dos comandos. O usuário poderá, então, começar a dialogar com o CP/M e a ativar o programa de aplicações desejado. Quando um programa de aplicações termina, o CP/M volta a supervisionar e espera o próximo comando. Poderíamos ver o CP/M como um empregado onipresente pronto para obedecer a comandos e gerenciar os recursos do computador, desde que o usuário não esteja executando um programa de aplicações. Especificamente, uma vez que um programa de aplicações é executado (por exemplo, um programa de mala direta), esse programa se apodera da memória do computador e todos os diálogos posteriores serão com esse programa. Contudo, quando o programa de aplicações termina, o CP/M é novamente ativado, e está pronto para aceitar novos comandos.

Em resumo, o CP/M é uma coleção de programas que reside em um disquete chamado disquete do sistema. O *monitor residente* ou o *programa bootstrap* (existente em qualquer computador) geralmente irá carregá-lo automaticamente a partir do disquete, logo que o sistema seja ligado. (Ocasionalmente, torna-se necessário uma intervenção manual pelo usuário.)

O CP/M oferece comandos específicos para transferir informações entre os dispositivos conectados ao sistema de computadores, executando programas e manipulando convenientemente os arquivos. Como qualquer outro bom sistema operacional, o CP/M oferece ainda muitas características adicionais. As mais importantes serão descritas neste capítulo e descrições abrangentes de todas as características serão oferecidas nos capítulos seguintes.

CP/M, MP/M e outras versões

CP/M e MP/M

Várias versões do CP/M têm sido divulgadas sucessivamente. Este livro apresenta, inicialmente, os aspectos padronizados do CP/M até a versão 1.4, e depois assinala os aperfeiçoamentos disponíveis em versões posteriores, como a versão 2.2 e a versão 2.1 MP/M. Várias outras versões de CP/M também foram divulgadas por outros fabricantes como "aperfeiçoamentos de CP/M". Por exemplo, o CDOS da Cromemco é "compatível com o CP/M" e fornece recursos adicionais. Todos os aspectos do CP/M descritos neste livro são aplicáveis a essas versões. No caso do CDOS da Cromemco, são apresentados comentários específicos nas seções pertinentes.

A diferença essencial entre CP/M e MP/M reside no fato de que o CP/M foi elaborado para servir como um sistema operacional de um único usuário. O MP/M, por outro lado, é um sistema operacional de múltiplos usuários que permite a vários terminais serem utilizados simultaneamente em um microcomputador. O MP/M fornece todos os recursos do CP/M e outros mais. Os meios adicionais fornecidos pelo MP/M serão descritos sistematicamente em cada capítulo.

O CDOS da Cromemco

Tem-se divulgado que o CDOS da Cromemco é compatível com a versão 1.3 do CP/M. Em outras palavras, os comandos da versão 1.3 do CP/M estão embutidos no CDOS. Contudo, o inverso não é verdadeiro: programas que dependem das facilidades do CDOS podem não rodar com o CP/M. Além do mais, o CDOS fornece um certo número de facilidades adicionais em comparação com o CP/M. O CDOS utiliza um sistema de arquivos idêntico ao do CP/M, de forma que um disquete lido por CP/M também pode ser lido por CDOS. Existem diferenças de menor importância: o *prompt* do sistema utilizado pelo CDOS é um ponto ao invés do sinal >. O programa especial CONPROC ("Console Processor") também deve estar presente em todos os disquetes do sistema. No CDOS existe outra versão do programa PIP, denominado XFER, que opera essencialmente como o PIP, com alguns aperfeiçoamentos. Não obstante, o PIP pode também ser executado sob CDOS.

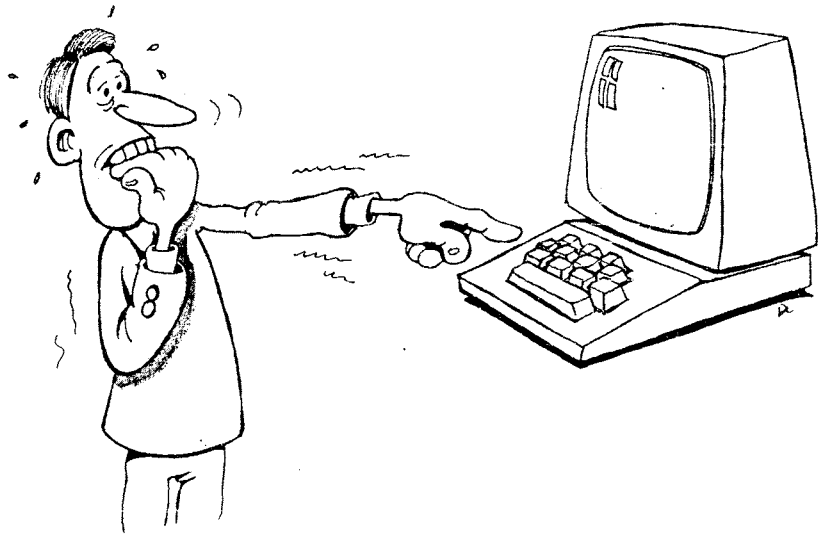
A maior diferença é que alguns caracteres de controle que não têm significado sob o CP/M são interpretados pelo CDOS e podem não ser utilizados por programas de aplicação escritos para rodar sob o CP/M. Tipicamente, o programa continuará a rodar, mas pode não ser possível usar alguns dos caracteres de controle.

Outros programas

O sistema operacional CP/M inclui apenas os programas necessários para dialogar com o computador e gerenciar o sistema de arquivos. A versão padronizada do CP/M é também acompanhada de vários programas utilitários padronizados como PIP e ED (descritos detalhadamente nos próximos capítulos).

Naturalmente, cada usuário do sistema de computadores irá executar um certo número de programas de aplicação. Vários exemplos específicos serão apresentados para demonstrar como tais programas são executados sob o CP/M, e também apresentaremos as definições pertinentes. Considerando que a maioria dos programas de aplicação pressupõem uma organização específica do sistema de arquivos, é importante relembrar que os programas de aplicação destinados a rodar no seu sistema devem ser compatíveis com o CP/M. Além do mais, se forem escritos numa linguagem específica como BASIC, exigirão um *intérprete de linguagem* como, por exemplo, um intérprete BASIC (discutido mais adiante neste capítulo).

Aprendemos, até agora, todas as definições básicas. Passemos então ao próximo passo; vamos ligar o computador e nos comunicar com o CP/M.



CHAMANDO O CP/M

A abordagem ao computador

A melhor maneira de superar o receio em relação aos computadores consiste em aprender a ligá-los e desligá-los, sem causar dano algum. Uma vez ligado corretamente o computador, o sistema operacional entra em ação e espera que seja digitado um comando (isto é, que o operador explique sua presença e faça algum pedido). Se o operador não disser nada de coerente ou apresentar instruções erradas, o sistema operacional irá solicitar-lhe que repita o pedido.

Tente fazê-lo, se o computador já estiver ligado. Tecle palavras ao acaso ou então letras e veja o que acontece. Se nada acontecer, pressione a tecla "RETURN". O sistema provavelmente irá repetir o que foi digitado, seguido de um ponto de interrogação. Irá então aguardar o próximo pedido. O operador não pode causar dano ao sistema operacional digitando no terminal. Não obstante, é possível que apague arquivos se continuar tentando. Portanto, espere e continue a ler o que escrevemos.

Ligando o sistema

Para ligar o sistema e "trazer" o CP/M, manipula-se um disquete (a não ser que o sistema seja baseado em discos rígidos). Portanto, cabe apresentar aqui algumas palavras de advertência a respeito de disquetes.

Disquetes

Normalmente, utilizam-se dois tipos comuns de armazenamento em discos: discos rígidos e discos flexíveis. Os *discos flexíveis*, também denominados *disquetes*, podem ser encontrados em dois formatos: 200 mm e 133 mm, apresentados nas Figuras 1.2 e 1.3. Os disquetes (discos flexíveis) podem ser usados para armazenar uma grande quantidade de dados a custo baixo. Contudo, os discos flexíveis são relativamente lentos, e apesar de sua grande capacidade ainda são pequenos demais para armazenar alguns arquivos (por exemplo, grandes arquivos comerciais). Os *discos rígidos* resolvem este problema; oferecem grande capacidade e grande velocidade de acesso, mas a um custo mais elevado. A maioria dos pequenos sistemas de computadores é equipada com um ou ambos os tipos de discos. Como os discos flexíveis são os mais utilizados, todos os exemplos do nosso livro se referirão a eles.

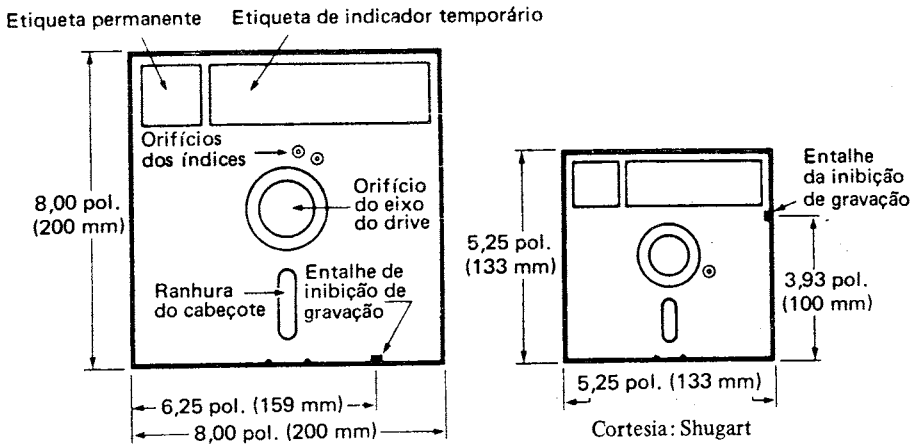


Figura 1.2: *Discos flexíveis do tipo "míni" e "regular": dimensões*

Os disquetes são meios magnéticos e fisicamente frágeis. Devem ser protegidos de influências magnéticas e tratados com carinho.

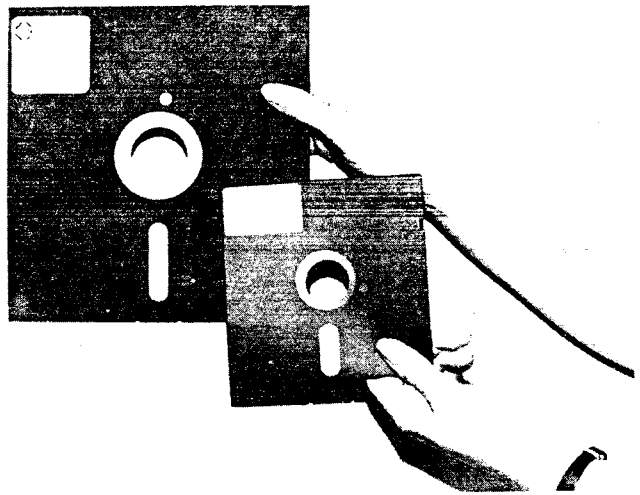


Figura 1.3: *Comparação de um disco flexível tipo "míni" e "regular"*

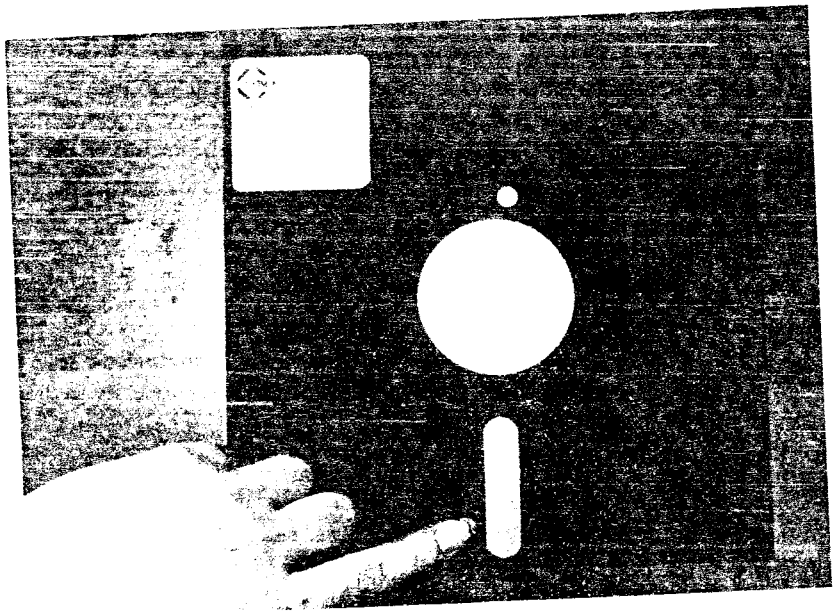


Figura 1.4: *O Slot permite ao cabeçote de leitura/gravação contatar o disco*

O quadrado de papelão (apresentado na Figura 1.4) contém um disquete flexível de *mylar* coberto com um óxido magnético. Quando usado, o disquete gira em alta velocidade dentro do papelão. O orifício central permite que o *drive* do disco faça girar o disquete. A abertura grande (apresentada na figura) permite ao cabeçote de leitura/gravação entrar em contato com a superfície do disco e nela ler ou gravar informações com uma operação semelhante à de um gravador. A informação é registrada em círculos concêntricos no disco, chamados *trilhas*. Cada trilha é dividida logicamente em *setores* pelo CP/M. (veja Figura 1.5)

Às vezes os disquetes são equipados com uma chave de *inibição de gravação*. No caso de um disquete padronizado de 200 mm, a chave é coberta por um pedaço de papel de alumínio. Se este for removido, a chave ficará exposta e o *drive* do disco não será mais capaz de gravar sobre ele.

No caso de minidisquetes (133 mm) o inverso é verdadeiro, sendo necessário remover o papel aluminizado para se poder gravar no disco. Uma vez estando o papel colocado sobre o cabeçote, só é possível ler. Esta característica é usada para proteger informações importantes. Por exemplo, disquetes-mestre, guardados e armazenados, geralmente são protegidos contra gravações. Contudo, o leitor deverá especificar esta opção ao comprar disquetes.

Manipulando disquetes

Manipule os disquetes sempre com cuidado. Não toque nas áreas expostas, não os contamine com poeira e nem os arranhe. Também não coloque nenhum objeto magnético próximo a um disquete (por exemplo, chaves de fenda e telefones), pois podem dani-

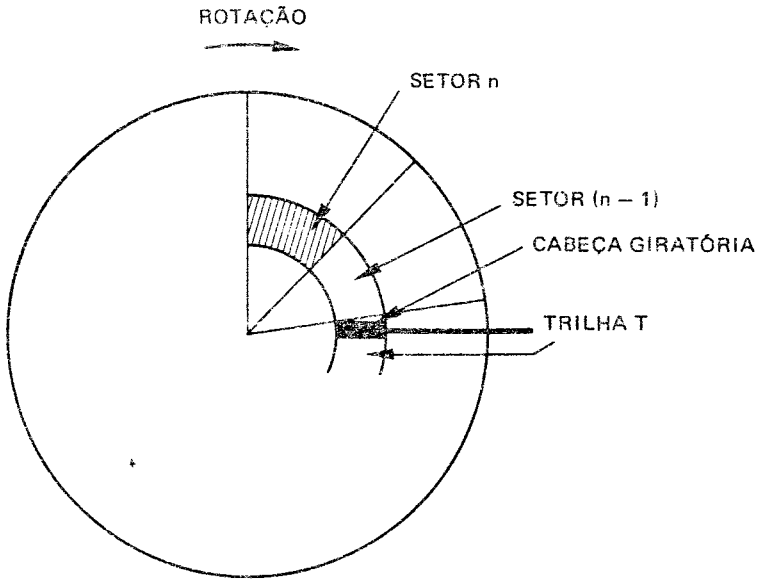


Figura 1.5: *Trilhas e setores*

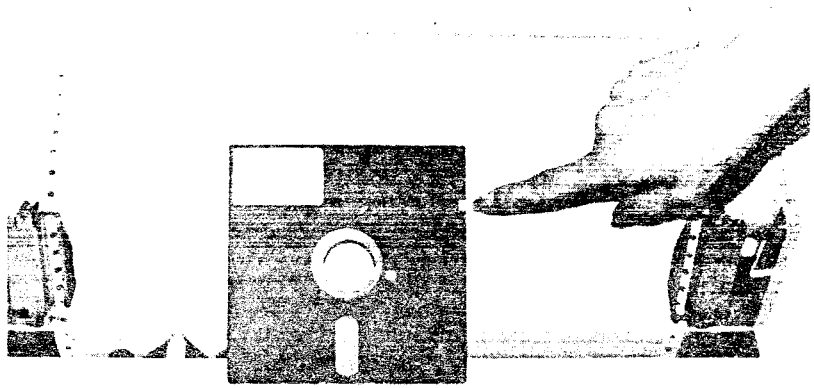


Figura 1.6: *O disquete de 133 mm é equipado com um cabeçote de inibição de gravação*

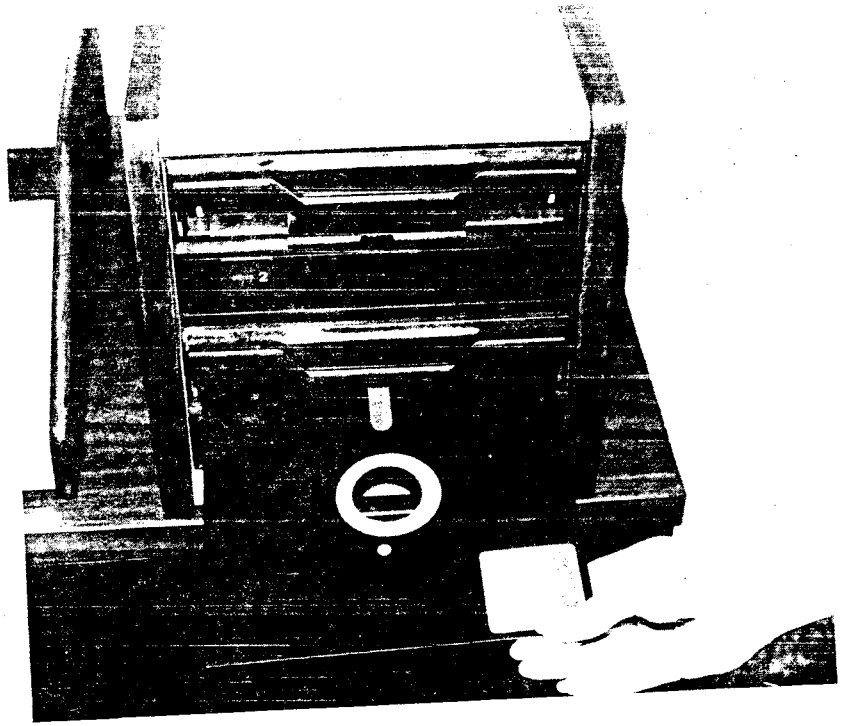


Figura 1.7: *Inserindo o disquete no drive 1 (SOL)*

ficar o disquete. É importante verificar como se deve inserir o disquete no seu *drive* específico. Em geral, podemos aplicar a seguinte "regra prática": segurar o disquete com o polegar sobre o quadrado para inseri-lo corretamente (veja Figura 1.7). Quem estiver praticando pela primeira vez deve usar uma *cópia* do disquete do sistema (para evitar danificar o original).

Desligando o computador (ou o *drive*, se este for separado do computador) enquanto o disquete ainda estiver no seu compartimento, é possível que o disquete se torne inutilizável. Distúrbios elétricos (de força) podem fazer com que o computador ou a eletrônica do *drive* enviem sinais indesejáveis ao disquete e gravem novas informações sobre as já existentes. Mantendo ligado o computador e o *drive* ao inserir ou remover disquetes, não se terá tal tipo de problema (a não ser que haja uma falta de força). Similarmente, desejando desligar o sistema, assegure-se sempre de que os disquetes foram removidos.

Agora apresentaremos as etapas necessárias para "chamar" o sistema. Novamente, por "sistema" entendemos a versão CP/M 1.4, a versão CP/M 2.2 ou a versão MP/M 1. A versão MP/M 1 é praticamente idêntica à versão CP/M 2.2. Descreveremos todos os três sistemas. Quando falarmos em "o sistema", poderemos nos referir a qualquer dos três, em caso contrário, vamos especificar a que sistema estaremos nos referindo.

Ligue-o, insira o disquete do sistema e dê a partida

O procedimento

Antes de começarmos, observe que o *Disquete do Sistema* é o disquete especial que contém o sistema operacional CP/M (ou MP/M). Provavelmente o aprendiz só recebeu um disquete do sistema, por isso deve solicitar ou fazer uma cópia para utilizar nas sessões de prática.

Se não conseguir alguém que lhe faça uma cópia do disquete do sistema e tiver que fazê-la sozinho, primeiro termine de ler esta seção e aprenda a ligar o sistema, e depois siga os procedimentos descritos no Capítulo 3 e resumidos abaixo. Nos *displays* da tela os caracteres sublinhados são aqueles já teclados. Um retorno do cursor (uma tecla especial do teclado) é representado por ↵. Eis o resumo do procedimento:

1. Insira o disquete do sistema no *drive* A.
2. Insira um disquete em branco no *drive* B.
3. Tecle os caracteres apresentados nesse *display*:

```
A > SYSGEN !
SYSGEN VER 1.4
SOURCE DRIVE NAME # (OR RETURN TO SKIP) A
SOURCE ON A, THEN TYPE RETURN !
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) B
DESTINATION B, THEN TYPE RETURN !
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) !
A > PIP B:=A:*.*[OV] !
(Copying Messages)
A >
```

4. Remova a cópia do *drive* B, rotule-a e insira-a no *drive* A.

Agora, tomando uma cópia do nosso disquete do sistema vamos aprender a ligar e desligar o microcomputador. Já que diferentes computadores têm métodos diversos para ligar o sistema, preste bem atenção para seguir as instruções que acompanham o computador.

Para "chamar" o CP/M é necessário:

1. Ligar o computador e os elementos periféricos.
2. Transferir o programa CP/M do disquete, onde está armazenado, para a memória do computador.

O procedimento exato varia ligeiramente de um computador para outro. Os computadores que não foram elaborados para lidar com o CP/M e que são equipados com seu próprio monitor ou sistema operacional (como o SOL) exigem duas operações sucessivas para "chamar" o CP/M. Por outro lado, os computadores elaborados para rodar o CP/M fazem-no com uma única operação simples. O programa monitor-residente do computador carrega automaticamente o CP/M a partir do disco.

Examinaremos agora um exemplo de cada caso, descrevendo a maneira de ligar dois *sistemas* diferentes de microcomputadores.

Ligando o Cromemco

Para ligar o computador Cromemco, aperte o interruptor ON/OFF na parte traseira da caixa, e vire a chave da frente para ON. Ligue o seu terminal, a impressora (se a tiver) e outros terminais (se estiver utilizando o MP/M). Os *drives* dos discos do Cromemco estão contidos dentro do móvel do computador e não precisam ser ligados separadamente. Se possuir outros meios de armazenamento, como um *drive* de disco rígido, ligue-os também.

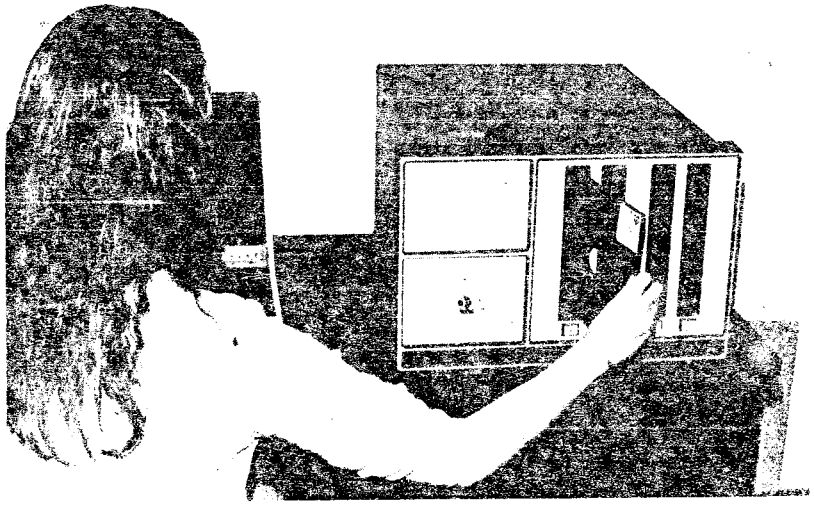


Figura 1.8: *Inserindo o disquete*

Agora insira o disquete do sistema no *drive* de disco A – o *drive* A é o que fica mais próximo à chave (como mostramos na Figura 1.8). Vire a chave para RESET e depois para ON (veja Figura 1.9). Examine o teclado do terminal, ache a tecla de retorno do cursor

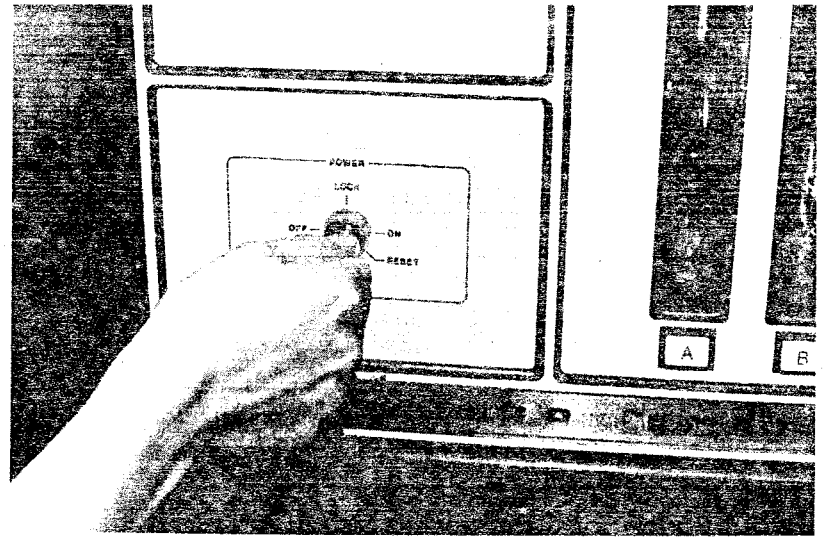


Figura 1.9: *Ligando o Cromemco*

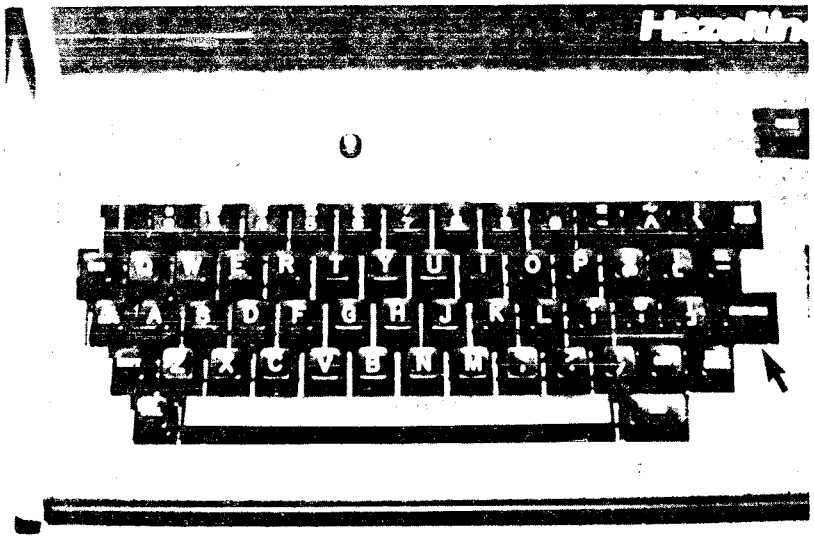


Figura 1.10: *O teclado do terminal*

(geralmente maroada RETURN ou CR) e pressione-a duas ou três vezes (veja Figura 1.10). De repente aparecerão na tela a mensagem do sistema e um *prompt*:

Mensagem do sistema: 48K CP/M
Prompt do sistema: A.

ou (com MP/M)

Mensagem do sistema: xxK MP/M
Prompt do sistema: 0A.

O CP/M está rodando agora à disposição, aguardando os seus comandos.

Ligue o SOL

Para ligar o computador SOL (um sistema mais antigo) utilize o interruptor na parte posterior do terminal e ligue o terminal de TV (CRT). (Veja Figura 1.11). Ligue os *drives*

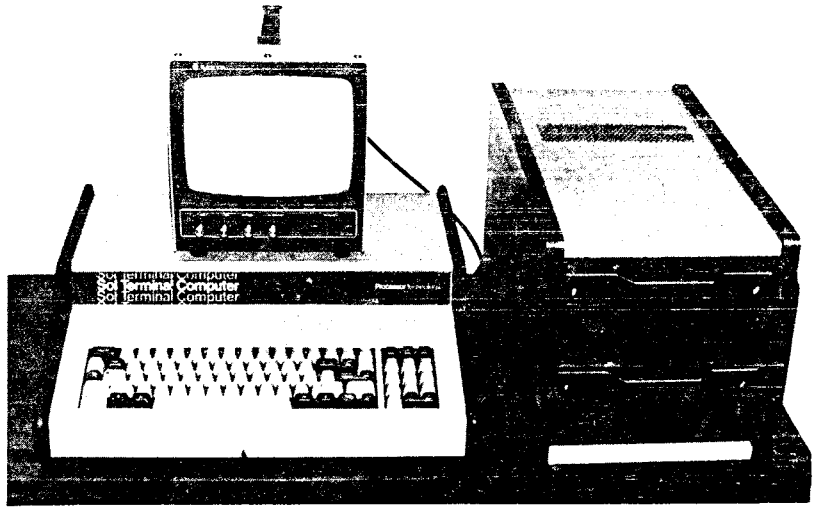


Figura 1.11: *O sistema SOL*

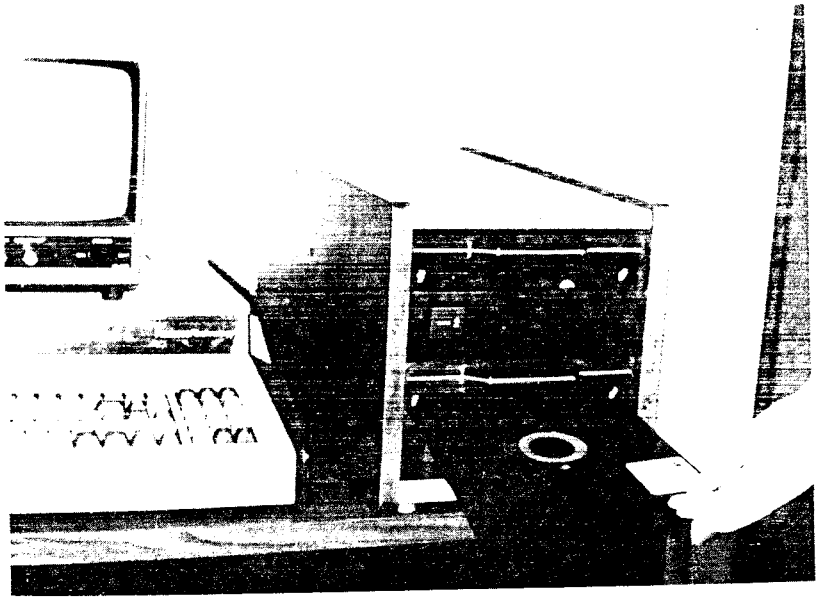


Figura 1.12: *Inserindo um disquete em A*

separados dos discos e insira o disquete do sistema CP/M no *drive* A (orifício inferior) (veja Figura 1.12). Este símbolo irá aparecer imediatamente na tela:

Este é um *prompt* do programa monitor do SOL, não o CP/M, que ainda está no disquete. Observe que, se a chave marcada LOCAL (no teclado do SOL) estiver em ON, você não estará conectado, de fato, com o sistema. Desligue LOCAL apertando essa tecla.

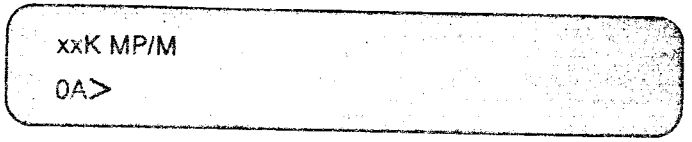
Para "chamar" o sistema, digite o seguinte comando:

> EX E000 ↵

O símbolo ↵ representa a tecla RETURN. O valor E000 é o endereço do programa que carrega automaticamente o CP/M a partir do disco. Este valor varia em cada controlador de disco. O fornecedor do controlador de disco dirá qual o endereço que deve ser utilizado com o seu sistema.

Depois de digitar o comando para "chamar" o sistema, a tela irá apresentar:

ou (com MP/M): '



O CP/M está pronto e esperando pelo aprendiz.

O que fazer se o SOL não funcionar ou se nada acontecer

Verifique primeiro se a tecla LOCAL está em ON. (Ela deve estar em OFF). Se LOCAL estiver ligado, desligue-o e tente digitar novamente 'EX E000', seguido de RETURN (↵).

Se LOCAL estiver desligado e o sistema não "der sinal de vida", verifique se a tecla UPPER está ligada. Essa tecla transforma todos os caracteres em letras MAIÚSCULAS (UPPER CASE) ao invés de minúsculas. O aprendiz deve teclar 'EX E000' em caracteres MAIÚSCULOS (UPPER CASE) com exceção dos zeros. Pressionando a tecla UPPER para a posição ON (não a tecla SHIFT LOCK) o seu comando 'EX E000' irá funcionar.

USANDO O CP/M

Pronto para dar a partida

Agora o aprendiz acabou de executar uma operação "bootstrap", ou partida "a frio". Algumas pessoas preferem pensar que as máquinas estão frias até ligá-las, ou que se "chame" um sistema operacional "dando-lhe um pontapé". O termo "bootstrap" derivou-se, na realidade, da idéia de que se a pessoa fosse suficientemente forte "poderia levantar-se a si mesma pelos cordões de suas botas". Na verdade, o monitor residente "retira o CP/M do disquete e dá a partida", isto é, o sistema "dá a partida a si próprio".

Uma partida "a frio" difere de uma partida "a quente", que será descrita mais adiante.

Agora, o que significa este 'A' (ou '0A'), e o que é um *prompt*?

Prompts do sistema

Um *prompt* é uma mensagem ou um símbolo que o sistema apresenta quando está pronto para receber o seu próximo comando. Todos os sistemas possuem *prompts*, mas cada um usa um símbolo diferente. Para a versão 1.4 do CP/M (ou abaixo) o símbolo de partida é 'A>'. Para a versão 2.2 do CP/M ou para MP/M, o símbolo é '0A>'. O A representa o *drive* de disquete A, e o '0' representa a área zero do usuário. As áreas do usuário são descritas no Capítulo 2, mas o aprendiz ainda não necessita esta informação (ainda não irá modificar a sua área de usuário).

O *prompt* do sistema sempre diz em que *drive* de disquete (ou de disco) o aprendiz "está", isto é, aquele que ele está usando; ele possui pelo menos um *drive*, e seu rótulo é 'A'. *Drives* subseqüentes seriam rotulados 'B', 'C' etc. Passemos para o *drive* B, admitindo-se que o aprendiz possua dois.

O aprendiz digita:

B: ↵

A resposta é:

B>

O sistema agora está rodando com o *drive* B, e o *prompt* agora é:

B>

Voltemos a A:

B> A: ↵

A>

Arquivos

Os disquetes armazenam as informações em *arquivos*. Para ter acesso a essas informações, o aprendiz deve dizer ao computador para dirigir-se a um disquete ou disco especial (por meio do *drive* do disquete ou disco) e encontrar um arquivo que possua um certo nome (um *filename*). Foi dada a partida ao sistema utilizando o *drive* do disquete A. Como o aprendiz não "se moveu" para outro *drive*, ele continua "em" A e, portanto, receberá o *prompt* 'A>'. O aprendiz só poderá se mover para outro *drive* se tiver outro disquete no *drive*.

Mostraremos como inserir outro disquete, mais adiante, neste capítulo.

Examinando o teclado

Pressione apenas a tecla RETURN. O *prompt* do sistema deve aparecer novamente. Pressione a tecla RETURN várias vezes, e observe como é fácil enviar linhas em branco ao computador. A tecla RETURN é sempre empregada para enviar um comando ao computador. O aprendiz sempre digita um comando e depois o segue com RETURN, simbolizado por ↵ neste livro. Só existem alguns casos especiais nos quais o praticante não iria empregar a tecla RETURN – por exemplo, quando usar a tecla CTRL (control) e outra tecla simultaneamente. Estes casos especiais serão explicados mais tarde, minuciosamente.

Primeiro, acostumemo-nos a usar o sistema: digite caracteres ao acaso e aperte a tecla RETURN, como no exemplo abaixo:

```
A> ANYTHING ↵
ANYTHING ?
A>
```

Se o sistema, em qualquer momento, deixar de apresentar uma mensagem de erro e o *prompt* do sistema, como mostramos acima, pressione a tecla CTRL (mantenha-a pressionada) ao mesmo tempo em que pressiona a tecla C. (veja Figura 1.13) Esta combinação (CTRL e C) produz uma partida "a quente" (ou "boot quente", ou um "reboot" do sistema). Uma partida "a quente", essencialmente, interrompe o que quer que o computador esteja fazendo e dá nova partida ao sistema operacional. O aprendiz irá, então, receber de volta o *prompt* do sistema.

Se isto não funcionar, consulte a seção "O que há de errado?" neste capítulo.

O aprendiz deve praticar os CTRL e C (abreviado para ↑ C neste livro; ↑ representa CTRL). Lembre-se de que a tecla CTRL deve manter-se pressionada ao mesmo tempo que

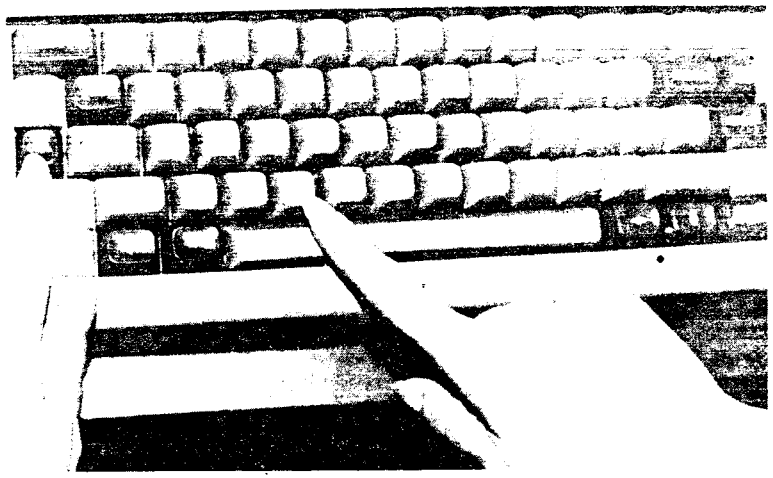


Figura 1.13: *O controle C causa um "boot" quente*

pressiona C. O aprendiz também deve praticar o uso de RUBOUT (DELETE ou '←' em alguns terminais). Ao teclar algo, o aprendiz pode apagar o último caractere que foi teclado pressionado a tecla RUBOUT (DELETE). O RUBOUT (DELETE) irá apagar o primeiro caractere à esquerda. Em alguns terminais o aprendiz pode pressionar essa tecla e apagar a linha inteira.

Em muitas versões CP/M, o RUBOUT (DELETE) reapresenta o caractere que apaga, ao invés de fazê-lo desaparecer. Se o RUBOUT (DELETE) reapresentar (ecoar) o caractere que apagar, teremos algo como o que segue:

```
A> THIS ISAN EXXE NASI SIHT
```

(Começamos a pressionar RUBOUT (DELETE) aqui.)

(Podemos começar a teclar novamente aqui.)

Outro modo de apagar uma linha é pressionar simultaneamente as teclas CTRL e U (isto é, pressionando e mantendo a tecla CTRL para baixo enquanto U é pressionada). CTRL e X farão o mesmo. Abreviamo-los para ↑ U e ↑ X. Quando se pressiona CTRL e U (ou CTRL e X), o sistema apresenta um sinal numérico (#), significando que "tudo o que está à esquerda desse sinal foi apagado":

```
A> THIS IS AN EXAMPLE#
```

(Começe a teclar novamente aqui.)

(CTRL e U (ou CTRL e X) pressionados aqui.)

Pode-se digitar novamente como se fosse uma linha nova. Naturalmente o aprendiz também pode pressionar RETURN para enviar uma linha em branco, reexibindo o *prompt* do sistema.

O que há de errado . . . ?

Às vezes acontece algo inusitado e pode não se ter certeza da causa. Se pressionar RETURN e o computador não responder, então ele está ocupado fazendo algo (por exemplo, rodando um programa). Se o aprendiz, inadvertidamente, digitar o nome de um programa que ignorava existir, e o programa começou a ser executado, pode fazer abortar o programa (isto é, fazê-lo parar) e voltar ao sistema com uma partida "a quente" ("boot quente"): Tecle ↑ C.

Se uma partida "a quente" (↑C) não trouxer de volta o *prompt* do sistema (A >), verifique se algumas das luzes dos *drives* dos disquetes estão acesas. Uma luz acesa significa que o computador está tentando *ler* o disquete desse *drive*. Se a luz estiver acesa e não houver disquete, pode-se tentar inserir um disquete no *drive* para que o computador tenha algo a ler. Se isto não trouxer de volta o *prompt* do sistema (ou resumir o programa), tem-se que voltar ao início e fazer uma partida "a frio" (consulte, no início deste capítulo, a seção intitulada "Ligue-o, insira o disquete do sistema e dê a partida").

Em alguns sistemas, existe um recurso de interrupção para parar o computador. No caso do SOL, pressionar simultaneamente UPPER CASE e REPEAT fará com que o operador volte ao monitor do SOL.

NOTA: Assegure-se de ter retirado os disquetes antes de desligar qualquer coisa.

Disquete do sistema

É importante lembrar que o disquete do sistema foi configurado para o seu sistema específico. Se qualquer elemento do hardware do sistema for alterado, o disquete do sistema original, comumente, não funcionará.

Em especial, se for mudada a impressora, o terminal CRT, o controlador de disco ou o tamanho da memória, será necessário um disquete de sistema diferente. Se alterar a configuração do seu hardware, de tempos em tempos, preste atenção e rotule corretamente os disquetes dos sistemas para não confundi-los entre si.

Examinando o diretório

O nosso disquete do sistema está no *drive* A. Contém CP/M e ainda outros arquivos. Vamos examiná-lo.

Poder-se-á descobrir quais os arquivos que se tem no disquete do *drive* A digitando o comando DIR ("diretório"):

```
A> DIR↵
A:PIP      COM
A:ASM      COM
A.LOAD     COM
A:PROGR    COM
A:STAT     COM
A:PROGR    INT
```

Se o seu *display* rodar depressa demais, pressione as teclas CTRL e S simultaneamente (isto é, ↑S). Isto fará com que o *display* pare. Quando estiver pronto para fazer continuar o *display* da lista, pressione novamente as teclas CTRL e S, simultaneamente, para dar nova partida ao *display*.

O exemplo acima é uma amostra do *display* do comando DIRectory. Cada disquete (ou disco) possui um "diretório" de *filenames*, com um nome para cada arquivo. Sabemos que esses arquivos estão no disquete do sistema por estarmos no *drive* A, e o disquete do sistema no *drive* A. Cada nome de arquivo (*filename*) é prefaciado por 'A:' para mostrar que o arquivo está no *drive* A. O primeiro nome 'PIP' e a palavra subseqüente 'COM' formam um *filename* completo 'PIP.COM'. 'PIP' é o nome primário, e 'COM' é uma *extensão*, indicando o tipo de arquivo; são separados por um ponto.

As *extensões* são também chamadas *file types* (tipos de arquivo). Por exemplo, todos os arquivos com 'COM' como extensão são *command files* (arquivos de comando); algumas vezes denominados *transient commands* (comandos transientes). Todos os arquivos com 'BAS' como extensões são programas fonte BASIC, e todos os arquivos com 'INT' como extensões são programas intermediários BASIC. O aprendiz não necessita utilizar uma extensão específica para um arquivo de dados ou um arquivo de texto (um arquivo que contém certos tipos de informação, como um texto, por exemplo). Podem-se criar extensões próprias para categorizar os arquivos de dados.

É importante separar os diferentes tipos de arquivo, já que um programa poderia aparecer no seu diretório com o mesmo nome e dois ou mais tipos.

Por exemplo:

TEXT.WRK	(arquivo de trabalho)
TEXT.BAK	(arquivo de reserva)

ou

PROG.BAS	(listagem em BASIC)
PROG.INT	(forma compilada)

Utilize sempre o nome de arquivo completo (isto é, um nome primário e uma extensão, com um ponto) ao se referir a um arquivo, exceto quando empregar um arquivo de comando como arquivo transitório (discutido mais adiante neste capítulo). Já que o aprendiz agora deseja criar um novo arquivo e não alterar arquivos já existentes, deve aprender a inserir um novo disquete e criar nele um novo arquivo.

EXECUTANDO UM PROGRAMA

Iremos agora criar um arquivo simples. Esse arquivo será empregado no restante do capítulo para demonstrar os procedimentos corretos e a utilização dos recursos do CP/M.

A melhor maneira de se criar um arquivo é executando um programa que possa criá-lo. Um bom exemplo é um programa de processamento de palavras ou um programa comercial. Se não houver ninguém disponível para mostrar como utilizar tal programa, pode-se empregar ED, o *editor* que é fornecido junto com o CP/M, para criar um arquivo. Rodaremos um programa comercial-amostra e depois mostraremos como empregar ED para criar um arquivo.

Em primeiro lugar, vamos inserir um disquete em branco no *drive* B (veja a Figura 1.14). Utilizaremos o sistema de correspondência NAD (da Structured Systems, Oakland,

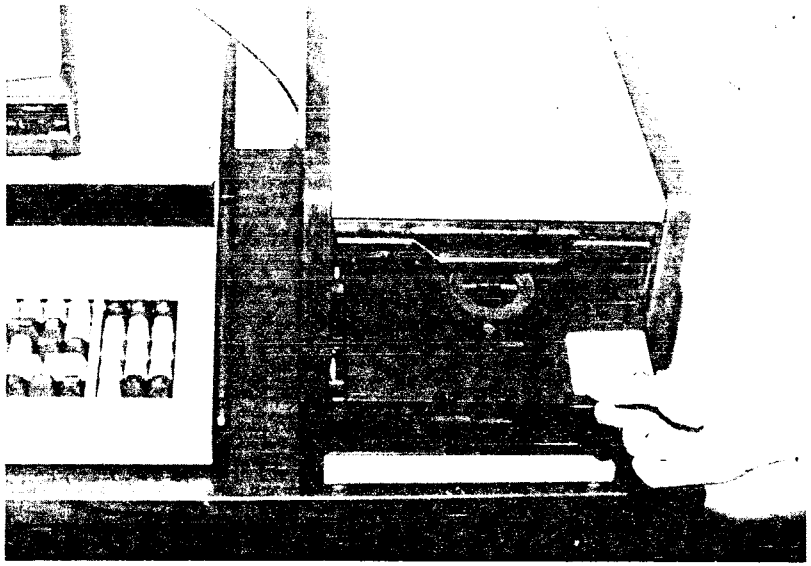


Figura 1.14: *Inserindo um disquete em B*

Califórnia) escrito em CBASIC e que é apresentado com vários programas. Para criar um novo nome e endereço para o arquivo, digitamos:

```
A > CRUN NADENTRY↵
```

Para que funcione, tanto o CRUN (o compilador CBASIC) como o NADENTRY devem estar no nosso disquete do sistema. Observe que usamos um nome de arquivo incompleto para NADENTRY. O programa CRUN admite, automaticamente, que NADENTRY é do tipo INT.

Para descrever o que acabou de acontecer, precisamos definir uma nova palavra: *compilador*. O CRUN é um compilador. Na instrução CRUN NADENTRY, NADENTRY refere-se a um programa escrito na linguagem BASIC. Para poder executar um programa escrito em BASIC, o computador exige um interpretador ou compilador BASIC. Aqui, empregamos o compilador CRUN. A diferença teórica entre um compilador e um interpretador é que um compilador executa o programa com mais eficiência, ao passo que o interpretador permite um desenvolvimento interativo do programa. Logo que o programa comece a rodar, o efeito é o mesmo, quer tenha sido usado um interpretador ou um compilador. A seqüência é apresentada abaixo (para maior clareza, o diálogo foi abreviado).

```
A > CRUN NADENTRY↵
CRUN VER 1.04
NAD  VER 2.0
```


Criamos um novo arquivo denominado NAMES no *drive* B (deve existir um disquete no *drive* B):

```
ENTER FILE NAME: NAMES
ENTER DISK DRIVE: B
```

Fazemos entrar um nome e um endereço (A especifica "add" – acrescente – um nome):

```
ENTER FUNCTION (A, C, D, E, S, OR STOP): A↵
RECORD NUMBER IS: 7
ENTER NAME: CHARLES FRIEND
ENTER LINE ONE OF ADDRESS: ABC COMPANY
ENTER LINE TWO OF ADDRESS: 123 LUNAR DR
ENTER CITY: PALO ALTO
ENTER STATE: CA
ENTER ZIP: 90010
ENTER PHONE: 408 123 4567
```

Guardamo-lo no disco. O comando 'S' é específico para NADENTRY e significa "Save" (guarde*):

```
ENTER FUNCTION (A, C, D, E, S, OR STOP): S↵
1 RECORD SAVED
ENTER FUNCTION (A, C, D, E, S, OR STOP): STOP↵
NADENTRY COMPLETED
A >
```

*Normalmente usa-se a tradução literal "guardar". (N. do T.)

Estamos de volta ao CP/M. A ação de *guardar* a entrada criou um arquivo em B denominado NAMES. Esse arquivo contém o nome e o endereço de CHARLES FRIEND.

CRIANDO UM ARQUIVO COM ED

Também é possível criar-se um arquivo-texto simples, utilizando o editor do CP/M, o ED. Possuindo-se outro editor mais fácil de se usar e que rode em CP/M, e se a sua documentação for mais fácil de ler, tente empregar o outro editor; caso contrário, continue com a nossa sessão ED simples.

O comando ED é um *comando transitório*, o que significa que ele existe em um disquete como um arquivo de comando (como ED.COM). O disquete no *drive* A deverá conter ED.COM. O operador pode verificá-lo usando o comando DIR:

Tecle:

A: ↵ (para voltar ao *drive* A)

Agora tecle:

DIR ↵ (para examinar o diretório, você deveria ver ED.COM na listagem)

Tecle:

B: ↵ (para voltar ao *drive* B)

Já que está agora no *drive* B, o operador deve especificar *drive* A no nome do arquivo quando se referir a ED.COM.

Observe que, utilizando o CDOS da Cromemco, não necessita especificar o *drive* ao teclar um comando. Pode teclar: 'ED' ↵ no *drive* B. CDOS irá procurá-lo no *drive* B, e, se não o achar, automaticamente procurará no *drive* A. Esta é uma característica que facilita o trabalho.

Ao teclar um comando transiente, o operador não deve teclar a extensão 'COM' com o nome primário. Para executar 'ED.COM', é suficiente teclar 'ED' ou 'A:ED', dependendo do disco em que se acha (na realidade, se digitar 'ED.COM' cometerá um erro).

Em primeiro lugar, pense em um nome de arquivo para o seu arquivo, como, por exemplo, SAMPLE.TXT (separe sempre o nome primário SAMPLE da extensão TXT por meio de um ponto). A extensão TXT não é exigida, mas ajuda a indentificar o arquivo.

Execute o comando ED digitando 'ED', um espaço, e depois o seu novo nome de arquivo:

```
B> ED SAMPLE.TXT ↵
ED?
B>
```


Espere! Esquecemos de especificar o *drive* A com o comando ED (porque ED.COM está no *drive* A, e nós estamos no *drive* B). Tentemos novamente:

```
B> A:ED SAMPLE.TXT ↵
NEW FILE
*
```

Funcionou. Não esqueça de teclar SAMPLE.TXT depois de ED!

O asterisco (*) é o *prompt do editor*. Indica que o programa ED está presente e pronto para o seu próximo comando ED. Os comandos ED são descritos em detalhe no Capítulo 4, mas o operador pode aprender sobre alguns aqui: o comando I *insere* um novo texto em um arquivo, o comando B coloca o operador no *início* de um arquivo, e o comando T apresenta o conteúdo do arquivo ('Text'). O comando E *salva* o arquivo e *termina* a sessão com ED ('END').

O aprendiz pode usar ED para criar e modificar arquivos-texto. Um arquivo-texto é igual a qualquer outro arquivo, com a informação representada em um código binário chamado ASCII. Cada caractere nas teclas do teclado do terminal tem um número de código ASCII – estes são apresentados no Apêndice C. Provavelmente o aprendiz nunca terá necessidade de usar o número de código, mas ajuda saber como são "codificados" – na eventualidade de ver números estranhos em seu arquivo, rodeados de símbolos estranhos.

Para inserir novos caracteres em seu novo arquivo, use o comando I e depois *digite* as linhas do texto. Termine cada linha com um "retorno do cursor" (RETURN ou CR), assim como faria em uma máquina de escrever. Use ↟ Z quando terminar de inserir o texto.

```
*I ↵
THIS IS MY NEW TEXT FILE, CALLED SAMPLE.TXT ↵
↟ Z
*
```

Agora, acrescente uma segunda linha:

```
'THIS IS LINE TWO'.
```

Se desejar reapresentar o que digitou, use esta seqüência de comandos:

```
*B ↵ (vá para o início do arquivo)
*#T ↵ (apresente o conteúdo)
THIS IS MY NEW TEXT FILE, CALLED SAMPLE.TXT.
THIS IS LINE TWO.
*
```

O comando B coloca o operador no início do arquivo-texto, e o comando T digita ou apresenta uma linha do texto, ou o arquivo completo (se teclar um '#' antes do T).

Agora deve guardar o arquivo usando o comando E:

```
*E ↵
B>
```

O comando E guarda o arquivo e faz parar o programa ED. Volte ao sistema e o arquivo estará no disquete no *drive* B. SE NÃO EXECUTAR O COMANDO E, PERDE A INFORMAÇÃO QUE ACABOU DE DIGITAR NO ARQUIVO. Use o comando E com freqüência! Isto é importante porque o texto que o operador digita é armazenado na memória interna do computador, onde é de fácil acesso e modificação. Ao teclar E, o texto é copiado em um meio de maior permanência, o disco.

Saindo de ED, desligando o sistema, ou por meio de um "reboot" (pressionando ↑ C), o operador perde o conteúdo da memória interna do computador. Só o que foi explicitamente guardado no disco continuará acessível. Isto é aplicável a qualquer programa de edição. Por este motivo foi que usamos 'S' no final de NADENTRY na seção anterior.

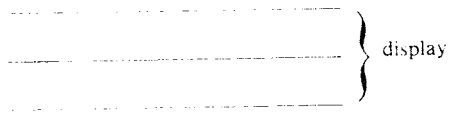
Como recomendação de natureza geral, guarde com freqüência, especialmente se tiver tendência a pressionar a tecla ↑ C por acaso, ou deixar o terminal sem que ninguém o vigie. O operador pode guardar o conteúdo de um texto quantas vezes quiser. Isto atualizará o arquivo que está no disco e não resultará em múltiplas cópias.

MANIPULANDO ARQUIVOS

Apresentação de arquivos

Para apresentar o arquivo quando voltar ao sistema operacional, use o comando TYPE, e especifique o nome do arquivo:

```
B> TYPE SAMPLE.TXT ↵
```



O aprendiz deve praticar a criação de arquivos utilizando ED e os comandos de ED. Encontrará uma descrição completa de ED no Capítulo 4. Observe, porém, que existem outros programas editores (e processadores de palavras), que rodam sob CP/M ou MP/M, e que os outros programas editores possuem comandos diferentes e diferentes estruturas de arquivos. Use o editor que melhor se adapte às suas necessidades. Alguns são muito simples, ao passo que outros são mais complicados e apresentam maior número de detalhes.

Se possuir mais de um programa editor, lembre-se de qual foi o editor que usou para criar o arquivo e use-o para modificar o arquivo.

RENOMEANDO ARQUIVOS (REN)

O comando REN é usado para modificar o nome de um arquivo. É escrito como

```
REN Newname = Oldname
```

Use sempre nomes completos. Por exemplo:

```
REN FILE2.TXT = OLD.TXT
```

Ou:

```
REN BETTER.NAD = NAMES.NAD
```

Depois de executar REN, 'OLD' não existirá mais. Terá sido substituído por 'FILE2'. Similarmente, 'BETTER' ficará no lugar de 'NAMES'.

Lembre-se de usar sempre o nome completo do arquivo, incluindo o ponto e a extensão.

CARREGANDO UM NOVO DISQUETE (EXECUTANDO UM "WARM BOOT")

Logo desejaremos fazer cópias de disquetes e mudar o disquete que está no *drive* B. Para fazê-lo, é necessário, em primeiro lugar, aprender a executar um "warm boot".

Quando se insere um disquete, o CP/M automaticamente lê o diretório na memória do computador e "registra" o disquete. Se o aprendiz trocar o disquete em B e depois, imediatamente, tentar gravar no novo disquete em B (para criar um arquivo, por exemplo), não poderá fazê-lo. Este procedimento não funcionará por causa de uma característica (de versões anteriores do CP/M) que tem como intuito proteger o disquete contra rasuras inadvertidas.

Portanto, uma vez trocado um disquete, deverá, especificamente, autorizar o CP/M a gravar nele. Isto é feito dando nova partida ao CP/M e se denomina um "warm boot".

Vejamos como funciona isto: Mantenha o disquete do sistema no *drive* A e insira um novo disquete no *drive* B (admitamos, por exemplo, que você usou o *drive* B anteriormente para fazer uma cópia). Agora, volte ao terminal e pressione C para fazer um "reboot" no sistema (fazer um "warm boot").

Um "warm boot" (↑ C) reorienta o computador para o novo disquete. O computador coloca a informação em um disquete seguindo um mapa em sua memória. Quando o operador retira um disquete anterior e coloca um novo, o mapa para o disquete anterior ainda se encontra no computador. Em função da característica (previamente mencionada) do CP/M, que não lhe permite escrever em um disquete se antes houve outro no mesmo *drive* (para evitar acidentes), a pessoa deve teclar ↑ C para autorizar o CP/M a escrever no novo disquete. Executa-se um "warm boot" (↑ C) para *modificar o mapa* de modo que o computador possa colocar informações no novo disquete.

Se apenas estiver *lendo* a partir de um novo disquete (por exemplo, examinando arquivos e seu conteúdo) não há necessidade de um "warm boot" para introduzir o novo disquete. Se estiver gravando (isto é, criando um arquivo, criando uma cópia, ou enviando dados para um arquivo) em um novo disquete, em um *drive* que anteriormente continha outro disquete, a pessoa *deve* executar um "warm boot" para criar um novo mapa para o novo disquete. Exige-se um "warm boot" sempre que se inserir um novo disquete com a intenção de gravá-lo, a não ser que as nossas instruções especifiquem algo diferente (apenas em operações especiais de cópia).

Deve-se assinalar que o MP/M não permite mudar, ou inserir novos disquetes, a não ser que a pessoa faça um *disk reset* (a ser descrito mais tarde). O CDOS da Cromemco excluiu também a característica que exige um "warm boot" e a pessoa já não é obrigada a fazer um "reboot" cada vez que mudar disquetes em um *drive*.

Depois de executar um "warm boot" (↑ C) o aprendiz deve obter o *prompt* do sistema (A >). Agora está livre para ter acesso a arquivos e neles gravar o que desejar, tanto no *drive* A como no *drive* B, ou em outros *drives* se os possuir. Observe que ↑ C cancela qualquer ↑ P prévio.

O operador pode passar do *drive* A para o *drive* B teclando 'B' como um comando:

```
A> B:↵
B>
```

O operador agora está no *drive* B. Use o comando DIR para determinar que arquivos possui no disquete no *drive* B:

```
B> DIR↵
NOT FOUND
B>
```

A mensagem 'NOT FOUND' significa que o DIR não achou nenhum arquivo no disquete. Isto aconteceu porque temos um disquete em branco no *drive* B. Quando nos referimos a um arquivo, o CP/M admite (a não ser que lhe digam o contrário) que o arquivo está no *drive* em que nos encontramos. Neste ponto, para se referir a um arquivo no *drive* A, a pessoa pode ou se mover de volta para o *drive* A ou especificar '*drive* A' no nome do arquivo. Por exemplo, se a pessoa quisesse se referir a PIP.COM no *drive* A, diria:

```
A:PIP.COM
```

O comando DIR pode ser utilizado para achar um arquivo específico. Para fazer isto, a pessoa deve digitar o nome do arquivo depois do comando DIR (separado por um espaço):

```
B> DIR A:PIP.COM↵
A:PIP      COM
B>
```

Outra maneira de achar PIP.COM no *drive* A é mover-se para o *drive* A e depois executar o comando DIR:

```
B> A:↵
A> DIR PIP.COM↵
A:PIP      COM
A>
```


Passando para um *drive* de disquete que não contém um disquete, o sistema irá "ficar pendurado" (a luz no *drive* do disquete se acende, e nada acontece, e o teclado fica inerte) até que a pessoa execute uma partida "a frio". (Veja "ligue-o, insira o disquete do sistema e dê a partida" no início deste capítulo.)

Se o aprendiz possuir apenas um *drive* de disquete, não tem escolha -- ele é obrigado a remover o disquete do sistema para inserir um novo disquete. Já que isto é mais complicado, veja o Capítulo 2 para instruções a respeito. Com o CP/M a pessoa não pode fazer cópias de arquivos a não ser que possua mais de um *drive*. Entretanto, programas especiais em seu computador poderão, talvez, permitir que copie um disquete inteiro.

Fazendo uma cópia de um arquivo

O aprendiz sempre desejará fazer cópias de arquivos. Na realidade, tão logo tenha criado ou obtido um novo arquivo ou programa, deve fazer uma cópia do mesmo e arquivá-la separadamente em outro lugar (para o caso de algum acidente ocorrer com o disquete original).

O procedimento correto é o seguinte:

- Ao obter um novo programa ou arquivo, faça uma cópia do mesmo antes de qualquer outra coisa. Coloque o disquete-mestre em um lugar seguro, e trabalhe apenas com a cópia.
- Ao criar ou atualizar um arquivo, faça uma cópia do arquivo antes de deixar o sistema. Rotule-o claramente. Indique a data. Arquive-o em lugar seguro.

O comando PIP é usado para executar as operações de cópia. PIP representa "Peripheral Interchange Program" (Programa Periférico de Intercâmbio). Trata-se de um programa (escrito em linguagem de máquina) que o operador executa digitando 'PIP'. Para executá-lo, PIP já deve existir como arquivo em seu disquete com o nome PIP.COM (geralmente residente no disquete do sistema).

Se o operador estiver seguindo os exemplos, terá então um arquivo-texto SAMPLE.TXT no *drive* B. Faça uma cópia desse arquivo e coloque-a no disquete no *drive* A. Se esse arquivo for valioso, deve copiá-lo em um disquete novo em branco. Isto será explicado mais tarde.

Agora, tentemos usar o PIP. A pessoa deve passar primeiro para o *drive* A, porque PIP.COM está no *drive* A:

```
B > A:↵
A >
```

Agora, pode executar o PIP:

```
A > PIP↵
*
```

O asterisco (*) é o *prompt* do programa PIP, e lhe diz que o PIP está à altura e sendo executado. O operador pode agora digitar uma expressão PIP que executa uma operação de cópia. Expressões PIP simples têm esta forma:

```
d:copyname = d:originalname
```

O 'copyname' e 'originalname' são argumentos que na realidade representam nomes de arquivos, e o 'd' representa a letra do *drive*. O PIP sempre copia algo de um arquivo original para um arquivo cópia. Desta forma, o arquivo original já deve existir; o PIP cria o arquivo cópia com o nome que a pessoa apresentar. Por exemplo:

```
*A:COPY.TXT = B:SAMPLE.TXT ↵
```

Esta expressão diz ao PIP que faça uma cópia de SAMPLE.TXT, que está no *drive* B, e que dê a essa cópia o nome COPY.TXT, colocando-a no *drive* A.

Já que o aprendiz está agora no *drive* A, pode abreviar a expressão acima:

```
*COPY.TXT = B:SAMPLE.TXT ↵
```

O PIP parte do pressuposto de que o arquivo-cópia deve ser criado no *drive atual* (isto é, aquele em que o aprendiz se encontra no momento). Como SAMPLE.TXT está no *drive* B, a pessoa deve especificar a letra do *drive*. Neste ponto, sempre use a letra do *drive* ao praticar o uso do PIP, para familiarizar-se com o problema de copiar de *drive* para *drive*.

Depois que o PIP termina de copiar SAMPLE.TXT no novo COPY.TXT, volta com o asterisco (*), o *prompt* do PIP. Pressione a tecla de retorno do cursor (RETURN ou CR) para sair do programa PIP e voltar para o sistema:

```
* ↵
A>
```

Não interessa qual é o *drive* do qual a pessoa copia ou para o qual copia; o PIP deve fazer o operador voltar para o *drive* a partir do qual executou o comando. Como executamos o PIP a partir do *drive* A, voltamos a A.

Se desejar executar apenas uma operação PIP, o operador pode empregar um comando mais curto e digitar a expressão e o comando PIP em uma só linha:

```
A> PIP A:COPY.TXT = B:SAMPLE.TXT ↵
A>
```

Quando o PIP pára de copiar, o *prompt* do sistema reaparece.

Observe que a luz do *drive* B acende, depois apaga, e a luz do *drive* A acende e apaga, à medida que se tem acesso a cada disco. O PIP copia o arquivo em segmentos chamados *blocos*. Se o arquivo for grande, ele se vê obrigado a voltar ao arquivo original para obter mais blocos.

O aprendiz agora tem uma cópia de SAMPLE.TXT chamada COPY.TXT no *drive* A. Façamos uma cópia de COPY.TXT e coloquemo-la no *drive* B:

```
A> PIP B:=A:COPY.TXT ↵
A>
```

Não tivemos que especificar um nome do arquivo-cópia ('B:') porque desejamos que a nossa cópia de COPY.TXT tenha o *mesmo nome* que o original. O aprendiz provavelmente desejará executar este tipo de operação de cópia mais freqüentemente do que qualquer outra. O comando acima é equivalente a:

```
A> PIP B:COPY.TXT = A:COPY.TXT ↵
A>
```

Ambos os comandos criam um novo COPY.TXT no *drive* B que é uma cópia de COPY.TXT existente no *drive* A. A pessoa não é obrigada a especificar um nome de arquivo para a cópia se desejar que esta tenha o mesmo nome que o arquivo original. Contudo, deve especificar o *drive*, e esse *drive* deve ser diferente daquele do arquivo original se o nome do novo arquivo for idêntico ao da fonte.

Isto acontece porque a pessoa não pode ter dois arquivos, no mesmo disquete, com o mesmo nome. Se tivesse tentado fazer uma cópia de um arquivo sem especificar ou um nome de arquivo ou um *drive* diferente, a pessoa obteria a declaração 'INVALID FORMAT', seguida da parte da expressão que causou o erro. Se obtiver essa mensagem de erro, pressione a tecla RUBOUT (ou DELETE) para retificar o erro.

A maioria das operações de cópia envolve transferências de disquetes (ou de discos para disquetes ou vice-versa), porque a pessoa há de querer fazer cópias de reserva de arquivos em disquetes ou discos. Já que vai aprender a utilizar o comando ERAse, a pessoa também deve aprender a fazer uma cópia-reserva de todo o disquete (no caso de vir a cometer um engano com o comando ERAse).

COPIANDO UM DISQUETE COMPLETO

Aprendeu-se a copiar o arquivo COPY.TXT no *drive* A para COPY.TXT no *drive* B:

```
A> PIP B:=A:COPY.TXT ↵
A>
```

Fez-se uma cópia-reserva de COPY.TXT.

Para copiar um disquete completo com um só comando o aprendiz necessita usar um *filename match*, ou seja, uma expressão que diz ao computador que "execute esta instrução para qualquer arquivo que possa ser associado a esse nome". O "filename match" a ser usado para todos os arquivos no disquete é '*.*'. Por exemplo:

```
A> PIP B:=A:*.* ↵
```

O símbolo '*' irá comparar qualquer nome em seu campo. Esse comando copia todos os arquivos comparáveis com '*.*' (todos os arquivos no disquete) no *drive* A para novos arquivos com os mesmos nomes no *drive* B. (Observe que com o MP/M e novas versões do CP/M, '*.*' só irá comparar todos os arquivos na área atual do usuário.)

Ao copiar um disquete completo para B, é conveniente, nesta etapa, ter um disquete vazio no *drive* B. De outra forma, dois problemas poderiam surgir:

1. Se B já possuir arquivos, deverá ter espaço suficiente de sobra para acomodar todos os de A. Um disquete-padrão pode armazenar até 270K bytes, incluindo o diretório (se o aprendiz executar DIR, ele lhe dirá quanto espaço do disquete já foi usado).

2. Se B já possuir um arquivo com o mesmo nome do que está sendo copiado para B, a operação de cópia será sustada.

Já que o aprendiz está no *drive* A (no exemplo anterior), ele pode abreviar o comando acima:

```
A> PIP B:=*.*/
```

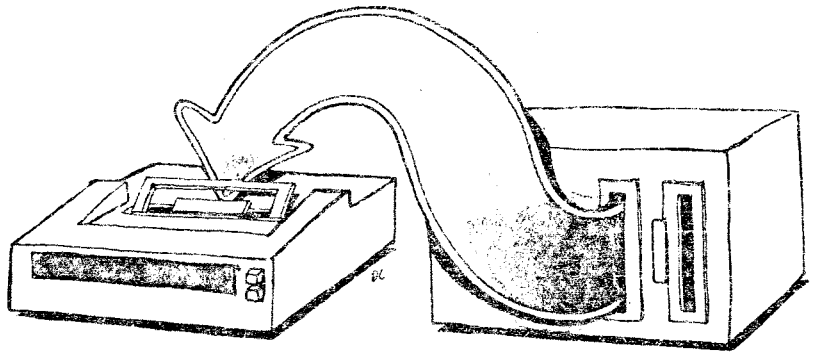
Ambos os comandos vasculham o *drive* A (o *drive* atual neste exemplo) em busca de arquivos que se comparem com '*.*' e criam cópias desses arquivos no *drive* B utilizando o mesmo nome de arquivo. Se já existir um arquivo no *drive* B com o mesmo nome, este será suprimido.

Lembre-se de que os nomes de arquivo podem ter oito caracteres à esquerda do ponto e três caracteres à direita do mesmo. O símbolo '*' compara quaisquer oito caracteres à esquerda do ponto e o símbolo '.*' compara quaisquer três caracteres à direita do mesmo.

Ao usar o comando acima, lembre-se de que todos os arquivos (e apenas arquivos) que estão no disquete serão copiados. O programa CP/M não é um arquivo. Ele é armazenado em duas trilhas reservadas do disco. Se o CP/M for a única coisa que estiver no disquete, o comando DIR mostrará um 'disquete vazio'. Isto ocorre porque o CP/M é um programa especial que não é armazenado como um arquivo. O PIP só irá copiar arquivos. Se for necessário copiar também o CP/M, é necessário usar um comando especial, isto é, o SYSGEN (descrito mais adiante).

Uma dica prática

Sempre que você editar um arquivo, é preferível usar a cópia na próxima vez: a cópia será carregada muito mais rapidamente pelo programa editor do que o disquete original. Isto se deve à técnica utilizada para alocar o armazenamento do disco em blocos. No disquete original, o arquivo é distribuído em blocos não adjacentes. Na cópia, é adequadamente compactado em trilhas comuns; portanto, há um acesso muito mais rápido aos dados (desde que tenha sido usado um disquete novo).



IMPRIMINDO UM ARQUIVO

Pode-se ter duas cópias de COPY.TXT (SAMPLE.TXT e COPY.TXT no *drive* B), mas antes de começar a brincar com o comando ERAse, é necessário aprender a enviar o arquivo para a impressora. Existem duas maneiras muito fáceis de fazê-lo.

A melhor maneira é pressionar simultaneamente as teclas CTRL e P. Agora tecle um "avanço de linha" (line feed). Observe que sua impressora (se estiver ligada) irá pular uma linha. Agora, veja, qualquer coisa que se tecle, e qualquer coisa que apareça na tela também será impressa no papel. Tente teclar e usar o comando DIR para verificar a operação correta da impressora. Depois, use o comando TYPE:

```
A> TYPE SAMPLE.TXT /
_____
_____
_____   } display
_____
A>
```

O comando TYPE imprime seu arquivo tanto na tela como na impressora. Agora, pressione de novo, simultaneamente, as teclas CTRL e P. Acabou-se de desligar a operação de impressão. Use de novo o comando TYPE e seu arquivo aparecerá apenas na tela do terminal (a uma velocidade muito maior).

Este método é fácil, mas oneroso, caso se deseje imprimir vários arquivos. Para imprimir vários arquivos, use uma forma do comando PIP que enviará um ou mais arquivos à impressora:

```
A> PIP LST:=COPY.TXT↵
```

Este comando envia o arquivo COPY.TXT para um "dispositivo de listagem" que é a impressora. 'LST:' sempre é usado como "dispositivo de listagem". Se não funcionar, verifique o comando STAT no Capítulo 2.

Pode-se enviar todos os arquivos do *drive* A à impressora substituindo '*.*' por 'COPY.TXT'. Para imprimir todos os arquivos do *drive* B a partir do *drive* A, use o seguinte comando:

```
A> PIP LST:=B*.*↵
```

APAGANDO ARQUIVOS

Agora que já se tem COPY.TXT no *drive* A, e COPY.TXT e SAMPLE.TXT no *drive* B, pode-se dar o luxo de apagar um deles. Verifique, sempre, em primeiro lugar, o seu diretório para saber o que existe:

```
A> DIR↵
_____
_____   } display
_____
A> B:↵
B> DIR↵
SAMPLE    TXT
COPY      TXT
B>
```

Utilize este comando para apagar o arquivo COPY.TXT:

```
B> ERA COPY.TXT↵
```

Se o arquivo não existir, obter-se-á 'File not Found' ("o arquivo não foi localizado") como mensagem de erro. Desejando apagar outro arquivo ou outro disquete, pode-se especificar a letra do *drive* com o nome do arquivo:

```
B> ERA A:COPY.TXT↵
```

Este comando apaga o arquivo COPY.TXT no *drive* A.

Para apagar *todos* os arquivos em um disquete, use um "filename match" para todos os arquivos:

```
B> ERA *.*↵
```

Este comando irá apagar todos os arquivos no *drive* B. No MP/M e novas versões do CP/M o *filename match* '*.*' irá apenas comparar todos os arquivos na *área atual do usuário* (confira o Capítulo 2 para uma explanação das áreas do usuário e de procedimento de *filename match*).

COMPREENDENDO O CP/M

O mecanismo interno

Já se aprendeu a fazer surgir um sistema CP/M (ou MP/M), criar arquivos, renomear e apagar arquivos, copiar arquivos e disquetes, e imprimir arquivos; mas o que foi que aconteceu na realidade?

O CP/M reage a uma grande variedade de instruções: "Examine este disquete para encontrar um arquivo com este nome, apresente-o, faça uma cópia dele etc.". Examinemos mais de perto duas destas operações: apresentar um arquivo e copiá-lo.

Quando o operador diz ao CP/M para 'TYPE SAMPLE.TXT", está executando uma série de instruções denominadas 'TYPE'. O programa do sistema operacional aceita o que o operador digita e o lê quando pressiona a tecla RETURN. Lê 'TYPE', e depois busca as instruções do programa TYPE. Depois, lê 'SAMPLE.TXT' e sai à procura de um arquivo com este nome. Apenas procura SAMPLE.TXT no *drive* atual porque o operador não o encarregou de procurar em outro *drive*. Uma vez localizado o SAMPLE.TXT, o sistema operacional começa a enviar partes dele para a "unidade de console". O arquivo é enviado bloco a bloco, até que todo o arquivo tenha sido mandado. Como a sua "unidade de console" é o seu terminal, recebe-se o arquivo em sua tela. Se a pessoa permitisse à impressora repetir (ecoar) tudo que é enviado à "unidade de console", o arquivo também seria impresso.

O CP/M é um programa complexo que executa programas utilitários mais simples. O sistema reserva uma área de memória dentro da memória interna do computador para armazenar temporariamente os programas e executá-los. Por exemplo, para copiar um arquivo, deve-se executar o programa PIP. Quando o operador tecla o PIP, o sistema carrega o programa PIP nessa memória interna e começa a executá-lo. O PIP é chamado de comando transitório ou programa transitório. Trata-se de um programa (escrito em linguagem de máquina) que é executado como se fosse um comando, com a exceção de que não faz parte do "núcleo" do CP/M e deve existir como um arquivo no seu disquete com uma extensão '.COM' (PIP.COM). Outros comandos, como STAT, SUBMIT, SYSGEN etc. também são comandos transientes.

Comandos transientes

Comandos transientes (arquivos de comando) na realidade são programas escritos em linguagem de máquina (linguagem Assembler). Depois de ter sido "montado" e testado, um programa de linguagem de máquina foi LOADed (carregado), utilizando o comando LOAD, na memória interna do sistema (denominada TPA – Transient Program Area – Área de Programa Transiente). O comando *load* também providenciou a extensão '.COM',

e o programa se transformou em um comando transiente. Agora já se pode executar o programa digitando apenas o seu nome principal (sem a extensão '.COM').

Por exemplo, PIP.COM é um comando transiente. Execute-o digitando:

```
A > PIP↵
```

Pode-se acrescentar os seus próprios comandos (ou adquiridos) ao disquete, como um processador de palavras, ou uma linguagem como BASIC (CBASIC, MICROSOFT BASIC etc.). Por exemplo, WORDSTAR, um processador de palavras da Micropro International em San Rafael, Califórnia, possui dois comandos transientes: WSU.COM e WSMSG.COM.. O WORDSTAR é executado, digitando:

```
A > WSU↵
```

O WORDSTAR assim se torna outra facilidade dentro do sistema CP/M (ou MP/M). Outro exemplo seria o Microsoft BASIC, que possui o comando transiente MBASIC.COM. Para executar o sistema Microsoft BASIC, basta digitar:

```
A> MBASIC↵
```

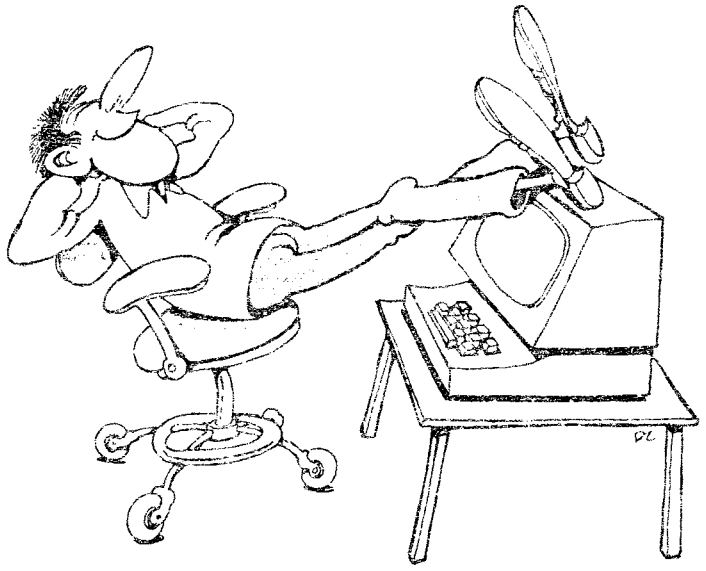
Consulte sempre os manuais fornecidos com os novos softwares para as verdadeiras instruções de execução.

Desligando o sistema

Estando pronto para desligar o sistema, em primeiro lugar retire o disquete-arquivo (no *drive* B) e depois o disquete do sistema (no *drive* A). Não desligue o sistema com os disquetes ainda nos *drives*, porque isto poderá apagá-los.

Depois de retirar os disquetes, pode-se desligar, sem susto, o sistema de computador desligando primeiro os dispositivos e depois o próprio computador.

NOTA: Não desligue o sistema sem primeiro fazer cópias de reserva de quaisquer arquivos novos que tenha criado.



UMA LISTA DE VERIFICAÇÃO DO USUÁRIO

A seguinte lista de verificação do usuário resume as precauções e os procedimentos que o aprendiz sempre deve seguir. São muito importantes. Estude com atenção as instruções desta lista antes de utilizar o seu computador.

LISTA DE VERIFICAÇÃO DO USUÁRIO

LIGANDO O SISTEMA

Assegure-se de que:

- ☐ Os disquetes estão fora dos *drives* ao ligá-lo.
- ☐ O disquete apropriado ao sistema está disponível.
- ☐ Há disponibilidade de um ou mais disquetes em branco.
- ☐ Todos os fios estão adequadamente conectados.
- ☐ Todos os posicionamentos na impressora e no terminal estão corretos.

USANDO O SISTEMA

É importante que o aprendiz:

- ☐ Tenha uma cópia de todos os disquetes que está usando.
- ☐ Reserve freqüentemente seu arquivo em um disco durante a edição.
- ☐ Rotule logo que possível os disquetes, com título, data e conteúdos, usando uma caneta hidrográfica.

USANDO UM NOVO PROGRAMA

O aprendiz deve:

- ☐ Proteger o novo disquete contra gravação, se possível.
- ☐ Fazer uma cópia antes de utilizar seu programa.
- ☐ Arquivar o original do programa novo em um lugar seguro.

SAINDO DO SISTEMA

É importante que o aprendiz:

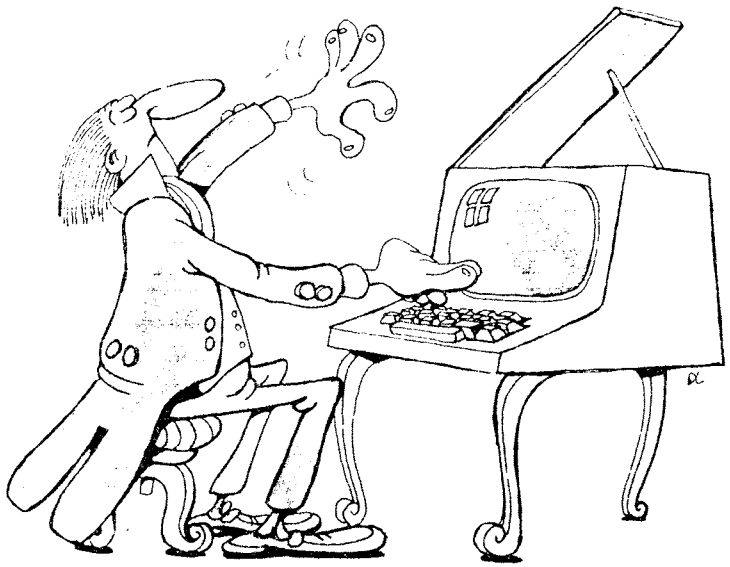
- ☐ Tenha uma cópia de reserva de todos os arquivos novos criados.
- ☐ Retire os disquetes dos *drives*.

RESUMO

Já agora aprendeu-se a ligar e desligar o sistema, a iniciar o CP/M, a usar os comandos e as utilidades básicas do CP/M, como DIR, REN, ERA, PIP, ED, assim como funções especiais, do tipo DELete e CTRL-C.

Aprendeu-se também a listar um arquivo na impressora ou na tela, e a fazer cópias de seus arquivos assim como do CP/M.

O aprendiz se surpreenderá ao saber que agora conhece o suficiente acerca do CP/M para rodar a maioria dos programas de aplicação, sem problemas. Contudo, se quiser saber mais detalhes sobre seu sistema operacional e seus recursos, continue a leitura deste texto.



Capítulo 5

Facilidades do CP/M e do MP/M

INTRODUÇÃO

Este capítulo lhe ensinará todos os comandos do CP/M, incluindo ERA, REN, STAT, DIR e os caracteres de controle. Também abordaremos a montagem, a carga, o despejo (dumping), a execução, a depuração e a salva de programas (ASM, LOAD, DUMP, DDT, SAVE) e a submissão de um arquivo para execução (SUBMIT e XSUB). Apresentaremos um resumo de facilidades de controle e dos comandos disponíveis com CP/M. Cada comando e seu respectivo emprego será examinado em detalhe. Não é necessário, a esta altura, memorizar os comandos, mas o aprendiz deve revê-los, porque poderão lhe ser úteis brevemente.

Mesmo sendo apenas um usuário casual do CP/M, o leitor deve aprender o seguinte:

- Os cinco caracteres de controle mais freqüentemente utilizados (descritos na seção seguinte)
- Como apagar arquivos com ERA
- Como alterar nomes de arquivos com REN
- Como conhecer seu espaço de arquivo com STAT
- Como copiar o seu "sistema" CP/M com SYSGEN

É útil, apesar de não ser absolutamente necessário, ler este capítulo por inteiro, uma vez só. Não obstante, à medida que se emprega o CP/M, pode-se reler este capítulo para se poder tirar todas as vantagens possíveis dos recursos oferecidos pelos CP/M e MP/M.

O material apresentado neste capítulo capacitá-lo-á a empregar todos os comandos do CP/M (e a maioria dos comandos do MP/M). Provavelmente o leitor sentirá necessidade de consultar sempre este capítulo, até que se familiarize completamente com as ações e as convenções de comando do CP/M. Passará, então, a utilizar o Capítulo 6 como um guia útil de consulta.

As descrições dos comandos são baseadas, principalmente, na versão 1.4 do CP/M. A versão 2.2 do CP/M possui alguns aperfeiçoamentos, ao passo que a versão 1.0 do MP/M possui muitos deles. Como a maioria dos usuários possui a versão 1.4, focalizaremos nossas discussões e exemplos do CP/M nesta última versão e descreveremos a versão 2.2 e o MP/M, mais tarde, em uma seção especial deste capítulo. Se o leitor estiver usando a versão 2.2 do CP/M ou então MP/M, deve ler, em primeiro lugar, a seção a que acabamos de nos referir. A versão 2.2 do CP/M e MP/M, com os seus comandos aperfeiçoados, também estão incluídas no resumo e ainda no Capítulo 6.

Descreveremos, inicialmente, as convenções empregadas pelo CP/M: caracteres de controle, comandos embutidos, comandos transientes e nomes de arquivos. Depois, exa-

minaremos cada um dos comandos embutidos: DIR, TYPE, REN, ERA e SAVE; e os comandos transientes: SYSGEN, PIP, ED, STAT, ASM, LOAD, DUMP, DDT, SUBMIT e MOVCPM. Também serão descritos vários aspectos importantes da versão 2.2 e do MP/M.

COMANDOS

Para descrever formatos de comandos no decorrer da nossa explanação, convencionalmente cada comando será apresentado em letras MAIÚSCULAS. "Os argumentos" que devem ser fornecidos com os comandos serão apresentados em letras minúsculas. Se os "argumentos" estiverem em *italic*, serão opcionais. Se dois "argumentos" estiverem incluídos entre chaves { }, então o aprendiz deve escolher um deles. Estas convenções também são usadas no Capítulo 6.

O símbolo ↵ representa a tecla RETURN, às vezes denominada CR, que confirma o comando para o sistema (isto é, faz com que o comando seja executado) e também move o cursor para a linha seguinte (isto é, gera um retorno do cursor e avanço de linha). O símbolo ↑ representa a tecla CTRL (controle), e é usado para simbolizar a pressão sobre a tecla CTRL enquanto se pressiona, simultaneamente, outra tecla (por exemplo, ↑ X significa "pressione CTRL e também a tecla X").

Comecemos por introduzir os caracteres de controle. Eles estão descritos na Figura 2.1.

Os caracteres de controle podem ser usados ao digitar qualquer linha de comando CP/M. Por exemplo, admitamos que o operador comece digitando:

```
A > PIP B:NEW.NCD =
```

Contudo, o que queria, na realidade, era digitar NAD, não NCD. Se pressionar a tecla DEL três vezes, poderá apagar os três últimos caracteres:

```
A > PIP B:NEW.NCD == DC
```

e completar o comando, exceto para RETURN:

```
A > PIP B:NEW.NCD == DCAD = A:OLD.NAD
```

Isto pode lhe parecer confuso, mas tente fazê-lo no seu computador. Use CTRL-R e obterá uma linha limpa, redigitada:

```
A > PIP B:NEW.NAD = A:OLD.NAD↵
```

O comando digitado agora está correto. Pode-se teclar RETURN, e o comando será executado.

Igualmente, se o operador digitou um comando incorreto, pode apagar a linha inteira com CTRL-U (Veja a Figura 2.1). Os cinco caracteres de controle disponíveis para entrada em todas as versões do CP/M estão resumidos na Figura 2.2. Não deixe de se familiarizar com todos eles.

Voltando à Figura 2.1, observe que ela faz uma listagem de todos os caracteres de controle disponíveis, não apenas para o CP/M, mas para todas as versões CP/M ao usar os

Função	Tecla(s) a pressionar	Operação
Apagar o último caractere digitando	RUBOUT (também chamado (DELETE) CTRL – H +	O cursor repete o caractere (ecoa)
Apagar toda a linha	CTRL-U	O CURSOR se movimenta para a próxima linha e apresenta '#' para sinalizar que está pronto para um novo comando
	CTRL-X +	O cursor se movimenta para trás para o início da linha, apagando a linha
Redigitar a linha de comando atual	CTRL-R	Apresenta uma linha limpa Utilizado após correções
Transmitir (executar) a linha de comando	RETURN (às vezes denominada CR) ou CTRL-M ou CTRL-J (LINE FEED)	O comando atual é executado O cursor se movimenta para a próxima linha aguardando novo comando
Digitar uma linha de comando longa (mais longa do que o comprimento da linha do terminal)	CTRL-E	O cursor se movimenta para a próxima linha sem nenhum comando de transmissão ou execução
Reiniciar o ("warm boot") sistema	CTRL-C	Dá nova partida ao CP/M Faz parar uma série de processos que estão sendo executados
Reinício do ("warm boot") sistema para permitir gravação em um disquete que acabou de ser inserido	CTRL-C	Permite gravar em um novo disquete que substitui o disquete anterior
Abortar um processo (programa em MP/M)	CTRL-C + +	Aborta o processo e dá nova partida no MP/M
Terminar operações PIP (cópia)	RETURN (CR) ou CTRL-M	Termina o PIP Faz voltar o controle para o CP/M ou MP/M

\+ indica versão 2.2 CP/M e MP/M apenas
\+ + indica apenas MP/M

Figura 2.1: *Caracteres de controle*

Função	Tecla(s) a pressionar	Operação
Terminar operações DDT (depuração)	GO seguido de RETURN	Devolve o controle ao CP/M ou MP/M
Terminar operações ED (programa editor)	E, seguido de RETURN	Salva o texto no arquivo fonte e na memória intermediária
Terminar operação ED (inserção)	CTRL-Z	Devolve o controle ao ED
Desvincular um programa que está rodando (processo) de um terminal	CTRL-D + +	Devolve o controle ao MP/M (o programa roda desvinculado)
Enviar tudo a ser digitado e mostrado no terminal à impressora	CTRL-P	A impressora repete o que se digita e o que é mostrado na tela
Parar de enviar tudo à impressora	CTRL-P	Usar o interruptor On/Off para a impressora
Parar os *displays* rápidos para maior facilidade de leitura	CTRL-S	Pára o *display* até que se pressione outro ↑ S
Iniciar o *display* depois da parada	CTRL-S	Usar o interruptor On/Off para parar o *display*

\+ Indica versão 2.2 do CP/M e MP/M apenas
\+ + Indica MP/M apenas

Figura 2.1: *Caracteres de controle (cont.)*

rubout/delete	suprima e ecoe o último caractere
CTRL-U ou CTRL-X	suprima a linha
CTRL-R	redigite a linha
CTRL-E	continue na próxima linha
CTRL-C	"reboot" do CP/M

(Nota: CTRL-P e CTRL-S também estão disponíveis para controle da impressora).

Figura 2.2: *Resumo dos controles de edição*

comandos PIP, ED e DDT. É importante conhecer os comandos disponíveis do CP/M. Dependendo da versão (CP/M ou MP/M), há pelo menos cinco comandos embutidos:

TYPE
DIR
REN
ERA
SAVE

e vários comandos "transientes" padronizados, que precisam estar presentes como arquivos no disquete do sistema para que possam ser executados quando chamados. Estes comandos transientes são:

SYSGEN
ED
PIP
ASM
LOAD
DUMP
DDT
SUBMIT
MOVECPM
STAT

Todos estes comandos estão listados na Figura 2.3. Cada um deles será descrito, detalhadamente, neste capítulo.

Manuseando os dispositivos	Comando	Operação
Apresente e altere dispositivos de atribuição	STAT ↵ Dispositivos de atribuição/ STAT ↵	*NOTA: STAT VAL: ↵ irá apresentar todos os comandos dos STAT possíveis
Passe de um *drive* de disco para outro	d: ↵	d é o símbolo do novo *drive* (A, B, C, D)
Copie de um disco para outro	PIP ↵ Destino de PIP = Fonte ↵	Peripheral Interchange Program Destino *filenames.*
Imprima, perfure, copie, combine e execute outras operações sobre arquivos com dispositivos	Parâmetros PIP	Veja explicação a respeito do PIP

Figura 2.3: *Comandos do CP/M*

Manipulando os dispositivos	Comando	Operação
Faça sair os arquivos para a impressora + +	SPOOL *filename* *filename* . . . ↵	
Faça parar e suprima a fila do "spool" + +	STOPSPLR ↵	
Apresente o número do console (terminal) + +	CONSOLE ↵	
Capacite o operador do sistema a trocar os discos + +	DSKRESET ↵	O comando pergunta aos outros usuários se é possível efetuar uma troca de disco
Faça uma cópia do sistema CP/M (faça um novo disquete do sistema)	SYSGEN ↵	Dá partida ao programa SYSGEN
Crie uma versão diferente do sistema CP/M (reconfigure para um tamanho diferente de memória)	MOVCPM ↵	Dá partida ao programa MOVCPM
Faça uma cópia do sistema MP/M ou reconfigure o sistema + +	Use os comandos MPMLDR ou SYSGEN *apenas* depois de ler a informação essencial no "Guia de alteração do MP/M" da documentação da Digital Research	
Apresente o estado de execução do processamento do sistema MP/M + +	MPMSTAT ↵	Apresenta informação relativa ao processo
Altere a área do usuário +	USER *n* ↵	*n* é o número da área do usuário — veja a seção "Áreas do usuário" neste capítulo Sem *n*, USER apresenta a área atual do usuário
Apresente ou marque a data e a hora + +	TOD ↵ TOD*mm/dd/yy* *hh:mm:ss* ↵	Apresenta a hora e data Marca a hora e data
+ Indica versão 2.2 do CP/M e MP/M apenas + + Indica MP/M apenas		

Lidando com arquivos	Formato do Comando	Operação
Criando um arquivo em um disco	ED *filename* ↵	O programa editor do CP/M cria arquivos-texto Pode-se também usar qualquer outro programa editor
Alterando o nome de um arquivo	REN newname = oldname ↵	Altera o nome do arquivo
Apagando (suprimindo) arquivos	ERA { *filename* / *filename match* } ↵	Procura um *filename* ou *match* para um "filename match string"

Figura 2.3: *Comandos do CP/M (cont.)*

Lidando com arquivos	Formato do comando	Operação
Copiando um arquivo	PIP newcopyname = = oldcopyname ↵	Copia um arquivo e lhe dá um nome
Copiando de um *drive* para outro (cópia simples)	PIP d: newcopyname = d: oldcopyname (onde d é a letra do *drive*) (consulte a descrição do PIP para abreviações)	
Fazendo muitas operações de cópia	PIP ↵ *newcopyname = oldcopyname ↵ *d:newcopyname = d: oldcopyname ↵ * ↵	Use RETURN para terminar
Apresentando o conteúdo de um arquivo-texto	TYPE (*filename*)	São apresentados os conteúdos do arquivo. Utilize P para imprimi-los
Modificando os conteúdos do arquivo-texto	ED (*filename*)	O programa ED permite que se editem arquivos-texto
Fazendo uma listagem dos nomes dos arquivos no diretório	DIR { *filename* / *filename match* }	Por si mesmo, DIR lista todos os nomes de arquivo
Apresentando os tamanhos do arquivo e do disco	STAT { d: *filename* / d: *filename match* } STAT *d:*	Este comando apresenta os tamanhos dos arquivos e o espaço ocupado Este comando apresenta o disco atual, ou, opcionalmente, o espaço d do disco que está livre
Apresentando os atributos do arquivo (indicadores) +	STAT *d: filename*	Apresenta atributos como R/O para leitura apenas ou SYS para o arquivo do sistema
Criando um programa em linguagem assembler	ED *filename*	O programa editor do CP/M cria arquivos-texto. Qualquer programa editor fundamentado em CP/M pode ser empregado para criar arquivos-texto
Lidando com programas	**Comando**	**Operação**
Monte um programa em linguagem assembler	ASM *filename*	Cria um "arquivo-objeto" em linguagem de máquina

Figura 2.3: *Comandos do CP/M (cont.)*

Lidando com programas	Comando	Operação
Produza um programa relocável	GENMOD *filename* HEX *filename*. PRL *$bytes* ↵	Produz um arquivo PRL a partir do arquivo HEX, com memória extra opcional expressa em dígitos hexadecimais para *$bytes* (consulte o Capítulo 3)
Crie um novo comando transiente	LOAD *filename*	Cria um arquivo de comando executável com uma extensão .COM a partir de um "arquivo-objeto" em linguagem de máquina
Execute um comando ou programa transiente	programname ↵	Programname é o nome de arquivo de extensão .COM sem o '.COM'
Imprima um "arquivo-objeto" (arquivo em linguagem de máquina)	DUMP *filename* ↵	Utilizado para arquivos hex
Reserve uma cópia de um comando ou programa transiente (após tê-lo carregado ou executado)	Save p *filename* ↵	p é o número de páginas. Cada página corresponde a 256 bytes
Depure um programa (utilizando o depurador CP/M ou MP/M)	DDT *filename* RDT *filename*	RDT é o nome do depurador relocável do MP/M
Submeta um lote de comandos (programas)	SUBMIT *filename* parm 1 parm 2 . . .	parm representa parâmetro. Este comando procura um arquivo de muitos comandos, substitui parâmetros e executa os comandos. O sistema apresenta os comandos que estão sendo executados (a não ser que se use + XSUB). Pode-se abortar a operação em qualquer momento que se desejar, empregando RUBOUT. + XSUB', ao ser utilizado como primeiro comando no arquivo submetido, irá processar os comandos e aceitará entradas para programas (se os programas forem elaborados para entrada com memória intermediária)
Faça um esquema cronológico para a execução de programas	SCHED mm/dd/yy hh:mm	mm/dd/yy é a data e hh:mm representa a hora

Figura 2.3: *Comandos do CP/M*

É necessário compreender-se dois conceitos para se utilizar os recursos do CP/M e seus comandos padronizados de maneira eficiente:

– Comandos embutidos vs. comandos transientes
– Convenções a respeito de denominação de arquivos

Passemos a examiná-los agora.

COMANDOS EMBUTIDOS VS. COMANDOS TRANSIENTES

Como já se sabe, ao ver o *prompt* do sistema CP/M (uma letra que indica o *drive* do disco, seguida de um colchete de ângulo reto, por exemplo, 'A > '), pode-se teclar qualquer comando embutido, e o computador responderá imediatamente. Simbolicamente, todos os comandos assumem a seguinte forma:

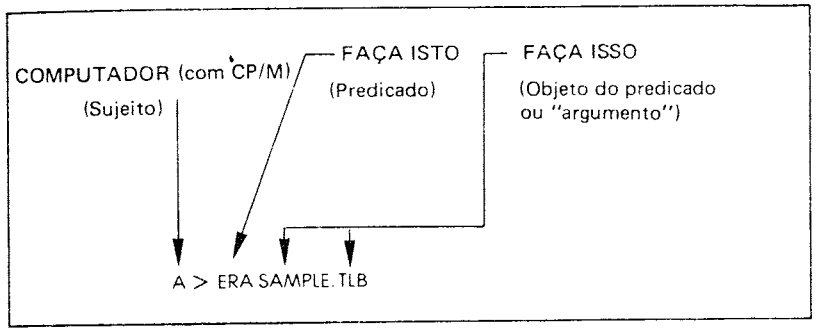


Figura 2.4: *"Computador, apague o arquivo SAMPLE.TLB."*

Alguns comandos são simples, outros complexos. Na realidade, todos os comandos são programas em linguagem assembler (escritos em uma linguagem que o computador "entende"). Alguns são "embutidos" (não listados no diretório) e outros "transientes" (listados no diretório). O operador pode executar os cinco comandos "embutidos" (DIR, ERA, REN, SAVE e TYPE) em qualquer ocasião, pois eles estão embutidos no programa de operação do sistema CP/M. No que se refere aos comandos transientes, só se poderá executá-los se existirem sob a forma de arquivos de comando (.COM) no seu disco. Um comando transiente nada mais é do que um programa, em linguagem assembler, que pode ser copiado, suprimido, movimentado e continuar a ser executado como um comando. O aprendiz pode criar o seu próprio comando transiente (programa) se souber como programar um computador. O nome de arquivo de um comando transiente sempre tem uma extensão '.COM' (por exemplo, PIP.COM), mas não há necessidade de digitar o '.COM' quando ele for executado como comando. Se desejar copiar, suprimir, ou movimentar o arquivo, não obstante, é necessário que especifique o nome inteiro do arquivo com a extensão '.COM'.

Para descobrir se o comando transiente que o aprendiz deseja se encontra em seu disco, utilize o comando DIR. Os comandos transientes padronizados que o CP/M fornece são os seguintes: ASM, ED, DUMP, LOAD, PIP, MOVCMP, STAT, SYSGEN e SUBMIT.

Os comandos do MP/M são *todos* transientes, ou "programas residentes", como descrevemos na seção especial "CP/M versão 2.2 e MP/M". Apesar de descrevermos todos os comandos transientes, o aprendiz provavelmente só irá utilizar alguns deles. A Figura 2.5 apresenta um diretório de sistema típico.

Diretório do sistema	Sistema necessário	Muito recomendado	Útil	Opcional
(CP/M existe mas é invisível para DIR)	✓			
SYSGEN	✓			
PIP	✓			
STAT		✓		
ED			✓	
MOVCPM				✓
ASM				✓
LOAD				✓
DUMP				✓
DDT				✓
SUBMIT				✓
XSUB				✓
WORD PROCESSOR			✓	
YOUR APPLICATION PROGRAMS		✓		

NOTAS:
1. Como medida de precaução, não armazene dados ou arquivos-texto no seu disquete do sistema.
2. Utilize, ainda, a característica proteção/gravação no disquete do sistema.

Figura 2.5: *Um diretório típico de disquete do sistema*

NOMES DE ARQUIVO

Cada diretório do sistema contém um nome de arquivo para todos os arquivos do disquete. Um exemplo de formato de nome de arquivo é NAME.TTT, onde NAME pode ter até oito caracteres, e .TTT representa o tipo de extensão. Os nomes de arquivo podem conter letras, números e caracteres especiais. Contudo, não podem conter os seguintes símbolos:

< > . , : = ; * ? [ ]

Por exemplo:

PROG/22.BAK é legal
PROG = 22.BAK é ilegal

Extensões

Os nomes de arquivo em geral possuem extensões (três caracteres à direita de um ponto). Essas extensões servem para diferençar diversos tipos de arquivos. Para vários tipos de arquivos, as extensões são *exigidas;* outras extensões são empregadas apenas por conveniência (veja Figura 2.6).

Extensão	Tipo	Exemplo
COM	Arquivo de comando exigido de um comando transiente (programa)	PIP.COM LOAD.COM
ASM	Exigido para arquivos-fonte (texto) em linguagem assembler utilizados com o comando ASM	PROG1.ASM PATCH.ASM
PRN	Exigido para o arquivo de listagem do programa em linguagem assembler	PROG1.PRN PATCH.PRN
PRL	Exigido para os programas relocáveis MP/M	RDT.PRL
HEX	Exigido para um arquivo de programa em formato 'hex' (linguagem de máquina) pronto para ser carregado	PROG1.HEX PATCH.HEX
RSP	Exigido para "programas residentes de sistema" MP/M	SPOOL.RSP
BAS	Exigido para arquivos-fonte (texto) de programas BASIC	PROGBAS.BAS
INT	Exigido por arquivos intermediários de programas BASIC, prontos para execução (já compilados)	PROGBAS.INT
BAK	Criado por ED (editor de texto) como cópia-reserva (back-up) de um arquivo antes de alterar o mesmo	LETTER.BAK
$$$	Arquivos temporários (auxiliares) criados e normalmente apagados por ED e outros programas	LETTER.$$$
SUB	Arquivo-texto com comandos ou programas CP/M embutidos ou transientes; a serem executados na modalidade de lote pelo programa SUBMIT	TRANSFORM.SUB

Figura 2.6: *Tipos de extensão predefinidos*

Sempre que utilizar um nome de arquivo como argumento para um comando, é necessário que o aprendiz digite o nome do arquivo *completo*, incluindo a extensão, se ela existir. A única exceção a esta regra é a seguinte: quando usar o arquivo como comando transiente (este arquivo deve possuir uma extensão .COM internamente, mesmo que tal extensão não necessite ser digitada para executar o comando).

Associação de nomes de arquivo

Em determinadas ocasiões a pessoa poderá querer atuar sobre vários ou todos os arquivos ao mesmo tempo. Se o comando permiti-lo, seu formato indica que um *filename match* (associação de nomes de arquivo), abreviado *filematch*, pode servir de substituto para um *filename* real, funcionando como se fosse um argumento para o comando. Por exemplo, o formato para o comando ERAse é:

ERA { filename / filematch }

Isto significa que a pessoa deve escolher entre um nome de arquivo real ou um *filematch* como argumento para o ERAse.

Um *filename match* é um grupo de caracteres utilizado para se referir a vários arquivos ao mesmo tempo. Os caracteres podem ser letras, dígitos, um ponto, e dois símbolos especiais: '*' e '?'. Estes caracteres e símbolos podem ser utilizados para selecionar, de acordo com a nossa conveniência, os nomes (arquivos de vários arquivos) que serão afetados pelo comando. Por este motivo, é importante escolher caracteres que sejam comuns a todos os nomes de arquivos.

Tente usar o símbolo '?'. Ele poderá se associar com qualquer caractere, desde que seja *um para cada '?'* e *na posição exata* na qual o símbolo '?' foi colocado. Aqui estão alguns exemplos:

Isto	Se associará com estes nomes de arquivo	Mas não com estes
S?MPL? A?B?C	SAMPLE SIMPLE SIMPLY AABBC ACBCC	SIMPL SAMPLEY AABBCC ABCCC

Nem o '*' nem '?' podem ser associados com um ponto. Abaixo apresentamos alguns exemplos de símbolos múltiplos em *filename matches*:

Isto	Se associará com estes nomes de arquivo	Mas não com estes
T???Y.*	TEDDY.COM TARBY.ASM TILLY	TINY.COM TARABY.ASM TONY
?????????.??? or *.*	any filename	

Lembre-se de que no Capítulo 1 utilizamos um *filename match* para copiar um disquete completo. O comando usado foi:

```
A > PIP B:=A:*.*[OV] ↵
```

onde '**' representava "todos os arquivos".

NOTA: Utilizar o símbolo '*' no início do nome de arquivo '*AB.*' é perigoso! Ao invés disto, use ?AB.* Tente evitar misturar '?' e '*' em associação de nomes de arquivos. Os *filename matches* são geralmente empregados com o comando PIP para fazer cópias de vários ou todos os arquivos ao mesmo tempo (ou para enviar vários arquivos para um dispositivo).

ESPAÇOS EM BRANCO

O CP/M padrão exige que um comando seja seguido de pelo menos um espaço em branco antes de utilizar quaisquer argumentos. No exemplo anterior:

```
A > PIP B:=A:*.*[OV] ↵
```

O PIP deve ser seguido de um ou mais espaços em branco (ele também pode ser precedido por um espaço em branco). O argumento 'B:' pode ser seguido de um espaço em branco, mas isto é opcional.

Da mesma forma, um espaço em branco pode ser usado, opcionalmente, depois de '=' ou depois de 'A:'. Em algumas versões do CP/M (como a da North Star), *todos* os espaços em branco são opcionais. No entanto, ao usar uma versão "padrão" de CP/M, é essencial não esquecer que os espaços em branco, seguem, obrigatoriamente, qualquer comando.

Podemos empregar espaços em branco adicionais para separar subelementos de um comando. Abaixo apresentamos um exemplo específico:

```
A > PIP B: = A: * .*[OV] ↵
```

COMANDOS EMBUTIDOS

Introdução

Certamente o leitor se lembra de que no Capítulo 1 apresentamos, sucintamente, todos os comandos embutidos padronizados do CP/M, exceto o SAVE. Agora forneceremos exemplos específicos de como usá-los.

DIR (Diretório)

O comando diretório é usado para apresentar uma lista de todos os arquivos presentes no disquete. Para fazer uma listagem de todos os arquivos no disquete do *drive* A, tecle:

```
A > DIR ↵
```

Para fazer o mesmo com todos os arquivos do disquete no *drive* B, tecle:

```
A > DIR B: ↵
```

(Este método é mais veloz, e o aprendiz permanece no *drive* A). Também é possível digitar a seqüência:

```
A > B: ↵
B > DIR ↵
```

(Este método é mais lento, mas o aprendiz permanece no *drive* B depois de executar o último comando). Para procurar um arquivo específico no disquete do *drive* B, digite:

```
A > DIR B:SPECIFIC.NAD ↵
```

O método a usar para listar todos os arquivos NAD no disquete do *drive* A é digitar:

```
A > DIR *.NAD ↵
```

Utilize o comando DIR com freqüência para verificar quais os arquivos que estão no disquete. O DIRectory também deveria ser verificado sempre que um arquivo for criado ou suprimido.

Com a versão 2.2 do CP/M, a listagem do diretório é apresentada em um formato de quatro colunas, como mostra o exemplo abaixo:

```
A> DIR ↵
A : MOVCPM  COM : ASM     COM : DDT     COM : DUMP    COM
A : ED      COM : LOAD    COM : PIP     COM : STAT    COM
A : SUBMIT  COM : SYSGEN  COM : XSUB    COM : DISKDEF LIB
A : DUMP    ASM : SINGLE  ASM : COPY    COM : LIST    COM
A : FORMAT  COM : SAVEUSER COM : USER   ASM : MEMR    COM
A : FILECOPY COM : SETDRIVE COM : READ-ME DOC : CONFIG COM
```

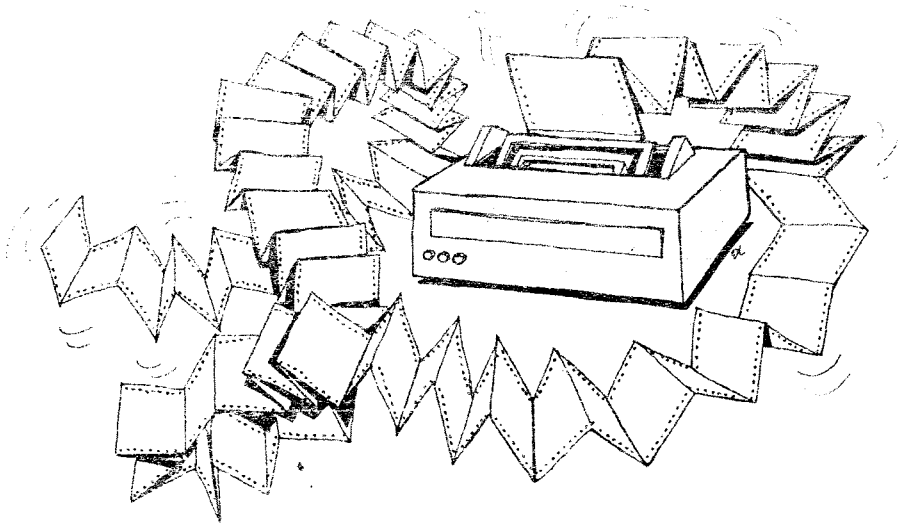
TYPE

O comando TYPE pode ser utilizado para apresentar qualquer arquivo ASCII na tela. Este é um método rápido e conveniente de examinar qualquer arquivo-texto. O formato é:

```
TYPE d:name.type
```

onde o nome do *drive*, d, é opcional. Caso deseje que o arquivo seja impresso, aperte CTRL-P antes de dar o comando. Não se esqueça de que a utilização da impressora irá diminuir a velocidade do *display*.

O comando TYPE pode ser empregado para verificar qualquer texto que se tenha criado. TYPE, apesar de não oferecer um formato conveniente como o de um programa de processamento de palavras ou de uma listagem de mala direta, imprime o arquivo assim como está armazenado na memória do computador. Geralmente TYPE é um meio rápido de examinar um arquivo-texto que foi acidentalmente danificado. Representa uma ferramenta de grande valor para examinar na tela um arquivo recém-criado.



REN (Rename)

Este comando permite-lhe alterar o nome de um arquivo. O seu formato é:

```
REN new = old
```

Por exemplo, admitamos que FILE11.TXT está no disquete do *drive* A. Se digitarmos:

```
A > REN FILE23.TXT = FILE11.TXT ↵
```

FILE11 passará a se denominar FILE23 e somente FILE23 estará no disquete do *drive* A. O nome FILE11 foi descartado mas o arquivo, em si, permanecerá intacto.

A especificação do *drive* é opcional. Por exemplo, a instrução:

```
A > REN B:FILE.TXT = FILE.BAK ↵
```

é equivalente a:

```
A > REN B:FILE.TXT = B:FILE.BAK ↵
```

Observe que ambos os arquivos devem residir no mesmo *drive*. A especificação do *drive* pode aparecer à esquerda ou à direita do sinal de igualdade, e o novo *filename* pode não existir ainda.

Uma prática aconselhável, ao trabalhar-se com um arquivo, é a de incluir a data no nome. Desta forma, por exemplo, o arquivo PUB pode ser rotulado PUBMAY21. Da próxima vez que for atualizado, o arquivo pode ser RENamed (redenominado) PUBMAY23.

No final de uma sessão com o computador, nunca deixe de REName o arquivo no qual estiver trabalhando, com o objetivo de:

1. Dar-lhe um nome que o identifique
2. Evitar possíveis confusões
3. Incluir a data

ERAse

ERA é usado para apagar um arquivo. O seu formato é:

```
ERA d:name
```

onde d é um nome de *drive* opcional. Abaixo apresentamos um exemplo de um comando ERA:

```
A > ERA PROG.TXT ↵
```

no qual PROG.TXT é apagado.
NOTA: O comando ERA pode ser perigoso. Utilize-o com cuidado!

ERA também pode ser empregado para apagar mais de um arquivo de cada vez. Por exemplo, se a pessoa desejar apagar todos os arquivos com a extensão '.ASM' em seus nomes de arquivo, deve substituir os nomes dos arquivos por um símbolo de associação.

```
A > ERA *.ASM ↵
```

Neste caso o *filename match* é '*.ASM' que se associa com todos os *filenames* que terminam com '.ASM'. O símbolo '*' é associado com qualquer número de caracteres, incluindo o caso de não haver caractere nenhum. Observe que em cada nome de arquivo só pode haver um ponto. Os últimos três caracteres no nome (ASM) só se associam com as letras 'ASM' que aparecem nessas posições exatas, a saber, as primeiras três posições à direita do único ponto existente. Portanto, neste exemplo, '*ASM' associar-se-á com 'SAMPLE.ASM', 'ANOTHER.ASM', '1.ASM' e '.ASM'. *Não* se associará com 'SAMPLE', 'ANOTHER.PRG', 'ASM.COM' ou 'ASM' (observe que o último 'ASM' não é uma extensão, porque não está situado nas três posições à direita de um ponto).

Não utilize o comando ERA para praticar associações de nomes de arquivo. Use, ao invés dele, o comando DIR. DIR irá listar apenas aqueles arquivos que se associam com o seu *filename match*.

Na prática, a pessoa poderá, muitas vezes, desejar criar múltiplas cópias de um arquivo, como VERSION1, VERSION2, VERSION3, ou COPY1, COPY2. Para evitar confusões, lembre-se de ERAse (apagar) todas as cópias desnecessárias antes de deixar o terminal.

SAVE

O comando SAVE é empregado para armazenar informações retiradas do TPA (memória principal). Este comando é mais complexo e será descrito, com maiores detalhes, mais adiante neste capítulo.

OS COMANDOS TRANSIENTES

Introdução

Os dez comandos transientes padrão do CP/M são: SYSGEN, PIP, ED, STAT, ASM, LOAD, DUMP, DDT, SUBMIT e MOVCPM. Todos são fornecidos com o CP/M, na forma de arquivos .COM. Serão descritos separadamente.

SYSGEN

Lembre-se de que no Capítulo 1 explicamos que SYSGEN é utilizado para copiar CP/M de um disco para outro. Mais especificamente, é usado para copiar um disquete de sistema. Quando o aprendiz recebe, pela primeira vez, um disquete do sistema e o utiliza para chamar o seu sistema CP/M, deve fazer, imediatamente, uma cópia do sistema, e depois armazenar o disquete do sistema em um lugar seguro. Pode-se empregar o comando transiente PIP para transferir todos os arquivos '.COM' (e '.SYS') para um novo disquete (consulte o Capítulo 3), mas também é necessário copiar "o sistema" que reside em trilhas reservadas no disquete, para que se possa transformar o novo disquete em um "disquete de sistema". Pode-se utilizar o comando (programa) transiente SYSGEN para fazê-lo.

O programa SYSGEN transforma um disquete comum em um "disquete de sistema" (no qual "o sistema" reside em trilhas reservadas). Para fazê-lo, SYSGEN traz "o sistema" *para a memória* a partir do disquete original (*para a memória* significa que é trazido para a área de execução de programas do computador, denominada TPA-Área Transiente de Programa), e depois o grava no disquete. SYSGEN traz "o sistema" para a memória e o movimenta para um novo disquete; ou, SYSGEN transfere o que já está na memória para outro disquete.

A seqüência para empregar SYSGEN é a seguinte:

```
A > SYSGEN ↵
    SYSGEN VERSION n.n
    SOURCE DRIVE NAME (OR RETURN TO SKIP) A
```

(Se o CP/M foi copiado para a memória, utilizando MOVCPM, por exemplo, o operador pode teclar RETURN em vez disto)

```
SOURCE ON A THEN TYPE RETURN ↵
```

(isto pretende lhe dar tempo para inserir um disquete com o CP/M no *drive* adequado)

```
FUNCTION COMPLETE
```

(o CP/M agora está na memória, pronto para ser gravado em um disco)

```
DESTINATION DRIVE NAME (OR RETURN TO
REBOOT) B
```

(coloque o novo disco em B)

```
DESTINATION ON B THEN TYPE RETURN ↵
```

(o CP/M está gravado em B)

```
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO
REBOOT) ↵
```

(O operador pode, agora, teclar *B* e fazer mais cópia. Neste caso, termine teclando RETURN. ↵ .)

```
A>
```

Lembre-se de que o disquete no *drive* B agora contém apenas o CP/M. Não possui arquivos (admitindo-se que se tratava de um disquete virgem). O CP/M pode ser copiado para um disquete que já possui arquivos; isto não irá danificar os arquivos. Se o disquete no *drive* B estiver vazio, o operador deve agora copiar os arquivos separadamente. Caso desejar copiar todos os arquivos do disquete no *drive* A para B, digite:

```
A > PIP B:=A:*.*[OV] ↵
```

Também pode copiar os arquivos desejados um de cada vez, à medida em que for necessário. O formato para essa operação é explicado no Capítulo 3.

Nesse momento o aprendiz deve testar o seu novo disquete de sistema.

Usuários do MP/M: é importante lembrar que, no exemplo acima, que utiliza o comando PIP para copiar vários arquivos de uma só vez, irá apenas copiar os arquivos na área atual do usuário. O disquete do sistema fornecido pode ser copiado empregando essa modalidade de PIP, pois todos os arquivos necessários estão na área zero do usuário, que é a "área atual do usuário" se o aprendiz *nunca utilizar* o comando USER. (Recomendamos não empregar a área do usuário ou o comando USER a não ser que tenha um sistema de múltiplos usuários.) (Consulte a seção "Aperfeiçoamentos da versão 2.2 do CP/M" no Capítulo 3).

SYSGEN pode ser usado para fazer uma cópia do sistema que pode ser deixada na TPA (Área Transiente de Programa) e *não* transferida para um novo disquete. Para fazer isto, apenas pressione 'RETURN' (para fazer o 'reboot') quando aparecer a mensagem 'DESTINATION DRIVE NAME (OU RETURN TO REBOOT)'.

Pode-se também empregar SYSGEN em conjunção com MOVCPM para configurar uma nova versão do seu sistema com um tamanho diferente de memória. Neste caso, quando aparecer a primeira mensagem 'SOURCE DRIVE NAME (OR RETURN TO SKIP)', apenas pressione RETURN (↵) e SYSGEN admite que o sistema já está carregado na memória (a TPA). Ao usar MOVCPM, o operador pode carregar o sistema na memória para se preparar para este tipo de SYSGEN. Isto é descrito no Capítulo 5.

PIP

PIP é o programa de transferência de arquivos, explicado, com detalhes, no Capítulo 3.

ED

ED é o editor, apresentado detalhadamente no Capítulo 4.

STAT

Usando o STAT para Display

STAT é usado para apresentar o estado ou para mudar a alocação de dispositivos.

O comando STAT é um meio fácil de apresentar o espaço disponível no disco. Também pode ser empregado para apresentar e modificar alocações de dispositivos.

O comando 'STAT' simples apresenta o tamanho do espaço restante do disco e o espaço alocado para os arquivos. A forma mais simples do comando STAT é:

```
A > STAT ↵
A:R/W, SPACE: 144K
A >
```

Este *display* mostra que existem 144K bytes restantes no disquete no *drive* A, isto é, 144K bytes que não estão sendo usados. A maioria dos disquetes de 200 mm possuem apenas 224K bytes de espaço para arquivos. 'R/W' lhe diz que é possível *gravar* no disquete (isto é, a pessoa pode criar novos arquivos, regravar, ou suprimir arquivos antigos). 'R/W' representa "read/write", opondo-se a 'R/O' que significa "read-only". Não se pode gravar um arquivo R/O, já que está protegido contra a gravação ("write-protected").

Bytes e Registros

Um *byte* é uma posição de 8 bits na memória (um bit é ou "1" ou "0"). 128 bytes formam um registro (não necessariamente um "registro de dados", mas um registro de arquivo CP/M). Este também é o tamanho de um setor no disco. Oito registros são equivalentes a 1024 bytes (ou 1K). Estes valores, em especial, são usados porque podem ser expressos como potências de 2 (números binários), isto é, podem ser representados por um número correspondente de bits (n bits representam 2^n).

Um registro de 128 bytes é uma convenção útil para ler e gravar seções em um arquivo. Os arquivos existem no disquete como "pedaços" de 16K denominados "extensões". Estas "extensões" não são contíguas (não estão umas ao lado das outras) no disquete. Cada uma contém sempre o endereço do início da próxima, ou seja, em outras palavras, elas estão "encadeadas". O aprendiz nunca é obrigado a achá-las sozinho; o Sistema Operacional Básico de Disco (BDOS), uma parte do sistema integral, lida com a alocação do espaço para arquivos utilizando blocos de controle de arquivos (descritos no Capítulo 5). A alocação do espaço é dinâmica – o sistema aloca novo espaço para o seu arquivo à medida que o aprendiz (ou o seu programa) gravam registros no arquivo. Não é necessário especificar um tamanho máximo.

Atribuindo dispositivos com STAT

O comando STAT oferece informações úteis sobre os discos (disquetes), a memória transiente ou "auxiliar" ("scratch-pad"), também denominada área de programa transiente ou TPA, e as atribuições dos dispositivos do sistema. Também permite que sejam *modificadas* as atribuições dos dispositivos e que os *drives* dos disquetes sejam protegidos contra gravações. Na versão 2.2 do CP/M e nos sistemas MP/M, podem-se posicionar indicadores (como apenas leitura) em *arquivos*, receber informações adicionais a respeito do estado dos tamanhos alocados aos arquivos e remanescentes nos discos e ainda sobre áreas do usuário.

A forma seguinte de STAT pode ser empregada para apresentar uma lista de possíveis atribuições para os nomes genéricos (lógicos) CON:, RDR:, PUN: e LST:

```
A > STAT VAL: ↵
```

O sistema responde apresentando:

```
CON:  = TTY:  CRT:  BAT:  UC1:
RDR:  = TTY:  PTR:  UR1:  UR2:
PUN:  = TTY:  PTP:  UP1:  UP2:
LST:  = TTY:  CRT:  LPT:  UL1:
```

Ao lado do nome de cada dispositivo lógico (CON, RDR, PUN, LST), o STAT apresenta uma lista de nomes físicos possíveis para cada um deles. Esses nomes correspondem aos nomes no sistema Intel MDS-800.

O nome físico do dispositivo pode ser utilizado para um outro dispositivo que desempenha as mesmas funções básicas; assim, por exemplo, o PTP (digitação em fita de papel) pode ser, na realidade, a operação "registro" (gravação) de um gravador de cassete.

As atribuições atuais dos dispositivos podem ser apresentadas a qualquer momento digitando o comando abaixo:

```
A > STAT DEV:↵
CON: = CRT:
RDR: = UR1:
PUN: = PTP:
LST : = TTY:
A >
```

Esta amostra demonstra que o dispositivo CON: é um CRT (tubo de raios catódicos), o RDR: é uma leitora definida pelo usuário (número 1), o PUN: é um dispositivo de perfuração em fita de papel e o LST: é um dispositivo de teletipo.

Ocasionalmente, o operador poderá alterar os quatro nomes físicos dos dispositivos. Tais atribuições podem ser modificadas usando o seguinte comando STAT:

STAT log: = dev:, log: = dev:, . . .

onde *log*: é um nome lógico para o dispositivo (CON:, RDR:, PUN:, ou LST:) e *dev*: é um nome físico (PTP:, CRT:, UR1: etc.). Por exemplo, o comando

A > STAT LST: = LPT:↵

altera o dispositivo LST: para uma impressora de linhas (LPT:), de modo que todas as operações de cópia para LST: irão agora para a impressora de linhas (a não ser que o sistema sofra realocação).

Observe que o dispositivo CON: deve ser um dispositivo de entrada e saída que tanto pode enviar como receber dados (por exemplo, um terminal com um *display* e um teclado). O RDR: deve ser um dispositivo capaz pelo menos de enviar dados (input) e os dispositivos PUN: e LST: devem ser capazes de receber dados (output).

Versão 1.4 do CP/M e versões posteriores

Na versão 1.4 do CP/M e outras subseqüentes, um comando STAT simples apresenta o número de bytes restantes no *drive* atual. Você também pode obter esta informação, para outros *drives*, usando este formato.

```
A > STAT B.
BYTES REMAINING ON B: 192K
A >
```

O *display* mostra que o disquete no *drive* B é 'R/O', o que representa "read-only" (só leitura). Só se pode ler este disquete; qualquer tentativa de gravá-lo resultará na mensagem de erro:

```
BDOS ERR ON B: READ ONLY
```

Quando obtiver esta mensagem na versão 1.4, basta pressionar uma tecla no teclado do terminal (como RETURN), e o disquete será recolocado para 'R/W' ("read-write", ou seja, leitura e gravação). Agora o aprendiz pode ler o disquete e gravá-lo também.

Pode-se recolocar o disquete em 'R/O', executando a seguinte forma do comando STAT:

STAT d: = R/O

onde d: é qualquer *drive* de disco.

Para apresentar o tamanho dos arquivos, use a seguinte forma de STAT:

STAT d: { filename | filename match }

onde d: é uma letra opcional para o *drive* relativa a arquivos que não estão no *drive* de disco atual. Se o operador especificar um *filename* STAT irá apresentar a informação para esse arquivo. Se especificar um *filename match*, STAT irá associar arquivos e os apresentará em ordem alfabética, com a informação relativa ao tamanho. O *display* abaixo é uma amostra:

```
A > STAT B:*.TXT ↵

RECS BYTS EXT D:FILENAME.TYP
8     1K    1   B:SAMPLE:TXT
4     1K    1   B:QUOTE:TXT
16    2K    1   B:CHAP1.TXT
A >
```

O campo RECS mostra quantos registros de 128 bytes foram alocados ao arquivo (até o presente momento), o campo BYTS mostra quantos kilobytes foram alocados ao arquivo (1K representa 1024 bytes e 128 vezes RECS é igual ao número de bytes alocados, de modo que BYTS é igual ao resultado de 128 vezes RECS dividido por 1024). O campo BYTS é o número de alocação mais preciso.

O campo EX mostra o número de extensões de 16K remanescentes no arquivo (o número em BYTS é dividido por 16.). Na versão 1.4, isto pode corresponder ao número de entradas de diretório em um disco para um mesmo arquivo, porque uma entrada de diretório (um bloco de controle de arquivos) só pode endereçar no máximo 16K. O CP/M cria, automaticamente, mais entradas e blocos de controle de arquivos quando o arquivo é ampliado.

Blocos e registros CP/M

O CP/M sempre trabalha com um mínimo de 8 registros ao alocar espaço. Nos disquetes-padrões da IBM, um registro possui 128 bytes. Por este motivo, a quantidade mínima de espaço alocada pelo CP/M é 1K, como mostramos no exemplo STAT acima.

Disquetes de dupla densidade, entretanto, podem possuir 256 registros. Neste caso, a quantidade mínima de espaço alocada pelo CP/M é 2K, mesmo que o arquivo só utilize uma fração pequena deste espaço. Se for usado um disco rígido, a quantidade mínima de espaço pode ser 4K.

STAT na versão 2.2 de CP/M e em versões subseqüentes

O programa STAT para a versão 2.0 ou 2.2 do CP/M possui um certo número de aperfeiçoamentos, incluindo *displays* mais complexos: Se a pessoa apenas digitar 'STAT ↵', o *display* irá corresponder aos das versões anteriores do CP/M, como mostra o exemplo abaixo:

```
A > STAT *.* ↵

Recs  Bytes  Ext  Acc
  64    8K    1  R/W  A:ASM.COM
   8    1K    1  R/W  A:BOOTHD.COM
  20    3K    1  R/W  A:CONFIG.COM
  22    3K    1  R/W  A:COPY.COM
   6    1K    1  R/W  A:SAVEUSER.COM
```


A forma 'STAT VAL: ↵', contudo, produz o seguinte *display*:

```
Temp R/O Disk  : d: = R/O
Set Indicator  : d:filename.typ $R/O $R/W $SYS $DIR
Disk Status    : DSK: d:DSK:
User Status    : USR:
Iobyte Assign  :
CON:   = TTY:   CRT:   BAT:   UC1:
RDR:   = TTY:   PTR:   UR1:   UR2:
PUN:   = TTY:   PTP:   UP1:   UP2:
LST:   = TTY:   CRT:   LPT:   UL1:
```

Observe que as últimas quatro linhas são idênticas às apresentadas na versão 1.4 do CP/M. As linhas acima delas mostram possíveis comandos STAT e aquilo que eles podem fazer. Para colocar o atributo R/O em um *disco* inteiro, use a versão do comando STAT 'd = R/O' (como na versão 1.4 do CP/M) onde d: é um especificador do *drive* (A:, B:, Y:). Para colocar os atributos $R/O, $R/W, $SYS ou $DIR de um *arquivo* deve-se usar o formato de STAT descrito na seção a respeito da versão 2.2 do CP/M e MP/M. Para apresentar o estado do disco ou disquete atual, ou um ou outro *drive*, use o formato 'STAT DSK:' ou 'STAT d:DSK:'. Para apresentar a área atual do usuário (e outras áreas de usuário presentes no sistema) use o formato 'STAT USR:' (daremos um exemplo mais adiante).

Na versão 2.2, pode-se acrescentar o argumento '$S' ao comando STAT para apresentar os tamanhos dos arquivos utilizando o seguinte formato:

STAT d: { filename / filematch } $S

$S é um campo opcional que apresenta o tamanho do arquivo. A pessoa pode especificar um *filename* (incluindo a extensão), ou um *filematch* (associação de nomes de arquivos para vários *filenames*). É também possível especificar, opcionalmente, um *drive* d: para apresentar arquivos em outro *drive* do disco. Aqui estão alguns exemplos:

```
A > STAT PIP.COM $S ↵

Size   Recs   Bytes   Ext   Acc
  55     55     12K     1   R/O A:PIP.COM
```

O campo 'Size' lhe diz quantos registros (unidades de 128 bytes) estão alocados ao arquivo, mas este é um tamanho "virtual", porque o arquivo pode não estar ainda usando este

espaço todo. O campo 'Recs' soma o número de registros em cada extensão (uma extensão representa um bloco de 16K). Se o arquivo tiver sido construído seqüencialmente, estes dois campos seriam idênticos (e corresponderiam ao campo 'RECS' na versão 1.4 do CP/M).

O campo 'Bytes' informa o único número preciso de alocação para arquivos de acesso randômico – o número atual de bytes alocado ao arquivo. Este número corresponde exatamente aos campos 'Size' e 'Recs' dos arquivos seqüenciais. Arquivos de acesso randômico crescem à medida que os dados são neles gravados, de modo que o campo 'Bytes' apresenta o tamanho, em bytes, alocado em um dado momento, e o campo 'Recs' soma o número de registros em cada ampliação – apesar do fato de que uma ampliação possa ter "buracos" sem alocação que ainda não foram preenchidos com dados. O campo 'Size' é mais preciso para o "número lógico de registros" em um arquivo.

O campo 'EXT' lhe diz quantas extensões (blocos de 16K) são alocadas ao arquivo. Diferentemente da versão 1.4, uma entrada de diretório (bloco de controle de arquivo) em um disco pode endereçar até 128K bytes (8 extensões lógicas) ao invés de apenas 16K (uma extensão lógica). O número apresentado, contudo, corresponde ao do *display* na versão 1.4 para manter compatíveis ambas as versões.

O campo 'ACC' lhe diz que tipo de acesso é permitido – R/O (read only) ou R/W (read-write). Estes tipos de acesso correspondem aos atributos de arquivo $R/O e $R/W descritos na seção sobre a versão 2.2 e MP/M (neste capítulo). Pode-se alterar os atributos do arquivo fornecendo um atributo como argumento (para substituir '$S' na definição acima) para o formato STAT. Por exemplo:

```
A>STAT PROG.ASM $R/O ↵
A>STAT FILE.TXT $R/W ↵
```

O primeiro comando STAT coloca o atributo $R/O no arquivo PROG.ASM. O segundo comando STAT aloca o atributo $R/W ao arquivo FILE.TXT. A forma genérica deste comando é:

```
STAT d: { filename  } { $R/O }
        { filematch } { $R/W }
                      { $SYS }
                      { $DIR }
```

A seguir damos outro exemplo onde ED.COM é "read-only", PIP.COM é "read-only" e um arquivo de sistema ($SYS), e DATA é um arquivo de dados de acesso aleatório com o atributo "read-write":

Size	Recs	Bytes	Ext	Acc	
48	48	6K	1	R/O	A:ED.COM
55	55	12K	1	R/O	(A:PIP.COM)
65536	128	2K	2	R/W	A:TEXT.NAD

Observe que PIP.COM está entre parênteses para denotar que possui o atributo $SYS (sistema). Os arquivos que não estão entre parênteses têm o atributo default $DIR (diretório). Os atributos são descritos neste capítulo, mais adiante, na seção sobre a versão 2.2 do CP/M e MP/M.

Para apresentar informações a respeito do disco (disquete) atual (ou alternativo), utilize a forma 'STAT d: DSK', onde d é um especificador opcional do *drive* (A, B, C, D, . . . P) para um *drive* alternativo. Abaixo damos uma amostra do *display* respectivo:

```
A>STAT DSK: ↵
A: Drive Characteristics
3888: 128 Byte Record Capacity
 486: Kilobyte Drive Capacity
  32: Byte Directory Entries
 128: Checked Directory Entries
 128: Records/Extent
  16: Records/Block
  52: Sectors/Track
   2: Reserved Tracks
```

Esta amostra de *display* refere-se a um disquete de dupla densidade no *drive* atual (A, neste caso). A capacidade total de registros (128 bytes por registro) é de 3888 registros (aproximadamente 486K bytes). A capacidade do *drive* em kilobytes é listada como sendo igual a 486 (isto não leva em consideração as áreas reservadas do disquete). O número de entradas no diretório (32-byte) corresponde ao número atual de blocos de controle de arquivos (FCBs) armazenados no disquete. O número de entradas "checadas" é, em geral, idêntico ao número de entradas no diretório para meios removíveis (como disquetes), porque o sistema necessita verificar as entradas para poder detectar uma alteração nos disquetes (este número geralmente corresponde a zero para discos não-removíveis). O sistema faz uma verificação das entradas, a não ser que haja uma partida "a quente" interveniente (↑ C) ou uma partida "a frio" ("system reset", ou "cold boot"). O número de registros por extensões ('Records/Extent") mostrado no *display* dá a capacidade de endereçamento de cada entrada do diretório; isto é, quantos registros (128 segmentos de bytes) podem ser endereçados por uma entrada de diretório (bloco de controle de arquivo no disco). O *display* mostra que 128 registros podem ser endereçados por uma única entrada de diretório (128 registros é igual a 128 vezes 128 bytes, ou seja, aproximadamente 16K bytes). O número de registros por bloco ('Records/Block') apresentado no *display* nos dá o número de registros alocados a cada setor (16 registros são 2048 bytes, ou 2K bytes por bloco). Esta informação é seguida pelo número de setores físicos ("blocos") por trilha ('Sectors/Track') e o número de trilhas reservadas no disco (disquete). O nosso exemplo mostra 52 setores (bloco por trilha), e duas trilhas reservadas para o sistema. Observe que, possuindo um disco grande ao qual vários *drives* lógicos de disco têm acesso, o número de trilhas reservadas será grande, já que o sistema utiliza essas trilhas para determinar quais as áreas do disco às quais cada *drive* lógico do disco deveria ter acesso.

Para apresentar informações acerca de áreas de usuários, use a forma 'STAT USR: ↵ '.

A seguir, damos um *display* à guisa de exemplo:

```
A > STAT USR:↵
Active User : 0
Active Files : 0 1 3 1 0
```

O 'Active User' neste *display* representa, na realidade, o número da área do usuário na qual o operador se "encontra" agora. Em MP/M, a área atual do usuário é apresentada junto com o *prompt* do sistema (por ex.: 'OA > ' significa *drive* atual A, área do usuário 0); na versão 2.2 do CP/M, o operador é obrigado a determinar sua área de usuário utilizando esta forma do STAT. No nosso exemplo, a área atual do usuário é zero (é a área default do usuário depois de uma partida "a frio" ou reinício do sistema). O termo 'Active Files' indica o número de áreas do usuário que atualmente contêm arquivos (no *drive* atual do disco ou disquete). Se o operador passar para uma destas áreas do usuário (utilizando o comando USER), descobrirá, pelo menos, um arquivo nessa área. As áreas do usuário que não contêm arquivos não são mostradas no *display* 'Active Files'. Descrevemos, com maiores detalhes, as áreas do usuário e o comando USER na seção a respeito da versão 2.2 do CP/M e do MP/M.

SUBMETENDO UM ARQUIVO DE COMANDOS PARA EXECUÇÃO (SUBMIT AND XSUB)

Introdução

Ocasionalmente é necessário e útil executar uma seqüência de comandos CP/M como se fossem instruções de um programa. Por exemplo, se um operador utiliza freqüentemente uma seqüência, seria conveniente dar-lhe um nome e executá-la por meio de um único comando, como se fosse um programa.

O CP/M oferece o comando transiente SUBMIT para executar, segundo a nossa conveniência, uma seqüência de vários comandos. O comando SUBMIT espera encontrar um arquivo com a extensão '.SUB' que contém linhas de comando atuais que podem incluir *argumentos* a serem substituídos por *valores* por ocasião da execução. O arquivo '.SUB' é criado como um arquivo-texto, utilizando ED ou outro programa editor, e lista linhas de comandos como seriam digitadas no terminal. Por exemplo, SAMPLE.SUB poderia conter estas linhas:

```
DIR $1:$2↵
PIP A:=$1:$2↵
```

Os '$1' e '$2' são argumentos opcionais e funcionam como variáveis em um programa – serão substituídos pelos valores atuais quando o operador os fornece por ocasião da execução (isto é, quando usa o comando SUBMIT). Aqui, o argumento '$1' é substituído por uma letra de *drive* para um *drive* de disco, e o argumento '$2' é substituído por um *filename* completo (incluindo a extensão, se esta existir).

Neste exemplo, o arquivo SAMPLE.SUB é executado utilizando o seguinte comando SUBMIT:


```
A > SUBMIT SAMPLE B FILE1.TXT↵
```

O programa SUBMIT inicialmente procura SAMPLE.SUB e depois começa a executar os comandos. Para executar DIR, coloca o valor 'b' em $1 e 'FILE.TXT' em $2, e apresenta o *filename* no diretório do *drive* B. Depois executa PIP para copiar B:FILE1.TXT para o *drive* A, usando o mesmo nome.

O comando SUBMIT assume a seguinte forma:

```
SUBMIT filename v1 v2 v3 . . .
```

onde *v1* é o valor a ser colocado no lugar de '$1' em qualquer posição do arquivo 'SUB', e *v2* substitui '$2' neste mesmo arquivo. SUBMIT aplica uma extensão '.SUB' ao nome do arquivo fornecido, de modo que você não precisa especificar a extensão '.SUB'.

Ao empregar argumentos ($1, $2, . . .) é necessário usar o sinal $ seguido de um número inteiro. O número deve começar do 1 para o 1º argumento, 2 para o 2º argumento, 3 para o seguinte, e assim por diante. Já que se está usando o sinal $ para denotar argumentos, é necessário empregar dois sinais $$ para significar um valor em dólares (ou cruzeiros) em um arquivo '.SUB' (dois sinais $ se tornam um em um arquivo '.SUB', ao passo que um sinal $ seguido de um número se torna um *argumento*). Pode-se também incluir um caractere para denotar uma combinação CTRL-tecla (por ex., ↑C ou ↑Z) em um arquivo '.SUB' – utilize a seta para cima ao invés do CTRL (Controle). Isto é necessário porque na maioria dos casos não se pode pressionar quaisquer seqüências de teclas CTRL em um programa editor ao criar o arquivo '.SUB'.

NOTA: qualquer que seja o *drive* escolhido para desempenhar a operação SUBMIT, o arquivo '.SUB' não será processado até ser inserido o disquete no *drive* A (ou use *drive* lógico A) e faça um "reboot" (partida "a quente") do sistema (um ↑C é suficiente). Pode-se especificar um *drive* alternativo para o arquivo '.SUB' em um comando SUBMIT, precedendo o nome do arquivo com um especificador de *drive*, mas a operação não se realizará até que o disquete com o arquivo '.SUB' esteja no *drive* A.

É possível abortar uma operação SUBMIT já em progresso pressionando a tecla RUBOUT (DELETE). SUBMIT cria, automaticamente, um arquivo temporário '$$$.SUB' para conter os comandos de um arquivo '.SUB'; este arquivo temporário é apagado quando se termina SUBMIT, ou se o sistema detectar um erro enquanto a operação SUBMIT está se processando, ou se o operador abortar a operação pressionando RUBOUT (DELETE). Se, por algum motivo, esse arquivo $$$.SUB ainda existir depois que o sistema execute os comandos contidos em $$$.SUB, ao invés de aguardar os seus comandos digitados. Se isto acontecer, aborte a operação SUBMIT, pressionando RUBOUT, e apague o arquivo $$$.SUB.

NOTA: para programadores que elaboram programas utilizando a facilidade SUBMIT: se o seu programa assumir a função CCP (Console Command Processor) de ler e interpretar as entradas do console e os erros do sistema, é necessário fazer o seu programa apagar (erase) o arquivo $$$.SUB que SUBMIT cria. Além disso, executando o seu programa mediante uma operação SUBMIT, assegure-se de que o arquivo $$$.SUB seja apagado (se você não desejar continuar a operação SUBMIT) ou preservado para uso futuro (dando-lhe novo nome). Qualquer novo comando SUBMIT irá substituir o arquivo $$$.SUB existente. Naturalmente é possível embutir um comando SUBMIT dentro do arquivo '.SUB' como "qualquer outro comando CP/M". Isto lhe permite "encadear-se" a outro conjunto de comandos submetidos.

SUBMIT com XSUB

Na versão 1.4 do CP/M, SUBMIT cria o arquivo temporário $$$.SUB a partir do arquivo '.SUB' que o operador fornece, e depois o CCP (Console Command Processor) executa cada linha de $$$.SUB como se fosse um comando digitado (o CCP é a parte do sistema que lê e executa os comandos que foram digitados).

Na versão 2.2 do CP/M existe uma capacidade adicional: o programa XSUB (comando transiente). O programa XSUB lhe permite incluir entradas (isto é, comandos) a programas (outros além do CCP) que utilizam a operação "entrada com memória intermediária" do CP/M. Não se precisa saber usar essa entrada com memória intermediária; basta saber se o programa que se deseja executar e ao qual irá fornecer entradas emprega este tipo de memória. Os programas ED, DDT e PIP o fazem. Portanto, por exemplo, pode-se usar os comandos ED em um arquivo '.SUB' como entrada para o programa ED, e desempenhar, automaticamente, uma operação ED complexa, repetidamente, com um único comando SUBMIT.

Abaixo apresentamos um exemplo de tal operação. O arquivo DOTHIS.SUB contém as seguintes linhas, incluindo o 'XSUB':

```
XSUB                    Execute XSUB
DIR $1.*                Display files
ED $1.$2                Use ED on file specified
#A                      Do an ED append to fill buffer
B                       Go to beginning of buffer
Copyright 1980, Sybex   Insert copyright notice
E                       Terminate ED
PIP $1.OLD = $1.BAK     Make copy of old backup
                        file, when CP/M system returns
                        after ED terminates
DIR $1.*                Display all files again
A > SUBMIT DOTHIS SAMPLE TXT ↵
```

Neste exemplo, SUBMIT inicialmente diz ao CCP para executar o XSUB. O programa XSUB realoca-se para a área diretamente abaixo do CCP e permanece ativo até a próxima partida "a frio" ("system reset" ou "cold boot"). Enquanto XSUB estiver ativo, apresenta a mensagem 'XSUB ACTIVE' acima do *prompt* do sistema; e enquanto houver comandos no arquivo DOTHIS.SUB, XSUB os executa (a não ser que você intervenha com um ↑C).

Depois, XSUB executa o comando DIR para apresentar os arquivos associados com SAMPLE. 'SAMPLE' é um substituto para '1$' e 'TXT' é um substituto para '$' em DOTHIS.SUB. SUB executa o programa ED no arquivo SAMPLE.TXT. SUB continua a providenciar entradas para o programa ED (utilizando entrada intermediária do console): primeiro o comando #A para dizer a ED para acrescentar, ao final, o arquivo completo (65535) linhas à entrada intermediária do editor, depois o comando B para localizar o indicador de caracteres ED no início da entrada intermediária, depois o comando I para inserir o texto 'Copyright 1980, Sybex' no início da entrada intermediária, e finalmente o comando E para salvar as edições e terminar o programa ED (Os comandos ED são descritos no Capítulo 4.)

O programa XSUB permanece ativo depois que termina ED, executando o comando PIP para copiar SAMPLE.BAK no arquivo antigo SAMPLE.OLD (fazendo uma cópia de reserva de SAMPLE.BAK). Depois, XSUB executa o comando DIR para apresentar todos os arquivos associados com o nome SAMPLE.

Quando se esgotaram os comandos para o arquivo 'submit', o CCP interfere e aguarda comandos adicionais a serem teclados no terminal. Todavia, o programa XSUB continua ativo (a mensagem 'XSUB ACTIVE' ainda reaparece depois de cada execução de comando) a não ser que seja executado um programa que intencionalmente regrava a área ocupada por XSUB (isto é, diretamente abaixo de CCP, ou até que o operador dê uma partida "a frio" ou faça um "system reset"). Enquanto XSUB permanecer ativo, o comando SUBMIT pode ser usado para executar outros arquivos '.SUB'. Os outros arquivos '.SUB' *não* necessitam possuir 'XSUB' como linha de comando se XSUB estiver ativo. Portanto, o aprendiz pode criar arquivos '.SUB' *sem* XSUB para mantê-los compatíveis com versões anteriores do CP/M e executar XSUB como comando, sempre que o desejar ativo, ao submeter arquivos '.SUB'.

PROGRAMAS DE MONTAGEM (ASSEMBLING-ASM), DE CARREGAMENTO (LOADING-LOAD) E DE DESCARGA (DUMPING-DUMP)

Introdução

No mundo da linguagem assembler, ASM, LOAD, e DUMP são termos padronizados para operações que permitem que programas em linguagem assembler possam funcionar como comandos. Podem-se usar os comandos ASM e LOAD para transformar um programa-fonte em linguagem assembler (escrito em um arquivo-texto) em um comando transiente do tipo "faça-o você mesmo", e pode usar o DUMP para apresentar os conteúdos do comando transiente. O termo "dump" é frequentemente empregado para descrever operações de cópia em larga escala (por ex.: a operação "daily dump" em minicomputadores e computadores de grande porte) e o ato de enviar conteúdo de um programa para a impressora, o comando DUMP do CP/M apenas "descarrega" a área de memória do programa (em notação hexadecimal) no vídeo do terminal. Para enviar um arquivo para a impressora, ou executar operações de cópia em larga escala, é necessário empregar o comando transiente PIP.

Para escrever programas em linguagem assembler para um sistema CP/M ou MP/M é necessário conhecer o tipo de linguagem assembler adequado ao computador utilizado. O CP/M padrão corre em microcomputadores que utilizam os microprocessadores Intel 8080, 8085 ou 8086, ou Zilog Z80 ou Z8000. Outros fornecedores de software ofere-

cem CP/M em computadores baseados no 6502, com o Apple ou o Pet, mas utilizam uma placa adicional 8080 ou Z80.

A montagem

O comando ASM executa o Assembler 8080 da "Digital Research", residente no arquivo ASM.COM. Este programa assembler traduz um arquivo-fonte em linguagem assembler (escrito na linguagem assembler do 8080) para um arquivo em *linguagem de máquina* (veja Figura 2.7) Um arquivo de linguagem de máquina contém instruções em *binário* – a linguagem do microprocessador; entretanto, a maioria dos *displays* ou das impressões do arquivo serão em *notação hexadecimal.*

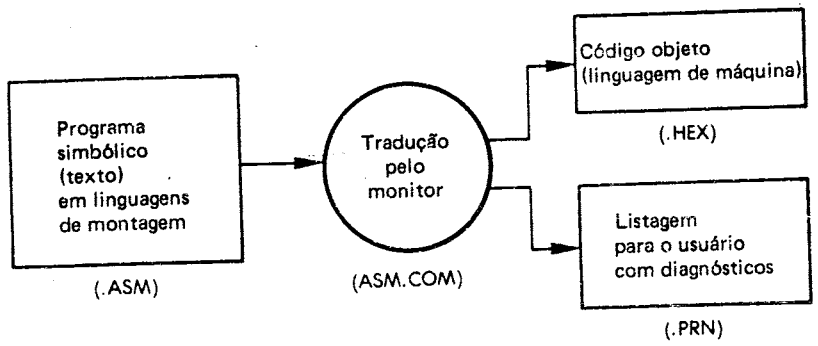


Figura 2.7: *O processo de montagem*

Existem outros programas de montagem que podem elaborar linguagens de montagem para outros microprocessadores, também, e existem, ainda, montadores mais potentes para o 8080. Esses montadores existiriam todos como arquivos '.COM' em seu sistema, e o operador pode executá-los assim como executaria ASM.COM.

O programa de montagem monta um arquivo-fonte que possui uma extensão '.ASM' (por ex.:, PROGR.ASM). O montador utiliza então a extensão '.HEX' para denotar o arquivo montado em linguagem de máquina que criou (por ex.:, PROG.HEX). O montador emprega, também, uma extensão '.PRN' (por ex.:, PROG.PRN) para criar um arquivo que poderá ser impresso. O operador pode enviar esse arquivo '.PRN' para a impressora com o objetivo de obter uma impressão do programa. O arquivo '.PRN' contém as linhas do arquivo-fonte '.ASM' original, junto com mensagens de erro e o código da máquina resultante (na notação padrão intel-hexadecimal).

O arquivo código de máquina '.HEX' produzido por ASM está agora pronto para ser carregado no sistema (usando o comando LOAD) e para se transformar em um programa transiente executado como se fosse um comando.

Para executar o ASM pode-se usar uma destas formas:

1. ASM filename
2. ASM filename.*shp*

Em ambas as formas, a pessoa só necessita especificar o nome primário do arquivo-fonte. O arquivo precisa ter uma extensão '.ASM' – o programa ASM espera encontrar uma extensão '.ASM' para cada nome de arquivo especificado. O ASM não irá montar um arquivo que não possua uma extensão '.ASM'.

A primeira forma (1.) simplesmente executa o programa ASM para montar o arquivo denominado por "filename". O ASM parte do pressuposto de que o arquivo está no *drive* atual, e que deve colocar os arquivos '.HEX' e '.PRN' nesse *drive.* Por exemplo:

```
A > ASM PROG↵
```

Este comando executa o ASM (que se pressupõe estar no *drive* A sob a forma de ASM.COM) para montar o arquivo PROG.ASM (também pressuposto estar no *drive* A). O ASM irá produzir os arquivos PROG.HEX e PROG.PRN e também colocá-los no *drive* A.

A segunda forma (2.) lhe permite especificar *drives* de discos diferentes do atual se o arquivo-fonte estiver em outro *drive,* se a pessoa desejar colocar os arquivos '.HEX' ou '.PRN' em outro *drive,* ou se desejar dizer ao ASM que pule a função de criar os arquivos '.HEX' ou '.PRN'.

No formato, *shp* consiste de três letras precedidas de um ponto seguindo o argumento do nome do arquivo. O *s* pode ser qualquer letra de A até P. O *h* pode ser qualquer letra de A até P, ou Z; o *p* é uma letra que indica o *drive* (A, B, ... , Y) que contém o disco com o arquivo-fonte. O *h* é uma letra que indica qual o *drive* (A, B, ... , Y) que deveria receber o arquivo '.HEX', ou, se a letra Z for usada, que diz ao ASM que deixe de gerar o arquivo '.HEX' e gere apenas o arquivo '.PRN'. O *p* é uma letra que indica que *drive* (A, B, ... W, Y) deveria receber o arquivo '.PRN'. Se for usado um X, o arquivo '.PRN' é enviado apenas para o *display* de seu terminal, ou, se for usado um Z, o ASM deixa de criar o arquivo '.PRN' e só cria o arquivo '.HEX'.

Aqui estão alguns exemplos da segunda forma:

```
A > ASM PROG1.ABZ↵
```

Este comando monta o arquivo PROG1.ASM, que é o arquivo-fonte no *drive* A. Também cria PROG1.HEX no *drive* B (e assim pula a função de criação do PRG1.PRN).

```
A > ASM PROG2.BZX↵
```

Este comando monta o arquivo PROG2.ASM, que é o arquivo-fonte no *drive* B, e envia o arquivo PROG2.PRN para o *display* do terminal (sem criar PROG2.HEX).

NOTA: As mensagens de erro do ASM podem assumir a forma de uma linha-fonte em linguagem assembler, na qual uma letra indica um código de erro (esses códigos são explicados na documentação do montador). Os erros são corrigidos modificando-se o programa com o depurador DDT (ou qualquer outro), ou corrigindo o arquivo-fonte e o remontando. Tais erros ASM também são sinalizados no arquivo '.PRN'. Outros erros que podem ocorrer são apresentados na Figura 2.8.

Carregamento

Quando se tem um arquivo '.HEX' produzido pelo ASM (ou outro montador), pode-se transformá-lo em um arquivo '.COM' (comando transiente executável) carregando-o no sistema utilizando LOAD (veja Figura 2.9).

NO SOURCE FILE PRESENT	— O ASM não consegue encontrar o arquivo fonte, ou você especificou um arquivo que não possui uma extensão '.ASM'.
NO DIRECTORY SPACE	— O diretório do disco (disquete) está lotado; apague os arquivos não essenciais (ou os copie) e tente de novo.
SOURCE FILE NAME ERROR	— O nome do arquivo foi inadequadamente digitado para o comando ASM. Não se pode usar *filename matches* em comandos ASM.
SOURCE FILE READ ERROR	— O arquivo-fonte não pode ser lido pelo ASM por alguma razão. Utilize o comando TYPE para descobrir a linha incorreta no arquivo-fonte.
OUTPUT FILE WRITE ERROR	— O arquivo de saída ('.HEX' ou '.PRN') não pode ser gravado no disco (disquete); provavelmente porque este está lotado.
CANNOT CLOSE FILE	— O arquivo de saída ('.HEX' ou '.PRN') não pode ser fechado (atualizado); verifique se o disco (disquete) está protegido contra gravação (isto é, "read-only").

Figura 2.8: *Erros de montagem*

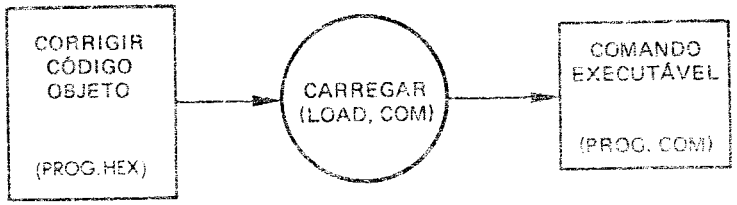


Figura 2.9: *Carregando o Código Objeto*

O comando LOAD é também um comando transiente, existindo sob a forma LOAD.COM. Pode-se executá-lo teclando 'LOAD', seguido de um nome de arquivo de um arquivo tipo '.HEX', como no seguinte exemplo:

```
A > LOAD PROG ↵
```

Este comando procura PROG.HEX no *drive* atual, cria uma imagem mnemônica deste arquivo (em formato hexadecimal) e apelida o novo arquivo de PROG.COM. Este novo programa pode ser executado digitando-se 'PROG ↵'.

Como LOAD cria um arquivo '.COM' você só precisa carregar um arquivo '.HEX' uma só vez. O arquivo '.HEX' deve conter um "formato hexadecimal" Intel válido que é criado pelo ASM ou outro montador similar.

Pode-se também carregar arquivos de outro *drive* de disco especificando o *drive* como parte do nome de arquivo. Por exemplo:

```
A > LOAD B:GAMES ↵
```

Este comando carrega o arquivo GAMES.HEX no *drive* B e cria GAMES.COM no *drive* A (o *drive* atual).

Como se pode usar o PIP para transferir arquivos em "formato hexadecimal" a partir de dispositivos de leitura, também se pode carregar um arquivo montado em alguma ocasião anterior e que foi transferido recentemente para o seu sistema via PIP.

NOTA: Para os aficionados de linguagem assembler: o arquivo '.HEX' deve possuir registros hexadecimais válidos que começam com 100('H' representa hexadecimal, isto é, endereçado 100_{16}). 100H é o início do TPA (Área de Programa Transiente). Os endereços do arquivo '.HEX' devem estar em ordem crescente (lacunas em regiões de memória não preenchidas são completadas com zero por LOAD à medida que lê os registros hexadecimais). Desta forma, LOAD só é usado para criar arquivos '.COM' que são executados no TPA. Programas que *não* irão ocupar o TPA (programas especiais) podem ser carregados usando DDT (ou outro depurador).

Descarregando

O comando DUMP apresenta os conteúdos de um arquivo no terminal sob forma hexadecimal. Qualquer arquivo pode ser "DUMPed" (descarregado) já que todos os dados assumem valores binários que também podem ser representados sob forma hexadecimal (base 16). Todos os números hexadecimais terminam em 'H' neste livro, da mesma forma que ocorre na mnemônica Intel. O 'H' representa a base 16 (por ex., 10H é 10_{16}).

O comando DUMP exige o nome de arquivo completo (incluindo a extensão):

```
A > DUMP B:PROG.HEX ↵
```

O exemplo acima apresenta os conteúdos de PROG.HEX, que está no *drive* B.

EXECUTANDO, DEPURANDO (DDT) E SALVANDO (SAVE) PROGRAMAS

Executando

Depois que um programa '.HEX' foi carregado (LOADed) e um arquivo '.COM' criado para ele, pode-se executar o programa digitando o nome do arquivo como um comando (sem a sua extensão '.COM'). Por exemplo:

```
A > PROG ↵
```

Admita que PROG.COM está no *drive* A.

Pode-se também executar o programa a partir de outro *drive*:

```
A > B: ↵
B > A:PROG ↵
```

Quando se executa um programa ou um comando transiente, ele é trazido para a memória principal do computador (memória de trabalho ou TPA).

Depurando (Debugging)

"Bugs" são erros em um programa. Os erros sinalizados pelo ASM (ou outro montador), PIP ou LOAD, ou erros de tempo de processamento (erros que ocorrem durante a execução) podem ser, geralmente, corrigidos usando um programa depurador ("debugger") como DDT. O DDT.COM é fornecido com a versão-padrão do CP/M (ou MP/M). O DDT irá trazer qualquer arquivo para a memória principal e desempenhar as operações disponíveis como comandos DDT. O DDT pode ser usado para corrigir erros, ou para trazer um arquivo para a memória principal com o objetivo de reservar uma imagem de memória do mesmo (usando SAVE). Para executar o DDT em qualquer arquivo, utilize a seguinte forma:

```
DDT { filename.HEX }
    { filename.COM }
```

A pessoa pode especificar uma letra de *drive* como parte do nome do arquivo (se o arquivo existir em outro *drive*).

Em ambos os casos o DDT substitui o CCP ("Console Command Processor") como o "sistema operacional" que lê o comando linha por linha. Descrevemos o CCP no Capítulo 5. Quando o programa DDT substitui o programa CCP do sistema operacional, o programa DDT assume a tarefa de ler a linha de comando (assim como o programa ED, DDT possui o seu próprio conjunto de comandos).

Se você especificar um nome de arquivo (do tipo '.HEX' ou '.COM') com DDT, o depurador traz o programa para o TPA (apagando o que quer que estivesse aí antes). Se a pessoa não especificar um nome de arquivo, o DDT irá ocupar a TPA e esperar por um comando 'I' DDT para colocar um arquivo na TPA.

Ao ser executado, o DDT apresenta uma mensagem "sinal ligado" (que pode variar de instalação para instalação) e depois o seu *prompt*: '–'. Agora já se podem digitar comandos DDT (use a documentação fornecida com DDT). Em versões posteriores do CP/M, o DDT também apresenta os valores 'NEXT' e 'PC', que são discutidos com o comando 'R' DDT, abaixo.

Vários comandos DDT serão discutidos aqui: o comando I (entrada), o comando R (leitura) e o comando GO (pare a depuração e volte ao sistema operacional).

O comando I coloca um arquivo '.HEX' ou '.COM' na TPA. Essencialmente, esta é a mesma operação que foi executada quando foi apresentado um nome de arquivo do tipo '.HEX' e '.COM' com o comando DDT. Pode-se ainda acrescentar outro arquivo ao existente na TPA usando esse comando.

O comando R lê o arquivo na TPA e apresenta uma "mensagem de carga" que consiste de:

```
NEXT    PC
nnnH    pc
```

Este *display* ocorre, automaticamente, em versões posteriores do DDT. O número sob a coluna NEXT é o próximo endereço depois do programa carregado. Pode-se usar esse valor para calcular o número de "páginas" (blocos de 256 bytes) a utilizar em um comando SAVE. O número está em notação hexadecimal.

O comando GO termina o DDT, mas deixa o programa na TPA para que se possa usar o comando SAVE para salvar uma imagem de memória do arquivo. Basta digitar:

```
GO!
```

Salvando

O comando SAVE retira uma ou mais "páginas" da TPA (memória principal) e coloca-as em um disco como arquivo, cujo nome a pessoa especifica (observe que uma "página" são 256 bytes). (Em MP/M, esta operação é implementada *dentro* do programa depurador – DDT ou SID).

Usa-se o comando SAVE para criar uma imagem mnemônica do arquivo, referente ao que está atualmente na TPA (o programa que foi executado por último). Se a pessoa usa o DDT em um programa, e ele começa a trabalhar corretamente (ou se mesmo uma *parte* do mesmo trabalha corretamente) irá desejar salvá-lo (SAVE) como um arquivo que pode ser copiado, depurado ou executado.

NOTA: Na versão 1.4, não se pode executar dois SAVEs consecutivos sobre os mesmos conteúdos da TPA, porque o primeiro SAVE ocasiona uma mudança de diretório que altera várias áreas da TPA. Na versão 2.2, contudo, este problema foi corrigido – é possível executar dois SAVEs consecutivos no mesmo conteúdo da TPA.

O comando SAVE assume esta forma:

```
SAVE p filename
```

O operador fornece o número de páginas da memória como p (um número decimal). p é calculado utilizando os comandos DDT e R de DDT para apresentar o endereço sob a coluna NEXT – o próximo endereço (o mais alto) que segue o programa na TPA. Este endereço é, na realidade, o último endereço da TPA *mais* um. Portanto, se a pessoa subtrair 1 do valor apresentado (subtraindo em aritmética hexadecimal), terá o último endereço da TPA (fim do programa), que poderá converter a "páginas" em decimal.

Um método fácil para converter o endereço NEXT no valor de p é: se o endereço NEXT termina em dois zeros (por exemplo, 1200H), subtraia 1H ('H' representa hexadecimal) para obter 11FFH, depois ignore os últimos dois dígitos ('FF') e converta 11H a decimal ($1 \times 16^0 = 1$, e $1 \times 16 = 16$, portanto $11H = 17_{10}$). Se o endereço NEXT *não* terminar em dois zeros, *não* subtraia 1H; simplesmente converta os dois primeiros dígitos (por ex., 1205H se torna 12H, que é igual a 18 páginas). Aqui o aprendiz tem um exemplo fácil:

(Valor sob NEXT no *display* do comando R de DDT):

```
3FFH
```

NOTA: 'F' é o valor decimal "15" em notação hexadecimal. 'H' representa hexadecimal, e *não* é um número.

Para converter em *páginas* (blocos de 256 bytes), ignore os primeiros dois dígitos 'FF' e utilize o dígito '3'. Este é o "byte de ordem superior", e pode ser expresso como 3H. 3H convertido em decimal é 3_{10}. Portanto, usa-se o 3 como o valor decimal de páginas no comando SAVE.

Aqui está um exemplo mais difícil:

(Valor sob NEXT no *display* do comando R de DDT):

```
1D00H
  -1H
------

1CFFH
```

NOTA: D é o valor decimal '13'. Subtraia 1H porque '1D00H' termina em dois zeros. D(13) se transforma em C(12).

Para converter em páginas, utilize o byte de ordem superior '1C' e o converta em decimal:

```
   CH = 12 vezes 16^0  =  12
+  10H = 1 vez 16^1    =  16
                          --

=  . . . . . . . . . .    28
```

NOTA: O valor decimal 28 é igual ao número de páginas de memória entre os endereços 100H (início da TPA) e 1CFFH.

Aqui estão alguns exemplos de comandos SAVE:

A SAVE 4 PROG.COM ↵ Salva a memória de 100H até 4FFH e a coloca em PROG.COM no *drive* A. O próximo endereço superior depois de 4FFH é 500H.

B SAVE 10 KLUDGE.COM ↵ Salva a memória desde 100H até 0AFFH no arquivo KLUDGE.COM no *drive* B. O próximo endereço superior depois de 0AFFH é 0B00H. AH é 10 decimal.

A SAVE 40B:WORKS.COM ↵ Salva a memória de 100H até 28FFH no arquivo WORKS.COM no *drive* B. O próximo endereço superior depois de a8FFH é 2900H. 28H é (2 vezes 16) + 8 = 40 decimal.

VERSÃO 2.2 DO CP/M E MP/M

Uma introdução do MP/M

O MP/M é um sistema operacional elaborado para tempo compartilhado. Um sistema de tempo compartilhado pode executar vários *processos* ou programas, simultaneamente. Não obstante, esta simultaneidade é apenas aparente. De fato, o computador executa um

programa de um usuário, e depois outro, em sucessão muito veloz, de modo que cada usuário tem a impressão de que ele é o único a usar o computador.

Portanto, quando o MP/M é utilizado em um único terminal, o sistema se comporta, essencialmente, como um sistema CP/M de um único usuário. Não obstante, o MP/M oferece facilidades adicionais, permitindo que um único usuário execute vários programas simultaneamente. Por exemplo, programas podem ser vinculados e desvinculados do console, permitindo que um único usuário interaja com o console enquanto outros programas estão sendo executados.

NOTA: Se a pessoa planejar usar o MP/M como usuário único, com facilidade para um só programa, não há necessidade de conhecer nada a respeito do MP/M. Ler o texto sobre a versão 2.2 do CP/M deve ser o suficiente para satisfazer suas necessidades.

Um dos problemas a serem resolvidos por um sistema de tempo compartilhado é o *escalonamento dos programas.* Cada programa deve ser escalonado para rodar por sua vez, no processador. A técnica de escalonamento de *round-robin* é a melhor para fazê-lo. Com esta técnica, cada programa recebe uma parcela igual de tempo no computador, por sua vez. Esta técnica é ilustrada por um ponteiro que gira, na Figura 2.10.

Para que se possa utilizar eficientemente o processador, alguns programas (ou processos) devem rodar antes dos outros. Em todo bom sistema de escalonamento, cada processo deve ser equipado com um *nível de prioridade* que determinará quando irá correr. Por exemplo, quando os dados se tornam disponíveis a partir do disco, devem ser lidos imediatamente por um programa de transferência; caso contrário, haverá um grande atraso à medida que o cabeçote passa o ponto apropriado no disco. Este processo de transferência deve ter uma alta prioridade.

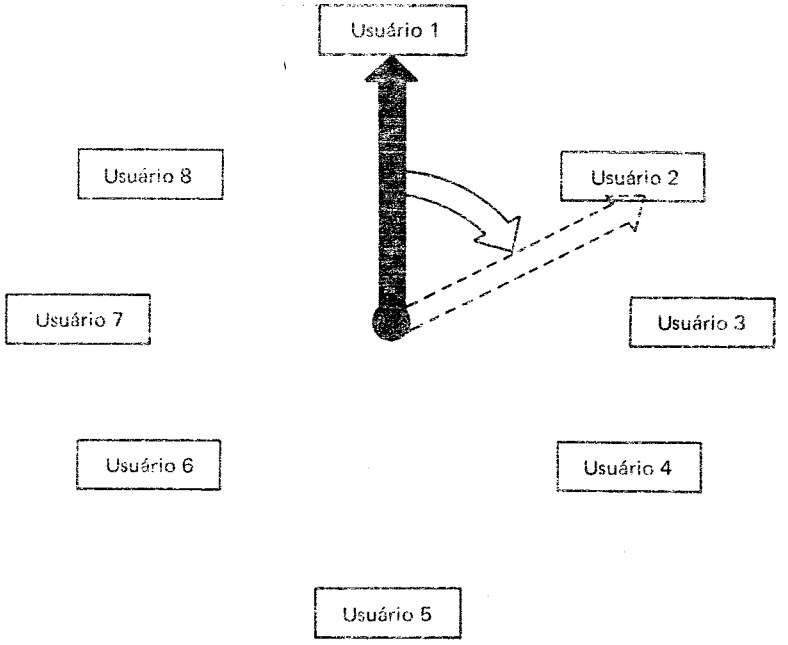


Figura 2.10: *Escalonamento Round-Robin*

Um processo que é impresso em uma impressora lenta pode levar alguns milissegundos, ou até segundos a mais, sem que se perca muita coisa. Portanto, o processo de impressão deve ter menos prioridade.

Para que se possam estabelecer prioridades, em geral distinguimos duas classes de processos: os que desempenham funções de entrada e saída – input/output – (operações I/O), e os que executam cálculos (operações CPU).

Sempre que um processo está sendo executado na CPU (processador) ele poderá solicitar uma operação de entrada ou de saída. Este tipo de operação é muito lento de acordo com os padrões da CPU, já que exige, tipicamente, vários milissegundos, e uma instrução na CPU opera em microssegundos. Quando é solicitada uma operação de I/O, o processo que está sendo executado é às vezes *bloqueado* e se torna dormente. Inicia-se então uma operação de I/O. O I/O procederá concorrentemente com a CPU. Quando a operação de I/O é completada, reativará o processo original dormente. O processo de CPU então reassume sua posição na lista de escalonamento, isto é, tornou-se novamente *ativo*.

A Figura 2.11 mostra uma lista de prioridades de processamento em níveis múltiplos. A prioridade mais alta costuma receber o número zero, com prioridades atuais decrescendo à medida que aumenta seu número de prioridade.

Na Figura 2.11, o processo 10 será executado inicialmente, e depois o processo 23.

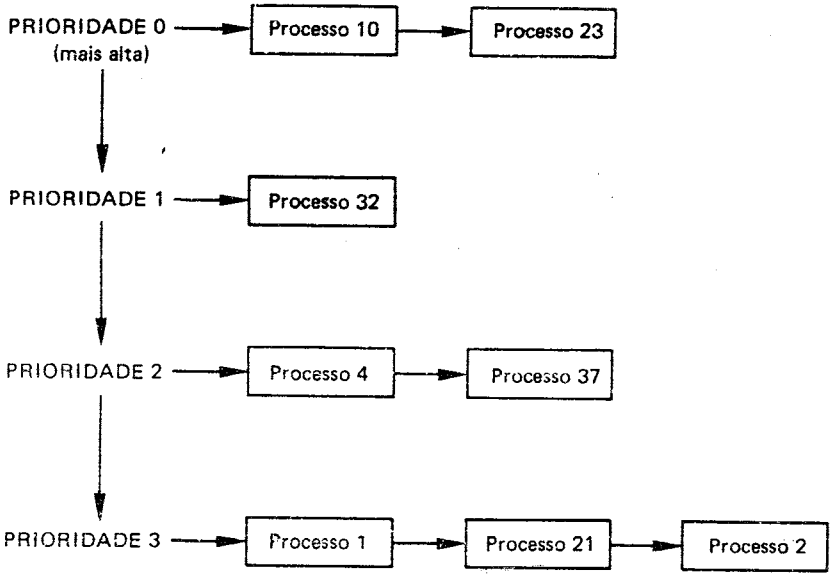


Figura 2.11: *Uma lista de prioridades de quatro níveis*

Enquanto o processo 23 está sendo executado, um novo processo pode entrar na prioridade 0 (veja a Figura 2.12). Neste caso, o processo 40 será o próximo a ser executado. Quando todos os processos no nível 0 terminarem, serão executados os do nível 1, e assim por diante.

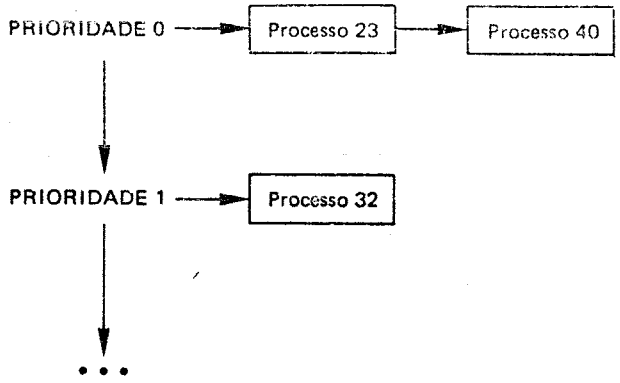


Figura 2.12: *Um novo processo entra na prioridade 0*

Um processo não foi definido formalmente aqui porque cada sistema operacional utiliza numa definição algo diferente. Um processo pode ser um programa, um subprograma ou outras entidades de programa. Desta forma, um usuário pode ativar uma variedade de processos.

No exemplo da Figura 2.11, só mostramos *processos ativos*. Mantém-se uma lista similar para *processos dormentes* que foram bloqueados. Quando "acordado", um processo dormente tornar-se-á ativo e entrará na lista ativa.

O sistema de escalonamento que foi descrito é geralmente utilizado para o processador. Contudo, poderá haver competição por outros recursos escassos dentro do sistema, como o disco ou a impressora.

De ordinário os processos formarão filas para a impressora e serão servidos na base do "quem chega primeiro é atendido em primeiro lugar". Isto se denomina lista FIFO ("First-in First-out"). O programa de escalonamento correspondente recebe o nome de "spooler".

Da mesma forma, solicitações para o disco geralmente entram em uma fila. Não obstante, um programa irá tentar otimizar o acesso ao disco lendo tantos setores quantos forem possíveis, ao mesmo tempo em que move o cabeçote tão pouco quanto possível. Disto resulta que os processos em uma fila de espera para o disco podem ser atendidos em qualquer ordem, se o bloco de que necessitam aparece sob o cabeçote do disco.

Processos simultâneos que compartilham recursos comuns exigem mecanismos adicionais:

- Os processos devem ser sincronizados. Isto é feito mediante o uso de sinalizadores e recursos de interrupção.
- Os processos devem operar em uma ordem razoável ao competir por um dispositivo comum. Esta é a função do programa de escalonamento para este dispositivo.
- É preciso providenciar mecanismos de proteção para que uma falha no funcionamento de um processo não prejudique os demais.
- O espaço de memória deve ser eficientemente alocado para que os processos só o ocupem quando necessário.

O MP/M é um sistema simples de tempo compartilhado que oferece um bom conjunto de facilidades. Não obstante, para que um sistema de tempo compartilhado possa resi-

dir em uma quantidade limitada de memória, muitos dos recursos são fornecidos de modo simplificado:

- o MP/M executará até oito processos
- cada processo possui seu próprio segmento de memória fixado (48K)
- os arquivos podem ter um destes quatro atributos:
  - R/O (read-only)
  - R/W (read-write)
  - SYS (sistema – o arquivo não está listado no diretório)
  - DIR (diretório – remove o atributo SYS).

Introdução ao CP/M 2.2

A nova versão do CP/M possui vários aperfeiçoamentos opcionais destinados a permitir uma transição fácil para o MP/M, um sistema de múltiplos usuários. Em um ambiente de múltiplos usuários, no qual várias pessoas estão utilizando a mesma máquina ao mesmo tempo, há necessidade maior de proteção para seus arquivos, pois a pessoa é obrigada a *compartilhar* os recursos de seu sistema – a impressora, os *drives* de disco (ou disquete), e o próprio computador (CPU). Possuindo a versão 2.2 do CP/M, a pessoa pode utilizar várias características que, na realidade, foram elaboradas para o MP/M, e ainda assim manter compatibilidade, isto é, seus arquivos e programas poderão ser utilizados em ambos os sistemas.

Os aspectos novos incluídos são:

- Áreas de usuários e comando USER para separar seus arquivos dos dos outros usuários.
- Atributos de arquivos (atribuídos pelo uso de START) para proteger seus arquivos de supressões ou regravações acidentais, incluindo um que armazena o arquivo secretamente.

NOTA: Se a pessoa *não* usar estes aspectos, seus arquivos e programas serão compatíveis com as versões prévias do CP/M assim como com a versão 2.2 do CP/M e MP/M.

Além destas características, o MP/M possui aspectos que não são encontrados na versão 2.2 do CP/M, e que *só* são usados em um ambiente de múltiplos usuários:

- Comando CONSOLE para apresentar o número atual do console (terminal) em um sistema com mais de um terminal.
- Comando DSKRESET para regular a mudança de discos em um sistema de múltiplos usuários.
- Comando GENMOD para produzir um programa relocável (essencial em sistemas de multiprogramação).
- Comando SPOOL para regular o tráfego para a impressora de linhas (STOPSPL é empregado para cancelar um SPOOL).
- Comando SCHED para escalonar programas a serem executados em outra ocasião.
- Comando TOD para apresentar ou atribuir a hora e data.
- Capacidade para desvincular um programa do seu terminal e revinculá-lo mais tarde; utilize junto com MPMSTAT para mostrar informações do sistema.

Características MP/M 1.0

O MP/M exige pelo menos 32K bytes de memória. Pode utilizar até dezesseis consoles e dezesseis discos, sendo que cada *drive* de disco contém até oito megabytes de informação. O MP/M permite até oito segmentos de memória de 48K.

Áreas do usuário e comando USER

Na versão 2.2 do CP/M e no MP/M pode-se segregar arquivos em um único disco (disquete) em "áreas de usuário", numeradas de zero a quinze. Os arquivos não são segregados; na realidade é necessário que sejam referidos utilizando-se um "número de área de usuário" que está prefixado internamente a seus nomes. Admite-se que a pessoa escolha uma área de usuário e coloque seus arquivos na mesma área em todos os discos (disquetes). A Figura 2.13 demonstra esta colocação.

	Disco A	Disco B	Disco C
Usuário 0	arquivos do sistema	arquivos do sistema	arquivos do sistema
Usuário 1			
Usuário 2 (a pessoa)	seus arquivos	seus arquivos	seus arquivos
Usuário 4			

Figura 2.13: *Colocação de arquivos em áreas de usuários*

Quando se tem o primeiro acesso ao MP/M, em cada console aparece a seguinte mensagem:

```
MP/M
nA >
```

onde n é o número do usuário para esse console, n é 0 para o console 0, 1 para o console 1 etc. Por exemplo, o *display* do console 3 (admitindo que quatro ou mais consoles estão conectados) é:

```
MP/M
3A >
```

Qualquer número do usuário na faixa entre 0 e 15 pode ser atribuído a qualquer console empregando o comando USER, descrito abaixo.

O 'A' refere-se ao *drive* de disco A, e pode ser alterado à vontade, assim como no CP/M.

Como colocar um arquivo em uma área do usuário

Quando o aprendiz está "em" uma "área atual do usuário", os arquivos que cria são colocados nessa área, devendo estar "nessa" área do usuário para ter acesso aos arquivos. Se necessitar "obter" uma cópia de um arquivo que se acha em outra área do usuário, utilize o parâmetro G do PIP, oferecido na versão 2.2 do CP/M e MP/M (descrito no Capítulo 3).

Quando o aprendiz chama, inicialmente, a versão 2.2 do CP/M ou MP/M, a sua área atual do usuário é a área zero. No MP/M, isto é assinalado pelo *prompt* do sistema, '0A >' (para *drive* A, usuário 0). Na versão 2.2 do CP/M deve-se usar a forma de STAT, 'STAT USR: ', para determinar o número da área atual do usuário.

Como passar para outra área do usuário

Ficando na área zero do usuário (isto é, não usando o comando USER), os seus arquivos e programas serão compatíveis com versões anteriores do CP/M e MP/M, no que se refere à área do usuário (área zero). Se desejar passar para outra área do usuário, é imprescindível que utilize o comando USER; os arquivos que a pessoa cria nessa área do usuário devem ser copiados para a área zero do usuário (empregando o parâmetro G do PIP) para que sejam compatíveis com versões prévias do CP/M.

O formato do comando USER é:

```
USER n
```

onde n é um argumento opcional do MP/M, mas necessário na versão 2.2 do CP/M. Em ambos os sistemas, se o operador fornecer uma faixa de zero a quinze, o comando USER irá movê-lo para essa área do usuário. Por exemplo:

```
0A > USER 2 ↵
2A >
```

NOTA: Este exemplo se aplica ao MP/M. No CP/M a pessoa deve usar 'STAT USR: ↵ ' para apresentar a área atual do usuário. Em ambos os casos, o usuário passa à área dois do mesmo *drive* do disco.

Quando o operador passa para outra área do usuário, permanece nela até que outro comando USER é executado ou até que o sistema seja reposicionado (reset). Depois de um reset, o operador será sempre colocado na área zero do usuário.

Áreas do Usuário: Consideração

Uma área do usuário não se torna ativa até que a pessoa passe para ela. Observe, porém, que o *drive* do disco permanece ativo, qualquer que seja o lugar para onde a pessoa vá – e as áreas do usuário são elaboradas tendo em vista discos de grande capacidade (pode-se usar discos rígidos). A área do usuário deixa de existir quando o operador apaga todos os arquivos, incluindo os que possuem os atributos $R/0 e $SYS (discutidos mais adiante neste capítulo). Apagam-se todos os arquivos utilizando o comando ERA com o *filename match* '*.*' que apaga todos os arquivos na área atual do usuário, exceto os que possuem o atributo $R/0 (read-only).

Já que o comando ERA apenas apaga arquivos na área atual do usuário, pode-se apagar arquivos em uma área do usuário de cada vez, a não ser que se escreva um programa anulando tudo que está no disco (disquete). A forma 'ERA *.* ↵ ' somente apaga arquivos na área atual do usuário (aquela em que a pessoa "está" no momento). Não apagará arquivos com o atributo $R/0 ("read-only") até que a pessoa mude o atributo para $R/W ("read-write").

Utilizando o comando DIR para verificar se ainda existem arquivos na área atual do usuário, a pessoa poderá omitir arquivos com o atributo $SYS (denominado 'arquivos do sistema"). Esses arquivos não aparecerão em um *display* DIR, mas apenas entre parênteses em um *display* produzido por 'STAT *.* $S ↵ '. Utilize o comando STAT para alterar atributos de arquivos (veja a seção seguinte) para apagar todos os arquivos e fechar uma área do usuário.

Atributos de arquivos

Em geral, os seus arquivos já são arquivos do "diretório" que podem ser *lidos* ou *gravados*. O aprendiz tem a opção de evitar que um arquivo seja listado em um *display* DIR, substituindo o atributo "Diretório" por um atributo "sistema". Da mesma forma, pode evitar uma operação de gravação ou supressão em um arquivo se alterar o seu atributo "read-write" para "read-only".

Na versão 2.2 do CP/M e do MP/M, cada arquivo é criado com o atributo $DIR para indicar que se trata de um arquivo "diretório" que o comando DIR pode encontrar e apresentar. Cada arquivo é também criado com o atributo $R/W ("read-write") para indicar que se pode *ler e gravar* (*e apagar*) o arquivo.

Quando um arquivo possui o atributo $SYS (o oposto de DIR), nem o comando DIR ou o comando PIP podem encontrá-lo. (O PIP não pode encontrar, a não ser que a pessoa utilize o parâmetro R descrito no Capítulo 3). O atributo $SYS é utilizado para "esconder" arquivos de DIR e PIP, e assim evitar que outros usuários do sistema conheçam o arquivo ou o copiem. Observe, porém que o comando STAT pode ser usado por outros usuários para apresentar até arquivos de "sistema", e, portanto, esta proteção (do PIP e DIR) não é suficiente. A pessoa *também* deve afixar o atributo $R/0 (read-only) (o contrário de SR/W) ao arquivo, e depois proteger a documentação do comando STAT. Essencialmente, esses arquivos estão apenas protegidos contra o *seu* emprego *errôneo* no sistema. Se a pessoa utilizar erroneamente os seus discos, irá enfrentar outro problema.

NOTA: O comando ERA *irá* encontrar arquivos de "sistema", mas não suprimirá arquivos "read-only". Deve-se usar STAT para alterar o atributo $R/0 para $R/W caso queira suprimir-se qualquer arquivo $R/0.

Aqui estão alguns exemplos do uso de STAT para alterar e apresentar atributos de arquivos:

```
A > STAT SAMPLE.TXT $R/0 ↵
```

Este comando transforma SAMPLE.TXT para ser apenas lido ("read-only").

```
A > STAT B:TEMP $R/W ↵
```

Este comando transforma TEMP no *drive* B para ser lido e gravado (de "read-only").

```
A > STAT SAMPLE.BAK $R/O ↵
A > STAT SAMPLE.BAK $SYS ↵
```

Este comando transforma SAMPLE.BAK em um arquivo "read-only" e do "sistema", que não pode ser encontrado por DIR:

```
A > DIR SAMPLE.BAK ↵
NOT FOUND
A > STAT SAMPLE.* $S ↵
  Size    Recs    Bytes    Ext    Acc
    48      48       6K      1    R/O    A:SAMPLE.TXT
    48      48       6K      1    R/W    A:SAMPLE.JNC
    48      48       6K      1    R/O    (A:SAMPLE.BAK)
```

O arquivo SAMPLE.BAK está entre parênteses por possuir o atributo $SYS, assim como o atributo $R/0 (mostrado na coluna 'Acc' e representando o método de acesso ao arquivo). Os outros arquivos têm o atributo $DIR, e SAMPLE.TXT tem o atributo $R/0 ("read-only"), ao passo que SAMPLE.JNC tem o atributo $R/W ("read-write").

Se a pessoa copiar um arquivo "read-only" ou "sistema", o PIP cria a nova cópia com default "read-write" ($R/W) e diretório ($DIR).

Deve-se usar STAT para dotar os arquivos de novos atributos.

Operações MP/M

Em um sistema MP/M, de múltiplos usuários, cada usuário tem um terminal e pode operar o sistema. Um sistema MP/M com um único usuário será idêntico à versão 2.2 do sistema CP/M. Com mais de um usuário, porém, o MP/M pode aparecer como um sistema CP/M separado para cada usuário. Em alguns ambientes, pode ser necessário que um "operador do sistema" lide com todos os recursos e gerencie o sistema para os usuários. Um operador de sistema poderia executar as seguintes operações:

- Mudar disquetes (usando o comando DSKRESET).
- Direcionar os arquivos para a impressora de linhas ou outro dispositivo (comandos SPOOL e STOPSPRL).
- Escalonar programas a serem executados posteriormente (comando SCHED).
- Fixar a hora e a data (comando TOD).
- Apresentar informações sobre o sistema (comando MPMSTAT).

DSKRESET (MP/M)

Embora o aprendiz possa aproximar-se de qualquer *drive* de disquete e retirar um disquete, isto não seria aconselhável no sistema MP/M, no qual outros usuários poderiam estar tendo acesso a arquivos no disquete. O MP/M possui o comando DSKRESET para informar os outros usuários que alguém deseja trocar um disquete. DSKRESET, assim como outros comandos do MP/M, existe em um disquete do sistema sob a forma de um arquivo 'COM' ou '.PRL' ("página relocável") e pode ser executado simplesmente digitando-se o seu nome primário:

```
0A > DSKRESET↵
Confirm reset disk system (Y/N)?Y
```

A mensagem "Confirme o reset do disco do sistema" aparece em cada terminal vinculado ao sistema. Cada usuário deve responder com um 'Y' (representa sim) para permitir o 'reset' do disco.

Spooling (MP/M)

Quando mais de um usuário deseja enviar arquivos à impressora, ou quando um usuário quer enviar vários arquivos, ele pode usar um *spool* (colocar em linha, um depois do outro, em uma *fila*), enrolar os arquivos ao dispositivo de dados, preenchidos com texto ASCII. (observe que arquivos-fonte, listagens, arquivos '.PRN' e quaisquer outros arquivos criados por um editor de textos como ED, ou um processador de palavras, são todos arquivos-texto).

A fim de enviar arquivos para a fila do *spool* utilize o comando SPOOL, que assume a seguinte forma:

```
SPOOL filename filename. . .
```

O primeiro *filename* é exigido; os subseqüentes são opcionais. O *filename* deve incluir a extensão, se houver alguma. Aqui está um exemplo:

```
0A > SPOOL PROG.PRN SAMPLE.TXT TEMP.LST↵
```

Este comando envia os arquivos PROG.PRN, SAMPLE.TXT, e TEMP.LST ao dispositivo LST (geralmente uma impressora).

Para parar uma operação de *spool* e esvaziar uma fila use o comando STOPSPLR. Aqui temos um exemplo:

```
0A > STOPSPLR↵
```

O MP/M tem a capacidade de manter a hora e oferecer a data precisa se a pessoa fixar adequadamente estes valores. Utilizando a hora e a data, pode-se escalonar os programas para que rodem em um horário especificado em uma data especificada. O sistema supervisiona constantemente a hora e a data para lidar com programas escalonados.

Execução de escalonamento (MP/M)

O programa SCHED pode ser empregado para escalonar a execução de outro programa. O programa SCHED possui ou uma extensão 'PRL' ou '.COM'. Quando o aprendiz o executa como um comando, deve fornecer os argumentos apresentados no seguinte formato:

```
SCHED mm/dd/yy hh:mm program
```

Introduza a sua data com mm/dd/yy, seu horário em horas (hh) (expresso de zero a vinte e quatro) e minutos (mm) (expressos de zero a sessenta). Admite-se que o seu programa seja um *filename* com uma extensão '.COM' ou '.PRL' ("página relocável"), a qual não é necessário fornecer. Aqui temos um exemplo:

```
0A > SCHED 12/31/80 23:59 EIGHTY↵
```

O programa EIGHTY.COM (ou EIGHTY.PRL) irá ser executado no dia 31 de dezembro de 1980, às 23 horas e 59 minutos, se o sistema estiver correndo e encontrar esta data. Observe que qualquer pessoa poderia interferir e alterar a hora e a data, já que o indicador de tempo (time) pode ser fixado e refixado a qualquer momento. Portanto, todos os usuários devem cooperar com o intuito de poder confiar nesta operação.

Hora do dia (MP/M)

Para apresentar a hora do dia, utilize a forma simples de comando TOD:

```
0A > TOD↵
Sat 12/29/80 02:20:14
```

Para refixar a hora e a data, use esta forma de comando TOD:

```
0A > TOD 12/29/80 02.22.00↵
Strike any key to set time
Sat 12/29/80 02:22:00
0A >
```

NOTA: O programa TOD apresenta a mensagem 'Strike any key'; portanto, pode-se pressionar qualquer tecla quando o aprendiz estiver pronto para fazê-lo.

Abortando um programa (MP/M)

Para abortar um programa vinculado, no momento, ao console tecle ↑C. (Isto não afeta nenhum programa destacado).

O comando ABORT irá abortar qualquer programa, mesmo que pertença a outro console.

Por exemplo:

```
2B > ABORT LISTING 2↵
```

aborta o programa LISTING, pertencente ao console no qual o comando foi digitado (console 2).

Também é possível digitar no console 4:

```
4A > ABORT LISTING 2↵
```

para abortar o programa LISTING. Observe que o número do console pode ser especificado, opcionalmente, depois do nome do programa.

Vinculando e desvinculando um programa (MP/M)

Um programa pode ser desvinculado do console teclando-se ↑D. Continuará a ser executado "invisivelmente" até que seja revinculado ao console. Isto libera o console para a execução de outro programa ou para a entrada de texto ou dados.

Teclar um ↑D irá desvincular o programa do console, desde que o programa verifique o estado do console, isto é, leia o comando. Também é possível desvincular um programa automaticamente, utilizando uma chamada de desvinculação XDOS.

Inversamente, um programa é revinculado a seu console com o comando ATTACH. Deve sempre ser revinculado ao mesmo console do qual for desvinculado.

Por exemplo:

```
0A > ATTACH PROG↵
```

NOTA: Teclar ↑D quando o TMP (Processo de Mensagem do Terminal) está sendo executado no console resulta na ativação do próximo processo, o que qualifica como pronto para rodas com a maior prioridade entre os que estão esperando pelo console. Observe, ainda, que o TMP está sendo executado sempre que um comando pode ser, ou está sendo digitado.

Console (MP/M)

Já que um número do usuário nem sempre corresponde, necessariamente, ao número do console, o comando CONSOLE foi providenciado para examinar o número do console sendo utilizado.

Por exemplo:

```
2A > CONSOLE↵
CONSOLE = 1
```

O exemplo acima mostra que o usuário número 2 está empregando o console número 1.

Diretório (MP/M)

O comando DIR trabalha de forma normal e possui uma extensão, a "opção S".

Por exemplo:

```
0A > DIR *.* S↵
```

irá incluir todos os arquivos que possuem o conjunto de atributos do sistema.

Erase (MP/M)

A forma usual do comando Erase é oferecida, assim como uma forma nova. O comando ERAQ pode ser usado para apagar um conjunto de arquivos que se associam a um padrão específico.

Por exemplo:

```
1B > ERAQ PROG. *↵
B: PROG COM? 4
B: PROG INT? 4
```

Vizualizando um arquivo (MP/M)

Para visualizar um arquivo, o sistema oferece o comando TYPE usual. Além disso, pode ser empregada uma modalidade de pausa. Quando esta opção é usada, o comando:

```
0A > TYPE PROG.TXT P15↵
```

irá apresentar quinze linhas de PROG.TXT, e depois fazer uma pausa até que seja teclado o ↵ .

Caracteres do controle do MP/M

O MP/M oferece as funções habituais de edição de linha do CP/M para que o usuário digite comandos, além de outras funções adicionais. Os caracteres de controle estão listados no Capítulo 6.

MPMSTAT (MP/M)

O MP/M oferece um comando STAT especial para apresentar o estado de execução

completo do próprio MP/M. O comando é:

```
0A > MPMSTAT ↵
```

Abaixo mostramos uma saída típica:

```
                ****** MP/M Status Display *****

Top of memory = FFFFH
Number of consoles = 02
Debugger breakpoint restart # = 06
Stack is swapped on BDOS calls
Z80 complementary registers managed by dispatcher
Ready Process(es):
  MPMSTAT  Idle
Process(es)      DQing:
  [Sched     ]  Sched
  [ATTACH    ]  ATTACH
  [CLiQ      ]  cli
Process(es)      NQing:
Delayed Process(es):
Polling Process(es):
  PIP
Process(es) Flag Waiting:
  01 - Tick
  02 - Clock
Flag(s) Set:
  03
Queue(s):
  MPMSTAT     Sched     CliQ        ATTACH     MXParse
  MXList      [Tmp0     ] MXDisk
Process(es Attached to Consoles:
  [0] - MPMSTAT
  [1] - PIP
Process(es) Waiting for Consoles:
  [0] - TMP0      DIR
  [1] - TMP1
Memory Allocation:
  Base = 0000H    Size = 4000H    Allocated to PIP     [1]
  Base = 4000H    Size = 2000H    *  Free  *
  Base = 6000H    Size = 1100H    Allocated to DIR     [0]
```

NOTA: Uma interpretação detalhada deste *display* de estado vai além do deste capítulo. O *display* foi incluído, primariamente, para tornar o texto mais completo, e pode ser omitido em uma primeira leitura.

O significado simplificado do *display* é o seguinte:

Processo(s) pronto(s): Esta lista mostra todos os processos prontos em ordem de prioridade. O processo com a prioridade mais elevada é aquele que está correndo.

Processo(s) DQing: Cada fila é mostrada, junto com os processos que executaram uma operação de leitura na fila. Os processos são listados em ordem de prioridade e estão aguardando uma mensagem a ser inserida na fila.

Processo(s) NQing: O mesmo que acima, exceto que os processos estão aguardando um "buffer" para escrever uma mensagem para a fila.

Processo(s) retardados: lista os processos retardados por um período de tempo especificado ("clock ticks").

Processo(s) de Polling: Lista os processos que estão "interrogando" um dispositivo de I/O aguardando um estado "pronto" (ready).

Processo(s) de espera de sinalizador: Lista os processos diante do número correspondente do sinalizador (plug).

Sinalizador(es) posicionados: Lista os sinalizadores (plugs) que estão posicionados.

Fila(s): Lista as filas no sistema. Caracteres em maiúsculas são usados para aquelas filas a que se pode ter acesso via comando de console. 'MX' no início do nome de uma fila significa exclusão mútua.

Processo(s) conectados com o console: Lista os processos e os números correspondentes do console.

Processo(s) aguardando consoles: Lista os processos de acordo com o console e a prioridade. Os processos foram desvinculados e agora estão aguardando o seu console para retomar a execução.

Alocação de memória: Apresenta um mapa da memória mostrando a base, o tamanho, o banco (se aplicável), e o processo residente, junto com o número do console.

Comandos MP/M adicionais:

O MP/M é equipado com três comandos adicionais que são complexos em aparência, e só são utilizados por programadores de linguagem assembler: GENMOD, GENHEX, e PRLCOM.

Estão listados aqui para completar o texto, mas podem ser omitidos em uma primeira leitura.

GENMOD (MP/M)

Este comando especial transforma o FILE 1 que contém dois arquivos hexadecimais concatenados (do tipo HEX), separados um do outro por 0100H bytes, em um FILE 2 que é passível de relocação por página do (tipo PRL).

O formato do comando é:

```
0A > GENMOD d: FILE1.HEX d:FILE2-PRL $DDDD ↵
```

onde $DDDD é um parâmetro opcional que especifica, em hexadecimal, a quantidade adicional de memória exigida pelo programa.

GENHEX (MP/M)

Este comando transforma um arquivo COM em um arquivo HEX. Este comando é muitas vezes usado antes de um GENMOD. Também pode ser especificado um deslocamento.

Por exemplo:

```
0A > GENHEX B:FILE.COM 200 ↵
```

PRLCOM (MP/M)

Este comando transforma um arquivo PRL em um arquivo COM:

```
0A > PRLCOM B: FILE1.PRL A: FILE2.COM ↵
```

GENSYS (MP/M)

Este comando é usado para gerar um sistema MP/M. Apresenta ao usuário um *prompt*, para que especifique todos os parâmetros e informações exigidas, e cria o arquivo MPM.SYS. O comando MPMLDR (descrito mais adiante) pode ser utilizado depois para carregar e executar o arquivo MPM.SYS.

O diálogo é apresentado abaixo. Um travessão indica uma resposta do usuário.

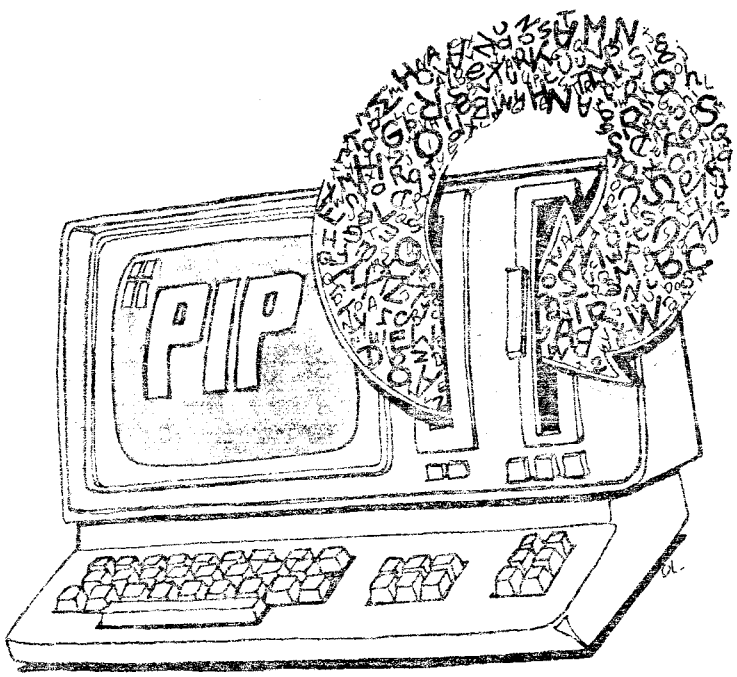
```
A > GENSYS ↵
                MP/M SYSTEM GENERATION

_____________________________________

_____,_______________________________

Top page of memory = ____

Number of consoles = __

Breakpoint RST# = __

Add system call user stacks (Y/N)? __

Z80CPU (Y/N)? __

Bank switched memory (Y/N)? __

Memory segment bases, (ff terminates list)
: __ __
: __ __
: __ __
: __ __
Select Resident System Processes: (Y/N)
ABORT       ? __
SPOOL       ? __
MPMSTAT     ? __
SCHED       ? __
```

NOTA: O diálogo acima varia, ligeiramente, se for especificado um Bank Switched Memory (Memória Comutada a Banco).

RESUMO

Todas as facilidades oferecidas pelo CP/M, desde os caracteres de controle aos comandos embutidos e transientes, foram apresentadas neste capítulo. Lembre-se de que devem ser conhecidos todos os caracteres de controle, quatro dos cinco comandos embutidos (DIR, TYPE, REN e ERA) e quatro dos nove comandos transientes, (PIP, ED, SYSGEN e STAT). Os outros comandos só serão úteis se o aprendiz planejar, elaborar e executar programas a nível de linguagem de máquina (ASM, LOAD, DUMP, DDT e SAVE) ou se verificar a conveniência de um arquivo SUBMIT.

No Capítulo 6, para referência, serão apresentados todos esses comandos, sob forma de resumo completo.



Capítulo 3

Manipulando Arquivos com PIP

INTRODUÇÃO

O comando PIP foi apresentado, brevemente, no Capítulo 1, onde foi utilizado para copiar um arquivo. O emprego primordial do PIP é copiar, ou mais precisamente, transferir arquivos. Porém ele pode fazer muito mais do que isso.

Neste capítulo, vamos aprender a respeito das facilidades oferecidas pelo PIP. Embora, provavelmente, possam ser utilizadas apenas algumas dessas facilidades, é importante que se conheçam todas as opções disponíveis. Portanto, aconselhamos a leitura deste capítulo por completo, e depois a volta ao estudo, detalhadamente, das seções de interesse.

O aprendiz ficará surpreso com as facilidades oferecidas pelo PIP. Mencionando apenas algumas delas, já aprender:

- Como juntar arquivos (concatenação).
- Como imprimir um texto formatado (utilizando tabuladores).
- Como cortar linhas que são longas demais para a tela (opção D).
- Como imprimir um grupo de arquivos com um único comando (utilizando uma especificação PRN).
- Como imprimir automaticamente um texto em páginas formatadas (opção P).
- Como ler um arquivo até uma palavra especificada, sem usar o editor (opção Q).

COMPREENDENDO O PIP

O PIP é um programa de transferência de arquivos. As letras PIP representam "Peripheral Interchange Command" (Comando de Intercâmbio entre periféricos). Como o seu nome indica, o PIP permite a transferência de arquivos entre quaisquer dois dispositivos. Até agora, efetuamos apenas transferências de arquivos para disco. Agora aprenderemos a utilizar opções adicionais disponíveis com o PIP. O PIP também pode processar os arquivos à medida que os transfere.

Apresentaremos, inicialmente, uma descrição completa da função mais importante do PIP: copiar arquivos. Depois, estudaremos as facilidades do PIP para transferir arquivos entre as várias unidades conectadas com o computador.

COPIANDO ARQUIVOS

Copiando um único arquivo

O PIP pode ser executado de duas maneiras:

1. Como um comando de linha única.
2. Como um "programa".

Eis aqui um exemplo que utiliza o PIP como um comando de linha única:

```
A> PIP B:COPY1.BAK = FILE1.TXT↵
A>
```

Admite-se que o FILE1.TXT encontra-se no *drive* atual. Este comando dreciona o PIP para fazer uma cópia de FILE1.TXT, denomina a nova cópia de COPY1.BAK e coloca o COPY1.BAK no *drive* B. Depois, o CP/M volta com o *prompt* do sistema (A >). Esta é uma maneira rápida de copiar um único arquivo.

O PIP pode também ser executado como um "programa", e utilizado para fazer uma seqüência de operações de cópia:

```
A> PIP↵
*B:COPY2.BAK = FILE2.TXT↵
*A: = B:SAMPLE.BAS↵
*A: = B:PROG.FOR↵
*↵
A>
```

Quando o PIP inicia sua execução, apresenta o *prompt* '*'. Agora os comandos PIP podem ser executados. A primeira linha no exemplo faz uma cópia de FILE2.TXT, que está no *drive* A, denomina a cópia COPY2.BAK, e depois coloca COPY2.BAK no *drive* B. A próxima linha faz uma cópia de SAMPLE.BAS, que está no *drive* B, utiliza o nome original (SAMPLE.BAS) para a cópia e a coloca no *drive* A. A próxima expressão também copia um arquivo do *drive* B para o *drive* A. Um simples RETURN termina o PIP, fazendo um retorno para o CP/M: o *prompt* de CP/M, 'A >' surge novamente na tela.

Apresentaremos, agora, as regras para se efetuar tais transferências. Para fazer uma cópia de um arquivo de um disquete para o outro, use esta forma de uma expressão PIP:

```
d:copy = d:original
```

onde 'd' é uma letra para o *drive*, 'copy' é o novo nome para a cópia do arquivo, e 'original' é o nome do arquivo original. Os dois 'ds' acima podem se referir ao mesmo *drive* ou a dois *drives* diferentes. O 'd' à direita pode ser omitido, já que o PIP admitirá que o arquivo se encontra no *drive* atual. O 'd' à esquerda também pode ser omitido, desde que se forneça o nome referente a 'copy'. Se desejar que o arquivo cópia tenha o mesmo nome do original, utilize esta forma abreviada:

```
d' = d:original
```

onde d' é uma letra de *drive* diferente de d: O nome da cópia pode ser omitido – O PIP admitirá que o nome do arquivo no novo *drive* é o mesmo do original. Não obstante, isto só funcionará se o *drive* do arquivo-cópia for diferente do *drive* do arquivo original, pois não se pode ter dois arquivos com o mesmo nome no mesmo disquete.

Por exemplo:

```
*B: = A:TEST.INT↵
```

irá copiar TEST.INT de A para B.

Estudemos agora alguns exemplos: Suponhamos estarmos no *drive* A. Admitindo-se que FILE1.NAD está no *drive* A e PROGRAM.TXT está no *drive* B, serão válidos estes comandos PIP?

```
A> PIP↵
(1) *A: = B:PROGRAM.TXT↵
(2) *B: = FILE1.NAD↵
(3) *A:FILEREV.NAD = A:FILE1.NAD↵
(4) *A:FILE1.TXT = FILE1.NAD↵
(5) *B:FILE1.NAD = FILE1.NAD↵
```

Sim, todos os comandos acima são válidos. Em (2) observe que o comando é equivalente a:

```
B: = A:FILE1.AND
```

Lembre-se de que o *drive* atual pode ser omitido. É então assumido pelo PIP que se trata do *drive* atual, isto é, A. O A poderia ter sido omitido em (3). FILE1.TXT em (4) *não* é o mesmo que FILE1.NAD. Isto é válido. O (5) poderia ter sido abreviado para ter a mesma forma que (2).

Copiando múltiplos arquivos

Múltiplos comandos PIP são empregados para copiar arquivos diferentes. Por exemplo, copiemos os três arquivos:

```
FILE1.NAD
LETTER.TXT
PROGRAM.INT
```

de B para A:

```
A> PIP↵
*A: = B:FILE1.NAD↵
*A: = B:LETTER.TXT↵
*A: = B:PROGRAM.INT↵
*↵
A>
```

Em casos especiais, porém, este procedimento pode ser simplificado com o uso dos *símbolos de associação* do PIP. Para facilitar a cópia de múltiplos arquivos, o PIP permite o uso de dois símbolos especiais: '?' e '*'. O '?' pode ser usado em um *file name* e irá associar com qualquer caractere que possa aparecer em seu lugar.

Por exemplo:

FILE?.NAD

Se associará com:

FILE1.NAD
FILE2.NAD
FILE3.NAD

mas não com: FILE44.NAD (um caractere a mais do que o permitido).

Copiaremos, agora, os três arquivos:

FILE1.NAD
FILE2.NAD
FILE3.NAD

do *drive* B para o *drive* A. Aqui está o comando:

A > PIP A:=B:FILE?.NAD ↵

que executará as três transferências com um só comando utilizando o caractere especial de associação. O diretório de B será examinado pelo PIP até que todas as possibilidades de fazer uma associação sejam esgotadas. Observe que, se houver outro arquivo em B denominado

FILES.NAD

nele também será transferido.

O segundo símbolo de associação, '*', é ainda mais potente. Irá associar qualquer coisa em seu campo, *sem se preocupar com o seu tamanho*. Por exemplo, admitamos que B contenha:

FILE1.NAD
FILE12.NAD
LETTER.TXT
CBASIC.INT
FILE1.BAK

Então, os caracteres *.NAD associarão

FILE1.NAD
FILE12.NAD

E FILE1.* associar-se-á com:

FILE1.NAD
FILE1.BAK

Também é possível escrever '* *' que simplesmente irá associar todos os arquivos do disquete no *drive* B. Isto será usado na próxima seção para copiar todos os arquivos.

Por exemplo, se desejarmos copiar todos os programas tipo COM do *drive* A para o *drive* B, podemos simplesmente digitar:

A > PIP B:=*.COM ↵

e todos os comandos serão copiados, sucessivamente, para o *drive* B.

Aprenderemos a copiar um único arquivo e um grupo de arquivos. Agora iremos aprender a copiar um disquete inteiro.

Copiando todos os arquivos

Observe que esta seção é intitulada "Copiando todos os arquivos" e não "Copiando um disquete inteiro". Isto ocorre porque o CP/M não está armazenado como um arquivo. Para poder copiar o CP/M, é necessário usar-se um comando especial, SYSGEN, descrito no Capítulo 2. Se um disquete contém apenas arquivos, então estaremos copiando todos os arquivos. Se o disquete, porém, contiver também o CP/M, estaremos copiando apenas os arquivos, não o CP/M.

Não se pode saber se o CP/M está em um disquete simplesmente examinando o diretório com o comando DIR; o CP/M não é um arquivo e não está listado como tal. Se o aprendiz quiser verificar se o CP/M está ou não em um disquete, deverá tentar executar o CP/M do mesmo, fazendo um CNTRL-C, por exemplo.

Utilizemos agora a facilidade de associação do pip copiando todos os arquivos. Um *filename match* pode ser utilizado como um nome de arquivo somente nos argumentos do *filename original*. Por exemplo, se o aprendiz desejasse copiar todos os arquivos do disquete do *drive* A para o disquete do *drive* B, digitaria este comando:

A > PIP B:=A:*.* ↵

Use esta forma do PIP para fazer cópias de disquetes:

PIP d':=d:*.*

onde *d'* é a letra do *drive* que contém o disquete novo e *d* é a letra do *drive* que contém o disquete antigo. O *filename match* '*.*' irá associar todos os *filenames*. *d'* deve ser diferente de *d*.

Na prática, ao copiar todos os arquivos, recomendamos que se digite:

A > PIP B:=A:*.*[OV] ↵

'V' é uma opção PIP que especifica 'verifique'; ela verifica se a cópia é idêntica ao original. Para maior segurança, este é o melhor comando a usar. Porém, o processo de cópia é

muito mais lento com a opção [V], de modo que muitos usuários não a empregam a não ser que o arquivo seja muito valioso. O 'O' é utilizado para arquivos que podem conter um símbolo de fim de arquivo, como os arquivos .INT.
NOTA: lembre-se de que o comando PIP

```
B: = A:*.*
```

irá copiar todos os arquivos em A, mas apenas os arquivos. Se A contém um "sistema", isto é, CP/M, este não será copiado por não ser um arquivo. Lembre-se de que o CP/M deve ser copiado com o comando SYSGEN.

Copiando um disquete

Em muitos computadores, existe um programa utilitário especial para copiar *disquetes inteiros* em alta velocidade. Se esta opção for usada, então o segundo disco será uma cópia completa do primeiro, incluindo o CP/M, se o sistema estiver no primeiro. Esta é, geralmente, a melhor maneira de copiar um disquete completo.

Por outro lado, quando o PIP copia um arquivo, ele o faz em "blocos" ou "setores" adjacentes no disquete. Isto faz com que programas como um editor (ou processador de palavras) ou um interpretador (BASIC) tenham acesso muito mais rápido ao arquivo.

Em resumo, usar PIP para copiar um disquete irá resultar em um arquivo "mais limpo". Utilizar um programa de cópia de discos, contudo, irá poupar tempo na operação de cópia.

CP/M (Versão 2.2) e MP/M

A expressão '*.*' para um *filename match* poderia não ser suficiente para sua instalação se o aprendiz tiver *áreas de usuário* (discutidas no Capítulo 2) separadas. Se fizer uso destas áreas, então o seu sistema deveria ter um programa especial que duplique disquetes. Caso obtenha uma mensagem de erro "formato inválido", pressione qualquer tecla *exceto* a tecla RETURN para trazer de volta o *prompt* do PIP. Se a operação PIP não ocorrer, tente de novo. Pressionando RETURN, o PIP termina, e o sistema operacional volta com o *prompt* do sistema.

Copiando para um disquete novo

Os dois métodos

Se a pessoa possuir três ou mais *drives* de disco, simplesmente deve inserir um disquete novo no *drive* C e copiar, utilizando seu arquivo de B para C, isto é, do segundo para o terceiro *drive*. Se possuir apenas dois *drives*, contudo, o processo de cópia se torna mais complicado. Admitiremos que o seu disco do sistema está no *drive* A, e que o arquivo a ser copiado está no *drive* B. Desejamos copiá-lo para um disquete novo. Dois métodos podem ser usados: transferir pelo A ou alternar disquetes.

Transferir pelo A

Este método é seguro, mas lento, e só funciona se o disquete no *drive* A tiver suficiente espaço de sobra. (Veremos mais adiante como verificar o espaço disponível com os comandos STAT ou DIR). Este método é bem simples:

1. O arquivo é transferido de B para A.
2. Coloca-se um novo disquete em B.
3. O arquivo é transferido de A para B, no disquete.

Lembre-se de que ↑C deve ser executado antes do passo nº 3, para que o CP/M possa gravar em um novo disquete. Por exemplo:

```
A > PIP A: = B:FILE.INT ↵
(place a new diskette in B)
↑C
A > PIP B: = A:FILE.INT ↵.
```

Verifique, então se seu arquivo está no disquete no *drive* B "checando" o diretório de B (utilizando 'DIR'), e apague a cópia extra de A ('ERA').

Alternar disquetes

Este método transfere o arquivo diretamente para um disquete novo. Para que a transferência possa ocorrer, três condições devem ser simultaneamente satisfeitas:

1. PIP deve estar executando.
2. O arquivo-fonte deve estar em um *drive*.
3. O disquete-destino deve estar em um *drive*.

Carrega-se o PIP do disquete para a memória do computador. Depois remove-se o disquete do sistema, e o PIP é executado na memória. Em outras palavras, uma vez que se tenha chamado o comando PIP, o programa PIP é carregado na memória do computador, e o disquete do sistema pode ser substituído por outro disquete para a transferência.

Admitindo-se que o disquete do sistema com PIP esteja no *drive* A, o processo é o seguinte:

1. Insira um disquete novo em B.
2. Aperte CTRL-C para fazer uma "partida a quente". Isto permite que CP/M reconheça o novo disquete em B, e possa gravar nele.
3. Chame o PIP digitando: 'PIP(CR)':

```
A> PIP↵.
```

O PIP agora está na memória, e pronto para ser executado.

Neste ponto, pode-se remover o disquete do sistema *momentaneamente* e inserir o disquete original (aquele que está sendo copiado) no *drive* A. Pode parecer surpreendente sugerir que se remova o disquete do sistema no qual reside o PIP. Lembre-se, todavia, de que sempre que um programa do tipo COM é executado, é carregado na memória do computador a partir do disco.

O PIP foi chamado, digitando

```
A> PIP↵.
```

Uma cópia do PIP está agora na memória do computador. Não necessitamos mais do disquete, e podemos removê-lo até desejarmos sair do PIP.
NOTA: Não termine o PIP antes de reinserir o disquete do sistema.

Não tecle o *return* depois de um *prompt*, se o PIP tiver sido ativado:

```
* ↵ (NÃO FAÇA ISTO!)
```

já que isto irá terminar o PIP. Não use, também, CTRL-C até ter posto o disquete de novo no lugar. Consulte as "dicas práticas" no Capítulo 7 para aprender uma proteção eficaz contra uma saída acidental do PIP.

Agora, coloque o disquete a ser copiado no *drive* A. Depois de aparecer o *prompt* do PIP, o aprendiz pode digitar uma expressão PIP. Lembre-se de que o disquete original está no *drive* A, e que o disquete novo está no *drive* B. Se estiver copiando o disquete original por completo, digite a expressão:

```
*B:=A:*.* ↵
```

ou, se você preferir:

```
*B:=A:*.* [V] ↵
```

com a opção de verificação.

O '*.*' é um "filename match" para todos os arquivos (na versão 2.2 do CP/M e em MP/M, o '*.*' [V]' associa todos os arquivos da sua área de usuário, e nenhum outro). Esta expressão PIP copia todos os arquivos do *drive* A (o disquete original) para o *drive* B usando os mesmos nomes para as cópias dos arquivos. Depois de executar, a pessoa terá uma cópia de cada arquivo no disquete novo, e os arquivos terão os mesmos nomes de antes.

Agora já se pode trazer de volta o sistema. Antes de terminar o PIP, retire o disquete original do *drive* A e recoloque o disquete do sistema nesse *drive*. Agora pode-se terminar o PIP, simplesmente pressionando a tecla RETURN:

```
* ↵
A>
```

Se o sistema não retornar, verifique o *drive* A. Se a luz do *drive* A estiver acesa, pode-se inserir o disquete do sistema, e o mesmo retornará. Se a luz estiver apagada, a pessoa precisa inserir o disquete do sistema, e dar nova partida, como descrevemos no Capítulo 1.

As instalações que só possuem dois *drives* devem utilizar o método que acabamos de descrever:

1. Coloque um novo disquete no *drive* B.
2. Tecle o CTRL-C.
3. Execute o PIP.
4. Retire o disquete do sistema.
5. Coloque o disquete original em A.
6. Execute o PIP para operações de cópia.
7. Ao terminar, retire o disquete original.
8. Recoloque o disquete do sistema.
9. Então, termine o PIP.

Lembre-se de colocar o disquete novo no *drive* B e o original no *drive* A. Se havia antes um disquete em B, execute um CTRL-C antes de fazer qualquer outra coisa, ou o CP/M se recusará a gravar em um disquete que "não conhece". O CTRL-C forçará o CP/M a ligar o disquete novo. Observe que isto não funciona às avessas. Se, ao invés do disquete do sistema, tivéssemos colocado o novo disquete em A o CP/M se recusaria a escrever nele, já que não o ligamos.

Uma vez removido o disquete do sistema, não se pode mais executar um CTRL-C, de modo que não é possível ligar um novo disquete em A. Não obstante, há uma "dica prática": faça uma cópia do CP/M e do PIP em seu novo disquete; depois poderá colocá-lo em um dos dois *drives* e fazer cópias à sua conveniência.

CDOS DA Cromemco

O CDOS não exige um CTRL-C para gravar em um disquete novo; portanto, o processo de cópia pode ser simplificado. Com o sistema no *drive* A, e o disquete a ser copiado no *drive* B, execute 'PIP ↵ ' como antes, e depois:

1. Retire o disquete do sistema.
2. Coloque o novo disquete em A.
3. Efetue a transferência.
4. Retire o novo disquete.
5. Recoloque o disquete do sistema em A.
6. Tecle "RETURN" para terminar o PIP.

O procedimento recomendado

Enquanto a pessoa ainda não tem experiência com o CP/M, deve transferir o seu arquivo usando o primeiro método (copiando por meio de A), já que este é mais seguro. Melhor ainda, copie o CP/M e o PIP em todos os seus disquetes para que não seja mais necessária a alternância. Embora a pessoa possa desejar tentar o segundo método de transferência, deve lembrar-se de que pode causar dano ao seu disquete original se sair cedo demais do PIP apertando "RETURN" sem antes ser removido o disquete original no qual residem os arquivos.

Abortando uma operação de cópia

Pressionando qualquer caractere do teclado durante uma transferência do PIP irá abortá-la. O PIP confirma-o apresentando a mensagem 'ABORTED' (abortado).

CÓPIA PARA DISPOSITIVOS

Introdução

A impressão de um arquivo é um exemplo de uma operação de transferência: o arquivo é copiado do disco para a impressora. O PIP oferece facilidades de transferência e permite que um arquivo seja transferido não apenas de disco para disco, mas, ainda, entre vários dispositivos.

Adiante descreveremos essas capacidades gerais. Aprenderemos como transferir entre quaisquer dois dispositivos que podem ser ligados ao computador. Introduziremos novos aspectos do PIP, como a concatenação, que pode ser usada em todas as operações de transferência, incluindo disco a disco. Mesmo que a pessoa não planeje empregar uma lei-

tura de cartões, é importante ler toda esta seção, já que ela também se aplica à impressão e à cópia de arquivos.

Consideraremos, inicialmente, a operação usada com mais freqüência: a impressão.

Imprimindo um arquivo

Um arquivo pode ser impresso pelo PIP ou outros programas. Se a pessoa estiver empregando um processador de palavras ou qualquer outro programa especial que possui uma "facilidade de impressão" (a capacidade de enviar o arquivo à impressora), deve usar tal programa para imprimir arquivos criados e aos quais o programa permite acesso. Por exemplo, se o aprendiz criou e adicionou dados a um arquivo "nome e endereço", utilizando um pacote de software "nome e endereço", é muito provável que também tenha um meio de imprimir o arquivo utilizando um programa especial fornecido como parte do mesmo pacote de software.

A principal vantagem de um programa de impressão especializado é que ele imprime o seu arquivo em um formato específico. Um programa de impressão, por exemplo, pode oferecer formatação, tabulações, espaçamento de linhas, paginação e outras opções automáticas de impressão.

Também podemos usar o comando TYPE do CP/M para listar um arquivo alfanumérico (consulte o Capítulo 1).

Por exemplo:

```
A > ↑P
A > TYPE FILE.TXT ↵
```

O CTRL-P liga a impressora, se ela estiver desligada. Este comando é fácil de usar, e oferece uma listagem "bruta"; em outras palavras, o arquivo aparece na impressora exatamente como está no disco sem nenhuma reformatação.

Apenas uma facilidade de formatação é oferecida por TYPE; ele irá expandir quaisquer caracteres de tabulação (CTRL-1) contidos no arquivo, e adota uma posição de tabulação para cada oitava coluna. O comando TYPE é usado para uma listagem rápida, para olhar para o início do arquivo ou, mais freqüentemente, apresentar depressa um arquivo na tela ao invés da impressora. O terminal CRT pode apresentar um texto a uma velocidade de 9.600 baud *versus* 300 a 600 baud para a impressora (aproximadamente). Utilizar o TYPE na tela irá, portanto, apresentar o texto muito mais rapidamente do que ocorreria se a listagem fosse realizada pela impressora.

Os arquivos também podem ser impressos como PIP, empregando suas facilidades gerais de transferência de arquivos. Descreveremos agora este processo.

Transferindo arquivos

Um arquivo pode ser enviado para qualquer dispositivo capaz de recebê-lo. Por exemplo, ele pode ser enviado a uma unidade de disco, a uma impressora, a um monitor de vídeo, a uma perfuradora de fitas de papel e para um gravador de cassete. Não pode ser enviado a uma leitora de cartões ou um teclado.

Uma impressora sem teclado só pode receber arquivos. Uma impressora com teclado se transforma em um terminal, e o teclado pode gerar um arquivo.

A Figura 3.1 mostra como um usuário pode "ler" a partir de um dispositivo ("input") e escrever ("output") para um dispositivo. O computador executa todos os programas e transfere todas as informações.

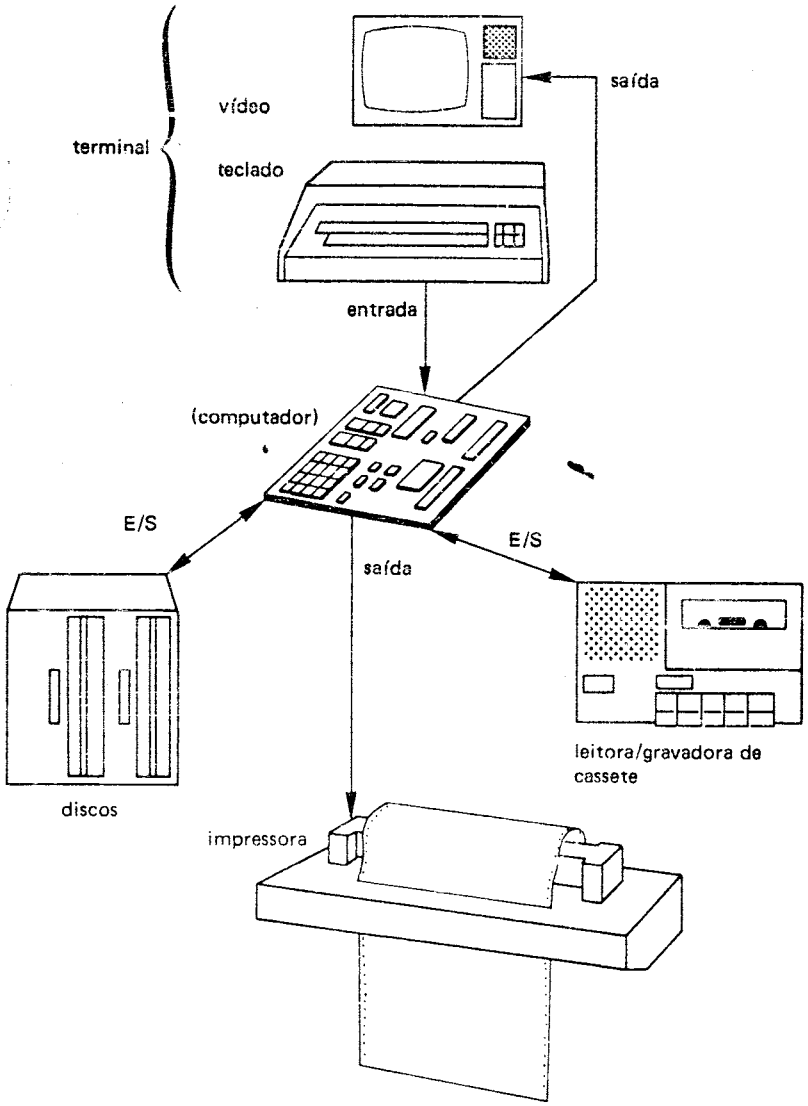


Figura 3.1: *Os elementos de um sistema*

Para listar um arquivo na impressora, o computador inicialmente lê o arquivo a partir do disco ("input") e depois o transfere para a impressora ("output"). A maioria dos programas lê arquivos dos discos em blocos (um setor de cada vez), de modo que podem transferir um arquivo longo utilizando apenas uma quantidade pequena de memória interna, como mostra a Figura 3.2.

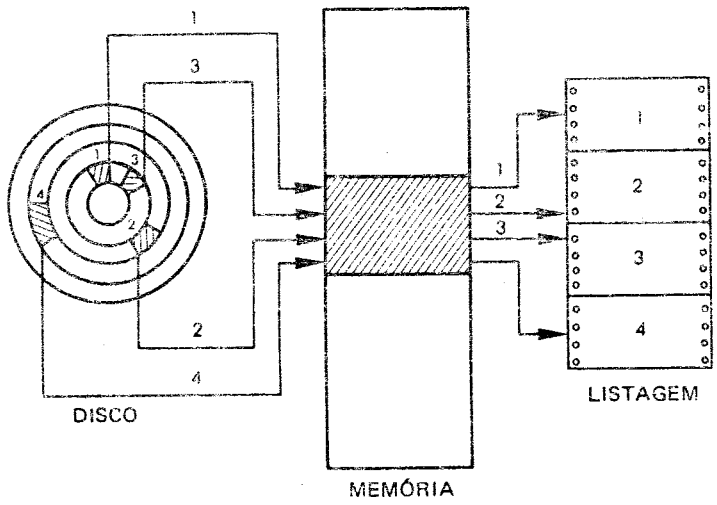


Figura 3.2: *Transferência pela memória*

Caracteres e arquivos também podem entrar a partir do teclado ou sair para o vídeo. Podem sair para o disco em uma operação de transferência disco-a-disco. Programas específicos de transferência de arquivos devem estar disponíveis para oferecer estas facilidades. Muitas vezes, pacotes de aplicação incluem programas especializados que executam uma destas funções. Não obstante, o programa PIP (um comando) pode ser empregado como se fosse uma facilidade geral para enviar uma cópia de um arquivo para um dispositivo, ou receber informação de um dispositivo para colocá-la no arquivo. Várias expressões PIP, combinadas com palavras-chave especiais para representar dispositivos, podem executar operações de cópia potentes e complexas. Vamos descrevê-las aqui.

Para aprender as expressões PIP válidas, é necessário designar apropriadamente os dispositivos. Examinaremos, em primeiro lugar, as convenções PIP para especificar dispositivos.

Especificando o dispositivo

As palavras-chave usadas nas expressões PIP são nomes "lógicos" além de físicos. Um nome *físico* para um dispositivo é o verdadeiro nome do dispositivo vinculado a seu sistema. Um nome *lógico* para um dispositivo é como se fosse um nome "genérico", que pode, na realidade, designar qualquer um dos vários dispositivos "list", que usualmente é a impressora, mas que também poderia ser outro dispositivo como um teletipo, ou um "modem" usado para se comunicar por intermédio de uma linha telefônica. A pessoa não precisa conhecer o nome comercial de tal dispositivo (como Teletype ou Hazeltine), pois apenas o nome lógico lhe interessa.

Os nomes lógicos de dispositivos permitidos em expressões PIP são:

CON: para "console" ou terminal, incluindo o teclado e o *display* ("INPUT/OUTPUT").

RDR: para leitora de cartões ou fita de papel (somente INPUT).
PUN: para perfuradoras de cartões ou fita de papel (somente OUTPUT").
LST: para um dispositivo de "listagem" como a impressora de linhas (somente "OUTPUT").

Observe que suas atribuições para RDR: e PUN: provavelmente não serão dispositivos de cartões ou fitas de papel perfuradas, já que estes estão, rapidamente, se tornando antiquados. Na maioria das vezes são usadas entradas e saídas "batch" (discutidas nos Capítulos 2 e 4). Da mesma forma, o seu dispositivo CON: geralmente é o seu terminal CRT (Tubo de raios catódicos) com um teclado; portanto, é um dispositivo de entrada e saída. O seu dispositivo LST: geralmente é a sua impressora. Ambos os dispositivos de console e listagem poderiam ser teletipos ou qualquer outra impressora equipada com um teclado.

A pessoa pode querer saber como é que o PIP conhece a rapidez com que deve dar saída aos caracteres, já que as impressoras operam com velocidades diversas. Isto ocorre porque o CP/M sempre "é feito sob medida" para uma instalação específica. Quando o CP/M for configurado para a sua instalação, já foram incluídas rotinas específicas para a sua impressora, o seu *display* CRT e o seu controlador de disco. Se o aprendiz trocar os seus dispositivos de entrada e saída, é necessário alterar o seu disquete CP/M para as exigências dos novos dispositivos.

O programa (comando) STAT descrito no Capítulo 2 apresenta as atribuições de dispositivos para cada nome lógico. Pode-se também alterar tais atribuições usando o comando STAT.

Exemplos práticos

Pode-se utilizar os nomes lógicos (previamente descritos) nas expressões PIP. Por exemplo:

```
A > PIP ↵
*CON: = SAMPLE.TXT ↵
*LST: = B:SIMPLE.BAK ↵
*PROG.BAS = RDR: ↵
*PUN: = PROG.BAS ↵
* ↵
A >
```

A primeira expressão PIP envia uma cópia do arquivo SAMPLE.TXT (que está no *drive* atual [A]) para o dispositivo console (provavelmente o seu terminal de *display*). Isto é mostrado pela Figura 3.3.

A segunda expressão transfere uma cópia do arquivo SIMPLE.BAK no *drive* B para o dispositivo atual de listagem LST: (geralmente a impressora de linhas ou um teletipo). Mostramos isto na Figura 3.4.

A terceira expressão PIP lê a informação vinda do dispositivo de leitura RDR:, e cria o arquivo PROG.BAS. Em geral, trata-se de um programa em fita de papel, cartão perfurado ou cassete, que pode ser lido para o sistema e armazenado em um arquivo de disco usando esta expressão PIP (veja Figura 3.5).

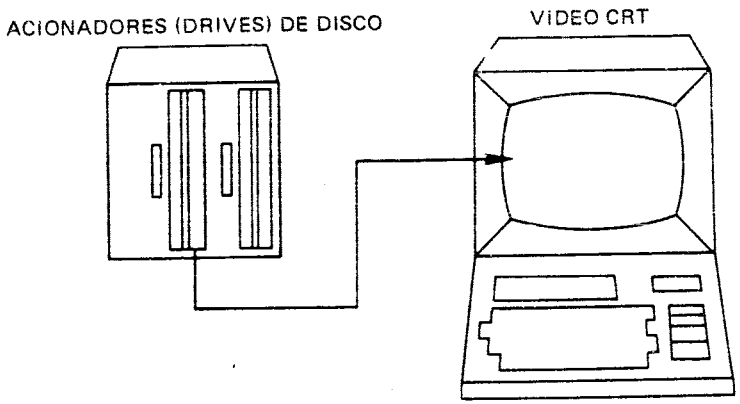


Figura 3.3: *Um arquivo é transferido para o console: CON: = SAMPLE.TXT*

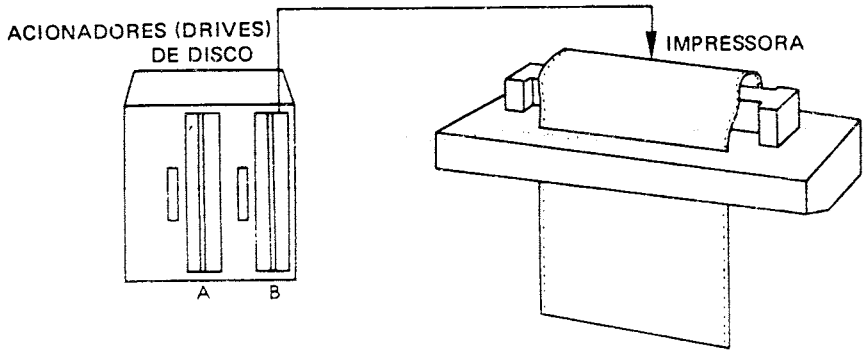


Figura 3.4: *LST: = B; SIMPLE.BAK*

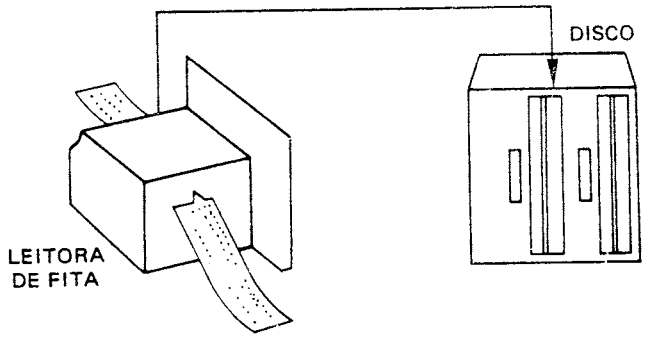


Figura 3.5: *PROG.BAS = RDR*

A operação oposta envia uma cópia do arquivo para a perfuradora de fitas de papel, para a perfuradora de cartões, ou para um gravador-cassete; é a última expressão PIP que envia uma cópia de PROG.BAS para o dispositivo de perfuração (PUN:), como mostramos na Figura 3.6.

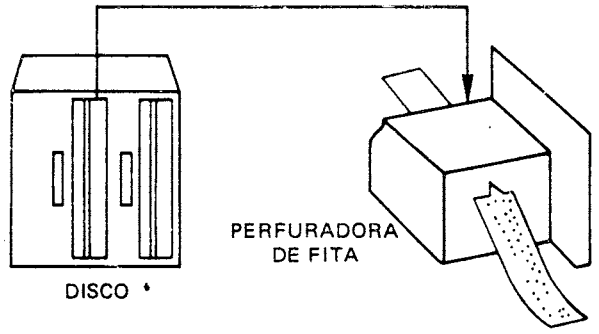


Figura 3.6: *PUN: = PROG.BAS*

Nomes físicos de dispositivos

Se for conveniente, também podemos utilizar nomes físicos de dispositivos nas expressões PIP. Os seguintes nomes físicos de dispositivos são válidos:

TTY:	para um console ou terminal, uma leitora, uma perfuradora ou um dispositivo de listagem (teletipo).
CRT:	para um console ou terminal, ou dispositivo de listagem (tubo de raios catódicos).
PTR:	para uma leitora de cartões ou fita de papel.
PTP:	para uma perfuradora de cartões ou fita de papel.
LPT:	para um dispositivo de listagem (impressora de linhas).
UCI:	para um console ou terminal definido pelo usuário.
UR1:	para uma leitora definida pelo usuário.
UR2:	para uma segunda leitora definida pelo usuário.
UP1:	para um dispositivo de saída (perfuração) definido pelo usuário.
UP2:	para um segundo dispositivo de saída definido pelo usuário.
UL1:	para um dispositivo de listagem definido pelo usuário.

NOTA: BAT: não é incluído, já que apenas reatribui os valores para RDT: e SLT:

OPERAÇÕES ESPECIAIS DE CÓPIA

Introdução

Aprendemos, agora, a executar todas as operações de cópia simples. Nesta seção aprenderemos a efetuar operações de transferência mais complexas em arquivos-texto e "arquivos hex". O PIP não é apenas um programa simples de "cópia", mas um programa

genérico de transferência equipado com um certo número de opções de processamento. Tais opções de processamento serão agora descritas, detalhadamente.

Nomes especiais de dispositivos

O PIP oferece vários nomes especiais de dispositivos que resultam num processamento especial de um arquivo. Os seguintes nomes de dispositivos adicionais podem ser utilizados ao fazer transferências com PIP:

NULL: Envia 40 "caracteres nulos" (ASCII código 0) ao dispositivo, geralmente uma perfuradora para saída. Exemplo (no qual PROG.HEX é enviado à perfuradora):

```
*PUN:=PROG.HEX,NULL:↵
```

EOF: Envia um fim de arquivo (ASCII ↑Z) ao dispositivo (enviado automaticamente pelo PIP durante transferências de arquivos-texto ASCII, e apenas necessário em casos especiais). Exemplo:

```
*PUN:=NUL:,X.ASM,EOF:,NULL:↵
```

Este exemplo envia 40 caracteres nulos para a perfuradora, seguidos de uma cópia do arquivo C.ASM, seguida do caractere fim de arquivo (↑Z) e mais 40 caracteres nulos.

PRN: O mesmo que LST: (envia a impressora), exceto que as tabulações são expandidas a cada oitavo caractere, as linhas são numeradas (como no programa ED) e "page ejects" (alimentação de formulários) são inseridas em cada 60 linhas (para fazer avançar a impressora para a próxima página), com um "page eject" inicial. Exemplo:

```
*PRN:=SAMPLE.TXT↵
```

INP: Código de dispositivo especial de entrada que pode ser vinculado ao próprio programa PIP (deve-se escrever o "patch" em linguagem assembler e acrescentá-lo ao PIP). O PIP recebe o caractere de entrada, chamando uma posição na memória (103H) e armazenando os dados iniciando na posição 109H (o bit de paridade deve ser zero – utilize o parâmetro Z).

OUT: Código de dispositivo especial de saída que pode ser vinculado ao próprio programa PIP, como INP: descrito acima. O PIP chama a posição 106H e envia os dados no registro C (cada um dos caracteres).
NOTA: para programadores em linguagem assembler: as posições 109H até 1FFH da imagem de memória do PIP são usadas e podem ser substituídas com códigos para *drives* de dispositivos de propósito especial (use DDT – o depurador de CP/M oferecido pela Digital Research com o CP/M ou MP/M). Exemplos:

```
*MODEL.CLK=INP:↵
```

(a entrada do dispositivo especial é armazenada no arquivo MODEL.CLK)

```
*OUT:=MODEL.CLK↵
```

(a cópia do MODEL.CLK é enviada ao dispositivo especial)

Arquivos-texto (ASCII) enviados a dispositivos

A maioria dos arquivos de dados e *todos* os arquivos-texto criados pelo programa-editor ou programas de processamento de palavras são arquivos-texto em formato ASCII. Outros arquivos, como os arquivos-comando (.COM), os arquivos-programa BASIC intermediário (.INT) e arquivos em linguagem de máquina (.HEX) são, na realidade, programas escritos em uma linguagem de alto nível (como BASIC) ou em linguagem assembler onde os códigos binários representam números reais ou instruções, mas *não* textos.

É importante que se conheçam as diferenças entre arquivos se o aprendiz executar operações de cópia especiais, como traduzir caracteres em MAIÚSCULAS para minúsculas ou suprimir caracteres ao copiar, ou copiar partes de um arquivo. Só se pode fazer isto com arquivos-texto ASCII, porquanto o PIP espera que um certo caractere (o caractere produzido pressionando simultaneamente CTRL e Z, isto é, ↑Z) esteja no fim do arquivo, para que possa procurar facilmente os caracteres. Pode-se também concatenar (juntar) vários arquivos-texto ASCII, utilizando o PIP (descrito mais adiante neste capítulo).

Certos dispositivos só podem receber ou enviar arquivos-texto ASCII. Pode apenas enviar arquivos-texto à impressoras ou *displays* (console), por exemplo, mas outros dispositivos, como RDR e PUN:, podem enviar ou receber qualquer tipo de arquivo. A informação em um arquivo pode ser codificada de várias maneiras. Por exemplo, um arquivo-texto é em geral codificado em formato ASCII, onde se utiliza um código de 8 bits (um "byte") para representar cada caractere, incluindo os caracteres de controle. Este código é apresentado na Figura 3.7.

Não obstante, programas que foram processados por um compilador, geralmente são representados em um código mais compacto, denominado hexadecimal, que usa apenas 4 bits para representar 16 símbolos.

Ao transferir um arquivo para a impressora ou para um *display*, é essencial especificar se é hexadecimal (dois dígitos por byte) ou ASCII (um caractere por byte). O PIP transfere um arquivo até chegar o caractere de fim de arquivo em arquivos-texto ASCII (↑Z), ou o fim real do arquivo em outros arquivos (exceto quando o PIP está apenas transferindo partes de um arquivo).

Concatenando arquivos-texto

A concatenação é um comando importante, utilizado para agrupar vários arquivos-texto em um só. Observe, entretanto, que o espaço disponível no disco deve ser suficiente para acomodar o arquivo final.

O exemplo mais simples é concatenar dois arquivos:

```
A>PIP↵
*BIG.TXT=PART1.TXT,PART2.TXT↵
```

NÚMEROS DE BITS								0	0	0	0	1	1	1	1
								0	0	1	1	0	0	1	1
								0	1	0	1	0	1	0	1
b_7	b_6	b_5	b_4	b_3	b_2	b_1	HEX 1 / HEX 0	0	1	2	3	4	5	6	7
			0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
			0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
			0	0	1	0	2	STX	DC2	"	2	B	R	b	r
			0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
			0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
			0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
			0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
			0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
			1	0	0	0	8	BS	CAN	(	8	H	X	h	x
			1	0	0	1	9	HT	EM	)	9	I	Y	i	y
			1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
			1	0	1	1	B	VT	ESC	+	;	K	[	k	{
			1	1	0	0	C	FF	FS	,	<	L	\	l	|
			1	1	0	1	D	CR	GS	-	=	M	]	m	}
			1	1	1	0	E	SO	RS	.	>	N	^	n	~
			1	1	1	1	F	SI	US	/	?	O	_	o	DEL

NUL – Caractere nulo
SOH – Início de cabeçalho
STX – Início de texto
ETX – Fim de texto
EOT – Fim de transmissão
ENQ – Consulta
ACK – Confirmação
BEL – Campainha
BS – Retrocesso
HT – Tabulação horizontal
LF – Avanço de linha
EM – Terminal do medium
ESC – Escape
GS – Separador de grupo
US – Separador de unidade
DEL – Supressão

VT – Tabulação vertical
FF – Alimentação de formulários
CR – Retorno do carro
SO – Mudança para números
SI – Mudança para letras
DLE – Dataline Escape
DC – Controle de dispositivo
NAK – Confirmação negativa
SYN – Inativo sincronizado
ETB – Fim de transmissão de bloco
CAN – Cancelamento
SUB – Substituição
FS – Separador de arquivos
RS – Separador de registro
SP – Espaço (em branco)

Figura 3.7: *Tabela de conversão ASCII*

DECIMAL	BINÁRIO	HEXADECIMAL
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Figura 3.8: *Tabela de conversão hexadecimal*

Pode-se também juntar vários arquivos-texto de uma só vez:

```
A> PIP↵
*FINAL.ASM = SUB1.ASM,SUB2.ASM,TEMP.ASM ↵
*B:NEW.ZOT = A:OLD.ZAP,B:OLD.ZOT,A:NEW.ZAP ↵
* ↵
A>
```

Neste exemplo, juntamos cópias dos arquivos SUB1.ASM,SUB2.ASM e TEMP.ASM (todos no *drive* atual A) nesta ordem (isto é, SUB1.ASM é o primeiro, SUB2.ASM o segundo etc.) em uma cópia denominada FINAL.ASM. A segunda expressão PIP junta uma cópia de OLD.ZAP no *drive* A com OLD.ZOT no *drive* B e com NEW.ZAP no *drive* A, e a cópia resultante é colocada no *drive* B com o nome de NEW.ZOT.

O PIP sempre admite que se trata de arquivos-*texto,* com cada um terminando com o caractere de fim de arquivo (ASCII ↑Z). Se são arquivos-texto, o PIP não terá problemas ao juntá-los (removendo os caracteres ↑Z e colocando um no fim da nova cópia para denotar fim do arquivo). Se *não* forem arquivos-texto, a pessoa deverá ler a próxima seção, "Concatenando arquivos não-texto".

Uma outra maneira de concatenar é especificar o arquivo *principal* como cópia.

Por exemplo:

```
A> PIP FIRST.TXT = FIRST.TXT,SECOND.TXT,THIRD.TXT ↵
A>
```

Este comando *não* irá modificar os conteúdos iniciais de FIRST.TXT ou dos outros arquivos, mas irá "append" (isto é, acrescentar ao fim) acrescentar ao FIRST.TXT os arquivos SECOND.TXT e THIRD.TXT. O último FIRST.TXT conterá, no seu início, os conteúdos do FIRST.TXT inicial. Observe, entretanto, que o PIP ainda irá usar um arquivo temporário denominado FIRST.$$$ durante a transferência, e que o disco deve ter suficiente espaço disponível.

A concatenação pode ser usada para fazer a listagem ou para examinar vários arquivos ao mesmo tempo, com um único comando.

Por exemplo:

```
A > PIP LPT: = FIRST.TXT,SECOND.TXT
```

irá imprimir os dois arquivos, em seqüência, na impressora.

Ao usar o XFER da CROMEMCO ao invés do PIP, os caracteres de controle são diferentes do PIP e deveriam ser aprendidos especificamente. Em particular, o XFER exige um caractere de controle para concatenar arquivos (ou o CTRL-Z permanecerá no fim de cada arquivo). Uma "dica prática": ao concatenar arquivos, assegure-se de que existe espaço suficiente no seu disquete para o arquivo resultante.

Concatenando arquivos não-texto

Quando se está concatenando (juntando) arquivos não-texto, cada arquivo deixa de possuir um caractere de fim de arquivo e portanto o PIP não irá copiar depois de alcançar o fim real do arquivo (isto é, se não houver um caractere de fim de arquivo, mas apenas o fim do mesmo). Para forçar o PIP a copiar o próximo arquivo e juntá-lo ao anterior, é preciso utilizar um "parâmetro de transferência" apresentado aqui como "X" (discutido na seção "Parâmetro em operações de cópia", neste capítulo). Os parâmetros são incluídos entre colchetes ([ ]) e precisam aparecer na expressão PIP depois do arquivo ou do dispositivo a que se aplicam.

Por exemplo:

```
A > PIP FINAL.HEX = TEMP1 [X], TEMP2 [X], TEMP3 ↵
```

Este comando PIP concatena cópias dos arquivos TEMP1, TEMP2 e TEMP3, e nomeia a cópia resultante FINAL.HEX. Os parâmetros X obrigam o PIP a deixar de lado os fins reais dos arquivos TEMP1 e TEMP2, e a efetuar a concatenação (só usado com arquivos não-texto).

Copiando arquivos Hex

Arquivos hexadecimais são criados por um *montador*. Um montador traduz um programa em linguagem assembler em um *código de máquina* (seqüências de números binários correspondentes a instruções) que é armazenado em um arquivo hexadecimal.

O montador CP/M cria um arquivo HEX, isto é, um arquivo com uma extensão HEX. A extensão HEX possui um significado especial para o PIP: o PIP admite que o arquivo esteja no "formato hex" Intel, e automaticamente confere o formato apropriado, valores hexadecimais válidos e verifica as somas. Portanto a extensão HEX deve ser utilizada com cuidado.

Se a pessoa quiser fazer uma cópia de um arquivo tipo HEX, pode usar uma expressão PIP com o parâmetro H ou I (para a transferência de dados *hexadecimais*). Quando se utiliza o parâmetro H, o PIP confere todos os dados para assegurar um formato hexadecimal apropriado para o Intel. Se houver um erro nos dados (isto é, se não tiverem o formato hex apropriado) o PIP emite um *prompt* no seu terminal para solicitar uma ação corretiva. O parâmetro H também remove caracteres não essenciais entre registros hexadecimais durante a operação de cópia.

O parâmetro I automaticamente põe em ação o parâmetro H (faz o que H faz, e ainda outras coisas). Se a pessoa usar o parâmetro I, o PIP ignora os registros ':00' no arquivo original formatado em hexadecimal (intel hex) e faz uma verificação de formatos hexadecimais não apropriados.

Se o aprendiz estiver copiando *de um dispositivo* para um arquivo com uma extensão 'HEX' explícita, o PIP confere a validade dos dados em formato hexadecimal Intel com os registros de soma de verificação. Em outras palavras, se você estiver copiando de uma leitora de fita de papel para um novo arquivo SAMPLE.HEX, você não precisa usar o parâmetro HEX para fazer a conferência do formato hexadecimal (mas você necessita do parâmetro I para ignorar os registros ':00').

Se o PIP sentir um formato não válido ou um erro de soma de verificação, transmite uma mensagem de erro para o seu terminal e aguarda uma correção. Caso esteja copiando de fita de papel, a pessoa poderá fazer retroceder a fita uns 50 cm e passá-la novamente. Quando a fita estiver pronta, pressione um simples RETURN e o PIP irá tentar copiar a partir da fita.

NOTA: se o dispositivo for o RDR:, pode-se dar entrada do caractere de fim de arquivo (↑Z) a partir do teclado do terminal enquanto a operação PIP está copiando. O PIP lê a partir do dispositivo enquanto supervisiona o seu teclado e espera que se tecle um ↑Z para terminar a operação de cópia.

Aqui estão exemplos de expressões PIP, utilizando essa extensão:

```
*X.HEX = CON:, Y.HEX [I], PTR: ↵
```

Hex format records typed at terminal

```
↑Z

*
```

Nesta expressão, o PIP copia para X.HEX primeiramente a partir do dispositivo CON: (o seu terminal) até que o operador tecle um Z. Depois, o PIP copia a partir de Y.HEX e ignora os registros '.00'. Finalmente, o PIP copia da leitora de fitas de papel PTR: até que encontre um fim de arquivo (↑Z).

```
*PROG.X = KLUDGE.HEX [H] ↵
*
```

Esta expressão copia o arquivo KLUDGE.HEX, em formato hexadecimal, para PROG.X e confere se existem formatos hexadecimais não válidos durante a transferência.

PARÂMETROS EM OPERAÇÕES DE CÓPIA

Os parâmetros

Descreveremos, agora, algumas das opções de processamento disponíveis durante as transferências. Apesar de só se usar algumas, é importante saber que elas existem. Parâmetros são letras especiais incluídas entre colchetes que seguem um *filename* em uma expressão PIP e afetam a cópia desse arquivo. A pessoa pode especificar mais de um parâmetro em uma expressão PIP. Alguns parâmetros exigem outra letra ou letras ou dígitos, que são operações avançadas de cópia, e o seu conhecimento não é essencial para o usuário que raramente utilizará o PIP.

B transferência em blocos. O PIP coloca os dados em um "buffer" até ler um caractere ASCII'X off' (↑S) a partir do dispositivo. O PIP, então, limpa o "buffer" do disco e volta para obter mais dados. O tamanho do "buffer" depende do tamanho do seu sistema (consulte a documentação oferecida junto com o seu sistema). Utilize esse parâmetro para transferir dados de um dispositivo de leitura contínua como um gravador cassete ou uma leitora.
Por exemplo:

```
*SERVE.TXT = RDR:[B]
```

Dn O PIP suprime caracteres que se estendem além da coluna 'n' (colunas verticais em seu terminal) enquanto copia os arquivos-texto. Utilize esse parâmetro para truncar linhas longas, caso esteja enviando o arquivo para um "dispositivo limitado", como uma impressora de baixo custo ou um monitor de 40 colunas. Exemplo:

```
*PRN: = LONG.TXT [D40]
```

Este comando também pode ser usado para "cortar" comentários de um programa, se eles aparecerem em uma posição específica.

E O parâmetro E reapresenta todas as operações de cópia na tela do terminal à medida que vão sendo executadas. Exemplo:

```
*COPY.TXT = SOURCE.TXT,S2.TXT,S3.TXT,S4.TXT [E]
```

Isto é útil no caso de uma seqüência de transferências.

F O PIP filtra alimentadores de formulários do arquivo (isto é, os remove). Pode-se também usar o parâmetro P para inserir novos alimentadores. Isto lhe permite "limpar" um arquivo depois de modificá-lo para uma impressão nítida.

H Transferência de dados hexadecimais: o PIP confere todos os dados para verificar se se ajustam ao formato hexadecimal Intel apropriado. Isto exige um arquivo .HEX.

I Ignore registros ':00' na transferência de arquivos com formato hex Intel (automaticamente posiciona o parâmetro H). Isto exige um arquivo .HEX.

L Traduza todos os caracteres em MAIÚSCULAS para minúsculas.

N Acrescente números de linhas a cada linha copiada para o novo arquivo (iniciando com a linha 1). Cada linha é seguida de dois pontos. Os zeros iniciais (por exemplo, 003) são suprimidos, a não ser que se especifique o parâmetro 'N2'. 'N2' não suprime os zeros iniciais e insere um espaço entre os números. Pode-se expandir os espaços de tabulação usando o parâmetro T. Isto é útil para poder consultar uma listagem.

O Transferência de arquivos-objeto (para arquivos que não estão em ASCII):
O PIP ignora o fim físico do arquivo durante a concatenação (veja "Concatenando arquivos") e a transferência.

Pn O PIP inclui "page ejects" em cada "enésima" linha (com um "page eject" inicial). Se n for 1 (ou se você não especificar 'n'), ocorrerão "page ejects" de 60 em 60 linhas (denominamos isto especificação *default*). Se a pessoa também usar o parâmetro F, o PIP remove os alimentadores de formulários antes de inserir "page ejects". Este é um método conveniente para fazer uma impressão num formato de página fixado.

Q string ↑Z O PIP pára de copiar do dispositivo ou do arquivo quando encontra 'string' de caracteres que a pessoa especifica (um 'string' é um grupo de caracteres; por exemplo, STRING 105%). O aprendiz termina seu 'string' com um ↑Z (CTRL e Z, simultaneamente). Veja "Copiando partes de arquivos" neste capítulo. Esta é uma maneira conveniente de se listar uma parte de um arquivo.

S string ↑Z O PIP começa a copiar do dispositivo ou do arquivo quando encontra o 'string' de caracteres que o aprendiz especifica. Termine o seu 'string' com um ↑Z. Veja "Copiando partes de arquivos" neste capítulo. Este é um meio conveniente de listar uma parte de um arquivo, começando em uma dada posição.

Tn Expande o espaço de tabulação para cada 'enésima' coluna durante a transferência de arquivos-texto. Você cria um espaço de tabulação em um arquivo-texto usando ↑I; este parâmetro expande o espaço de tabulação a partir da sua coluna usual fixa (colunas verticais na sua tela do terminal).

U Traduza todos os caracteres em letras minúsculas para letras MAIÚSCULAS durante a cópia de arquivos-texto.

V O PIP verifica se os dados foram corretamente copiados, relendo o novo arquivo-cópia depois (o arquivo-cópia pode ser um dispositivo), e apresentando uma mensagem se a operação de cópia foi bem-sucedida. Este parâmetro deveria ser usado sempre que se faz um *backup* importante.

Z Põe o bit de paridade em zero para entradas de caracteres ASCII. Utilize este parâmetro especialmente se estiver fazendo entradas a partir do dispositivo de correção INP.

Aqui estão exemplos de expressões PIP com parâmetros:

```
*LST: = SAMPLE TXT [NT8P60]
```

Esta expressão envia o arquivo SAMPLE.TXT para o dispositivo de listagem (LST:), com tabulações de números de linhas ('N') expandidas para cada oitava coluna de caracteres ('T8)') e "page ejects" a cada 60 linhas ('P60'). O dispositivo PRN: assume estes parâmetros; se o dispositivo de listagem tivesse sido atribuído a PRN:, o exemplo acima poderia ser reescrito:

```
*PRN: = SAMPLE.TXT ↵
```

Você pode cancelar os "pressupostos" de PRN: (parâmetros *default*) fornecendo os seus próprios parâmetros:

```
*PRN: = SAMPLE.TXT [P59] ↵
```

Esta expressão SAMPLE.TXT para o dispositivo PRN: com os parâmetros usuais *default* (isto é, NT8) *exceto* que o parâmetro P usual é modificado para 59 linhas.

Eis aqui outro exemplo:

```
*LPT: = PROG.ASM [NT8U] ↵
```

Esta expressão envia PROG.ASM para o dispositivo de listagem com números de linhas ('N'), tabulações expandidas para cada oitava coluna ('T8'), e com os caracteres em minúsculas traduzidas para MAIÚSCULAS ('U').

Copiando partes de arquivos

Inevitavelmente a pessoa irá interromper, alguma vez, uma listagem longa na sua impressora por acaso (pressionando um caractere no teclado) ou por causa de um problema de impressão (falta de papel ou outros problemas mecânicos). A pessoa gostaria de reiniciar a sua listagem no ponto em que houve a interrupção, e não listar novamente todo o arquivo. Este é o exemplo mais simples de uma transferência parcial. O PIP oferece a opção conveniente de listar e transferir partes de arquivos.

Pode-se instruir o PIP para copiar apenas *partes* de arquivos-texto especificando um 'string' de caracteres de *início* e de *fim*. (Lembre-se de que um 'string' é uma seqüência de caracteres.) Utilize o parâmetro S para especificar um 'string' de início (isto é, onde o PIP deveria iniciar a cópia) e o parâmetro Q para especificar um 'string' de fim (isto é, onde o PIP deveria parar de copiar). O PIP irá procurar automaticamente cada 'string' de caracteres.

O aprendiz deve terminar ambos os 'strings' com o caractere ↑Z (pressionando CTRL e Z, simultaneamente). Aqui está um exemplo que copia o arquivo SAMPLE.TXT do seu início até o 'string' 'Extra':

```
A> PIP ↵
*NEWSAMPL.TXT = SAMPLE.TXT[QExtra↑Z] ↵
* ↵
A>
```

Esta operação do PIP termina quando encontra o 'string' 'Extra'. Observe que 'Extra' está escrito em letras MAIÚSCULAS e minúsculas e que nós executamos o PIP como um programa, e não como um comando de uma só linha. Se tivéssemos digitado:

```
A > PIP NEWSAMPL.TXT = SAMPLE.TXT[QExtra↑Z] ↵
```

o PIP iria traduzir o 'string' 'extra' para 'EXTRA' automaticamente. SE O OPERADOR EXECUTAR O PIP COMO UM COMANDO DE UMA SÓ LINHA, ELE IRÁ SEMPRE TRADUZIR O SEU 'STRING' PARA LETRAS MAIÚSCULAS. Se executar o PIP como um programa e digitar uma expressão PIP, seu 'string' permanecerá assim como foi digitado.

Eis aqui outro exemplo do uso dos parâmetros S e Q:

```
A > PIP ↵
*EXTRA.TXT=SAMPLE.TXT [SExtra↑Z QAnother extra↑Z] ↵
* ↵
A>
```

Esta operação do PIP começa a copiar SAMPLE.TXT quando encontra o 'string' 'Extra' e *pára* de copiar quando encontra o 'string' 'Another Extra'. O arquivo EXTRA.TXT contém a parte do arquivo entre as séries 'Extra' e 'Another Extra', incluindo 'Extra', mas não 'Another Extra'.

Observem que executamos PIP como um programa para preservar os 'strings' em letras MAIÚSCULAS e minúsculas. SE OS 'STRINGS' ESTIVESSEM APENAS EM LETRAS MAIÚSCULAS NO ARQUIVO, então os 'strings' com letras MAIÚSCULAS e minúsculas não seriam encontrados pelo PIP.

APERFEIÇOAMENTOS NA VERSÃO 2.2 DO CP/M

Aperfeiçoamentos

Se a pessoa possui a versão 2.2 do CP/M, existem certas restrições que impedem a efetuação de operações de cópia que são "normais" na versão 1.4 do CP/M. Por exemplo, se estiver utilizando as *áreas do usuário* da versão 2.2, a pessoa talvez já tenha percebido que não pode criar uma cópia na área de outro usuário, ou fazer uma cópia de um arquivo que está em outra área do usuário.

Os aperfeiçoamentos no PIP lhe permitem "incorporar" os outros *aperfeiçoamentos* ao resto da versão 2.2 do CP/M e MP/M. A seção a respeito da versão 2.2 do CP/M e MP/M explica áreas do usuário e atributos dos arquivos, mas aqui temos um breve sumário:

Áreas do usuário: Qualquer disco (disquete) pode conter arquivos separados em áreas diversas do usuário (isto é, usuário 0, usuário 1, usuário 2 etc.). Naturalmente o seu disco (disquete) poderia ter apenas uma área de usuário (usuário 0) para ficar compatível com versões anteriores do CP/M, mas a versão 2.2 lhe permite separar arquivos entre áreas do usuário, e pular de uma para outra, utilizando o comando USER, para a pessoa se *prepa-*

rar para o uso do MP/M (de modo que seus discos e disquetes sejam compatíveis com futuros aperfeiçoamentos do MP/M). O MP/M é um sistema de múltiplos *usuários* e, portanto, há necessidade de se separar os arquivos dos vários usuários. Não é preciso utilizar esta característica, e isto não é recomendado, a não ser que seu sistema tenha muitos usuários simultâneos.

Atributos de arquivos: Na versão 2.2 do CP/M e do MP/M, você pode escolher colocar um indicador especial em seu arquivo, denominado um *atributo de arquivo.* Atributos de arquivo regulam o uso do arquivo e incluem: apenas leitura ("read-only"); leitura-gravação ("read-write"); sistema (system) e diretório (directory). Estes atributos são ativados com o comando STAT (descrito na seção referente à versão 2.2 do CP/M e MP/M).

Se o arquivo é "read-only" (abreviado 'R/O'), ele não pode ser *modificado* ("updated") nem *suprimido;* isto é, o sistema não pode gravar no arquivo (nem regravá-lo). Se a pessoa desejar fazê-lo, é necessário mudar, inicialmente, o atributo R/O para R/W (read-write) usando STAT.

Se um arquivo for um "arquivo sistema" (abreviado '$SYS') ele será apresentado pelo comando DIR, e a pessoa não pode ler a partir do arquivo (o que também significa que não pode copiá-lo). Caso também use o atributo R/O (isto é, $SYS e R/O) o seu arquivo fica bem protegido. Deve-se mudar o atributo $SYS, utilizando STAT, para o atributo '$DIR' (abreviação de 'diretório'). Se também usar o atributo R/O, ele permanece ativo até ser alterado para R/W. Atributos de arquivo são discutidos na seção que lida com a versão 2.2 de CP/M e MP/M (no Capítulo 2).

O PIP possui várias novas características para mudar estes atributos de arquivo, e para mudar arquivos de uma área do usuário para outra. Estas características estão sob a forma de parâmetros:

Gn Pegue o arquivo da área do usuário 'n', onde 'n' pode ser qualquer número de zero a quinze.

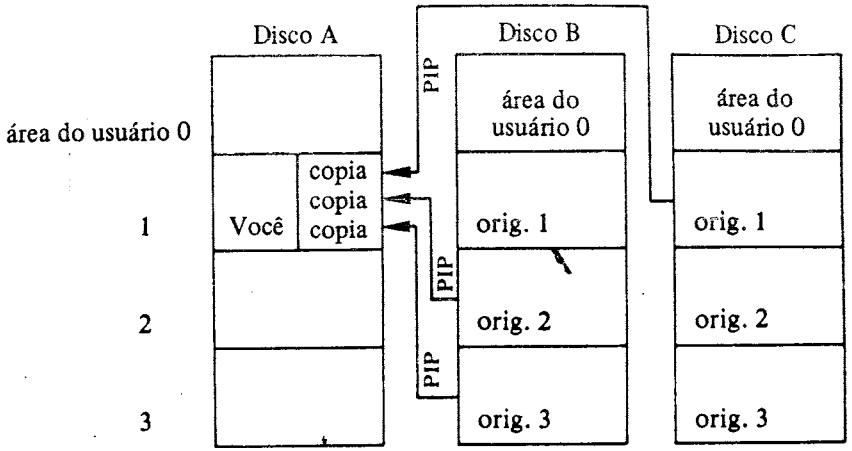
W Regrave (delete) arquivos read-only (ignore o atributo R/O).

R Leia (copie) arquivos de sistema (arquivos com o atributo $SYS). Este parâmetro também permite um W (ignore atributos R/O).

Copiando de áreas de usuário

Utilize o parâmetro G para copiar um arquivo que está na área de outro usuário (lembre-se: não se pode criar uma cópia em outra área de usuário, a não ser a sua própria). Não se necessita deste parâmetro para copiar um arquivo de um disco (disquete) para outro se o arquivo a ser copiado (e a cópia a ser criada) tiver o mesmo número de área do usuário. Por exemplo, se ORIG existir na área de usuário 2 no *drive* A, pode-se copiá-lo para um novo arquivo COPY na área do usuário 2 no *drive* B. A partir da área do usuário 2, pode-se copiar qualquer arquivo em qualquer outra área do usuário usando o parâmetro G, mas a sua nova cópia só pode ser criada na *área atual do usuário* (isto é, a área na qual a pessoa se encontra).

A pessoa altera a sua área atual do usuário com o comando USER, descrito, detalhadamente, na seção a respeito da versão 2.2 do CP/M e MP/M.



Aqui está um exemplo de uma expressão PIP usando o parâmetro G:

```
*A:=B:JIM.TXT [G3]
*
2A>
```

Esta expressão PIP copiou o arquivo JIM.TXT, no *drive* B e área do usuário 3, para o *drive* A e a área atual do usuário. A área atual do usuário é 2, como mostra o *prompt* do sistema (depois que o programa PIP foi terminado por um RETURN).

NOTA: para permanecer compatível com versões anteriores do CP/M e *ainda* ficar compatível com novos e futuros lançamentos do CP/M e MP/M, use apenas a área zero do usuário. Esta é a área *default* do usuário que ficará inalterada em novas e futuras versões e, ainda, não causará problemas com versões anteriores.

Se a pessoa usar outras áreas do usuário, é necessário que primeiro se copie PIP.COM para cada área do usuário que recebe cópias de arquivos. Uma vez que tenha uma cópia de PIP.COM em cada área do usuário de um *drive* de disco, é fácil chamar o programa PIP a partir de outro *drive,* especificando a letra do *drive.* Se o PIP.COM não existir na sua área atual de usuário em pelo menos um *drive* de disco ativo, então a pessoa não pode executar o comando PIP. Para copiar, inicialmente, o PIP para as áreas de usuários, utilize a seguinte seqüência de comandos envolvendo DDT (depurador dinâmico) e SAVE (ambos os comandos foram descritos no Capítulo 2):

```
A>USER 0                    (Especifique área zero do usuário – mude da área 2 do
                             usuário)
A>DDT PIP.COM               (Execute o DDT em PIP.COM – depurador)
                            (Mensagem DDT)
```

```
DDT VERS. xx.xx
NEXT        PC
1C80        xxxxxx
```

(O DDT apresenta o novo endereço depois do término de PIP.COM. Usa-se este endereço para calcular o número de páginas a empregar com SAVE).

```
-GO ↵
```

```
A > USER 3 ↵
```

(Mude para a área três do usuário, onde se quer colocar uma nova cópia do PIP.COM).

```
A > SAVE 28 PIP.COM
```

(SAVE cria um novo arquivo, PIP.COM, na área três do usuário no disco do *drive* A, com 28 páginas de memória – igual ao PIP.COM original. A pessoa deriva '28' do valor hexadecimal '1C' onde 'C' é igual a 12, e '1C' é igual a 28. 1C é o "byte de ordem superior" do valor hexadecimal 1C80, abaixo do *display next* do DDT).

Arquivos read-only e de sistema

O PIP não irá regravar (deletar e recriar) um arquivo que tem o atributo read-only ('R/O'). Se a pessoa tentar fazê-lo, o programa PIP responde com uma pergunta:

```
A > PIP B: COPY = ORIG ↵
DESTINATION FILE IS R/O, DELETE (Y/N)? Y
A >
```

Neste exemplo, tentamos fazer uma cópia de ORIG chamada COPY – mas COPY (o antigo COPY) já existe no *drive* B com o atributo R/O (se ele não tivesse tal atributo, poderia ter sido deletado por PIP e substituído pelo novo COPY). O PIP mostra a mensagem de que COPY (arquivo-destino) é read-only, e pergunta se queremos deletar o antigo COPY ('Y/N' representa "sim" ou "não"). Respondemos 'Y' para deletar o antigo COPY e substituí-lo pelo novo COPY (quando se responde 'Y' ou 'N', não há necessidade de pressionar a tecla RETURN ↵).

NOTA: O novo COPY *não* terá, automaticamente, o atributo R/O.

Se tivéssemos respondido 'N' ("não"), o antigo COPY *não* seria deletado e o PIP teria mostrado a mensagem:

```
**  NOT DELETED  **
```

Se a pessoa quiser *anular* esta ação do PIP, de mostrar uma mensagem se o arquivo é R/O e pedir que verifique, poderá usar o parâmetro W. O parâmetro W diz ao PIP que ignore o atributo R/O. Pode-se usar o parâmetro W no fim de uma expressão PIP se a pessoa quiser que ele se aplique a todos os arquivos em uma concatenação de arquivos:

```
A > PIP WHOLE.TXT = PART1.TXT,PART2.TXT,PART3.TXT[W] ↵
A >
```

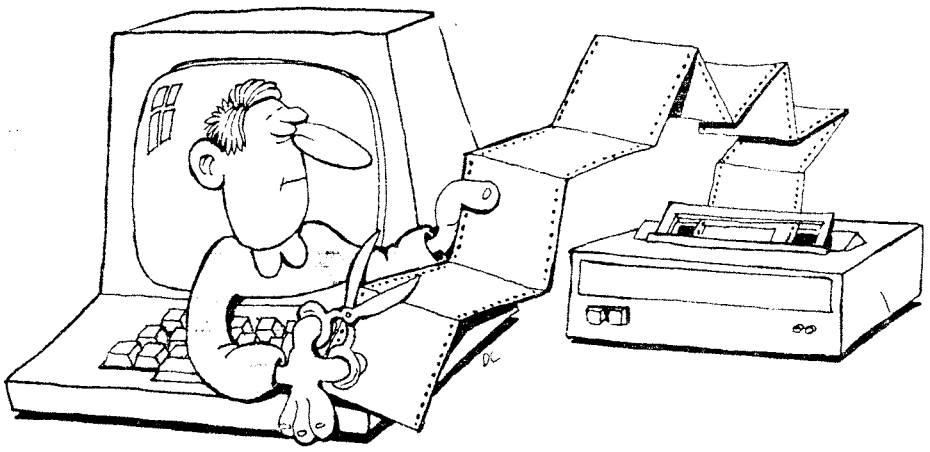
Caso o arquivo original ou sua cópia possuam o atributo $SYS (sistema) então o PIP não pode achar os arquivos no diretório do disco. Pode-se usar o parâmetro R para ignorar os atributos $SYS (e R/O) de modo que o PIP possa achar o arquivo-original ou criar o arquivo-cópia. Usa-se o atributo R da mesma maneira em que o atributo W foi usado no exemplo anterior.

RESUMO

O PIP é uma poderosa facilidade de transferência de arquivos. Embora a maioria dos usuários do CP/M só o utilize para simples transferências disco a disco, ele pode fazer muito mais:

- Transferências entre dispositivos e discos.
- Transferências de arquivos múltiplos.
- Processamento, conferência e formatação de arquivos.

O PIP pode ser utilizado, vantajosamente, para obter partes de um arquivo ou juntar vários arquivos. Também pode ser utilizado para obter impressões limpas, paginadas e tabuladas. Um conhecimento prático dos parâmetros relevantes é *uma vantagem definitiva.* Portanto, cada usuário do CP/M deve ler este capítulo, por completo, uma vez e depois reler os detalhes das opções que irá utilizar com maior freqüência.



Capítulo 4

Utilizando o Editor

INTRODUÇÃO

Os Capítulos 1 e 2 foram escritos para ensinar tudo o que o aprendiz precisa saber para começar a usar o CP/M. O Capítulo 3 descreve o utilitário mais importante: o PIP. Este capítulo vai ensinar-lhe a usar outro programa de grande importância, que vem com o sistema operacional CP/M: o editor, ED. Mostraremos como usar um programa editor e o aprendiz seguirá as transferências de dados entre o disco, a memória do computador e o terminal.

Não é importante decorar os comandos específicos oferecidos pelo ED, pois estão resumidos no apêndice. O que é importante, todavia, é que se compreenda como o editor opera, e o que ele pode fazer. Se o aprendiz atingir esse objetivo, descobrirá que a maioria dos outros programas de aplicação são simples de compreender (se forem bem elaborados e bem documentados). Certamente o aprendiz também será capaz de usar, facilmente, um programa de processamento de palavras, outro programa de aplicação muito útil em qualquer computador.

Este capítulo é útil, mas não indispensável. Se o aprendiz sentir que não está interessado em aprender algo a respeito do editor, pode, então, passar para o próximo capítulo.

O QUE É UM PROGRAMA-EDITOR?

Um bom programa-editor permite-lhe criar e editar arquivos-texto: cartas, novelas, poemas, formulários comerciais, ou qualquer outra coisa constituída de caracteres. O programa também deve deixar que o aprendiz se mova de linha para linha com facilidade, e alterar caracteres, redigitando-os ou deletando e inserindo caracteres de uma só vez. Deve ser capaz de descobrir qualquer grupo de caracteres que especifica em um arquivo e efetuar substituições. Um bom programa-editor deve lhe permitir agrupar dois arquivos, além de trocar linhas de texto desses dois arquivos.

Quando se digita um texto usando um bom programa editor, a pessoa irá digitar uma linha e terminá-la com um "carriage return" (RETURN ou CR em alguns teclados), assim como o operador faria em uma máquina de escrever elétrica. No futuro, um programa-editor poderá tornar a edição ainda mais fácil; certamente ainda não temos o melhor programa editor possível nem o mais fácil de usar.

ED, o programa-editor fornecido com o CP/M, é apenas um editor simples e não é tão fácil de usar como a maioria dos outros programas-editores. Se o aprendiz planeja realizar uma quantidade significativa de edição ou processamento de palavras, deve obter um edi-

tor mais poderoso. Existe uma variedade de programas-editores ou de "processamento de palavras" disponíveis e que podem rodar no CP/M ou MP/M.

Um *processador de palavras* é um programa que inclui tanto um programa-editor (para digitar textos) como um programa que aciona a impressora (para imprimir textos), fazendo-a retroceder, sublinhar, alinhar margens, expandir tabulações e imprimir em negrito. Observe, porém, que a pessoa deve saber ao certo o que está comprando. Existem alguns "processadores de palavras" que são apenas programas de impressão a serem usados com o programa ED do CP/M. Eles são apenas formatadores, não sendo suficientemente práticos ou poderosos para uso genérico como "processadores de palavras".

Procure comprá-los como se estivesse comprando uma máquina de escrever. A pessoa pode desejar um modelo altamente sofisticado, com tecla corretora e graduação para o número de cópias, ou então pode querer um modelo portátil, de baixo custo, que exige mais esforço de sua parte, mas dá conta do recado. Contudo, a facilidade de utilização *deve ser* a primeira preocupação (especialmente se a pessoa for usá-lo para escrever). Depois de ler este capítulo, o leitor conhecerá o conjunto mínimo de facilidades a serem oferecidas por um editor. O ED é suficiente para a maioria das aplicações simples.

O EDITOR ED

O programa-editor ED.COM reside em geral no disquete do sistema, e é executado digitando 'ED', seguido do nome do arquivo-texto que o aprendiz está criando ou modificando. Por exemplo, se desejar criar ou modificar o arquivo SAMPLE.TXT, digite:

A > ED SAMPLE.TXT ⏎

Não digite apenas 'ED'. Este erro é comum, e isto não funciona. É necessário fornecer um nome de arquivo.

NOTA: Caso obtenha a mensagem 'FILE IS READ/ONLY' ou 'SYSTEM FILE NOT ACCESSIBLE', então deve usar o comando STAT no arquivo em primeiro lugar (versão 2.2 do CP/M e MP/M). Consulte a seção sobre a versão 2.2 do CP/M e MP/M no Capítulo 2.

Se o ED não conseguir achar o arquivo especificado, ele presume que está sendo criado um novo arquivo. Esse arquivo, especificado pelo seu comando, se torna o arquivo-fonte. Os comandos ED subseqüentes irão, então, colocar o texto do arquivo-fonte na *memória intermediária do editor*, como mostramos na Figura 4.1.

O "*buffer*" é um bloco de memória dentro do computador, reservado para o processamento de textos pelo ED. Se o seu arquivo-texto for longo, a pessoa só poderá carregar um bloco de cada vez no "buffer" (observe que esta é uma desvantagem inerente ao ED que não existe em editores mais poderosos).

O operador só pode digitar um novo texto para ser colocado na memória intermediária do editor, ou alterar um texto que esteja lá. Contudo, o "buffer" não é copiado de volta para o disco automaticamente. Se a pessoa terminar o ED (ou se desligar o sistema) sem *salvar* o texto que está no "buffer" em um arquivo de disco, perderá o texto que está na memória intermediária. Já que o ED *copia* o arquivo-texto para a memória intermediária do editor, o seu arquivo original (isto é, antes de ter sido usado pelo ED) não foi tocado, mas o seu novo texto e as modificações foram perdidas. Portanto, a pessoa deve, periodicamente, salvar o texto que está na memória intermediária do editor, e sempre lembrar-se de salvar o "buffer" antes de sair do programa ED. Salva-se o "buffer" utilizando o comando 'E' do ED (E ⏎). O comando 'E' também copia o resto do arquivo-fonte para um

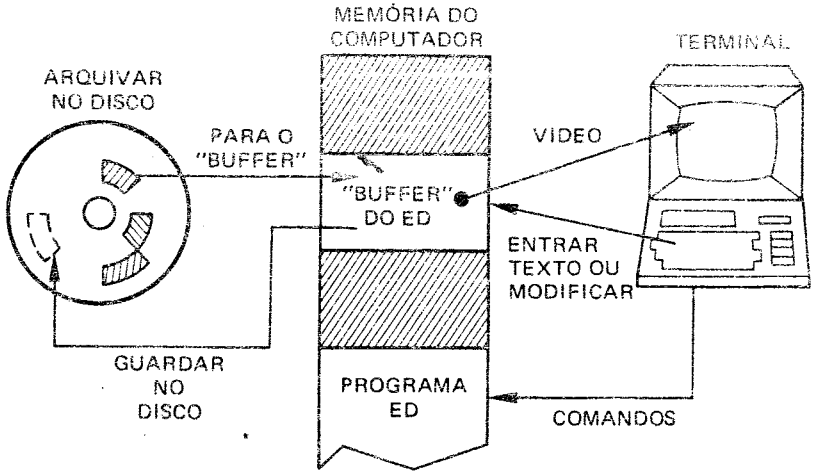


Figura 4.1: *O "buffer" do ED*

novo arquivo-fonte, que é criado. (Isto será explicado com maiores detalhes adiante neste capítulo.)

A maioria dos comandos ED consiste de uma letra precedida por um número ou símbolo determinando uma quantidade. Esses comandos são executados por meio de digitação, assim como ocorre com os comandos CP/M: digita-se o comando, seguido de um RETURN (⏎), que o transmite. Outros "comandos" são combinados especiais de teclas (como CTRL e Z (↑Z), CTRL e C (↑C) etc.) que são transmitidos automaticamente e não exigem um RETURN.

O programa ED apresenta, continuamente, o *prompt* ED:

O comando 'E' seria digitado como segue:

*E ⏎

Vários comandos atuam sobre o texto na memória intermediária do editor, ao passo que outros transferem o texto do "buffer" e para o "buffer" do editor. A Figura 4-2 ilustra o "buffer" do editor.

O 'CP' (INDICADOR DE CARACTERES) E NÚMEROS DE LINHAS

O 'CP' na ilustração é o indicador de caracteres. Não é apresentado na tela, mas usado pelo programa ED, e movimentado por comandos ED. O indicador sempre indica algum caractere, e os comandos ED geralmente se referem aos caracteres que seguem (incluem) o caractere para o qual o CP aponta, ou os caracteres anteriores. Em outras palavras, a pessoa move o 'CP' para a *direita* para avançar no texto, e o move para a *esquerda* para retroceder. Seções subseqüentes deste capítulo ilustrarão exemplos.

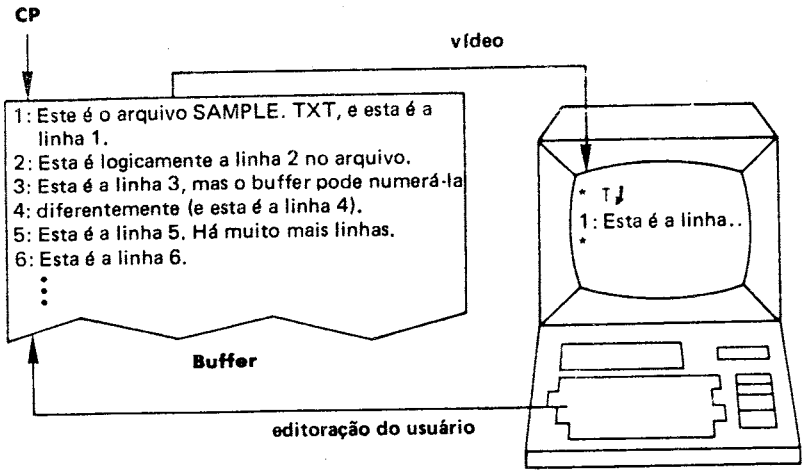


Figura 4.2: *Processamento de texto*

Além do 'CP' imaginário, cada linha tem um *número de linha* imaginário que de fato não faz parte do texto. Se a pessoa possui a versão mais nova do ED, ele automaticamente apresenta os números das linhas com o texto (e a pessoa pode deslocar-se para qualquer linha do texto, especificando o número da linha como se fosse um comando, como mostraremos mais adiante). Se a pessoa possui a versão antiga do ED, é indispensável que *ligue* o *display* dos números de linhas, executando o comando V (O comando-V – V negativo – o desliga). Se a pessoa possuir uma versão ainda mais *antiga*, talvez não tenha números de linhas (o que não é bom, porque elas facilitam o deslocamento no "buffer" do editor). Os números de linhas, como o 'CP', só existem no "buffer" do editor e são utilizados apenas para se deslocar nele. Não aparecerão numa impressão do arquivo.

O QUE ED FAZ COM SEU ARQUIVO-TEXTO

Por exemplo, suponhamos que o arquivo SAMPLE.TXT já existe, e que contenha o texto apresentado na Figura 4.3.

Quando se executa o comando CP/M:

```
A> ED SAMPLE.TXT ↵
```

O ED reserva um espaço na *área transiente de programa* ("scratch-pad memory" – isto será explicado no Capítulo 5) no "buffer" do editor, cria um arquivo de saída temporário chamado SAMPLE.$$$ (para armazenar o arquivo editado sem danificar o original), e prepara o SAMPLE.TXT para ser copiado no "buffer" do editor. A cópia é feita utilizando o comando A ("append") – isto irá acrescentar um certo número de linhas ao "buffer" do editor, como mostra a Figura 4.4.

O comando ED '2A' acrescentou as primeiras duas linhas do arquivo-texto ao "buffer".

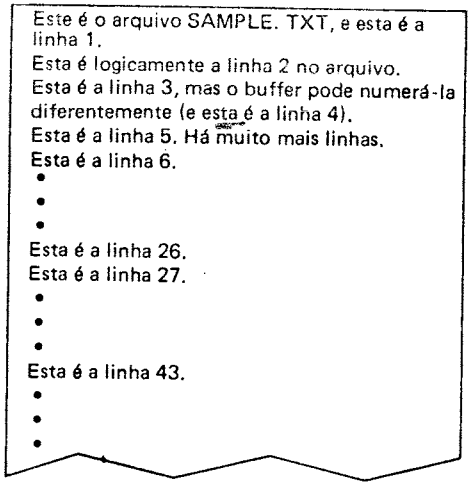


Figura 4.3: *Uma amostra de um arquivo que está no "buffer"*

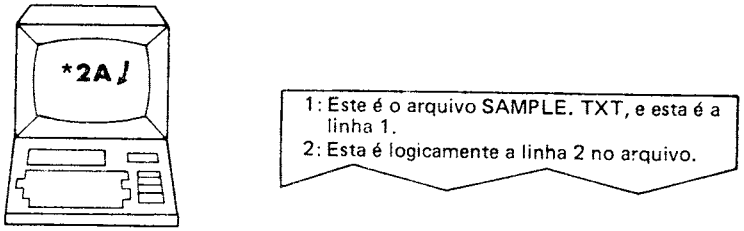


Figura 4.4: *O comando "Append" está em A*

Se a pessoa desejar acrescentar mais linhas, é necessário executar outro comando A, como mostramos na Figura 4.5.

Agora é possível modificar estas linhas que estão na memória intermediária do editor. Pode-se também acrescentar novas linhas a partir do teclado (usando o comando I, descrito mais adiante neste capítulo), como mostramos na Figura 4.6.

O leitor agora aprendeu a colocar linhas no "buffer" a partir do arquivo que está no disco ou a partir do teclado. Consulte a Figura 4.7.

Poder-se-ia ter acrescentado todo o arquivo ao "buffer" (a não ser que o arquivo tivesse mais de 6.000 caracteres e o leitor estiver usando uma versão de 16K do CP/M), mas, por motivos didáticos, este exemplo envia o texto modificado (com seus novos acréscimos) a um *arquivo temporário de saída*, para que a pessoa possa acrescentar mais texto ao "buffer" do editor. O ED denomina este arquivo temporário de saída de SAMPLE.$$$. Para enviar o texto recém-modificado para o arquivo temporário de saída, usa-se o comando W (grave), ou então executa-se um término "normal" com o comando E (termine a edição) ou, ainda, o comando H (termine e reabra). A Figura 4.8 demonstra o uso comando W.

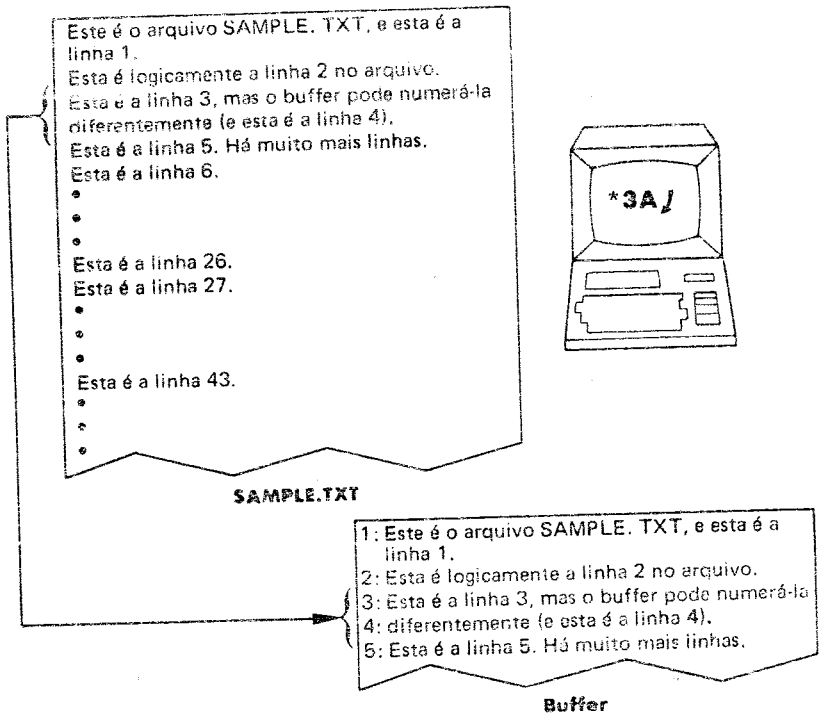


Figura 4.5: *Acrescentando mais três linhas*

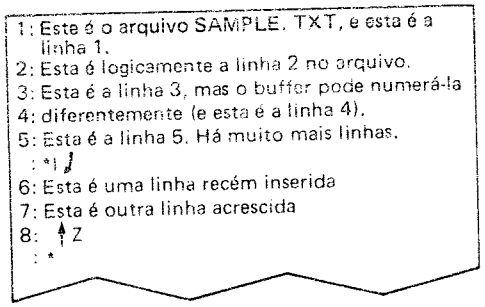


Figura 4.6: *Acrescentando duas linhas novas a partir do teclado*

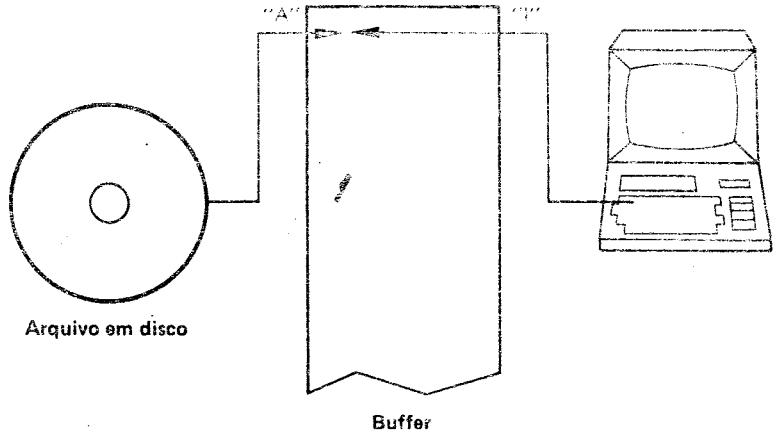


Figura 4.7: *Colocando linhas no "buffer"*

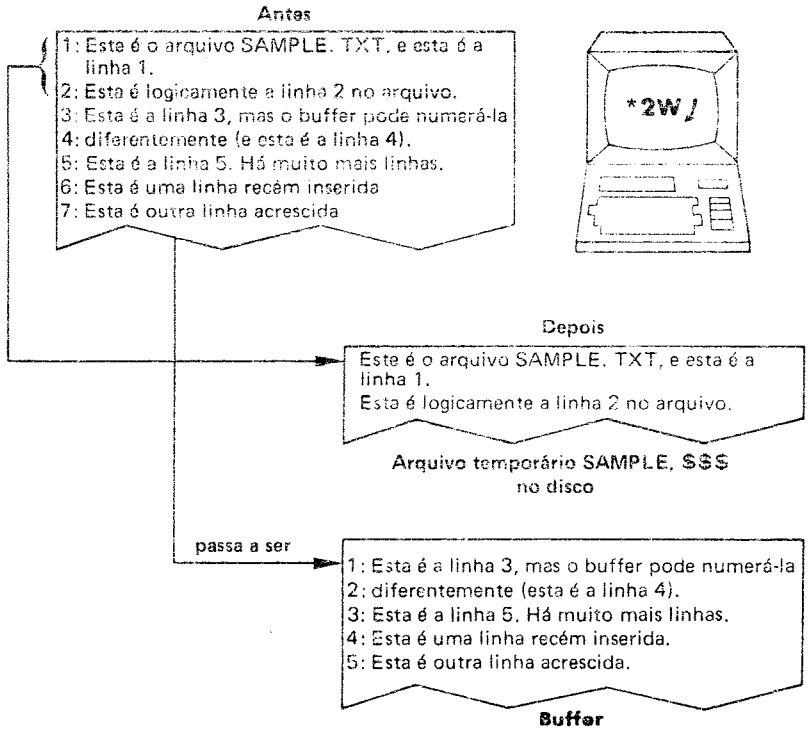


Figura 4.8: *Reservando o "buffer" no disco*

O exemplo da Figura 4.8 mostra uma operação de gravação de duas linhas. As linhas finais de edição no "buffer" se deslocam para o começo, liberando mais espaço para acrescentar outras linhas. O próximo exemplo mostra uma operação em que são acrescentadas mais vinte linhas.

Depois de modificar o texto no "buffer", pode-se terminar a sessão ED (e gravar o resto do "buffer" no arquivo de saída) utilizando o comando E, como mostramos no exemplo da Figura 4.9. O comando E também copia o *resto* do arquivo-fonte – as linhas que não foram acrescentadas ao "buffer" – para o arquivo temporário de saída. Observe que o arquivo temporário de saída contém as linhas do texto em sua ordem adequada.

Tão logo o "buffer" de edição é adequadamente copiado para o arquivo temporário de saída, que se denomina SAMPLE.$$$, o ED faz duas coisas: altera o nome original, SAMPLE.TXT para SAMPLE.BAK, e altera o nome de SAMPLE.$$$ para SAMPLE.TXT. Ao fazer isto, ED cria um backup do original SAMPLE.TXT (denominada SAMPLE.

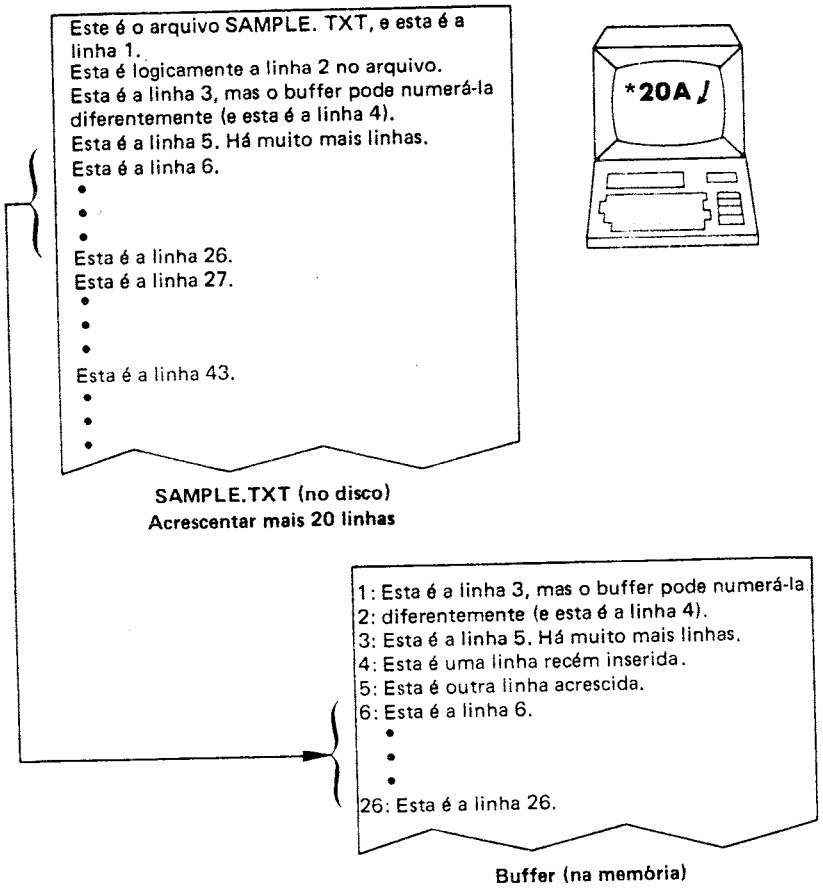


Figura 4.9: *Acrescentando vinte linhas ao "buffer"*

BAK) e altera o nome do recém-modificado SAMPLE.$$$ para SAMPLE.TXT. (Veja a Figura 4.10.) É por esta razão que ficamos com a impressão de que o ED modifica o ar-

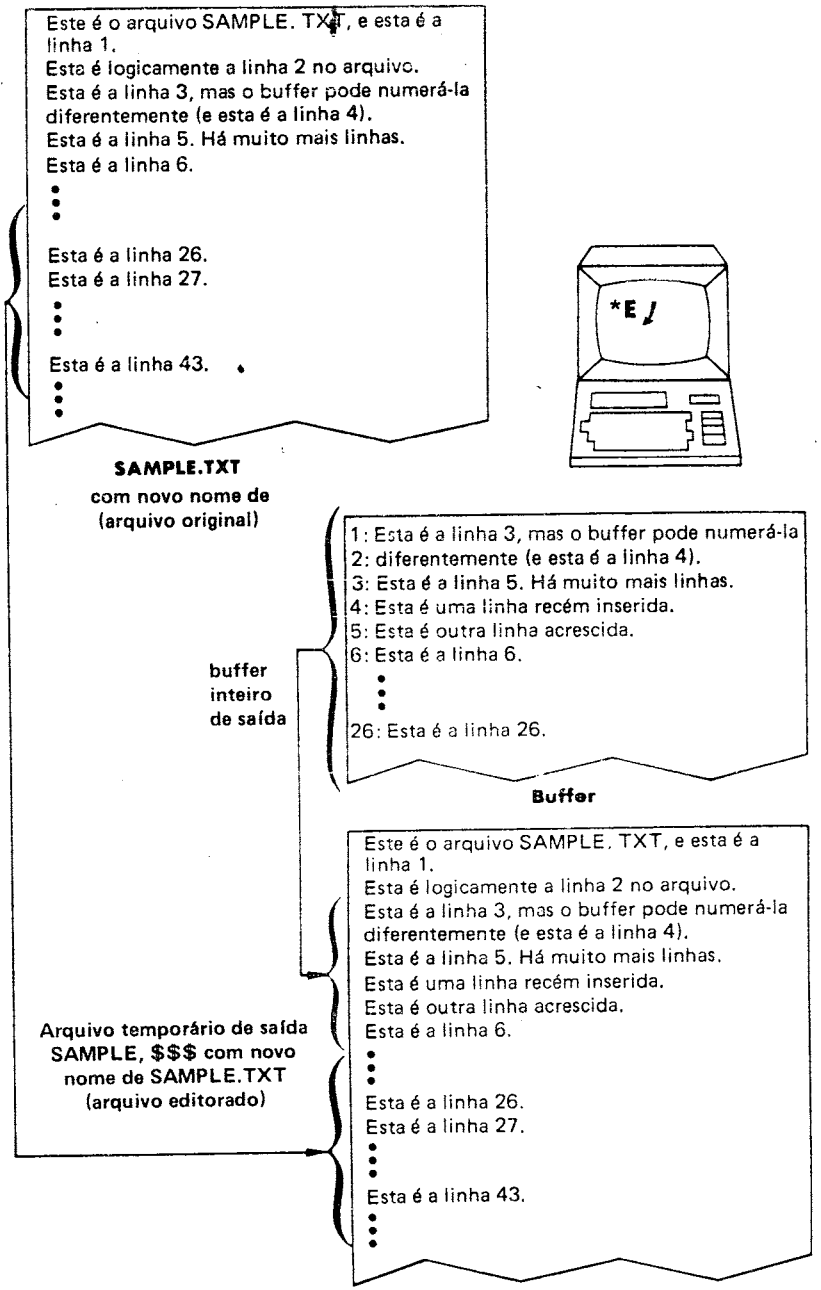


Figura 4.10: *Terminando uma sessão de edição*

quivo SAMPLE.TXT – na realidade, o ED modifica o texto no "buffer" e utiliza SAMPLE.TXT como cópia (backup) e SAMPLE.$$$ como uma versão futura do SAMPLE.TXT.

NOTA: Quando se executa o ED em um arquivo, o ED automaticamente deleta qualquer arquivo 'BAK' associado com o arquivo-texto (como medida de preparação para o novo arquivo 'BAK' que o ED cria). Os comandos 'O' e 'Q' não prevêem esta ação; portanto, seja cuidadoso.

GERENCIAMENTO DE ARQUIVOS

Quando você dá ao ED um nome de arquivo, ele o procura; se o ED não consegue achá-lo, ele o cria. Esse arquivo recebe o nome de arquivo-fonte na documentação do CP/M. Quando a pessoa acrescenta textos ao "buffer" a partir do arquivo-fonte (utilizando o comando A), o ED copia as linhas do texto a partir do topo do arquivo, para o "buffer", contando o número de linhas especificadas no comando A. Ao gravar linhas no arquivo temporário de saída criado pelo ED (*filename*.$$$), a pessoa libera espaço no "buffer" para poder acrescentar mais linhas de texto do arquivo-fonte. À medida que se grava no arquivo de saída, as linhas são acrescentadas para permanecerem na mesma ordem (pode-se escrever linhas específicas para o arquivo de saída com o objetivo de alterar essa ordem). Quando a pessoa termina de editar, ou termina o ED utilizando o comando E (ou H), o ED automaticamente rebatiza o arquivo original (arquivo-fonte) para um arquivo com extensão BAK (por exemplo, SAMPLE.BAK) para caracterizá-lo como arquivo de backup, e D rebatiza o arquivo temporário de saída (por exemplo, SAMPLE.$$$) com o nome do seu arquivo original (fonte); por exemplo, SAMPLE.TXT, de modo que o seu arquivo-texto contenha o texto recém-alterado.

INTERRUPÇÃO ACIDENTAL

Se for interrompido o ED acidentalmente, sem se usar o comando E (ou H): isto é, se houver um erro de sistema, ou a pessoa inadvertidamente teclar ↑C para fazer um 'reboot' do sistema, ou a força for desligada, o arquivo temporário de saída (por exemplo, SAMPLE.$$$) continuará com esse nome de arquivo, e o seu arquivo-fonte original (por exemplo, SAMPLE.TXT) continuará a ser a cópia antiga. O arquivo $$$ conteria apenas o texto que já tivesse sido gravado (isto pode não ser útil), perder-se-ia o "buffer" de edição, e o arquivo original seria a versão não editada (isto é, antes de se chamar o ED).

NOTA: Use o comando Q (quit – interromper) para fazer isto propositadamente.

Pode-se agrupar linhas de texto de outro arquivo-texto com o texto que já está no "buffer" de edição, usando-se o comando R, que será discutido mais adiante neste capítulo (o outro permanece sem alteração).

UMA SESSÃO COM O EDITOR

Para criar um arquivo de texto novo, pense primeiro em um nome (um que ainda não tenha sido usado com outra extensão). Usaremos QUOTE.TXT como exemplo. Execute o comando ED e crie QUOTE.TXT digitando:

```
A > ED QUOTE.TXT ↵
```

Neste exemplo, estamos no *drive* A; portanto, QUOTE.TXT estará nesse *drive*. Estamos no *drive* A porque o programa ED (ED.COM) está nele. O exemplo do Capítulo 1 mostrou como se pode executar o ED a partir do outro *drive*, colocando a letra do *drive* e dois pontos antes do nome do arquivo ED. Também se pode executar o ED a partir do *drive* A e colocar QUOTE.TXT no *drive* B, inserindo 'B': antes de 'QUOTE.TXT'.

O ED irá apresentar a mensagem 'NEWFILE' se estiver criando um arquivo novo. Apresentará o *prompt* do ED '*' quando estiver pronto para receber outro comando. Agora pode-se digitar qualquer comando do ED. Para inserir um novo texto a partir do teclado, deve-se usar o comando I. O comando I começa a inserir o texto imediatamente depois do CP (indicador de caracteres). Como não foi colocado explicitamente o CP (com um comando ED), ele está no início do "buffer". Para começar a inserir um novo texto, digite:

```
*I ↵
```

Para inserir o texto, a pessoa digita-o como faria em uma máquina de escrever. Para parar de inserir texto, pressiona-se simultaneamente as teclas CTRL e Z (↑Z).

Quando o operador digita 'I ↵', o ED faz o cursor (o indicador que acende e apaga na tela, ou qualquer outro símbolo usado pelo seu terminal) se movimentar para a próxima linha em branco. Agora pode-se digitar o texto que será automaticamente inserido no "buffer" de edição à medida que se transmite cada linha. *Cada linha deve terminar com um* RETURN (como na máquina de escrever) – RETURN (representado por um ↵ neste livro) transmite a linha para o "buffer" do editor. A linha se torna, então, uma *linha no "buffer"* (identificada por um número de linha). Desejando digitar uma linha no "buffer" que é, na realidade, maior que o limite que a pessoa pode digitar no terminal; utilize a combinação de CTRL e E (↑E) para movimentar o cursor para próxima linha em sua tela sem transmitir a linha para o "buffer" do editor. Quando o operador finalmente aperta RETURN, a linha inteira é transmitida para o "buffer". As linhas maiores não podem exceder a 128 caracteres.

Começaremos o exemplo, de novo, a partir do início. Observe que se pode utilizar as teclas de edição, do teclado, RUBOUT (DELETE), ↑U (delete a linha), ↑E (retorne o cursor sem transmitir a linha), e ↑R (reescreva a última linha) tanto nas versões antigas como novas do ED. Só se pode usar ↑H (retrocesso e deleção de um caractere) e ↑X (volte ao início da linha) na nova versão do ED.

Aqui temos de novo o exemplo, a partir do início:

```
A > ED QUOTE.TEXT ↵
NEW FILE
*I ↵
"We have not ceased from exploration"
wrote T.S. Eliot.
↑Z
*
```

Utilizamos a combinação CTRL e Z para parar de inserir o texto (↑Z). Agora temos duas linhas de texto – a primeira é a citação, e a segunda é 'wrote T.S. Eliot'.

Se desejar, a pessoa pode utilizar o comando U antes do comando I, e tudo que for digitado será convertido, automaticamente, para letras MAIÚSCULAS (mesmo que, ao digitar o texto, ele apareça em letras MAIÚSCULAS e minúsculas). Para desligar essa tradução automática para MAIÚSCULAS, use o comando – U (U negativo).

Agora que temos o texto no "buffer", podemos apresentá-lo. Em primeiro lugar, é necessário mover o 'CP' (indicador de caracteres) de volta para o início do "buffer", já que o comando I inseriu um texto e moveu o 'CP'. Aqui temos uma ilustração mostrando o texto e a posição do 'CP' no "buffer":

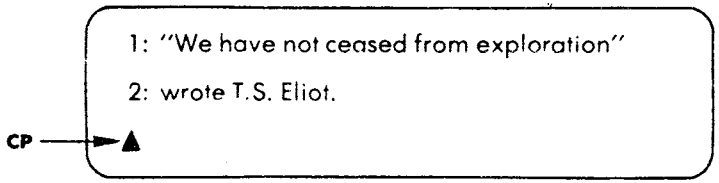


Figura 4.11: *O CP está no fim do "buffer"*

O manual do ED explica que o 'CP' está *entre* dois caracteres. Isto é difícil de visualizar e usar, de modo que alteramos nossa descrição para facilitá-lo. O 'CP' é um objeto "imaginário", um ponteiro de referência a ser usado com comandos ED. Ao consultar a documentação da Digital Research, lembre-se de que suas descrições se referem ao fato de o 'CP' estar entre dois caracteres, e a nossa se refere ao fato de que o 'CP' está apontando para o último caractere à direita. A ilustração da Figura 4.12 esclarece isto:

Figura 4.12: *Mostrando a posição do 'CP'*

A Digital Research diz que o 'CP' está *antes* do primeiro caractere no "buffer" quando movido para o início, e dizemos que o 'CP' aponta *para* o primeiro caractere do "buffer" quando movido para o início.

MOSTRANDO O TEXTO QUE ESTÁ NO "BUFFER"

Utilizaremos o comando T para mostrar o texto que está no "buffer". O formato do comando T é:

± nT

onde n pode ser zero, ou qualquer número, ou o sinal (#) (Figura 4.13). Se n for zero, T irá apresentar a linha atual até (sem incluí-la) o 'CP' (a linha atual é a que contém o 'CP'). Se a pessoa não especificar n admite-se que ele seja igual a 1. Se n for igual a 1 (ou assume-se que ele seja 1) a linha atual é apresentada a partir do 'CP' até o fim da linha. Se n for um número positivo, T irá apresentar n número de linhas a partir do 'CP' (linha atual); se n for um número negativo, T irá apresentar as linhas anteriores ao 'CP' (sem incluí-lo). Se a pessoa usar o sinal (#), T irá substituir o sinal por '65535' (número máximo de linhas permitidas no "buffer"), isto é, será apresentado o "buffer", por exemplo. Pode-se digitar o "buffer" inteiro movendo o 'CP' para o início do mesmo (usando o comando B), como mostramos na Figura 4.14) e utilizando o sinal (#) com T; de outra forma, o sinal (#) irá apresentar todas as linhas que *se seguem* ao 'CP' (incluindo-o), e um sinal negativo (– #) irá apresentar todas as linhas anteriores ao 'CP' (*sem* incluí-lo).

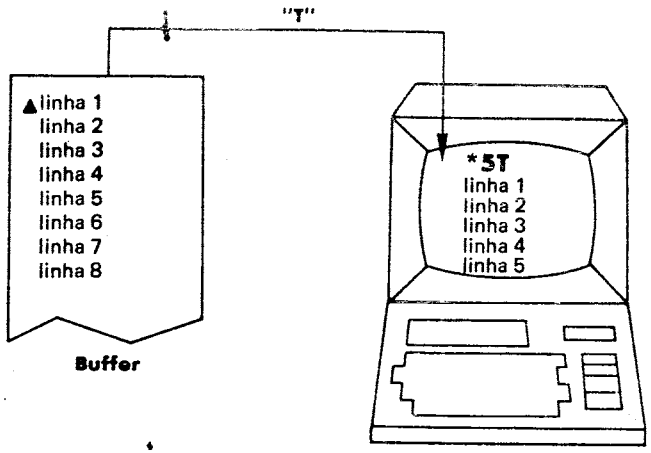


Figura 4.13: *Mostrando o texto*

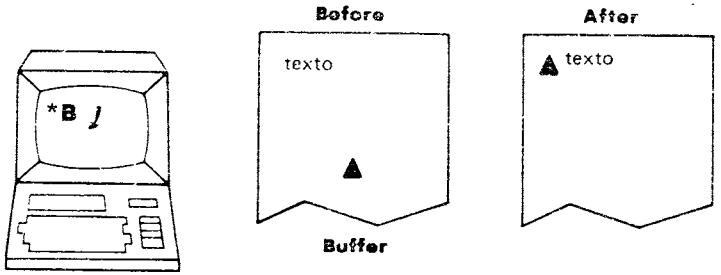


Figura 4.14: *Movendo o cursor*

Aqui temos um exemplo:

```
*-2T
1: "We have not ceased from exploration"
2: wrote T.S. Eliot.
: *
```

Se o seu *display* não incluir o número das linhas (números seguidos de dois pontos), tente executar o comando V (versão 1.4 do CP/M ou outra posterior):

```
*V↵
: *
```

O ':' lhe diz que o 'CP' está no início da linha3; contudo, essa linha ainda não tem texto, de modo que o 'CP' está, na realidade, apontando para o fim da linha 2 (depois da seqüência 'Carriage Return').

O comando −2T diz ao ED que apresente apenas as duas linhas anteriores àquela que contém o 'CP' (isto é, a linha atual, a número 3).

Nas versões 1.4 do CP/M ou outras posteriores, pode-se especificar o número da linha atual em um comando T para apresentar essa linha. Por exemplo, se a pessoa quisesse apenas apresentar a linha número 1, digitaria '1:T' como comando:

```
2: *1:T↵
1:  "We have not ceased from exploration"
1: *
```

Note que o comando '1:', digitado com 'T' moveu o 'CP' para a linha 1 (a linha atual agora é a linha 1). Pode-se especificar um número de linha como um comando para movimentar o 'CP' para essa linha.

Por exemplo:

```
1: *2:↵
2: *
```

Um simples comando T irá apresentar a linha atual:

```
2: *T↵
2: wrote T.S. Eliot.
2: *
```

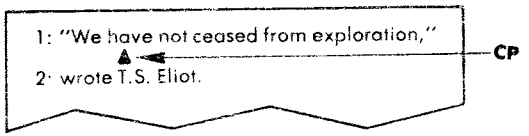
Uma maneira fácil de mostrar o "buffer" completo é executar dois comandos: o comando B para movimentar o 'CP' para o início do "buffer" e o comando T com o sinal (#) para mostrar o "buffer" por completo. Observe que se pode colocar os dois comandos na mesma linha e executá-los:

```
1: *B#T↵
1:  "We have not ceased from exploration"
2: wrote T.S. Eliot.
1: *
```

Nos exemplos acima, o 'CP' aponta para o primeiro caractere da linha. É possível movimentar o 'CP' na linha atual usando o comando C:

± nC

C irá mover o 'CP' +n caracteres para a frente ou −n caracteres para trás. Por exemplo, se o 'CP' estiver no início da linha 1, e a pessoa digitar esse comando:




```
1: *5C↵
1: *
```

O 'CP' estará apontando para o sexto caractere ('a', já que uma frase é contada como se fosse um caractere) na linha 1. O comando 'T' mostraria a linha desde o 'CP' até o fim (incluindo o 'CP'):

```
1: *T↵
ave not ceased from exploration"
1: *
```

O comando 'OT' mostraria a linha um desde o início até o 'CP', porém *sem* incluí-lo:

```
1: *0T↵
"We h
1: *
```

SALVANDO O ARQUIVO E TERMINANDO A SEÇÃO ED

Agora, deve-se aprender a salvar o conteúdo do "buffer" e sair do ED. A maneira mais simples de salvar o "buffer" e sair do ED ao mesmo tempo é utilizar o comando E:

```
1: *E
```

Sempre que o operador estiver no "buffer", esse comando irá esvaziá-lo (e também o resto do arquivo-fonte, se o operador não tiver acrescentado o arquivo-fonte por completo) num arquivo temporário de saída e irá redenominar o arquivo de saída com o nome de seu arquivo-fonte (original). Se o operador começou com um QUOTE.TXT vazio e acrescentou o texto, como mostramos nos exemplos precedentes, deve terminar com um QUOTE.TXT contendo uma linha de T.S. Eliot, e um QUOTE.BAK vazio (uma backup do arquivo antes de tê-lo editado).

ACRESCENTANDO LINHAS AO "BUFFER" (EDIÇÃO DE UM ARQUIVO JÁ EXISTENTE)

Quando se tem um arquivo-texto já existente e a pessoa executa o ED para editá-lo, o comando A trará as linhas do texto para o "buffer" do editor. De outra forma, não haverá nenhum texto no "buffer" exceto o texto que a pessoa insere utilizando o comando I. Pode-se querer inserir um texto novo *antes* da primeira linha do seu arquivo-texto existente – utilizando o comando I para inserir o texto novo antes de acrescentar o texto do seu arquivo existente ao "buffer" com o comando A; todavia, o aprendiz poderá fazê-lo facilmente *depois* que o texto antigo estiver no "buffer". Portanto, o aprendiz usará o comando A, em primeiro lugar, para ver o texto antigo. O comando A acrescenta ("appends") linhas ao final do "buffer".

O formato do comando A é:

```
nA
```

O comando A irá acrescentar n linhas do texto a partir do arquivo-texto fonte (original). Se não for especificado n, A acrescenta apenas uma linha. Se for usado um sinal (#) ao invés de n, A irá trazer todo o arquivo-fonte (até 65.535 linhas) para o "buffer" do editor. Já que a maioria dos arquivos não é tão grande, pode-se utilizar a forma '# A' para dar entrada a qualquer arquivo que provavelmente caberá no "buffer" do editor. Especificando um zero para n, o comando A irá acrescentar no sentido de preencher *metade* do "buffer" (isto é útil para arquivos longos). Utilize a forma 'OA' junto com 'OW' para acrescentar e gravar metade do "buffer" (discutido neste capítulo na seção "Gravando linhas no arquivo").

Se a pessoa acrescentar apenas uma parte de seu arquivo-fonte (original) ao "buffer", o seu *próximo* comando A irá começar a acrescentar a partir do arquivo-fonte no ponto em que o último A parou. Poder-se-á, facilmente, acrescentar dez linhas, alterar alguma parte do texto, enviar essas dez linhas para o arquivo temporário de saída (com o comando W a ser discutido mais adiante) e acrescentar mais linhas do arquivo-fonte. Ou, se tiver espaço suficiente no "buffer", a pessoa poderá acrescentar mais linhas do arquivo-fonte ao fim das linhas que já estão no "buffer".

Eis um exemplo simples, utilizando QUOTE.TXT como arquivo-fonte:

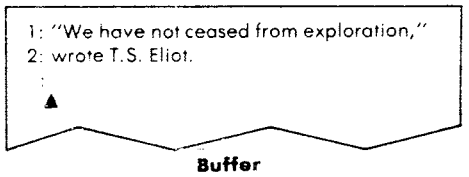
```
A>ED QUOTE.TXT
  *V  (if using version 1.4, not needed in version 2.2)
 :*A  (appends only one line)
1:*2A  (appends next two lines, but there is only one
        more line in the file anyway)
2:*B#T  (go to beginning and display all lines)
1:"We have not ceased from exploration"
2:wrote T.S. Eliot.
1:*
```

Como só existem duas linhas em QUOTE.TXT, um comando '#A' simples será o suficiente para acrescentar todas as linhas.

MOVIMENTAÇÃO DENTRO DO "BUFFER"

Com o texto no "buffer", a pessoa tem liberdade para se movimentar como desejar e alterar qualquer texto, assim como inserir um novo texto em qualquer lugar. Pode-se também localizar e substituir partes do texto, e acrescentar outras linhas de texto a partir de um arquivo especial, a *biblioteca-fonte* (a ser discutida mais adiante).

Se a pessoa estiver seguindo os exemplos e editando QUOTE.TXT, deve agora ir ao fim do "buffer" e inserir mais textos. O comando –B (B negativo) é utilizado para ir até o fim da última linha, como no seguinte exemplo:

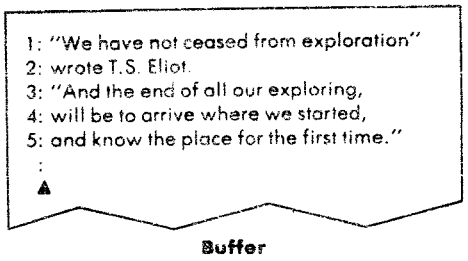



```
1: * -B
 : *
```

O "fim da última linha", na realidade, é um retorno do cursor, o que no código ASCII representa uma combinação de RETURN e LINE FEED: dois caracteres que executam a operação de fazer um 'Carriage Return' e gerar uma nova linha. Portanto, o "fim da úl-

tima linha" é, em essência, o início da próxima; porém, o número da próxima linha nova só será apresentado quando a pessoa inserir caracteres utilizando o comando I. Você pode inserir caracteres que formarão a nova linha.

Por exemplo:




```
: *I ↵
3: "And the end of all our exploring, ↵
4: will be to arrive where we started, ↵
5: and know the place for the first time." ↵
6: ↑Z ↵
: *
```

Agora você pode mostrar o "buffer" completo utilizando os comandos B e T:

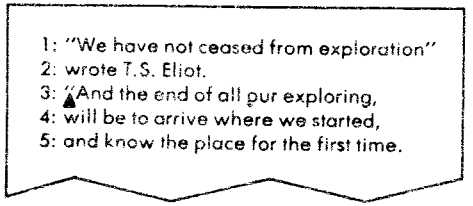
```
: *B#T ↵
1: "We have not ceased from exploration"
2: wrote T.S. Eliot.
3: "And the end of all our exploring,
4: will be to arrive where we started,
5: and know the place for the first time."
1: *
```

A maneira mais simples para se movimentar em direção a outra linha é a utilização de números de linhas. Se a pessoa possuir a versão 2.2 do CP/M, o ED irá mostrar automaticamente os números das linhas. Se a sua versão for a 1.4, essa característica estará desligada; ligue-a novamente executando o comando V (—V a desliga novamente). Se a sua versão for anterior à 1.4, a pessoa está sem sorte — não pode apresentar os números das linhas e necessita usar o comando L para se movimentar para outra linha.

Para escolher uma linha, digite o número da linha, seguido de dois pontos como um comando ED:

```
1: *2: ↵
2: *
```

Para escolher uma *faixa* de linhas, digite o número da primeira linha, seguido por *duas* seqüências de dois pontos e o número da segunda linha, como no exemplo abaixo (o comando T também pode ser usado para mostrar uma faixa de linhas):




```
2: *3::5T ↵
3: "And the end of all our exploring,
4: will be to arrive where we started,
5: and know the place for the first time."
3: *
```

Quando se especifica uma única linha, o 'CP' é movimentado na direção do início desta. Quando a especificação se refere a uma faixa de linhas, o 'CP' é movimentado para o início da primeira linha da faixa, e o ED conta o número de linhas na faixa (neste caso, três) e aplica esse número ao comando T (isto é, '3T') para mostrar a faixa solicitada. O 'CP' permanece na linha 3.

Pode-se também passar as linhas para cima ou para baixo, usando o comando L. O formato para esse comando é:

± nL

O comando L irá mover o 'CP' +n linhas para a frente ou −n linhas para trás, no "buffer", e colocar o 'CP' no início da linha escolhida. Aqui está um exemplo:

```
3: *1L ↵
4: *-2L ↵
2: *
```

A forma 'OL' (onde n é zero) faz o 'CP' se movimentar para o início da linha atual.

Pode-se usar números de linhas para escolher uma faixa de linhas que começa na linha atual, colocando *dois pontos antes do número da linha final:*

```
2: *:5T
2: wrote T.S. Eliot.
3: "And the end of all our exploring,
4: will be to arrive where we started,
5: and know the place for the first time."
```

ALTERANDO, INSERINDO E DELETANDO TEXTOS

Altera-se um texto em uma linha movimentando o 'CP' em direção ao texto, deletando o texto existente e inserindo um novo. Pode-se encontrar também um grupo de caracteres e substituir outro grupo, como veremos na seção seguinte.

Quando se deleta uma linha de um texto, os números das linhas refletem a deleção e se modificam nesse sentido, como demonstra o seguinte exemplo:

```
2:*1::5T
1:"We have not ceased from exploration"
2:wrote T. S. Eliot.
3:"And the end of all our exploring,
4:will be to arrive where we started,
5:and know the place for the first time."
1:2::
2:K
2:*1::4T
1:"We have not ceased from exploration"
2:"And the end of all our exploring,
3:will be to arrive where we started,
4:and know the place for the first time."
1:*
```

O comando K acima deleta a linha 2, e as outras linhas "movem-se para cima", para utilizar o espaço. Ao inserir uma linha, ocorre o oposto – as outras linhas "movimentam-se para baixo" para acomodar a nova linha recém-inserida.

O formato do comando K é:

+ nK

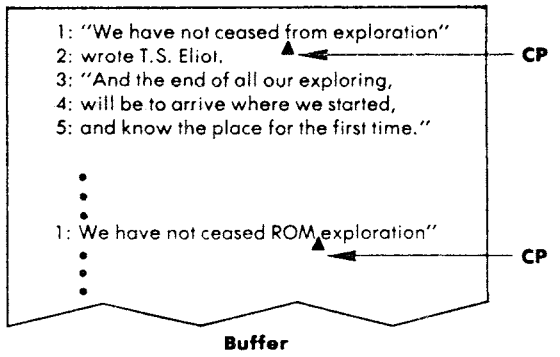
Este comando irá deletar ("kill") a linha atual se não for especificado nenhum valor para n. De outra forma, o comando K irá deletar + n linhas ou −n linhas a partir da linha atual. Se a pessoa fornecer o sinal (#) para n, K suprime 65.535 linhas incluindo (e as que seguem) a linha atual (+ # K), ou 65.535 linhas que ficam antes da linha atual (sem incluí-la). (−# K). Use estas formas para retirar linhas indesejáveis do seu "buffer", mas lembre-se de que não se pode trazer de volta essas linhas (a não ser que elas já existam no arquivo-fonte ou de backup).

Para deletar *caracteres* (e não linhas completas), use o comando D:

+ *n*D

Este comando deleta (apaga) o caractere para o qual o 'CP' aponta, se não for especificado nenhum n. De outra forma, o comando D deleta + n caracteres após (e incluindo) o 'CP', ou −n caracteres atrás do 'CP' (*não* o incluindo).

Aqui está um exemplo: alteramos a palavra 'from' para 'ROM' (e depois retornaremos a 'from'). Faremos isso pelo método mais difícil – movendo o 'CP' com o emprego do comando C, suprimindo 'from' usando o comando D, e inserindo 'ROM' por meio do comando I:




```
1: *1::4T
1: "We have not ceased from exploration"
2: "And the end of all our exploring,
3: will be to arrive where we started,
4: and know the place for the first time."
1: *T
1: "We have not ceased from exploration"
1: *20C
1: *4DIROM↑ZOLT
1: "We have not ceased ROM exploration"
1: *
```

Neste exemplo, movemos o 'CP' para o caractere nº 21 usando o comando '20C'. Contando com o caractere para o qual aponta o 'CP', deletamos 4 caracteres usando o comando D. Depois, fizemos uso do comando I para inserir 'ROM' (terminado por Z) -- observe que o comando I faz a inserção antes do 'CP' e o move. O comando D deleta o caractere 'CP' e também move o 'CP'. Usamos o comando L para trazer o 'CP' até o início da linha, e usamos o comando T para mostrá-lo.

Esta é a nova forma do comando I:

Iinsertions ↑Z

Isto desempenha a mesma função que I ↵, embora não insira retornos do cursor, automaticamente, para gerar novas linhas. Se a pessoa for alterar um grupo de caracteres incluindo uma seqüência de 'Carriage Return', deve usar o comando S (substituir) descrito na seção seguinte, por ser bem mais fácil fazê-lo do que contar caracteres para movimentar adequadamente o 'CP'.

PESQUISANDO E SUBSTITUINDO UM TEXTO

Um meio melhor de movimentar o 'CP' para um grupo de caracteres no texto é "encontrar" esse grupo usando o comando F. O comando S, discutido a seguir, pode ser empregado para "encontrar e substituir". O comando F é uma versão primitiva do comando S, e será descrito em primeiro lugar.

O exemplo mostra o formato do comando F. Executaremos F para achar 'ROM' na linha 1:

```
1:*FROM↵
1:*
```

A ilustração abaixo mostra onde está o 'CP' nesse momento:

```
"We have not ceased ROM ▲ exploration"
                        CP
```

O comando F movimenta o 'CP' para o caractere que fica logo depois do último caractere encontrado, para facilitar o uso do comando D (para deletar os caracteres encontrados). Iremos movimentar o 'CP' de volta para o início da linha, e executar o comando F juntamente com o D e o I no próximo exemplo:

```
1:0L↵
1:*FROM ↑Z-3DIfrom ↑Z0LT↵
1:"We have not ceased from exploration"
1:*
```

-3 caracteres do 'CP' (isto é, 'MOR'). O 'CP' está agora posicionado para que possamos usar o comando I para inserir "from". O comadno 'OLT' movimenta o 'CP' para o início da linha e a apresenta.

Observe que se pode terminar um comando F com um RETURN se a pessoa não for incluir outro comando na mesma linha.

O comando F também pode começar com um número (isto é, nF, onde n é um número positivo) que lhe dirá para achar n ocorrências do grupo de caracteres. Por exemplo, se o aprendiz digitar o comando '5Fthis ↵', o comando não irá parar nem movimentar o 'CP' antes de encontrar a quinta ocorrência de 'this'. O comando F pode examinar o texto que está no "buffer"; para examinar o *arquivo-fonte completo* deve-se usar o comando N (discutido na seção "Operações de ED avançadas").

Caso esteja tentando encontrar um grupo de caracteres, incluindo a seqüência de retorno do carro, pode-se usar, em substituição, uma combinação especial de teclas, ↑L, para representar a seqüência de RETURN e LINE FEED. No nosso exemplo com o comando S, demonstraremos o uso de ↑L.

O comando S (substituir) é o de mais freqüente emprego para pesquisar e substituir grupos de caracteres. O comando S combina as ações dos comandos F, D e I, apresentados anteriormente. O formato para o comando S é:

$$\text{nSoldtext}\uparrow\ \text{Znewtext}\ \left\{\begin{matrix}\uparrow Z\\ \text{↵}\end{matrix}\right\}$$

O comando S examina o "buffer" para localizar o grupo de caracteres *oldtext* e substituí-lo por *newtext*. Pode-se usar ↑Z ou RETURN para terminar a seqüência *newtext* (use ↑Z se a pessoa desejar acrescentar outro comando ao comando S). Pode-se executar esta substituição *n* vezes, e a execução se fará até atingir a *enésima* vez, ou até chegar ao fim do "buffer".

Por exemplo, se o operador digitou o comando "#S Peking ZBeijing ↵", a palavra "Beijing" seria colocada no lugar de "Peking" no texto que se acha no "buffer" (o sinal # representa 65.535 vezes).

Usaremos o comando S para alterar nosso exemplo e mudar o texto de prosa para poesia. Inicialmente, corrigiremos o fim da primeira linha e o começo da segunda em uma única substituição S:

```
1:*2T↵
1:"We have not ceased from exploration"
2:"And the end of all our exploring,
1:*S" ↑L" ↑Z/ ↑L ↑ZB2T↵
1:"We have not ceased from exploration/
2:And the end of all our exploring,
1:*
```

Nesse exemplo, usamos o comando S para localizar o grupo de caracteres começando com 'unquote' (retirar aspas), e terminando com um 'quote' (colocar aspas) na linha seguinte (↑L significa um retorno do cursor) e substituímos por um sinal (/) seguido de um retorno do cursor. Depois executamos a combinação 'B2T' para movimentar o 'CP' para o início do "buffer" e mostrar as duas linhas seguintes.

Nosso próximo exemplo irá colocar um sinal (/) no lugar da vírgula (,), em duas linhas:

```
1:*2::4T↵
2:And the end of all our exploring,
3:will be to arrive where we started,
4:and know the place for the first time."
4:*2:↵
2:*2S,↑Z/↑ZB4T↵
1:"We have not ceased from exploration/
2:And the end of all our exploring/
3:will be to arrive where we started/
4:and know the place for the first time."
1:*
```

Neste exemplo, primeiro movemos o 'CP' para o início da linha 2 ('2:'), e depois fazemos a substituição ('/' no lugar de ','). Depois movemos o 'CP' até o início do "buffer" e apresentamos quatro linhas (B4T).

Necessitamos apenas colocar os primeiros caracteres das linhas 3 e 4 em MAIÚSCULAS:

```
1:*3:↵
3:*DIW↑Z1LDIA↑ZB4T↵
1:"We have not ceased from exploration/
2:And the end of all our exploring/
3:Will be to arrive where we started/
4:And know the place for the first time."
1:*
```

Neste exemplo, movimentamos o 'CP' para o início da linha 3 e deletamos o primeiro caractere (o comando D suprime 'w'). Depois inserimos (comando L) um W MAIÚSCULO. Movimentamos o 'CP' para a linha seguinte (comando 1L), deletamos o primeiro caractere (o comando D suprime 'a') e inserimos um A MAIÚSCULO. Finalmente, movimentamos o 'CP' para o início e mostramos as quatro linhas.

GRAVANDO LINHAS EM UM ARQUIVO

Normalmente, a pessoa terminaria a sua seção ED utilizando os comandos E ou H. Cada um executa uma operação de gravação completa no "buffer"; isto é, *grava* as linhas do texto no arquivo temporário, de saída. Ambos os comandos também copiam o resto do arquivo-fonte (isto é, as linhas que não foram acrescentadas ao "buffer") para o arquivo de saída, e redenominam os arquivos de modo que a pessoa termina com o texto que acabou de ser modificado gravado no arquivo-fonte. O comando E também termina o ED, ao passo que o comando H faz com que o operador permaneça no ED e prepara o novo arquivo-fonte para outras edições.

Se o operador desejar apenas gravar linhas no arquivo de saída sem copiar o "buffer" completo ou o resto do arquivo-fonte, deve usar o comando W ("write"). O comando W assume a seguinte forma:

nW

Este comando grava as n linhas seguintes a partir da linha atual (incluindo-a) no arquivo de saída. O arquivo de saída é o arquivo temporário com um '$ $ $' como extensão. Se o operador não especificar n, a linha atual é gravada no arquivo.

Para ilustrar o emprego do comando W, acrescentaremos mais linhas ao nosso exemplo e gravaremos várias delas no arquivo temporário de saída:

```
1:*-B↓
5:↓
6:                              — wrote T.S. Eliot↓
7:↑Z
6:B#T↓
1:"We have not ceased from exploration/
2:And the end of all our exploring/
3:Will be to arrive where we started/
4:And know the place for the first time."
5:
6:                              — wrote T.S. Eliot
1:*3W↓
1:#T↓
1:And know the place for the first time."
2:
3:                              — wrote T.S. Eliot
1:*#W↓
:*
```

Neste exemplo, primeiro acrescentamos um novo texto ao final do "buffer" (comandos '-B1') (utilize um RETURN para gerar uma linha em branco, e um I para inserir um espaço de tabulação). Depois apresentamos o "buffer" completo com 'CP' no início do mesmo (comandos 'B #T'). A seguir apresentamos as três linhas seguintes, incluindo a atual (linhas 1, 2 e 3). Estas linhas estão agora em QUOTE.$$$, o arquivo temporário de saída. As linhas restantes do "buffer" são agora renumeradas. O último comando ('#W') dá saída às próximas 65.535 linhas, incluindo a atual, o que faz sair o que ainda estava no "buffer". Então, o "buffer" agora está vazio, e o ED mostra "nenhum número de linha" (': *').

Ainda temos que salvar o *resto* do arquivo-fonte (se não houver quaisquer linhas não acrescentadas) e *redenominar* o arquivo temporário de saída QUOTE.$$$ de QUOTE.TXT (e redenominar o antigo QUOTE.TXT para QUOTE.BAK). Os comandos E e H executarão estas operações automaticamente.

Uma forma especial do comando W se alia à forma '0A' do comando A – '0W' (onde n é zero) irá gravar metade do "buffer" e '0A' permitirá acrescentar linhas para preencher a última metade do "buffer". O número de linhas que são iguais à metade do "buffer" depende do comprimento das linhas do seu texto e do tamanho do seu sistema. Estas formas são principalmente usadas para "fazer entrar metade do "buffer", fazer sair metade, fazer entrar a outra metade etc."

Se a pessoa desejar rearrumar linhas no "buffer" ou intercalar linhas de outro arquivo, necessitará então utilizar comandos especiais (descritos em "Operações Avançadas") para trazer linhas de arquivos para o "buffer", e *depois* gravar a versão final para um arquivo, usando o comando W (ou reservar o "buffer" completo e o que resta do arquivo-fonte com os comandos E ou H).

OPERAÇÕES ED AVANÇADAS

Examinando o arquivo-fonte

Pode-se utilizar o comando N para executar uma operação – "encontrar algo" no "buffer" (como o faz o comando F) *e* o resto do arquivo-fonte. O comando N opera da mesma forma que o comando F, mas não pára no fim do "buffer" – executa um acréscimo automático e continua a acrescentar linhas do arquivo-fonte até encontrar o grupo de caracteres que foi especificado no comando.

O formato para o comando N é:

nNtext { ↑Z / ↵ }

OBSERVAÇÃO: As chaves { } designam uma opção entre os dois comandos. O comando N examina o "buffer", procurando a *enésima* ocorrência do *texto* nas linhas que seguem o 'CP'; se não encontrar essa ocorrência, irá acrescentar linhas do arquivo-fonte ao "buffer" até que a encontre. A pessoa pode terminar sua seqüência de *texto* com ↑Z se você for acrescentar outro comando; caso contrário, pressione a tecla RETURN.

O comando N irá colocar o 'CP' depois do último caractere da *enésima* ocorrência do *texto*, assim como nos comandos F.

Inserindo texto de uma biblioteca de arquivos-fonte

"A biblioteca de arquivos-fonte" é um termo conveniente na documentação da Digital Research para qualquer arquivo com uma extensão 'LIB' (por exemplo, SAMPLE.LIB). Pode-se usar uma "biblioteca de arquivo-fonte" como um arquivo-fonte alternativo – um arquivo com texto que se deseja inserir no "buffer" para intercalar-se com as linhas do arquivo-fonte original. Para fazê-lo, é necessário, em primeiro lugar, haver um arquivo com uma extensão 'LIB' que já possua o texto gravado.

O formato para o comando R é:

Rfilename

O comando R insere linhas do arquivo 'LIB' que a pessoa especifica no *filename*. Irá inserir linhas na posição atual do 'CP' de modo similar ao do comando I. R irá inserir todo o arquivo 'LIB' até encontrar o indicador fim-de-arquivo (↑Z).

Transferindo e Inserindo Linhas num Arquivo Temporário

Se a pessoa possuir a versão 1.4 do CP/M ou outra posterior, poderá usar o novo comando X para transferir linhas do "buffer" para um arquivo temporário "de retenção", e usar o comando R para inserir linhas desse arquivo no "buffer". O arquivo de "retenção" recebe o nome X$$$$$$$$.LIB e só existe enquanto o programa ED está sendo executado. Qualquer término normal do ED irá suprimir esse arquivo, mas se a pessoa terminar o ED com um ↑C ("warm boot"), o arquivo permanecerá (mas se executar novamente o ED, o arquivo será deletado).

O formato do comando X é:

nX

O comando X transfere (copia) as n próximas linhas a partir da linha atual para o arquivo temporário X$$$$$$$$.LIB. As linhas no "buffer" permanecerão: elas são apenas copiadas para o arquivo temporário. Se a pessoa desejar, poderá usar o comando K para deletar as linhas após copiá-las. As linhas transferidas acumulam-se no arquivo temporário na ordem em que elas são copiadas por comandos X sucessivos. Usando esse comando, o novo texto pode ser construído em blocos sucessivos.

As linhas podem ser recuperadas usando-se o comando R na forma 'R' sem o *filename*. Todas as linhas transferidas são então inseridas depois do 'CP' (como acontece com o comando I). Contudo, o comando R não esvazia o arquivo temporário, mas simplesmente copia as linhas. Pode-se recuperá-las quantas vezes se desejar (o que é útil para linhas que se repetem). Pode-se esvaziar o arquivo temporário (isto é, deletá-los) executando a forma '0X' (onde n é zero) do comando X.

Se a pessoa desejar preservar o arquivo temporário X$$$$$$$$.LIB, deve usar ↑C para abortar o ED, e imediatamente dar um novo nome ao arquivo (para se livrar da extensão 'LIB') para que ele não seja destruído ao executar novamente o ED!

JUSTAPOSIÇÃO

O comando J é usado para fazer a justaposição de três grupos de caracteres em um texto. Ele localiza um grupo no texto, justapõe um segundo grupo e deleta os caracteres

que se seguem a esse par até encontrar um terceiro grupo, efetuando desta forma a justaposição de todos os três grupos do texto.

O formato para o comando J é:

nJstring1↑Zstring2↑Zstring3 {↑Z / ↵}

O comando J começa por procurar a primeira ocorrência do 'string 1' a partir do 'CP' no "buffer". Se encontra 'string 1', o comando J insere o 'string 2' imediatamente após o 'string 1', e move o 'CP' para o fim do 'string 2'. Depois, procura o 'string 3'; se encontrar, esse comando deleta todos os caracteres entre o 'string 2' e o 'string 3', e deixa o 'CP' apontando para o primeiro caractere do 'string 3'. Se não encontrar, nada é suprimido. O comando J faz isto n número de vezes, ou até que tenha esgotado todas as linhas do "buffer".

Uma das utilizações do comando J é diminuir as linhas do texto. Escolha um grupo de caracteres que serão o fim da linha a ser encurtada. Estes formarão o 'string 1'. A idéia é justapô-los à seqüência do retorno do cursor, representada por ↑L ('string 3'). Este resultado é obtido inserindo um 'string 2' em branco ou nulo.

Repetindo um grupo de comandos

Pode-se juntar vários comandos ED em um só "comando" que então poderá ser executado repetidamente. O comando M (que representa "macro") assume a seguinte forma:

nMstringofcommands {↑Z / ↵}

Agrupe os seus comandos em 'string-of-commands', precedido de 'M', e o M irá executar os comandos n vezes, se n for maior do que um. Se n for zero ou um, os 'string-of-commands' são executados repetidamente até que ocorra um erro (como alcançar o fim do "buffer").

Eis aqui um exemplo que transforma todas as ocorrências de 'Peking' para 'Beijing' no "buffer" atual, e envia cada linha alterada:

```
MSPeking↑ZBeijing↑ZOTT
```

CONDIÇÕES DE ERRO DO ED

Sempre que ocorrer um erro no programa ED, é mostrado o último caractere antes do erro e uma das mensagens de Figura 4.15.

As novas versões do CP/M apresentam 'BREAKxATc' onde x é um dos símbolos de erro, e c é o comando ED que estava sendo executado.

Ocasionalmente, o sistema operacional (isto é, o CP/M ou o MP/M) detecta uma condição de erro de sistema (como um de disco). Dependendo da condição, a pessoa geralmente pode terminar o ED com um C (partida "a quente"), mas deve antes recuperar a

?	Não reconheço o comando, que comando é?
>	O "buffer" está cheio. Use um dos comandos D, K, S, W, E ou H para retirar caracteres. Ou, o seu 'string' com F, N, ou S é comprido demais.
#	Não posso executar o comando tantas vezes (como no comando F ao se atingir o fim do "buffer").
0	Não posso abrir o arquivo 'LIB' em um comando R.
	(Verifique se existe o arquivo 'LIB', ou se foi usado o *filename* apropriado).

Figura 4.15: *Mensagens de erro do ED*

cópia original do arquivo-fonte. Por exemplo, se o CP/M detectar um erro CRC ('cyclic redundancy check'), irá mostrar:

```
PERM ERR DISK d
```

onde d é o seu *drive* de disco atual. Pode-se ignorar o erro teclando qualquer caractere no seu terminal (embora deva verificar o seu "buffer", para achar erros que possam existir) ou então a pessoa pode fazer um "warm boot" (↑C) ou um "cold boot" (dar nova partida "a frio") e recuperar o arquivo .BAK (arquivo com a extensão BAK, por exemplo, QUOTE.BAK).

Resumo dos aperfeiçoamentos da versão 2.2 do CP/M

Na versão 2.2 do CP/M o ED assume que a opção de numeração das linhas esteja sempre ligada. Para eliminar os números das linhas, digite: –V. (Naturalmente, este modo pode ser restaurado com :V.).

Quando se usa o modo 'insert' (I) podemos empregar todos os caracteres usuais de controle do CP/M: DEL, C, E, H, J, M, R, U, X. Esses caracteres estão descritos no apêndice D.

Finalmente, o ED respeita os atributos de arquivo da versão 2.2. Por exemplo, um arquivo 'read/only' pode ser examinado, mas não alterado. Se for necessário modificar esse arquivo, então o seu estado 'read/only' deve, inicialmente, ser alterado para R/W, usando-se um comando STAT.

RESUMO

Neste capítulo, aprendeu-se a usar o editor, ED. O ED é um editor de texto geral que permite modificar um texto com apenas algumas teclas a serem acionadas. A utilização mais freqüente do ED é a de criar um arquivo simples, como um programa (se não existir outro editor especializado). Embora os comandos do ED sejam menos convenientes do que um processador de palavras especializado, ele pode ser empregado para redigir cartas ou documentos.

O ED é uma ferramenta apropriada para corrigir e modificar arquivos existentes (em particular, o ED pode ser usado para "limpar" um disquete do sistema que contém caracteres errados devido a um erro de manipulação). O ED pode também ser usado para substituir novas palavras ou linhas em um arquivo. Os apêndices D e E apresentam resumos dos caracteres de controle e comandos do ED.

Neste capítulo, o leitor não só aprendeu a usar o editor ED, mas também a servir-se de muitos comandos, caracteres de controle e outras convenções. Também aprendeu a trabalhar com arquivos, e a seguir transferências de texto entre o disco, a memória e o vídeo. Este conhecimento ajudá-lo-á a compreender a operação da maioria dos outros programas no computador.

Capítulo 5

Examinando o Interior do CP/M (e MP/M)

INTRODUÇÃO

Este capítulo irá descrever a operação interna do sistema operacional CP/M. O aprendiz desejará ler este capítulo se estiver interessado em aprender a respeito do funcionamento de um sistema operacional, ou se pretender alterar ou usar algumas das rotinas do CP/M. Contudo, se desejar apenas usar o CP/M para executar uma tarefa específica, a informação apresentada neste capítulo não será necessária.

O maior valor deste capítulo será para um programador de sistemas, ou qualquer pessoa que já esteja familiarizada com programação e que deseje saber como funciona o CP/M. Considerando que um livro a respeito do CP/M não seria completo sem tais informações, este capítulo destina-se a esta classe específica de leitores. Se o leitor quiser saber "o que está acontecendo dentro do sistema", continue a sua leitura.

Apresentaremos, inicialmente, uma síntese simplificada da operação do CP/M, introduzindo os seus componentes lógicos, suas funções e o modo pelo qual interagem. Depois descreveremos a *alocação de memória*, ou seja, a maneira pela qual esses módulos de software são distribuídos na memória, assim como a organização do sistema de arquivos. A seguir, cada um dos três módulos do CP/M será apresentado, detalhadamente, junto com os comandos fornecidos para fazer uso de suas capacidades. Outra seção discutirá os problemas que surgem quando se tenta adaptar o CP/M a novas configurações de hardware e mostraremos, ainda, um exemplo de alterações do CP/M para que o sistema se comporte como um sistema orientado a "menus". Finalmente, dedicaremos uma seção especial ao MP/M.

UMA VISÃO SIMPLIFICADA DA OPERAÇÃO DO CP/M

Fluxo de Controle

O CP/M possui três módulos funcionais, que se denominam CCP (Console Command Processor), BIOS (Basic Input/Output System), e BDOS (Basic Disk Operating System). (Veja a Figura 5.1.)

A função do CCP é a de se comunicar com o usuário e interpretar os comandos digitados no teclado. O CCP é, primordialmente, um *interpretador de comandos* e emprega recursos oferecidos pelos outros dois módulos, BIOS e BDOS. Isto representa apenas uma descrição simplificada, pois, como veremos mais adiante, o CCP também executa um processamento interno. Conceitualmente, o CCP pode ser visualizado como a "parte inteligente" do sistema operacional, ao passo que os outros dois são módulos de serviço.

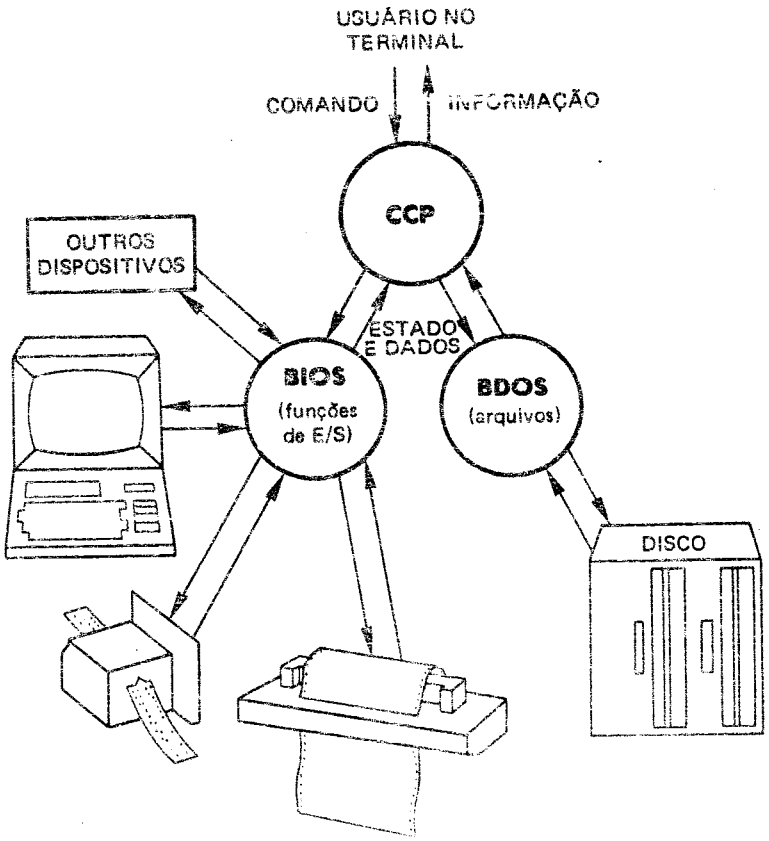


Figura 5.1: *Fluxo de controle*

O BIOS inclui uma coleção de *drives de periféricos*, isto é, rotinas que se encarregam da comunicação com os vários dispositivos ligados ao sistema. A função do BIOS é enviar ou receber informações a respeito do estado e dos dados entre um dispositivo e o depurador do CCP. O BIOS é chamado pelo CCP, com parâmetros específicos para indicar o serviço solicitado. Isto está ilustrado na Figura 5.1.

O BDOS está encarregado de gerenciar os arquivos de disco. Inclui um certo número de rotinas utilitárias que executam as funções desejadas no disco. Como qualquer bom sistema operacional de disco, o objetivo do BDOS é tornar o gerenciamento dos arquivos invisível (ou "transparente") para o usuário. O BDOS irá executar todas as tarefas necessárias para alocar os vários blocos de informação contidos no disquete, verificar a validade do acesso e a integridade dos dados, e alocar e liberar eficientemente o armazenamento.

Alocação de Memória

O CP/M divide a memória disponível em quatro zonas, como mostra a Figura 5.2. O topo da memória está reservado para as rotinas do próprio CP/M, isto é, para o CCP, BDOS e BIOS, como mostra a Figura 5.2. Algumas posições na parte inferior da memória estão reservadas para o sistema, como as primeiras 256 posições de memória, isto é, *página 0* (descrita mais detalhadamente em outra parte deste capítulo). Finalmente, a maior parte da memória, entre a posição 0100 hexadecimal e CBASE (denominada TPA) está disponível para a execução dos programas. Esta alocação padronizada da memória carrega programas em 100H nos computadores de vias S-100, como os da Cromemco, Imsai, Altair e North Star. Porém, no caso de outros computadores (por exemplo, o TRS 80 e o Heath H8) com programas pré-armazenados na memória baixa ROM, os programas se iniciam em 4300H.

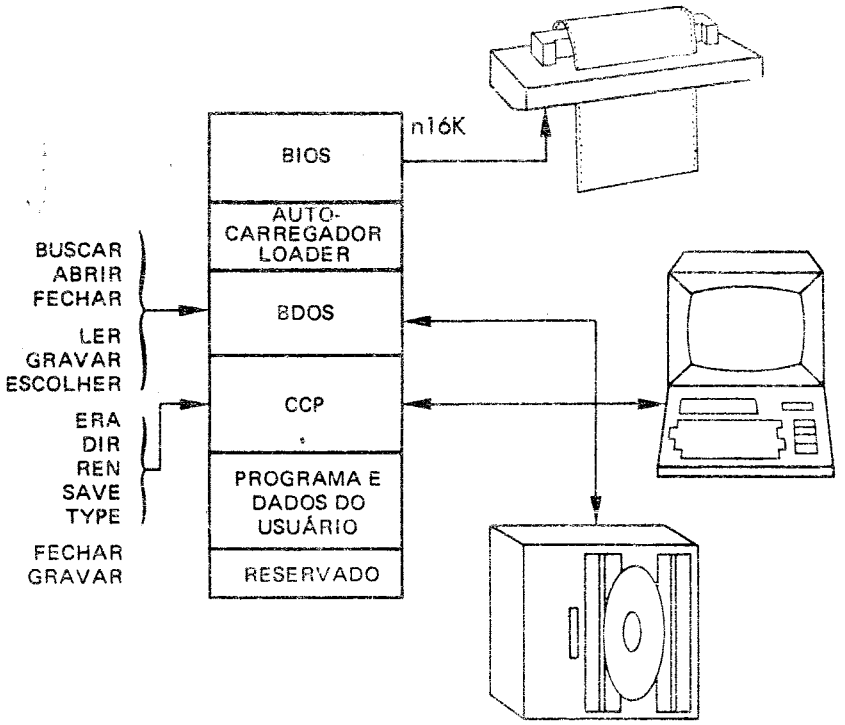


Figura 5.2: *O mapa de memória do CP/M*

A TPA ou *Transient Program Area* é o nome utilizado na documentação do CP/M para qualquer programa a ser executado. O CP/M assume que o sistema tenha 16, 32, 48 ou 64K. Em um sistema de 16K a base do CCP, denominada CBASE, está no endereço de memória 2900 (hexadecimal). Para cada 16K acrescentado ao sistema, esse endereço é incrementado de 4000 (hexadecimal). Isto significa que cada 16K de memória acrescentados ao sistema aumenta a TPA em função desta quantidade.

Veremos, mais tarde, que um programa do usuário pode utilizar quase toda a memória, sobrepondo-se ao CCP ou outras áreas de memória que pertencem às rotinas do CP/M. Mais adiante estenderemos a discussão deste aspecto. Mas isto significa que, quando o programa terminar a execução, ele deve trazer o CP/M de volta à memória.

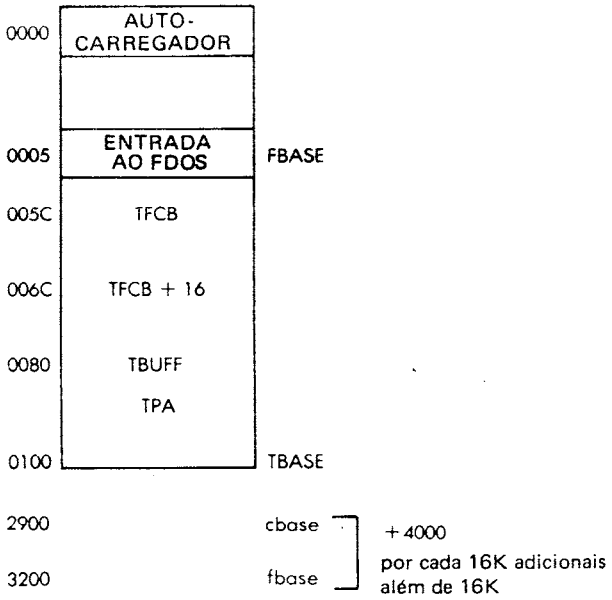


Figura 5.3: *Mapa CP/M padrão*

DESCRIÇÃO DETALHADA

O Sistema de Arquivo

Uma das funções primárias de qualquer *sistema operacional de discos* (DOS) é oferecer um gerenciamento eficaz e conveniente dos arquivos baseados em discos. Uma compreensão da organização geral de um sistema de arquivos é, portanto, exigida para entender a operação do próprio sistema operacional (BDOS para o CP/M), assim como a operação do CCP, que constrói e manipula descritores de arquivos. Todo o processamento de arquivos é realizado entre o CCP e o BDOS, e a porção BIOS do CP/M não precisa se preocupar com a natureza explícita de um arquivo. O BIOS essencialmente irá transmitir e receber simples fluxos de dados. Examinemos então a estrutura de arquivos no CP/M.

Um *arquivo* é uma unidade lógica que contém um texto, dados ou programas. A tarefa do sistema operacional de disco é a de suplementar esta facilidade lógica com os recursos físicos fornecidos pelo meio de armazenamento, isto é, o disquete, como no nosso caso. Vejamos, agora, como isto é feito.

Já explicamos que os disquetes são organizados em trilhas e setores. Cada setor, em um disquete-padrão de 400 mm, contém 128 bytes de informação. Na nomenclatura CP/M, isto é denominado um *registro.* Cada arquivo em um disquete é uma coleção de registros. Como não é possível manter um arquivo como uma seqüência completa de registros em um disco, é necessário usar setores que estão espalhados sobre toda a superfície do disco. Torna-se então necessário estabelecer algum tipo de estrutura de listas para se poder saber quais são os registros associados a um arquivo. Muitas técnicas são usadas para tal fim. No caso do CP/M, a lista dos setores que pertencem a um arquivo de disco está contida em uma entidade especial, chamada de *descritor,* na terminologia da ciência da computação, ou *o bloco de controle de arquivos,* na nomenclatura CP/M.

Com o CP/M, cada arquivo pode ter até 16 unidades. Cada unidade contém de 0 a 128 registros, isto é, de 0 a 16K bytes. O maior arquivo, no CP/M, pode, portanto, ter no máximo 16 unidades de 16K bytes ou 16 x 16K = 256K bytes. Isto é um pouco mais do que a capacidade máxima de um disquete-padrão de 400 mm. Cada *bloco de controle de arquivo* (FCB) descreve até 16K bytes de um arquivo determinado. São fornecidos mecanismos adicionais para vincular até 15 extensões adicionais do arquivo.

A utilização do espaço do disquete é representada na Figura 5.4. As duas trilhas externas dos disquetes são usadas para armazenar o sistema CP/M. O resto do disco é empregado para armazenar os arquivos e o diretório de arquivos.

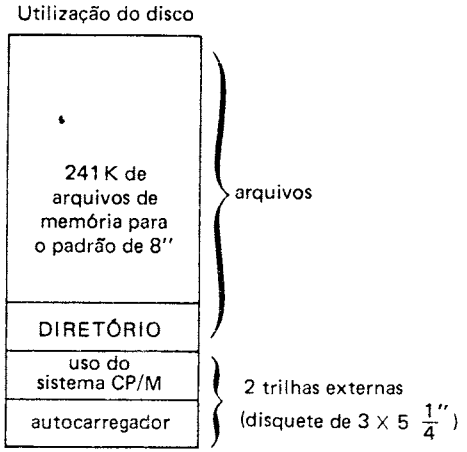


Figura 5.4: *Utilização do espaço do disquete*

O próprio bloco de controle de arquivos usa 33 bytes e é armazenado em uma área de diretório do disquete, reservada para este fim (veja Figura 5.4). Sempre que o arquivo estiver ativo, isto é, se o CP/M passa a ter acesso a ele, seu FCB é trazido para a área transiente do programa para que o sistema operacional possa ter acesso a ele de forma rápida e adequada.

Executamos nossa tarefa: a de mapear um arquivo lógico nos setores disponíveis em um disquete-padrão. Conseguimos isto mantendo um diretório dos blocos, alocados ao arquivo, em um local especial chamado FBC.

Uma exigência básica de qualquer bom sistema operacional é a de que o usuário possa designar um arquivo por meio de um *nome simbólico* (os usuários geralmente não gostam de se referir a arquivos por meio de números). O CP/M oferece esta facilidade, junto com a proteção adicional, que também exige que o usuário especifique o *tipo* do arquivo. Portanto, um mecanismo para localizar um arquivo *real,* quando o seu *lógico* é fornecido, deve ser providenciado, de modo que se possa examinar o diretório de arquivos até que se associe o nome fornecido pelo usuário à entrada no diretório. Esta associação entre um arquivo real e seu nome é realizada dentro do FCB. A outra parte essencial do bloco de controle de arquivo (FCB) é o nome do arquivo. Mostramos isto na Figura 5.5.

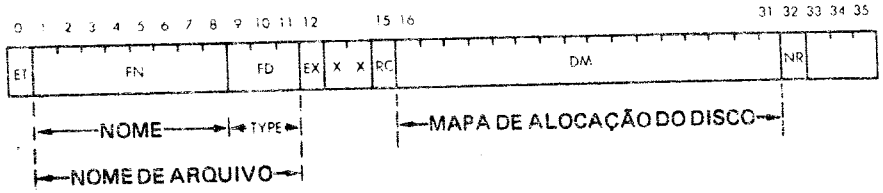


Figura 5.5: *O bloco de controle de arquivo*

A próxima tarefa associada com um planejamento eficiente de um sistema de arquivos é oferecer características de segurança e proteção para o período no qual se tem acesso aos arquivos. Por exemplo, os arquivos podem ser especificados como *Read-Only* ou *Read-and-Write*. Também é possível especificar que o arquivo só possa ser objeto de acesso mediante uma *senha*, ou um arquivo pode ser *executável*, mas não pode *ser lido*, e equipado com vários dispositivos de proteção que evitam execução ou acesso não autorizado. A versão 1.4 do CP/M praticamente não oferece tais medidas de proteção, mas possui a capacidade de reservar vários bytes no FCB (Bloco de Controle do Arquivo) para incorporá-las. A versão 2.2 do CP/M agora oferece quatro atributos de arquivo: R/O, R/W, SYS, e DIR.

Finalmente, são necessários alguns bytes para funções básicas do sistema. Elas são mostradas na Figura 5.5. Em especial, a localização NR na posição 32 contém o número do próximo registro a ser lido ou gravado.

Revisemos os campos do bloco de controle do arquivo apresentados na Figura 5.5. Contém oito campos que são, a partir da direita:

ET (posição 0): Descreve o tipo da entrada. Normalmente é 0, e não é usado no CP/M 1.4. Na versão 2.2 do CP/M e no MP/M representa o código do *drive*.

FN (posições 1-8): É o nome do arquivo, que pode conter até 8 caracteres. Qualquer caractere não fornecido pelo usuário fará com que se coloque um espaço em branco em ASCII.

FD (posições 9-11): É o tipo do arquivo. Possui de um a três caracteres alfanuméricos e também é completado se necessário com espaços em branco em ASCII. Sempre que o diretório de um disquete for listado, o nome e o tipo do arquivo são mostrados. Na versão 2.2 do CP/M, dois bits indicam o atributo, e um bit indica que o arquivo foi atualizado.

EX (posição 12): É a extensão do arquivo, geralmente 0.

XX (posições 13 & 14): É um campo não utilizado no CP/M 1.4, geralmente 0.

RC (posição 15): É o contador de registros. Um módulo de arquivos pode ter de 0 a 128 registros (até 16K bytes). É chamado de tamanho atual da extensão.

DM (posições 16-31): O mapa de alocação do disco mantém estado dos setores do disco usados por esse módulo de arquivo.

NR (posição 32): Contém o número do próximo registro a ser lido ou gravado, geralmente 0, e denominado CR na versão 2.2 do CP/M. R0-1-2 (posições 33-35) só é usado na versão 2.2 do CP/M para acesso randômico e representa o número do registro opcional (0 a 64K).

Operação do Sistema

No sistema operacional CP/M, cada programa em linguagem de máquina é instalado ou carregado na área transiente de programa (apresentada na Figura 5.2). No "CP/M padrão", a execução inicia-se no endereço 100H.

Na terminologia CP/M, o programa é chamado "transiente" e pode utilizar os recursos de operação do sistema executando um JMP para o endereço 05H.

Esse endereço é a porta de entrada para o sistema operacional. Trata-se de um ponto de entrada fixo, único, independente do tamanho atual da memória e resulta em uma transferência de controle para o CP/M. As rotinas do CP/M residem na extremidade superior da memória (como foi mostrado na Figura 5.2). A chamada do sistema operacional deve ser acompanhada de um parâmetro especificando o serviço solicitado. Este parâmetro é passado como um número de função e está contido no registrador C8080 ou do Z80. A versão 1.4 do CP/M oferece vinte e sete códigos (36 para o CP/M 2.2) que permitem o acesso aos vários dispositivos de entrada e saída, incluindo os arquivos em discos.

Execução do CP/M

Do ponto de vista do usuário, o sistema funciona da seguinte maneira: o CCP (Console Command Processor) apresenta o *prompt* do sistema e aguarda um comando. As transmissões são manejadas pelo BIOS (Basic Input/Output System) e a linha de comando surge no "buffer" do CCP.

Quando o CCP recebe o comando (e os *filenames* associados), executa o comando imediatamente se este for embutido (isto é, residindo permanentemente na parte da memória do computador reservada ao sistema). Se o comando não for embutido, o CCP assume que este é um programa transiente com um *filename* do tipo "COM" (por exemplo, PIP.COM), e solicita a BDOS que procure o arquivo e leia uma cópia dele para a TPA (Transient Program Area), a parte não utilizada da memória do computador (isto é, não usada pelos componentes do sistema CP/M). O CCP então constrói um bloco de controle de arquivo para o(s) arquivo(s) sobre o qual (ou quais) o comando (ou programa) irá agir.

Se existir mais de um *filename*, o CCP constrói um bloco de controle de arquivo para cada arquivo, mas que o novo programa na TPA (isto é, o comando transiente) irá tratar de achar e ter acesso aos arquivos. O CCP só olha para o primeiro arquivo chamado BDOS e solicitando uma cópia do arquivo.

O CCP termina, então, liberando assim espaço na memória do computador; o programa transiente agora pode ocupar o espaço usado pelo CCP.

Do ponto de vista do sistema, todos os arquivos são iguais. O BDOS acha o arquivo verificando o bloco de controle de arquivo criado pelo CCP. Este FCB está quase vazio e só contém o *filename*. Cada arquivo no disquete tem o seu próprio FCB (uma cópia do último criado pelo CCP), de modo que o BDOS simplesmente compara o *filename* do FCB do CCP com os *filenames* nos FCB dos arquivos.

Quando o BDOS acha o arquivo certo, fornece mais informações para o bloco de controle de arquivo que o CCP criou. Cada vez que o programa tem acesso a este determinado arquivo, o BDOS altera alguma das informações no bloco de controle de arquivo. Quando o programa fecha o arquivo, o BDOS efetua uma modificação final e depois copia a FCB da memória do computador (junto com a informação atualizada do arquivo) para o disquete, atualizando tanto o arquivo como a cópia da FCB do arquivo.

O BDOS assume que todos os arquivos são iguais, identificados por *filenames*. O CCP olha para a primeira palavra digitada (depois de apresentar o *prompt* do sistema) e assume que se trata ou de um comando embutido ou de um comando (programa) que reside em um disquete com um *filename* que é o nome do comando, incluindo a extensão "COM" (por exemplo, PIP.COM, LOAD.COM etc.). Se a primeira palavra não for um comando embutido e o CCP não achar um arquivo do tipo COM com um nome que se associa com a primeira palavra, o CCP mostra novamente a palavra que foi digitada e coloca no final um ponto de interrogação.

Se uma extensão 'COM' for colocada em um arquivo que não é um programa propriamente dito, e o nome desse arquivo for digitado como se fosse um comando, o CCP irá tentar executar o arquivo como se fosse um programa.

O CCP também verifica quaisquer *filenames* associados que poderiam ser digitados com o comando, mas não os julga nem decide como usá-los. Quando o comando (ou programa) começa a ser executado, ele mesmo decide o que fazer com os arquivos associados. Por exemplo, o comando ASM é um programa transiente que só trabalha com arquivos que tenham a extensão 'ASM'. O comando LOAD é outro programa transiente que espera um arquivo do tipo "HEX" para carregar. O comando LOAD produz um arquivo 'COM' a partir de um arquivo "HEX" em linguagem de máquina (esse arquivo é um programa).

Para se comunicar com os dispositivos periféricos, o sistema emprega um "portador de mensagens", o BIOS. O BIOS executa operações simples como "leia um caractere a partir do teclado do terminal" e "imprima um caractere na impressora". Para aquele que programa e instala o sistema, o BIOS é a parte do sistema que deve ser modificada para se ajustar ao hardware ("ambiente"). A Digital Research fornece um BIOS que irá funcionar em um Intel MDS-800 com dispositivos-padrão que se conectam ao MDS-800 (correspondente às definições periféricas de hardware do Intellec MDS). Para operar essa versão do CP/M em outro ambiente de hardware, o programador do sistema necessita apenas alterar o módulo BIOS.

FDOS E CCP: SUAS OPERAÇÕES

Organização geral

O mapa de memória do CP/M (depois de armazenado na memória principal do computador) é apresentado na Figura 5.6. O principal ponto de entrada para FDOS está na posição *"boot"* + 0005H na "página zero", a área da memória principal que o sistema reserva para os valores do próprio sistema.

A posição *"boot"* inicia instruções em código de máquina que executam uma partida do sistema ("warm boot"); portanto, os programas transientes apenas necessitam pular para *"boot"* para chamar de volta o CCP a fim de reiniciar as operações do CP/M. Os valores exatos de *"boot"*, *"tbase"*, *"cbase"* e *"fbase"* dependem da versão do CP/M, mas os endereços são usados – um programa do usuário (um programa transiente) ou um comando transiente vai ocupando o espaço de *tbase* para *cbase*, e possivelmente para *fbase* se gravar sobre o CCP.

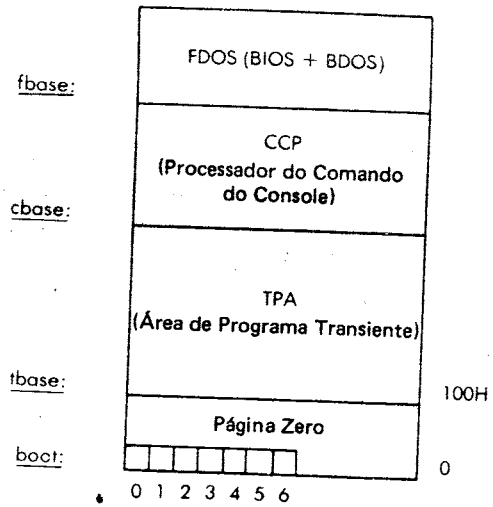


Figura 5.6: *Entrada para Endereço Base Atual (tamanho do espaço TPA + CCP disponível). O FDOS representa um pulo para fbase*

Quando o CCP recebe uma linha de comando, procura um arquivo COM para associar com o nome do comando, a não ser que este esteja embutido, quando irá iniciar imediatamente a execução. O CCP também constrói blocos de controle de arquivo para quaisquer *filenames* que também apareçam na linha de comando. Depois que o CCP termina, a imagem de memória do arquivo COM é trazida para a TPA (talvez até gravando sobre o CCP), onde poderá ter acesso às operações FDOS.

O FDOS está dividido nas duas partes que já apresentamos: O BIOS, que controla todos os periféricos e transmissões, e o BDOS, que examina os discos para localizar arquivos e gerenciar o bloco de controle de arquivo para cada arquivo, com o propósito de permitir um acesso randômico. Examinaremos, agora, cada uma destas partes mais detalhadamente.

Operações BIOS

Obtemos acesso tanto ao BIOS como ao BDOS através do principal ponto de entrada para FDOS. Um BIOS é especificado passando-se um número de função e um endereço de informação. Por exemplo, se o caractere 'B' do ASCII fosse enviado ao terminal, o número de função para a operação "write console" (função 2) seria colocado no registrador C, e o valor de 'B' seria colocado no par de registradores D e E.

As operações do BIOS são resumidas abaixo com cada número de função. (É aconselhável verificar, junto à Digital Research, se houve aperfeiçoamento ou alterações):

1. Read console (terminal). Devolve um caractere ASCII.
2. Write console (terminal). Envia um caractere ASCII.
3. Read reader. Devolve um caractere ASCII do dispositivo de leitura (RDR:).
4. Write punch. Envia um caractere ASCII para o dispositivo de perfuração (PUN:).
5. Write list. Envia um caractere ASCII para o dispositivo de listagem (LST:).

6. Direct console input/output (somente na versão 2.2 do CP/M e no MP/M). Envia 'FF' para receber caractere ou estado, ou envia um caractere para o console.
7. Get I/O status. Devolve um byte com o estado do dispositivo.
8. Set I/O status. Envia um byte com um estado para o dispositivo.
9. Print "buffer". Envia um 'string' inteiro (começando com o endereço e terminando com '$').
10. Read "buffer". Envia o endereço do "buffer" de leitura e retorna com o "buffer" preenchido.
11. Interrogate console ready. Se o bit menos significativo do byte for 1, então o console está disponível e pronto para entrar em ação.

Blocos de Controle de Arquivos

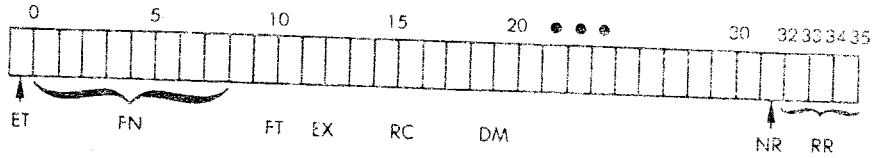
Quando o BDOS cria um arquivo, também constrói um bloco de controle de arquivo com informações, incluindo nome e alocação de memória. Cada bloco de controle de arquivo pode descrever 16K bytes, o que significa que temos 128 registros de 128 bytes cada; esta unidade é denominada uma "extensão" ou um "domínio" ("extent"). Um bloco de controle de arquivo pode endereçar até 256K bytes usando mecanismos automáticos de CP/M.

Os blocos de controle são armazenados junto com os arquivos no disquete, e o BDOS os traz para a memória principal antes de serem realmente acessados (Operações BDOS: "open file" e "make file"). O BDOS e o programa transiente estão continuamente atualizando os blocos de controle de arquivo, enquanto o programa transiente está sendo executado. Quando o programa transiente executa as operações do BDOS de "close file", o BDOS registra as versões finais dos blocos de controle de arquivo no disquete. Desta forma, toda a atualização de arquivos individuais ocorre na memória principal do computador, enquanto os arquivos originais não alterados permanecem no disco.

O CCP constrói blocos de controle de arquivo transientes para os *filenames* em uma linha de comando, e deixa a maior parte da informação em branco, para que sejam providenciadas pelo comando ou programa transiente. O CCP compara o *filename* encontrado na linha de comando (armazenado em um "buffer" de leitura para que o CCP possa inspecioná-lo) com os *filenames* encontrados pelo BDOS nos blocos de controle de arquivos com o propósito de recuperar a informação referente à alocação adequada da memória (isto é, para achar os domínios de um arquivo). O CCP sempre traduz caracteres em minúsculos na linha de comando para MAIÚSCULOS, a fim de se adaptar às convenções de nomes de arquivo do CP/M.

OPERAÇÕES BDOS

Podemos também ter acesso ao BDOS usando a entrada FDOS e passando um número de função e um endereço de informação (o ponto de entrada para o FDOS é *boot* + + 0005H). Por exemplo, se for necessário executar a operação leitura de um arquivo em disco, o programa deve enviar o número da função para um "disk read" (20 ou 33), junto com o endereço do bloco de controle do arquivo para o arquivo que se deseja ler. O BDOS executaria a função e depois retornaria com uma indicação de êxito da operação ou um indicador de erro (indicando que a leitura não foi bem-sucedida).



Informação das Posições de Campos

ET	0	Tipo de entrada (zero assumido em CP/M 1.4, posicionado para "drive code" em CP/M 2.2 e MP/M).
FN	1-8	*Filename* preenchido com espaços em branco (caracteres ASCII).
FT	9-11	Tipo de arquivo, por vezes chamado de extensão de *filename* (também preenchido com espaços em branco).
EX	12	Extensão de arquivo, normalmente parte em zero.
	13-14	Reservado para uso do sistema.
RC	15	Contagem de registro, ou tamanho de extensão corrente (0 a 128 registros).
DM	16-31	Mapa de alocação do disco preenchido e usado pelo CP/M.
NR	32	Número do registro seguinte a ler ou gravar (número do registro corrente no CP/M versão 2.2).
RR	33-35	Número de registro aleatório opcional (só CP/M versão 2.2 e MP/M) na faixa 0-65535.

Figura 5.7: *Informação a respeito das posições dos campos*

Resumimos, abaixo, as operações BDOS e seus números de função. (Novamente, entre em contado com a Digital Research para verificar se houve aperfeiçoamentos ou alterações.)

12. Lift Disk Head (versão 1.4). A função obtém o head do disco atual. Return version number (versão 2.2 do CP/M e MP/M).
    Isto devolve o número da versão do seu sistema CP/M para possibilitar uma versão de programação independente.
13. Reset disk system. Esta função inicializa o BDOS, reposiciona o estado leitura/gravação para todos os discos, seleciona o *drive* A e coloca o endereço default do DMA para *boot* + 0080H (usado pelos programas para mudar discos sem exigir um ↑C do terminal ou uma nova partida do sistema).
14. Select disk. O operador designa (1 para A, 2 para B, e 3 para C etc.) um *drive* do disco como sendo o atual para as operações subseqüentes nos arquivos.
15. Open file. Envie um endereço de bloco de controle de arquivo, e o BDOS irá encontrar um bloco de controle que se associa a este na área de diretório do disco, e voltará com o código adequado do diretório, indicando que a informação adequada foi copiada para o bloco de controle do arquivo. Isto permite acesso subseqüente ao arquivo.

16. Close file. Envie o endereço de um bloco de controle de arquivo, e o BDOS irá registrar a nova informação do diretório no bloco de controle do arquivo no disco (é o inverso da função open file, com os mesmos códigos devolvidos).
17. Search for file. Envie o endereço do bloco de controle de arquivo que contém um *filename,* e o BDOS irá procurar o primeiro *filename* que se associa a este e devolverá o endereço do bloco de controle de arquivo no disco que se associa com o que foi colocado pelo CCP (ou pelo programa).
18. Search for next occurrence. Utilize esta função *depois* da função 17 para achar a próxima ocorrência de uma associação para o *filename,* e o endereço do próximo bloco de controle do disco será devolvido.
19. Delete file. Envie o endereço do bloco de controle do arquivo que contém um *filename* (nome de arquivo), e o BDOS irá deletar este arquivo do disco.
20. Read sequentially. Se o arquivo tiver sido aberto ou ativado por uma função *make,* essa função irá ler os próximos 128 bytes (registro) na memória no endereço do DMA atual, e voltar com uma indicação de que a leitura teve êxito, um fim de arquivo, ou de dados não gravados durante um acesso randômico.
21. Write sequentially. Se o arquivo tiver sido aberto ou ativado por uma função *make,* essa função irá gravar 128 bytes, começando no endereço do DMA atual para o arquivo indicado pelo bloco de controle de arquivo. Essa função irá gravar por cima dos dados já existentes no arquivo (se houver algum).
22. Make file. Semelhante à função open file, esta cria um novo arquivo e o abre. Envie o endereço de um bloco de controle de arquivo com um novo *filename,* e essa função irá criar o arquivo e iniciar o seu bloco de controle de arquivo (na memória principal, assim como no disco) como um arquivo vazio. A pessoa deve assegurar-se de estar criando uma duplicata de um arquivo no mesmo disco, e tornando ambos acessíveis. Tente usar a função delete antes no novo *filename* (nome de arquivo).
23. Rename file. Envie o endereço do bloco de controle de arquivo e o BDOS irá dar um novo nome à área do *filename* no bloco e registrá-lo no disco.
24. Return log-in vector. Esta função determina quais os *drives* (guias) de disco que estão "em linha".
25. Return current disk (versão 2.2 e MP/M apenas). Esta função devolve um número correspondente à letra (A, B, C etc.) do *drive* do disco que está sendo selecionado.
26. Set DMA address. Esta função coloca o "Direct Memory Address" (endereço onde o ponteiro do arquivo parou depois de uma operação de leitura ou gravação) em outro valor com o objetivo de encontrar registros de dados em outro lugar da memória. Uma partida "a frio" ou "a quente", ou um "disk reset" irá colocar o DMA em *boot* + 0080H.
27. Get address of allocation vector. O sistema mantém um vetor de alocação na memória principal para cada drive "em linha". Esta função devolve o endereço do vetor para o *drive* atual. Vários programas (como STAT) utilizam esse vetor para determinar a quantidade de espaço de armazenamento disponível.
28. Write-protect disk. Esta função oferece proteção temporária contra a gravação. Qualquer tentativa de gravar no disco (sem uma partida "a frio" ou "a quente") irá gerar uma mensagem de erro.
29. Get read/only vector. Esta função devolve um vetor que indica quais os *drives* que estão protegidos contra gravação; isto é, quais os que possuem a atribuição "read/only".
30. Set file attributes. Esta função permite-lhe posicionar ou liberar indicadores vinculados a arquivos, que oferecem atributos "read/only" e de sistema.
31. Get disk parameter block address. Esta função devolve o endereço do bloco de parâmetro BIOS do disco, e é útil para calcular espaço e alterar valores de parâmetros do disco quando o conteúdo do disco se altera.
32. Get or set user code. Pode-se usar esta função para achar ou alterar o código do usuário atualmente ativo (as áreas do usuário estão na versão 2.2 do CP/M e no MP/M, mas não nas outras).
33. Read randomly. Esta função utiliza o campo RR do bloco de controle do arquivo para selecionar um número de registro e o ler. Retorna com o DMA apontando para o registro desejado, e o número do registro *não* avança como nas operações seqüenciais de leitura.
34. Write randomly. Esta função é iniciada da mesma maneira que uma operação de leitura randômica, exceto que grava os dados do DMA atual para o disco. Se o espaço do arquivo ainda não tiver sido alocado, a operação faz a alocação antes de gravar. O número do registro *não* avança.
35. Compute file size. Envie a esta função um endereço de bloco de controle de arquivo e ela devolverá o endereço do registro do "registro lógico" que segue o fim do arquivo (tamanho virtual do arquivo). A pessoa pode acrescentar dados a um arquivo existente utilizando esta informação para posicionar randomicamente o registro antes de executar uma série de operações de gravação randômica. O tamanho virtual corresponde ao tamanho físico se o arquivo tiver sido gravado seqüencialmente; em outro caso, o arquivo pode ter espaços não alocados em função das operações de gravação randômica.
36. Set random record position. Esta função devolve a posição randômica do registro depois de uma série de operações seqüenciais de leitura ou gravação. É útil para alternar operações seqüenciais e randômicas, ou para fazer um exame seqüencial inicial do arquivo antes de operações randômicas de leitura ou gravação.

INSTALANDO E ALTERANDO O CP/M

O CP/M é sempre elaborado para uma configuração específica de entrada/saída (e de memória). A Digital Research distribui uma forma do CP/M que funciona instantaneamente no sistema de desenvolvimento de microcomputadores Intel MDS-800. Outros vendedores de hardware e software fornecem versões do CP/M que funcionam em outros sistemas de hardware. Provavelmente, o leitor irá adquirir uma versão do CP/M que funciona automaticamente sem necessitar de alterações. Mas, se possuir alguma versão do CP/M e desejar adaptá-la a um novo hardware por causa de novos dispositivos de entrada e saída, a pessoa necessitará corrigir a parte BIOS do CP/M; caso possua um MP/M, necessitará corrigir as partes BIOS e XIOS. Corrigir o BIOS significa inserir as novas rotinas de entrada e saída exigidas pelos seus dipositivos específicos. Esta tarefa não é difícil, mas depende do dispositivo e da instalação. Por isso, não podemos apresentar aqui instruções específicas. Consulte a versão aplicável do *"CP/M Alteration Guide"* da Digital Research.

Se o leitor estiver criando uma nova versão do CP/M a partir da estaca zero, o problema se torna mais complexo. Se o seu sistema já possuir os elementos rudimentares para um desenvolvimento e execução de programas, pode escrever suas próprias rotinas (denominadas GETSYS e PUTSYS) para ler o "sistema" a partir de um disquete na memória do computador e gravar uma versão "corrigida" do sistema em um novo disquete a ser usado como disquete do sistema. Em outro caso, é preciso empregar outro sistema para gerar o novo disquete a ser usado com seu sistema.

Se a pessoa possuir uma versão CP/M que já esteja funcionando, é fácil elaborar programas em linguagem assembler para executar tarefas especiais; Pode-se também usar SYSGEN.COM e MOVCPM.COM para ajudar a fazer as alterações no CP/M, de maneira a não necessitar elaborar seus próprios programas GETSYS e PUTSYS. No entanto, a sua versão de SYSGEN.COM pode não funcionar com o seu tipo de disco ou disquete (minidisquetes, discos rígidos ou outros). Então vai ter que alterar primeiro o CP/M antes de poder usar o SYSGEN.COM.

A Digital Research fornece versões mínimas do GETSYS e PUTSYS (*"CP/M Alteration Guide" ou "MP/M User's Guide"*). Inicialmente, deve-se escrever um programa GETSYS para se ler as duas primeiras trilhas do disquete do sistema fornecido (as duas primeiras trilhas não são apresentadas por DIR, mas pertencem ao próprio sistema). Pode-se localizar a parte BIOS do sistema e modificá-la (a isto se chama corrigir ou remendar um programa). A pessoa pode salvar o sistema alterado no disquete escrevendo um programa PUTSYS. Finalmente, pode-se escrever uma versão do GETSYS, um programa "autocarregador", e colocá-lo na trilha 0, setor 1 utilizando o seu programa PUTSYS. Depois de testá-lo, a pessoa deverá estar com um sistema que funciona corretamente e que se inicializa automaticamente ao "dar uma partida 'a frio' no seu computador".

Se a pessoa estiver usando um sistema CP/M para alterar um sistema CP/M para um outro hardware, e os meios (os discos) forem compatíveis, então há meios mais rápidos para se criar o seu novo disquete do sistema: MOVCPM(MOVCPM.COM) e SYSGEN (SYSGEN.COM). Na Digital Research isto se denomina "uma regeração de sistema de segundo nível".

Pode-se combinar uma reconfiguração da memória (MOVCPM) e uma inicialização do disquete (PUTSYS) utilizando-se MOVCPM ao invés de GETSYS para ler, no sistema existente, e SYSGEN ao invés de PUTSYS para colocar a versão alterada no seu novo disquete do sistema.

MOVCPM é o comando transiente MOVCPM.COM, e a pessoa deve fornecer argumentos:

```
MOVCPM bb*
```

onde bb é o número de kilobytes (por exemplo, 32K, 64K, 20K) para a nova imagem de memória do sistema. A pessoa fornece um asterisco (*) para dizer a MOVCPM que deixe esta imagem de memória na memória (a TPA). Poderá também fornecer um asterisco para substituir bb, e o MOVCPM irá calcular a maior quantidade de memória que pode dedicar ao novo sistema. O comando MOVCPM será descrito com maiores detalhes na próxima parte desta seção:

Eis aqui um exemplo de uma operação MOVCPM:

```
A > MOVCPM 32* ↵
CONSTRUCTING 32K CP/M VERS x.x
READY FOR "SYSGEN" OR
"SAVE 34 CPM32.COM"
A >
```

Como o *display* sugere, o novo sistema de 32K está na TPA, pronto para a sua próxima operação, que deve ser um SYSGEN ou um SAVE. Já que se deseja alterar a parte BIOS do sistema, a pessoa também desejará salvar (SAVE) uma versão dele (denominando-a 'CPM32.COM' para um sistema de 32K, se o desejar). Uma vez salva (SAVED), pode-se carregá-la novamente na memória usando DDT (o programa depurador de CP/M) e alterar a parte BIOS.

Se o leitor planeja fazer alterações de maior importância (ou outras alterações em outra ocasião), será mais fácil criar o seu próprio programa autocarregador (a pessoa pode denominá-los CBIOS e BOOT se o desejar). A pessoa usaria então o programa ED (o programa editor do CP/M) para criar CBIOS.ASM e BOOT.ASM, e o programa ASM (o assembler do CP/M) ou outro assembler para criar CBIOS.HEX e BOOT.HEX, que poderiam ser carregados, utilizando LOAD para criar CBIOS.COM e BOOT.COM – programas que poderiam ser testados antes que a pessoa os intercalasse com o seu novo sistema.

Quando já tiver o novo sistema CPMbb.COM (onde bb é o tamanho de memória que se usou com MOVCPM) na memória empregando DDT, a pessoa pode intercalar CBIOS.COM e BOOT.COM com ele, ou alterar essas partes ao testá-las, e, finalmente, usar SYSGEN para colocar o sistema alterado nas primeiras duas trilhas do seu novo disquete do sistema. Aqui está um exemplo:

```
A > SYSGEN ↵
SYSGEN VERSION xx.xx
SOURCE DRIVE NAME (OR RETURN TO SKIP) ↵
```

(Responde com um RETURN para pular a operação read do SYSGEN, uma vez que já se tem o sistema alterado na memória).

```
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) B
```

(Assume que o novo disquete do sistema está no *drive* B).

```
DESTINATION ON DRIVE B, THEN TYPE RETURN ↵
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) ↵
A > PIP B:=A:*.*[V] ↵
• • •
```

(Copying messages)

```
• • •
A >
```

Se o leitor possuir uma cópia do CP/M com um arquivo .COM no disco, pode então usar um atalho:

```
A > SYSGEN CPM.COM ↵
    SYSGEN VERSION 2.2
    DESTINATION DRIVE NAME (OR RETURN TO REBOOT):
    (etc, . . .)
```

A pessoa deverá seguir as instruções fornecidas no *"CP/M Version 2.2 Alteration Guide"*, da Digital Research, para alterar o BIOS e criar os programas GETSYS, PUTSYS e BOOT.

RECONFIGURANDO (AJUSTANDO O TAMANHO DA MEMÓRIA) USANDO MOVCPM

Freqüentemente o tamanho de um sistema é aumentado ou reduzido (remoção de placas de memória). O CP/M deve ser alterado de acordo com isto. O programa MOVCPM (MOVCPM.COM) permite ao usuário reconfigurar de maneira simples o sistema CP/M para qualquer tamanho de memória.

O programa MOVCPM pode ser executado como se fosse um comando com argumentos opcionais:

MOVCPM { * / bb } *

O argumento opcional bb diz ao MOVCPM o quanto de memória ele deve gerenciar no novo sistema CP/M; se não for especificado bb, ou se especificar um asterisco (*) ao invés de bb, o MOVCPM irá configurar o novo sistema para gerenciar *toda* a memória RAM disponível do computador que a pessoa irá usar (RAM significa "random access memory"). Na maioria dos casos, a pessoa desejará que o CP/M gerencie (e tire proveito) de toda a memória RAM disponível.

O segundo asterisco (*) opcional, se a pessoa o fornecer, dirá a MOVCPM que mantenha o novo sistema configurado na memória (TPA) para se preparar para um SYSGEN ou um SAVE. Na maioria dos casos, a pessoa irá desejar reservar este novo sistema, ou gravá-lo em um disquete usando SYSGEN. Não fornecendo o segundo asterisco (*), o MOVCPM irá dar partida a esse novo sistema sem gravá-lo. Aqui estão alguns exemplos:

A > MOVCPM ↵ — Este comando reloca o sistema CP/M para tirar proveito de toda a memória RAM (começando em 100H, o início da TPA) no computador que será usado, e depois executa o sistema sem gravá-lo no disquete.

A > MOVCPM 32 ↵ — Este comando reloca o CP/M para gerenciar uma memória de 32K. e executa o sistema sem gravá-lo.

A > MOVCPM 32* ↵ — Este comando reloca o CP/M para gerenciar uma memória de 32K, e deixa a imagem de memória do sistema na memória (TPA) como preparação para um SYSGEN ou um SAVE.

A > MOVCPM ** ↵ — Este comando reloca o CP/M para gerenciar toda a memória RAM do computador que será usado (começando em 100H), e deixa a imagem de memória como preparação para um SYSGEN ou um SAVE.

As duas últimas formas empregam o segundo asterisco (*) para deixar a nova versão do sistema na memória de modo que a pessoa pode salvar (SAVE) o conteúdo da memória em um arquivo em disco ou SYSGEN o sistema na memória para gravar as duas primeiras trilhas do disquete (criar um disquete do sistema). Quando MOVCPM ** ↵ ou MOVCPM bb * ↵ é executado, a mensagem READY FOR SYSGEN ou SAVEbb CPMbb.COM é apresentada, depois de completar o programa MOVCPM (bb é o número de kilobytes).

A linha de comando fornecida pode conter um programa "menu" que oferece acesso a outros programas, encadeando-os, dependendo do programa que o usuário escolher a partir do "menu". Sempre que um programa termina, ou depois de cada partida "a frio" ou "a quente", o "menu" será novamente apresentado, e o usuário fará outra seleção. Ao usar esta versão do sistema, o usuário não poderá executar comandos CP/M, já que, depois de cada execução ou interrupção de um programa, o sistema executará automaticamente o programa "menu" mais uma vez.

Para que a pessoa possa inserir sua própria linha de comando para execução automática, deve pôr o sistema na memória usando MOVCPM ou SYSGEN, e salvá-lo. Uma vez que se tenha salvo (SAVEd) a imagem de memória do sistema, pode-se inserir a correção usando o DDT. A posição para a correção deverá começar no endereço 0980H; usando o comando D do DDT, a pessoa pode mostrar o conteúdo da memória nesta posição:

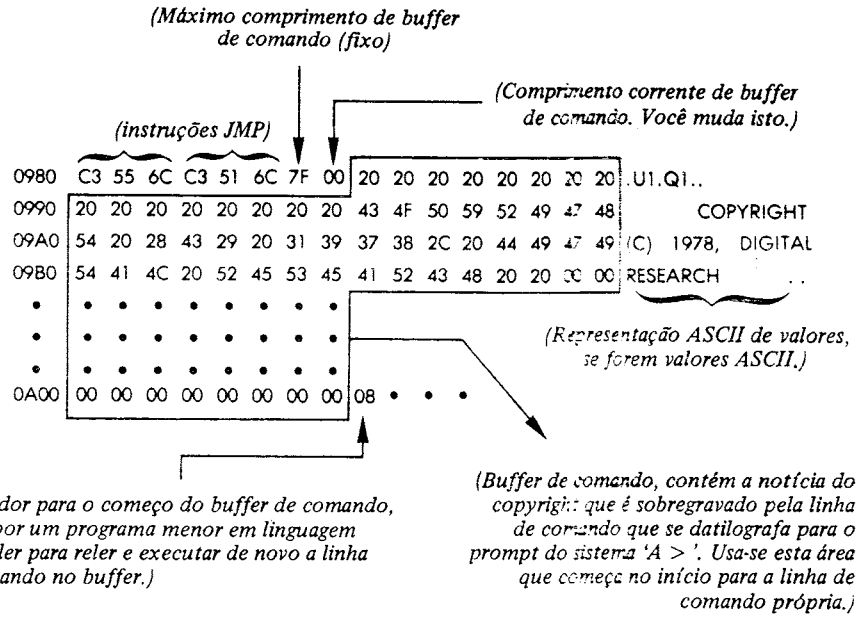


Figura 5.8: *Mostrando o conteúdo da memória*

UM EXEMPLO DE ALTERAÇÃO DO CP/M: UM SISTEMA DE "MENUS"

Este exemplo é fornecido para mostrar como usar os recursos do CP/M. Uma aplicação útil, tanto doméstica como comercial, é um sistema "turn-key" (ligue a chave) que

imediatamente começa a executar programas quando se dá uma partida "a frio" (por exemplo, depois de pressionar a tecla RETURN, o sistema automaticamente executa um programa que por sua vez está vinculado a outros programas para executar operações de contabilidade doméstica ou de controle). Outra ferramenta útil para ambientes domésticos ou comerciais é um *programa "menu"* (mencionado previamente), que dá ao usuário uma escolha de programas a executar; o programa *"menu"* seria executado automaticamente em um sistema "turn-key" para manter protegidos os programas do sistema.

Pode-se fornecer ambos, modificando posições especiais no sistema CP/M – as posições geralmente ocupadas pelo comando que o operador digita quando obtém o *prompt* do sistema. Estão na base do CCP (Console Command Processor), e alteram-se essas posições usando DDT (ou outro depurador, como o SID). Essencialmente, o operador insere uma linha de comando nessas posições e insere um flag para dizer ao CP/M que execute a linha de comando, ao invés de mostrar o *prompt* do sistema. Então, quando o usuário faz uma partida "a frio" ou "a quente" ("system reset" ou ↑C) a linha de comando inserida é executada automaticamente.

Os valores apresentados são hexadecimais. Cada caractere ASCII possui um valor hexadecimal – o 'C' de 'COPYRIGHT' tem o valor 43H ('H' representa hexadecimal), o 'O' de 'COPYRIGHT' tem o valor 4 FH e um espaço em branco tem o valor 20H (o "buffer" do comando se inicia com muitos espaços em branco). Estes valores podem ser encontrados na tabela ASCII.

Observe que o valor 00 para "tamanho atual do 'buffer' de comando" depois do valor 7F para "tamanho máximo do 'buffer' de comando", na linha de endereço 0980, diz ao CP/M que, nesse momento, não há comando no "buffer" de comando (em uma partida "a frio" ou "a quente"). O sistema portanto mostra o *prompt* do sistema, 'A >', e espera que o usuário digite um comando; essa linha de comando que o usuário digita é armazenada (depois do valor de comprimento atual) no "buffer" do comando, gravando em cima do que estivesse lá, em uma partida "a frio" (a observação a respeito do "copyright" ocupa o "buffer"). Isto é uma operação CP/M normal.

Para transformar o CP/M em um sistema "turn-key", o operador terá que induzir uma condição *anormal*: se alterar o valor do "tamanho atual do 'buffer' de comando" para um número diferente de zero, o sistema, depois de uma partida "a frio" ou "a quente", pensaria que *já* houve um comando no "buffer", e executaria esse comando. Depois de completar ou interromper a operação (ou após uma partida "a quente" ou "a frio"), o sistema voltará novamente a esse valor, e pensará que já existe um comando no "buffer". Em síntese, o sistema nunca chegaria a uma posição onde pudesse apresentar o *prompt* do sistema, 'A >'!

Naturalmente a pessoa terá que providenciar uma linha de comando para o "buffer" de comando, gravando sobre a observação a respeito do *copyright* (a pessoa também poderia movê-la para o fim do "buffer" de comando). Pode-se usar todas as posições até 0A07H, inclusive, mas não se pode, de modo nenhum, alterar posições acima desta. A posição 0A08H contém o ponteiro para o começo do "buffer" de comando, que o operador necessitará se escrever um programa *"menu"* em linguagem assembler.

Para inserir a linha de comando, digite o comando 'S' do DDT (como é descrito na documentação para o DDT fornecida pela Digital Research), utilizando valores hexadecimais para os caracteres ASCII da linha de comando. Essa linha de comando deve terminar com o valor 00, e o operador deve contar todos os caracteres (incluindo os espaços em branco) da sua linha de comando, sem incluir o 00 final, e colocar essa contagem de caracteres na posição para "tamanho atual do 'buffer' de comando", para que o CP/M saiba quão longa é a sua linha de comando.

Uma maneira fácil de implementar um sistema *"menu"* é usar BASIC. A maioria das linguagens BASIC tem a declaração "CHAIN program", para que um programa possa se vincular a outro, e o outro possa fazer o retorno para o programa *"menu"*. Esse programa *"menu"* pode ser escrito em BASIC, e o interpretador da linguagem BASIC tomaria conta do "buffer" do comando para executar programas (isto é, o aprendiz não necessitaria usar o ponteiro para o início do "buffer" de comando na localização 0A08H). CBASIC2 (SOFTWARE Systems) ou Microsoft BASIC (MBASIC.COM, da Microsoft Consumer Products) poderiam ser usados, já que ambos lhe permitem executar um programa BASIC como parte da linha de comando CP/M para executá-lo.

Por exemplo, o MICROSOFT BASIC é fornecido como MBASIC.COM, um comando que o aprendiz pode executar para "trazê-lo" (executar o interpretador) ou pode também executá-lo *com um argumento* que é o nome de um programa BASIC, como neste exemplo:

```
A > MBASIC PROG ↵
```

Neste exemplo, MBASIC é o programa interpretador BASIC que por sua vez executa o programa BASIC PROG.BAS ('BAS' é uma extensão esperada em *filenames* para programas Microsoft BASIC).

Se o aprendiz escrever um programa BASIC MENU.BAS, poderá usá-lo na seguinte linha de comando:

```
MBASIC MENU
```

O aprendiz pode inserir essa linha de comando no "buffer" do comando. O comprimento atual do "buffer" do comando seria 11, então a pessoa colocaria o valor 0B (hexadecimal) na posição "tamanho atual do 'buffer' de comando" e inseriria a linha de comando "MBASIC MENU" no "buffer" de comando, como mostramos na Figura 5.9.

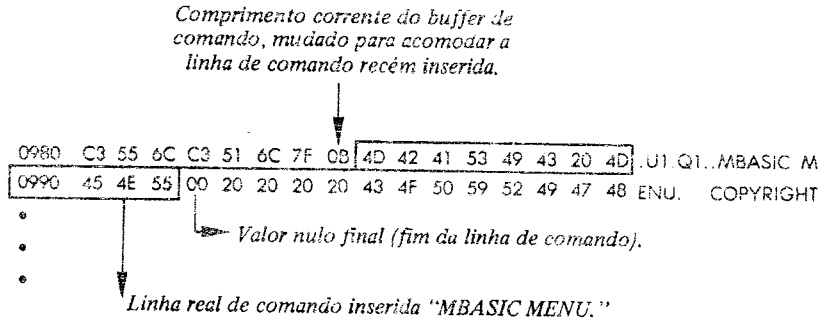


Figura 5.9: *A memória mostra a linha inserida*

Quando o aprendiz terminar de aplicar a correção acima, deve executar o comando G0 do DDT para terminar o DDT. Sem fazer nada que destrua o conteúdo da TPA, usa o comando SAVE para "salvar" a nova versão do seu sistema:

```
A > SAVE 34 AUTOCPM.COM ↵
```

(Denomine a sua nova versão de algum modo, por exemplo 'AUTOCPM.COM' para diferenciá-la das outras versões do seu sistema.)

Se estiver usando a versão 1.4 do CP/M, o aprendiz deverá, *em primeiro lugar*, executar a seguinte operação SYSGEN para colocar o seu sistema recém-alterado nas duas primeiras trilhas do seu novo disquete do sistema, e *depois* executar a operação SAVE acima (porquanto o comando SAVE da versão 1.4 destrói o conteúdo da TPA). Se estiver usando a versão 2.2 do CP/M, o aprendiz pode utilizar, sem susto, o comando SAVE em primeiro lugar (para salvar uma imagem de memória do seu novo sistema), e depois usar o SYSGEN para colocar o novo sistema nas duas primeiras trilhas do seu novo disquete do sistema.

```
A > SYSGEN ↵
SOURCE DRIVE NAME (OR RETURN TO SKIP) ↵
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) B
```

(Isto pressupõe que o seu novo disquete do sistema está no *drive* B).

```
DESTINATION IN DRIVE B, THEN TYPE RETURN ↵
FUNCTION COMPLETE
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) ↵
```

(Tecle RETURN para terminar o programa SYSGEN).

```
A >
```

Agora tudo o que o aprendiz necessita fazer é copiar os arquivos apropriados (Arquivos de comando CP/M, MBASIC.COM e MENU.BAS) para o novo disquete do sistema. Numa partida "a frio", o CP/M irá executar o MBASIC, e o MENU.BAS, ao invés de apresentar o *prompt* usual do sistema. O seu MENU.BAS deve ser capaz de providenciar acesso (por meio de declarações CHAIN ou SWAP) a outros programas BASIC.

Se quiser escrever o seu programa "*menu*" em linguagem assembler, o programa deverá ser suficientemente sofisticado para reconstruir outra linha de comando no "buffer" do comando para permitir o desvio a outros programas. Esse programa usaria o "ponteiro para início do 'buffer' de comando" na posição 0A08H para dizer ao sistema que volte e leia (execute) a linha de comando reconstruída.

MP/M

Instalando e Alterando o MP/M

Para instalar um sistema MP/M de múltiplos usuários, o aprendiz necessita de um sistema CP/M, pois o carregador MP/M (MPMLDR.COM) necessita ter uma versão do BIOS do CP/M incluída (LDRBIOS.COM). O MP/M pode ser executado ("trazido") a partir do CP/M, executando-se o MPMLDR.COM, que carrega o sistema MP/M (MPM.SYS) na memória a partir do disquete (similar a SYSGEN.COM, que carrega um sistema CP/M na memória). A partir do CP/M, o aprendiz precisa primeiro *gerar* o seu sistema MP/M utilizando o programa GENSYS.COM, que é fornecido com o MP/M, e rodará em CP/M.

O programa GENSYS faz diversas perguntas e utiliza a informação fornecida para construir o MPM.SYS. O programa MPMLDR, então, carrega o MPM.SYS na memória e o realoca automaticamente.

Para responder a todas as perguntas do GENSYS, a pessoa necessita saber que tipo de sistema deseja, se quer uma "memória agrupada" ou não, e que "processos residentes no sistema" vai desejar (esses processos estão descritos na seção seguinte). Eis aqui uma amostra de uma execução do GENSYS:

```
A > GENSYS ↵
MP/M 1.0 SYSTEM GENERATION
TOP PAGE OF MEMORY = 0 ↵
NUMBER OF CONSOLES = 2 ↵
BREAKPOINT RST # = 5 ↵
ALLOCATE USER STACKS FOR SYSTEM CALLS (Y/N) ?Y
MEMORY SEGMENT BASES, (FF TERMINATES LIST)
: 00,0 ↵
: 00,1 ↵
: 00,2 ↵
: FF ↵
Select Resident System Processes: (Y/N) Y
TIME        ?Y ↵
SCHED       ?Y ↵
ATTACH      ?Y ↵
SPOOL       ?Y ↵
MPMSTAT     ?Y ↵
A >
```

As respostas acima são descritas aqui:

Top page of memory (*Página superior de memória*): o aprendiz entra com o endereço hexadecimal da página superior de memória RAM do seu sistema. Se a entrada for zero, o carregador do MP/M determina o tamanho da memória em tempo de carregamento encontrando esta página superior na memória RAM.

Number of consoles (*Número de consoles*): o aprendiz entra com o número de consoles a serem conectados com o seu sistema; cada console usa 256 bytes de memória. A versão 1.0 do MP/M tem capacidade para até 16 consoles.

Breakpoint RST # (*Ponto de interrupção RST #*): o aprendiz entra com o número de reinício do ponto de interrupção a ser utilizado pelo DDT ou SID (programas de depuração). Não se permite uma nova partida em zero e nem os pontos de reinício usados pelo sistema MP/M. Consulte a documentação do MP/M fornecida pela Digital Research.

Allocate user stacks for system calls (*Aloque pilhas de usuário para chamadas do sistema*): Responda 'Y' (sim) se desejar utilizar os arquivos 'COM' do CP/M como comandos no seu sistema MP/M. O MP/M exige mais espaço de pilha do que o CP/M.

Memory segment bases (*Bancos de segmento de memória*): o aprendiz pode especificar de um a oito segmentos de memória do usuário com o mesmo espaço de endereço, mas números diferentes de bancos, como está descrito na documentação do MP/M. A primeira posição de memória a especificar deverá ser a sua primeira posição atual RAM (se o aprendiz tiver ROM começando em 0000H). O número do banco segue a posição, separado por uma vírgula. Essa lista termina com 'FF'.

Select resident system processes (*Selecione processos residentes do sistema*): Responda 'Y' (sim) para cada programa se desejar que sejam "residentes" ao invés de "relocáveis" ou "transientes". Processos residentes são programas que residem dentro do sistema operacional, e não são mostrados em um diretório (de forma similar aos comandos embutidos do CP/M).

Esta rotina parece complicada, e é! O MP/M é um conceito novinho em folha para microcomputadores – os sistemas compartilhados de múltiplos usuários ainda não proliferaram no mercado dos microcomputadores. O MP/M é complicado demais para operações normais em microcomputadores. Obviamente, este processo de geração será mais fácil de usar em novas versões do MP/M a serem ainda liberadas. (A maior parte desta informação pode ser achada em uma forma de fácil atualização do "*User's Guide to MP/M – Guia dos usuários do MP/M –* " da Digital Research).

Uma vez que se tenha gerado o MPM.SYS, o aprendiz pode usar o MPMLDR.COM para carregá-lo na memória e executá-lo. O MPMLDR.COM não exige respostas – execute-o simplesmente como um comando.

O MP/M foi elaborado para rodar num sistema de microcomputador Intel MDS-800, mas possui uma parte chamada XIOS (Extensão de BIOS) que a pessoa pode alterar para outros ambientes de hardware. Além de reescrever a parte XIOS, também é necessário alterar o programa MPMLDR.COM para carregar o MPM.SYS e executá-lo. Observe que o MPMLDR.COM usa o BIOS CP/M padrão (chamado LDRBIOS) para relocar e executar o sistema MP/M; portanto, a pessoa necessita pelo menos de uma versão do BIOS do CP/M (pode escrever a sua ou modificar uma existente). Usando o seu BIOS alterado (LDRBIOS), corrija a parte BIOS de MPMLDR.COM e recoloque-a no disquete usando SYSGEN.COM. (Se a pessoa não puder SYSGEN.COM, será necessário escrever programas GETSYS e PUTSYS, como estão descritos na documentação do MP/M e na seção anterior deste capítulo.) Para fazer esta correção, leia o MPMLDR.COM da memória usando o DDT ou o SID (programas de depuração), e/ou faça a alteração manualmente, ou faça a

fusão com o seu LDRBIOS.HEX. Quando terminar, salve (ou reserve) o conteúdo com o MPMLDR.COM e execute MPMLDR.COM para dar partida ao MP/M a partir de um CP/M que está sendo executado.

Para alterar a sua parte XIOS do MP/M, siga as instruções detalhadas na Documentação MP/M fornecida pela Digital Research. Use o programa GENMOD para produzir o arquivo XIOS.SPR (system page relocatable) a partir de dois arquivos HEX concatenados.

Operação do MP/M

O MP/M é composto de vários componentes: o XIOS (BIOS e IOS estendidos) para poder fazer uma "interface" com o hardware (o qual pode ser alterado), o BDOS e XDOS (operações básicas e estendidas de discos) para desempenhar operações com arquivos, e o CLI/TMP (Command Line Interpreter e Terminal Message Process) para lidar com a entrada e a saída do console.

O CP/M é um sistema "seqüencial", no qual um só programa é executado de cada vez. O MP/M, ao contrário, tem que acomodar vários programas que estão sendo executados "ao mesmo tempo", e compartilhando os mesmos recursos: a CPU (computador), os discos, os consoles e as impressoras de linha. O MP/M é um sistema "dirigido por prioridades", o que significa que o processo (programa sendo executado) que tiver a prioridade maior obtém a CPU. Um processo mantém um recurso até que seja terminado, que seja emitida uma chamada do sistema, interrompido, ou o relógio de tempo-real dê um sinal (opcional). A luta pelos recursos, que se segue então, é denominada "*escalonamento*".

O "*escalonador*" examina o "*descritor*" do processo para determinar a sua prioridade e para decidir se ele deve ser executado antes dos outros processos. O escalonador também utiliza o descritor do processo para armazenar informações temporárias a respeito do processo, e em que estado estava quando foi interrompido.

Quando todos os processos tiverem prioridades iguais, o escalonador executa-os na ordem "primeiro a entrar – primeiro a sair" ("first-in – first-out"). *Filas* são usadas para sincronizar processos fazendo com que um processo envie uma mensagem em uma determinada ocasião durante a sua execução, enquanto espera para receber a mensagem. O processo de espera é suspenso até que a mensagem chegue.

Uma *fila* é uma lista de espera, organizada como um arquivo especial que pode ser aberto e fechado e no qual a informação pode entrar seqüencialmente. Uma fila pode ser empregada para receber temporariamente informação para transferir para outro programa, que a grava no disco. As *filas* também são usadas para encaminhar arquivos para a impressora de linhas, e fornecer uso exclusivo de um recurso por um processo. Por exemplo, se um processo organiza uma fila que só contenha informação enquanto a impressora de linha não estiver ocupada e outro processo tenha que primeiro receber informação da fila antes de ter acesso à impressora de linhas, então esse mecanismo providencia uso exclusivo do recurso compartilhado. O segundo processo espera até receber a mensagem comunicando que a impressora de linhas está livre.

O MP/M oferece outro modo de sincronizar processos empregando *flags* posicionados por um processo e examinados por outro. Os flags fornecem um método de sincronização e interrupção de processos que é independente de dispositivos adicionais de interrupção de hardware e de mecanismos de interrupção de software.

Como opção, podemos usar um relógio de tempo-real (que pode ser reposicionado) para fornecer uma medida que precisa da hora do dia, e sincronização do sistema para es-

calonar processos, ou para a execução de programas em disco. Também é capaz de adiar a execução de um processo.

Não existem comandos "embutidos" no MP/M; todos os "comandos" são programas '.COM' transientes, programas '.PRL' (página relocável) ou processos residentes no sistema criados a partir de programas '.RSP' na geração do sistema (execução GENSYS). Os programas transientes exigem o uso da área "absoluta" da memória TPA, e também exigem espaço extra de pilha; a maioria dos programas devem ser feitos tipo relocável para que possam ocupar virtualmente *qualquer* espaço disponível na memória, e ainda assim serem executados adequadamente. A maioria desses programas utilizam "*macros*" que são muitas instruções englobadas em uma só, e o aprendiz necessitará do MAC (programa Macro-Assembler vendido separadamente pela Digital Research) para montar programas. O programa GENMOD, fornecido com o MP/M, transforma dois arquivos '.HEX' concatenados em um arquivo do tipo "página relocável" com uma extensão '.PRL'. Este programa '.PRL' será executado corretamente se você usar as declarações ORG de modo adequado, como estão descritas no "*MP/M User's Guide*" da Digital Research.

O GENMOD aceita um arquivo que contém dois arquivos '.HEX' concatenados separados um do outro por 100 bytes hexadecimais. O formato do comando GENMOD é:

```
GENMOD file.hex file.pr1$bbbb
```

O argumento file.hex deve ser uma *filename* incluindo a extensão '.HEX' de um arquivo com dois arquivos '.HEX' concatenados separados por 100 bytes. O argumento file.pr1 deve ser um *filename* incluindo a extensão '.PRL' para o novo programa do tipo página relocável. O argumento opcional $bbbb fornece memória adicional além do espaço explicitamente permitido pelo código, para fornecer espaço extra para "buffers". Se o seu programa necessitar dessa memória extra, forneça um cifrão ($) seguido de quatro dígitos hexadecimais. Aqui está um exemplo:

```
1A > GENMOD B:FINAL.HEX PERFORM.PRL $1000 ↵
```

Este comando converterá FINAL.HEX no *drive* B para PERFORM.PRL no *drive* atual, e fornece 1000H bytes de memória extra para o programa.

O aprendiz pode também criar o seu próprio processo residente no sistema usando GENMOD, substituindo um argumento file.rsp no lugar de file.pr1. Um processo residente no sistema começa como um programa '.RSP'; quando a pessoa gera o MP/M usando GENSYS, tem a opção de incorporar todos os arquivos '.RSP' dentro do sistema como processos residentes no sistema. Os arquivos '.RSP' fornecidos incluem MPMSTAT, SPOOL, hora do sistema (TOD) e escalonador (SCHED). O aprendiz pode criar o seu próprio arquivo '.RSP', desde que os torne "página relocável", os dois primeiros bytes do arquivo para o endereço de BDOS/XDOS estejam reservados e o descritor do processo esteja construído de acordo com as instruções fornecidas com a documentação do MP/M.

O MP/M lida com cada console, empregando um processo denominado TMP (Terminal Message Process). Quando se digita uma linha de comando, o TMP a envia ao CLI (Command Line Interpreter) onde ela é reconhecida e examinada. O CLI é, na realidade, um CCP (Console Command Processor – Processador de Comando de Console) mais avançado que o usado pelo CP/M. O CLI obtém a primeira palavra da linha de comando e tenta abrir uma fila com esse nome, assumindo, em primeiro lugar, que é apenas um pedido de colocar uma mensagem em uma fila (já que o CLI também lida com essa operação). Se

não houver nenhuma fila com esse nome, o CLI inicialmente procura um arquivo '.PRL' com esse nome; se houver uma fila com esse nome, o resto da linha de comando é copiado na fila como uma mensagem. Se o CLI encontrar um arquivo '.PRL' com esse nome, faz um pedido de memória relocável na qual possa carregar e executar o programa; depois, carrega e executa o mesmo. Se o CLI não encontrar um arquivo '.PRL', procura um arquivo '.COM'; se o encontrar, faz um pedido de memória absoluta TPA, carrega e executa o programa. Se o programa contiver especificações de arquivo ou for seguido por *filenames* na linha de comando, o CLI também cria blocos de controle de arquivo (como faz o CCP no sistema CP/M).

No console, o aprendiz pode "destacar" um programa que está sendo executado (isto é, a saída do programa não aparece no terminal, e este fica livre) para realizar outros programas usando D. Quando se pressiona novamente a tecla ↑D, o próximo processo esperando pelo console (o processo com a prioridade mais elevada) voltará. Pode-se, ainda, usar o programa ATTACH para vincular o console a um programa específico. Aqui temos um exemplo:

```
1A > ATTACH PROG1 ↵
                    (PROG1 assume o console)
```

Um programa só pode ser vinculado ao console do qual foi destacado. A pessoa pode usar o processo residente MPMSTAT (se o gerou com o seu sistema) ou o programa MPMSTAT.RSP (se redenominado MPMSTAT.PRL) para fornecer um *display* dos processos no sistema:

```
1A > MPMSTAT ↵

      * * * * * MPM 1.0 STATUS DISPLAY * * * * *

Ready Process(es):
MPMSTAT      cli      Idle
```

(Os processos prontos são aqueles que estão prontos para serem executados e estão esperando tempo da CPU. O primeiro tem a prioridade mais elevada, e está sendo executado nesse momento.)

```
Process(es) DQing:
[Sched]      Sched
[ATTACH]     ATTACH
[SPOOL]      Spool
```

(Estes processos aguardam mensagens das filas que estão entre colchetes. Estão organizados numa ordem de prioridade da mais elevada para a mais baixa.)

```
Process(es) NQing:
```

(Geralmente semelhante ao *display* acima, este mostra que não há processos, nesse momento, gravando em filas.)

```
Delayed Process(es):
```

(Não há processos atrasados nesse momento. Um processo atrasado é o que está esperando uma quantidade específica de sinais de relógio do *timer* do sistema.)

```
Polling Process(es):
    PIP
```

(O processo PIP está sondando o dispositivo do console.)

```
Swapped Process(es):
```

(A troca não é implementada na versão 1.0 do MP/M.)

```
Process(es) Flag Waiting:
01 - Tick
02 - Clock
```

(Estes são processos que posicionam e alteram *flags* para sincronizar outros processos.)

```
Flag(s) Set:
03
```

(O *flag* de "um intervalo de um minuto" está posicionado.)

```
Queue(s):
tod SCHED ATTACH STOPSPLR SPOOL MPMSTAT
Cliq Parseq   ListMQ   DiskMQ
```

(Estas são todas as filas no sistema. As filas em letras MAIÚSCULAS podem receber mensagens do CLI ou do console, por intermédio do CLI. Por exemplo, a fila SPOOL pode receber *filenames* digitando-se 'SPOOL', seguido de *filenames*, no console.)

```
Process(es) Attached to Consoles:
[0] - MPMSTAT
[1] - PIP
```

(Os processos vinculados a consoles são listados por número de console e pelo nome do processo.)

```
Process(es) Waiting for Consoles:
[0] - TMP0        DIR
[1] - TMP1
```

(Estes processos estão aguardando consoles dos quais foram destacados; por exemplo TMP0 está esperando o console 0, e DIR está esperando que TMP0 termine com o console 0. Considerando que TMP0 é o processo de mensagem do console, DIR irá esperar até que apareça um ↑D ou que ATTACH seja executado.)


```
Memory Allocation:
Base = 0000H    Size = 4000H    Allocated to PIP 1
Base = 4000H    Size = 2000H    *  Free  *
Base = 6000H    Size = 1100H    Allocated to DIR 0
```

(Esta tela mostra a base, tamanho e propriedade dos segmentos de memória, junto com o número do console do proprietário em colchetes. Segmentos não alocados estão livres para serem utilizados.)

RESUMO

Neste capítulo, explicamos as operações internas do CP/M e do MP/M. Os princípios destas operações não são complexos, e devem ser compreendidos para que se possa modificar o CP/M.

Agora aprendemos todos os recursos disponíveis dos sistemas CP/M e MP/M, mas isto não significa, automaticamente, que o usuário já possua um treinamento suficiente para usar o computador eficientemente e com êxito: é necessário praticar.

À medida que pratica, o aprendiz irá perceber o valor das recomendações apresentadas no capítulo seguinte.

- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Abortar um programa em processamento

(ABORT.COM ou ABORT.PRL)

FORMATOS:

1. ABORT programname
2. ABORT programname consolenumber

ARGUMENTOS:

programname O nome do programa em processamento a ser abortado.

console number Número do console a partir do qual o programa foi iniciado. Deve ser especificado se aquele console não é o mesmo em que o ABORT é especificado.

DESCRIÇÃO:

Este comando interrompe a execução do programa especificado. Deve ser utilizado cuidadosamente, já que qualquer usuário pode abortar qualquer programa iniciado em qualquer console.

COMO UTILIZÁ-LO:

Se o programa foi iniciado em seu console, simplesmente especifique o ABORT seguido pelo nome do programa. Se o programa foi iniciado em outro console, utilize o segundo formato e especifique o número do console.

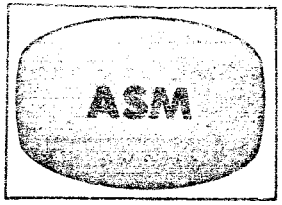
EXEMPLOS:

0A > ABORT COMPUTE ↵

(*COMPUTE foi iniciado neste console*).

2A > ABORT TEST 1 ↵

(*TEST foi iniciado no console 1*).



- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Montar um arquivo

(ASM.COM) fornecido com o CP/M (ou MP/M)

FORMATOS:

1. ASMfilename
2. ASMfilename.*shp*

ARGUMENTOS:

filename O nome de um arquivo-fonte '.ASM' (arquivo-texto) que contém instruções em linguagem assembler como textos ASCII. O ASM procura o 'filename.ASM'; a extensão não precisa ser especificada no *filename*.

.shp Os parâmetros opcionais para o ASM, que consistem de três letras precedidas por um ponto. O s deve ser a letra do *drive* (A, B, . . . P) contendo o arquivo-fonte '.ASM', se não estiver no *drive* atual. O h deve ser a letra do *drive* (A, B, . . . P) a receber o arquivo '.HEX' criado pelo ASM, ou 'Z', para dizer ao ASM que pule a função de criar o arquivo '.HEX' (descrita abaixo). O p deve ser a letra do *drive* (A, B, ... P) a receber o arquivo '.PRN' criado por ASM, ou 'X', para enviar o arquivo '.PRN' para o *display* do terminal, ou 'Z', para dizer a ASM que pule a função de criar o arquivo '.PRN' (descrita abaixo).

DESCRIÇÃO:

O programa de montagem (ASM.COM) transforma um arquivo-fonte de linguagem assembler (escrito em código 8080 ou Z-80) em um arquivo em código de máquina do tipo '.HEX' que pode ser posteriormente carregado (LOADed), utilizando-se o comando LOAD, no sistema, como um comando transiente (programa executável). O ASM também cria um arquivo de listagem com uma extensão '.PRN' que contém as linhas-fonte em linguagem assembler com *flags* de erros e notações hexadecimais (código de máquina).

COMO UTILIZÁ-LO:

Utilize o formato 1 se o arquivo-fonte '.ASM' estiver no disco atual e o usuário quiser criar arquivos '.HEX' e '.PRN' também no *drive* atual do disco. Caso contrário, deve usar o formato 2, e especificar explicitamente s como o *drive* para o arquivo-fonte, h como

Capítulo 6

Guia de Referências para Comandos e Programas em CP/M e MP/M

INTRODUÇÃO

Este capítulo é um rápido guia de referência para comandos CP/M e MP/M e programas utilitários (apresentados nos Capítulos 1 a 4). Este guia é organizado de tal forma que o usuário pode buscar a palavra-chave de um comando ou programa. Estas palavras-chave são mostradas em ordem alfabética.

Descreveremos, agora, o formato utilizado neste capítulo. Apresentamos a seguir um exemplo de um descritor que aparece depois de cada comando:

- • CP/M versão 1.4
- ○ CP/M versão 2.2
- ○ MP/M versão 1.0

Vendo o exemplo acima, o ponto ao lado do CP/M versão 1.4 indica-nos que o comando ou o programa se aplica àquela versão específica do CP/M. O pequeno círculo ao lado do CP/M versão 2.2 e do MP/M versão 1.0 indica que o exemplo não se aplica a elas. Vários comandos e programas se aplicam a todos os três sistemas.

Em seguida à descrição do objetivo de cada comando ou programa, encontramos entre parênteses a indicação de sua natureza, isto é, comando embutido, arquivo '.COM', arquivo '.PRL' ou processo residente (MP/M).

Em seguida, *mostra-se um* formato conciso que demonstra os possíveis *argumentos* utilizados quando o comando ou programa é executado. Este tipo indica que um argumento é essencial (por exemplo, *filename*), ao passo que este *TIPO* indica que um argumento é opcional (por exemplo, *FILENAME*). As chaves { } indicam uma escolha, onde pelo menos um dos argumentos é essencial (exceto se um dos argumentos for opcional dentro das chaves).

Em alguns casos, uma parte do argumento é opcional, ao passo que outra é essencial (por exemplo, *d filename*) (em que *d* é opcional e o *filename* é essencial). De qualquer forma, leia as descrições dos argumentos abaixo dos formatos.

No decorrer deste guia, fazemos algumas suposições. Por exemplo, assume-se que um argumento de um nome de arquivo, quer seja opcional ou essencial, pode conter uma letra opcional do *drive* (por exemplo, B:FILE), que especifica um *drive* de disco alternativo do atual. Isto sempre ocorre, exceto quando o argumento é definido de modo a excluir especificadores do *drive*. Nos casos em que os especificadores do *drive* são mostrados como parte do argumento, deve-se ler as instruções para aquele argumento em particular.

Uma outra suposição geral é que todos os arquivos '.COM' podem ser executados se estiverem em um disco (no *drive* do disco atual ou em outro alternativo), bem como todos os arquivos '.PRL'. Por exemplo, para executar SAMPLE.COM, o aprendiz digitaria 'SAMPLE ↵'. Se SAMPLE.COM existisse no *drive* B, e a pessoa estivesse no *drive* A, então poderia digitar 'B:SAMPLE ↵' para executá-lo.

Finalmente, tudo o que o aprendiz deveria digitar no sistema está sublinhado. O símbolo ↵ representa o retorno do cursor (RETURN ou tecla CR), e o símbolo ↑, combinado com uma letra, como por exemplo o C (isto é, ↑C) indica a tecla de controle (CTRL) usada simultaneamente com a tecla da letra.

o *drive* a receber o arquivo '.HEX' e p como o *drive* a receber o arquivo '.PRN'. Se o usuário quiser que o ASM monte o arquivo e crie apenas o arquivo '.PRN' (isto é, pular o arquivo '.HEX'), então especifique um 'Z' para h. Se quiser que o ASM crie apenas o arquivo '.HEX' quando ele montar o arquivo-fonte, então especifique um 'Z' para p (para pular o arquivo '.PRN'). Se desejar que o ASM envie o arquivo '.PRN' apenas para o *display* do terminal (e não reserve uma cópia no disco), então especifique um 'X' para p.

Em ambos os tipos de formato, o ASM traduz ("monta") as linhas-fonte em linguagem assembler para a notação hexadecimal Intel para denotar o código de máquina (código binário). Se o ASM encontrar erros no arquivo-fonte, ele indica no *display* a linha que contém o erro e um código de erro.

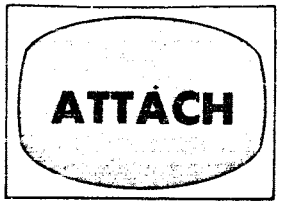
EXEMPLOS:

A > ASM PROG ↵

(Execute o ASM no arquivo PROG.ASM no drive atual.)

A > ASM DOTHIS.ABZ ↵

(Execute o ASM no arquivo DOTHIS.ASM no drive A; coloque o novo arquivo DOTHIS.HEX no drive B, e pule a criação de DOTHIS.PRN.)



- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Vincular um console a um programa que foi desvinculado
(Processo residente ou ATTACH.PRL)

FORMATO:

ATTACHprogname

ARGUMENTO:

progname O nome de arquivo (*filename*) de um programa que foi desvinculado do console que no momento está executando o ATTACH.

DESCRIÇÃO:

O programa ATTACH vincula um programa desvinculado ao console. O programa desvinculado deve ter sido desvinculado do mesmo console (terminal). Pode-se desvincular um programa de um console pressionando as teclas ↑D enquanto o programa estiver sendo executado. Um processo que está esperando pelo console é automaticamente vinculado quando se desvincula outro.

COMO UTILIZÁ-LO:

ATTACH pode ser executado como um comando, juntamente com o argumento progname, se o usuário dispõe do ATTACH como um processo residente no sistema gerado com o MP/M, ou se dispõe de ATTACH.PRL acessível em um disco.

EXEMPLO:

1A > ATTACH PROG1.PRL ↵

(PROG1.PRL se apodera do console.)

CONSOLE

- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Apresente o número do console (terminal) que está atualmente executando o comando CONSOLE.
(CONSOLE.COM ou CONSOLE.PRL)

FORMATO:

CONSOLE

DESCRIÇÃO:

Quando o comando CONSOLE é digitado em um terminal, ele retorna com o número do console do terminal que está sendo utilizado. Isto pode ser útil para determinar que console (terminal) possui programas desvinculados aguardando-o, como é mostrado em *display* MPMSTAT.

EXEMPLO:

```
0A > CONSOLE ↵
Console = 1
0A >
```

(O console usado no momento é o console 1, o segundo console depois do console 0.)

DDT

- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Depurar – Execute o programa depurador para carregar, alterar e testar programas.
(DDT.COM ou DDT.PRL)

FORMATO:

DDT *filename*

ARGUMENTO:

filename Argumento opcional para o nome de um arquivo '.COM' ou '.HEX'; deve-se também especificar a extensão.

DESCRIÇÃO:

O DDT substitui o CCP na memória e carrega o arquivo na TPA; se não houver especificação de nome de arquivo, o DDT ocupa a TPA e aguarda que um arquivo seja colocado na memória. O DDT também mostra o endereço NEXT (endereço em seguida ao último endereço no programa que está sendo depurado) e o PC (o contador do programa). O DDT possui seus próprios comandos para inserir valores, apresentar posições de memória, salvar comentários, estabelecer pontos de interrupção e outras funções de depuração.

COMO UTILIZÁ-LO:

Para executar o DDT, ele deve existir como programa acessível em um disco (isto é, como DDT.COM ou DDT.PRL em sistemas MP/M). Para terminar corretamente o DDT, use o comando GO.

EXEMPLO:

```
A > DDT PIP.COM ↵
```

OBSERVAÇÃO: Caso necessite de informações adicionais, consulte o "*CP/M Dynamic Debugging Tool (DDT) User's Guide*" (Guia do usuário do DDT do CP/M) fornecido pela Digital Research.

- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Diretório – Apresenta uma lista de nomes de arquivo no diretório do drive atual do disco.
(Comando embutido no CP/M, DIR.COM, ou DIR.PRL em MP/M)

FORMATO:

DIR { *filename* / *filematch* }

ARGUMENTOS:

filename Argumento opcional para indicar a DIR que localize apenas o arquivo indicado pelo *filename.*

filematch Substituição opcional para o nome do arquivo, com o intuito de indicar ao DIR que encontre diversos arquivos e faça uma listagem de seus nomes. Tanto o *filename* como o *filematch* podem ter especificadores de *drive.*

DESCRIÇÃO:

Se um *filename* ou um *filematch* não forem especificados, o DIR presume que o que se deseja é uma listagem de todos os nomes de arquivo encontrados no diretório atual do *drive* (somente arquivos com o atributo $DIR, e não arquivos com o atributo $SYS). Note-se, contudo, que o DIR somente mostra os arquivos na área atual do usuário, tanto na versão 2.2 do CP/M como na versão MP/M.

Se for fornecido um *filename,* o DIR mostra apenas esse arquivo, se estiver no *drive* atual do disco, e na área atual do usuário. Sendo fornecido um indicador de *drive* (isto é, A, B, C, . . . P), o DIR examina o *drive* especificado, na área atual do usuário.

Caso se forneça um *filename match* (*filematch*) em lugar de um *filename,* o DIR procura todos os arquivos que combinam com o *filematch* no *drive* atual do disco (ou no *drive* especificado) e na área atual do usuário.

COMO UTILIZÁ-LO:

Nas versões 1.4 e 2.2 do CP/M, o DIR é um comando embutido (isto é, faz parte do sistema operacional) e pode ser executado a partir de qualquer *drive* do disco e qualquer área do usuário. No MP/M. o DIR pode ser fornecido como DIR.COM ou DIR.PRL. Os DIR.COM e DIR.PRL precisam existir no *drive* atual, a não ser que se especifique outro *drive* como prefixo para DIR. Ele deve igualmente estar na área atual do usuário para poder ser executado dentro do sistema MP/M.

OBSERVAÇÃO: O *display* do DIR na versão 1.4 é apenas vertical, ao passo que na versão 2.2 (e MP/M) os *displays* dispõem de linhas horizontais e colunas verticais.

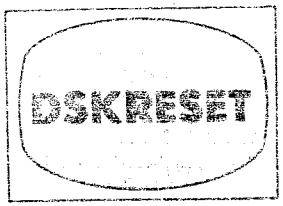
EXEMPLOS:

```
A> DIR ↵

ASM.COM     DUMP.COM     ED.COM       PIP.COM
LOAD.COM    PROG.HEX     BASIC.COM
STAT.COM    SAMPLE.TXT   FILE.TXT
TONE.TXT    SYSGEN.COM   32CPM.COM
GAME1.INT   GAME1.BAS    GAME2.INT
GAME2.BAS   SOURCE.BAS   TEST.SYS
•
•
•
```

(Display no CP/M versão 2.2 de todos os arquivos com o atributo $DIR no drive A, usuário 0.)

```
B> DIR *.TXT ↵

SAMPLE.TXT
PROG.TXT
POEM.TXT
NAME.TXT
BOOK.TXT
ONE.TXT
LETTER.TXT
FILE.TXT
```

(Display no CP/M versão 1.4 de todos os arquivos no drive B com extensões '.TXT'.)



- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Restaurar (mudar) o disco em um sistema de múltiplos-usuários.
(DSKRESET.COM ou DSKRESET.PRL)

FORMATO:

DSKRESET

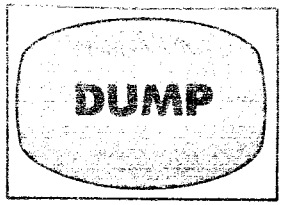
DESCRIÇÃO:

Quando o DSKRESET é executado, ele envia uma mensagem para os outros terminais conectados ao sistema: "Confirm reset disk system (Y/N)?" Se qualquer terminal responder com um 'N' (não), então o pedido de retirada do disco é negado. Se todos os terminais responderem com 'Y' (sim), o usuário pode mudar o disco (disquete). É importante que o usuário indique aos outros usuários que ele irá mudar um disco, ou um disquete, já que outros usuários podem ainda estar no processo de atualizar ou tentar obter acesso a arquivos nesse disco.

EXEMPLO:

0A > DSKRESET ↵

(Esta mensagem aparece em cada terminal conectado ao sistema.)



- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Descarrega o arquivo no terminal
Apresenta o conteúdo de um arquivo de disco sob forma hexadecimal.

FORMATO:

DUMP *filename*

ARGUMENTOS:

filename O nome (incluindo a extensão) de qualquer arquivo de disco.

DESCRIÇÃO:

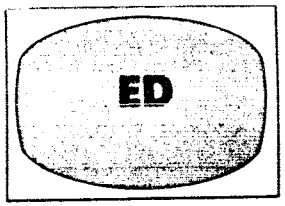
O DUMP apresenta, em notação hexadecimal, o conteúdo de qualquer arquivo de disco na tela do terminal, listando dezesseis bytes de cada vez, com o endereço absoluto do byte de cada linha à esquerda.

COMO UTILIZÁ-LO:

Execute o programa DUMP (DUMP.COM) como um comando, fornecendo um nome de arquivo com sua extensão. Se DUMP.COM não estiver no *drive* atual, especifique uma letra de *drive* para proceder o comando.

EXEMPLOS:

A > DUMP SCRATCH.HEX ↵
A > DUMP B:NONAME.COM ↵



- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Editar um arquivo
(ED.COM ou ED.PRL)

FORMATO:

ED *filename*

ARGUMENTO:

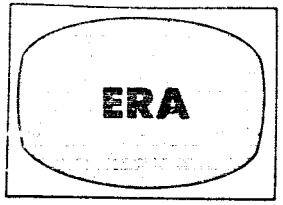
filename Nome de um arquivo a ser editado; é necessário que seja um arquivo-texto (ou outro tipo de arquivo ASCII). A pessoa deve também fornecer a extensão.

DESCRIÇÃO:

O programa ED cria um "buffer" de edição e permite ao usuário alterar o texto nesse "buffer". Inicialmente, o ED deleta qualquer arquivo '.BAK' que se associe com o nome primário do *filename* (por exemplo, SAMPLE.BAK para SAMPLE.TXT). A seguir permite ao usuário acrescentar texto ao "buffer" para modificar o texto original. O texto pode ser colocado em um arquivo temporário enquanto outro texto está sendo alterado no "buffer". Pode-se inserir texto de arquivos "biblioteca". Quando se finaliza a operação do ED com o comando E, o ED atualiza o arquivo-fonte e cria um arquivo de backup do arquivo-fonte original.

COMO UTILIZÁ-LO:

É necessário que ED.COM ou ED.PRL estejam em um disco (disquete) acessível. O Capítulo 4 descreve o emprego de ED. Nos apêndices D e E apresentamos um resumo adicional dos comandos ED.



- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Apaga um ou mais arquivos de um disco ou disquete
(Comando embutido no CP/M, ERA.COM ou ERA.PRL em MP/M)

FORMATO:

ERA { filename / filematch }

ARGUMENTOS:

filename É necessário dar ao ERA um argumento *filename* para indicar que ele deve apagar um determinado arquivo. Um *filename* deve incluir a extensão do arquivo, e também pode incluir um especificador do *drive.*

filematch Pode-se fornecer um *filename match* para um nome de arquivo (como descrevemo no Capítulo 2) para indicar ao ERA que apague vários arquivos de uma só vez. O *filematch* '*.*' com ERA irá apagar todos os arquivos no *drive* atual na versão 1.4, ou todos os arquivos no *drive* atual e na área atual do usuário, mas não em outras áreas do usuário, na versão 2.2 no MP/M.

DESCRIÇÃO:

O comando ERA apaga qualquer arquivo que é fornecido como um argumento de um nome de arquivo, a não ser que o arquivo seja "read-only" (tenha o atributo $R/0), ou que o disco atual (ou disco especificado) seja "read-only". Se o ERA não achar o arquivo, apresenta a mensagem "No file". Na versão 2.2 do CP/M e no MP/M, o ERA só apaga arquivos na área atual do usuário. A pessoa pode apagar arquivos em um *drive* alternativo, especificando o *drive* como parte do *filename* ou *filematch.* (Por exemplo 'ERA B:FILE1.*' apaga todas as ocorrências de FILE1 com quaisquer extensões existentes na área atual do usuário no *drive* B.)

COMO UTILIZÁ-LO:

O ERA é um comando embutido nas versões 1.4 e 2.2 do CP/M que se pode executar a partir de qualquer *drive* no MP/M. O ERA é fornecido como ERA.COM ou ERA.PRL

(arquivo de comando para memória absoluta ou arquivo recolocável para memória recolocável). O ERA.COM ou ERA.PRL deve existir no *drive* atual, ou pode ser utilizado a partir de outro *drive* usando-se um especificador (por exemplo, B:ERA).

Para apagar um disco completo na versão 1.4 do CP/M, apenas necessita-se utilizar o *filematch* "*.*" para se associar todos os arquivos no disco. Para apagar um disco completo na versão 2.2 do CP/M e no MP/M, o usuário se verá obrigado a escrever um programa que irá preencher o disco com dados absurdos, ou então será necessário usar-se a forma 'ERA*.*' em cada área do usuário, certificando-se de que não existem arquivos "ready-only" (com o atributo $R/O) que deixaram de ser suprimidos.

EXEMPLOS:

```
A > ERA SAMPLE.TXT ↵
```

(Este comando apaga o arquivo SAMPLE.TXT no drive A.)

```
A > ERA B:JUMP.TXT ↵
```

(Este comando apaga o arquivo JUMP.TXT no drive B.)

```
0A > ERA *.* ↵
```

(Este é um exemplo do MP/M, onde ERA apaga todos os arquivos na área 0 do usuário no drive A.)

```
A > ERA*.HEX ↵
```

(Este comando irá apagar todos os arquivos que possuem extensões 'HEX' no drive A.)

- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Apaga um ou mais arquivos de um disco ou disquete
(ERAQ.COM ERAQ.PRL em MP/M)

FORMATO:

ERAQ *filematch*

ARGUMENTO:

filematch Fornece-se um *filename match* para indicar a ERAQ que apague vários arquivos, um depois do outro.

DESCRIÇÃO:

O comando ERAQ apaga, seqüencialmente, todos os arquivos que se associam com o argumento *filematch*, a não ser que o arquivo seja "read-only" (tenha o atributo $R/O), ou se o disco especificado for "read-only". Diferentemente do ERA, o ERAQ solicita ao usuário que confirme antes de apagar cada arquivo sucessivo.

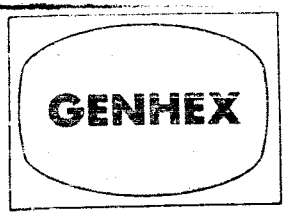
COMO UTILIZÁ-LO:

O ERAQ é fornecido como ERAQ.COM ou ERAQ.PRL no MP/M (arquivo-comando para memória relocável). O ERAQ.COM ou ERAQ.PRL deve existir no *drive* atual, ou ser chamada para outro *drive* utilizando um especificador (por exemplo, B:ERAQ).

Para apagar uma área do usuário, utilize o *filematch* '*.*' para combinar com todos os arquivos da área atual do usuário. Para apagar um disco completo, deve-se escrever um programa que preencha o disco com alguns dados, ou usar a forma 'ERAQ*.*' em cada área do usuário, certificando-se de que não deixaram de ser deletados arquivos com o atributo $ R/O ("read-only").

EXEMPLO:

```
0A > ERAQ PROG.* ↵
A: PROG TXT? Y
A: PROG INT? Y
```



- ○ CP/M versão 1.4
- ○ CP/M versão 2.2
- • MP/M versão 1.0

Transforma um arquivo COM em um arquivo HEX
(GENHEX.COM ou GENHEX.PRL)

FORMATO:

GENHEX programname.COM offset

ARGUMENTOS:

programname O nome do programa (deve ser do tipo *.COM*). Pode ser precedido de um especificador de disco.

offset Deslocamento para que o arquivo '.HEX' seja gerado (hexadecimal).

DESCRIÇÃO:

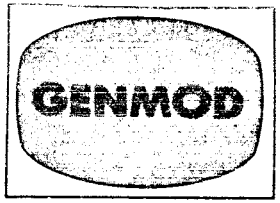
Este comando gera um arquivo do tipo 'HEX' a partir de um arquivo do tipo 'COM' e desloca o arquivo resultante em função de uma quantidade especificada.

COMO UTILIZÁ-LO:

Este comando costuma ser empregado para gerar um arquivo do tipo PRL, usando-se um comando GENMOD. Neste caso, o deslocamento é 0 ou 0100 bytes (hexadecimal).

EXEMPLO:

1A > GENHEX ACTION.COM 100 ↵



- ○ CP/M versão 1.4
- ○ CP/M versão 2.2
- • MP/M versão 1.0

Gera um programa modificado.
Cria um programa relocável a partir de dois arquivos '.HEX' concatenados.
(GENMOD.COM ou GENMOD.PRL)

FORMATO:

GENMOD file.hex file.prl *$bbbb*

ARGUMENTOS:

file.hex Este arquivo 'HEX' deve conter dois arquivos '.HEX' concatenados e separados um do outro por 100H bytes.

file.prl Este é o nome do arquivo '.PRL' a ser criado. Pode-se substituir a extensão por '.RSP' para criar um processo residente no sistema.

$bbbb Este é um número hexadecimal opcional de bytes para memória adicional, necessária para este programa.

DESCRIÇÃO:

O programa GENMOD produz um programa relocável com uma extensão '.PRL' (ou '.RSP') a partir de dois arquivos '.HEX' separados por 100H. Se *$bbbb* for fornecido, GENMOD também reparte a quantidade de memória adicional para o programa.

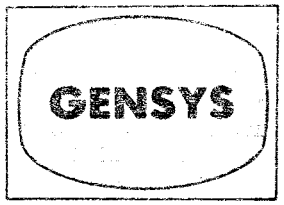
COMO UTILIZÁ-LO:

O GENMOD deve existir em um disquete como GENMOD.COM ou GENMOD.PRL. O GENMOD.COM também pode ser executado na versão 2.2. do CP/M, mas o arquivo '.PRL' só pode ser realizado no sistema MP/M.

EXEMPLO:

1A > GENMOD FINAL.HEX PERFORM.PRL *$1000* ↵

(Este comando produz o programa relocável PERFORM.PRL a partir do arquivo final. HEX (que é uma concatenação de dois arquivos '.HEX' separados por 100H) com uma memória adicional de 1000H.)



- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Gera um sistema MP/M a partir do CP/M
(GENSYS.COM)

FORMATO:

GENSYS

DESCRIÇÃO:

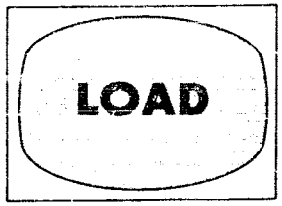
O programa GENSYS faz várias perguntas acerca do novo sistema, e depois constrói o arquivo MPM.SYS para conter o sistema. O programa MPRMLDR.COM carrega o sistema (MPM.SYS) na memória. O GENSYS também pode ser usado para gerar novas versões do sistema e permite ao usuário incorporar processos residentes no sistema. GENSYS, procura arquivos com a extensão '.RSP', e solicita ao usuário que escolha processos residentes a partir de uma listagem.

COMO UTILIZÁ-LO:

O GENSYS.COM pode ser executado a partir de um sistema CP/M ou MP/M. Responda às perguntas com um valor e um RETURN. Para uma descrição mais completa de cada pergunta, veja o Capítulo 5, "Instalando e alternando o MP/M".

EXEMPLO:

```
1A> GENSYS ↵

MP/M 1.0 System Generation

Top page of memory = C0 ↵
Number of consoles = 2 ↵
Breakpoint RST #   = 5 ↵
Allocate user stacks for system calls (Y/N) Y ↵

Memory segment bases, (ff terminates list)
  : 00 ↵
  : 40 ↵
  : 60 ↵
  : ff ↵

Select Resident System Process: (Y/N)
TIME    ?Y ↵
SCHED   ?N ↵
ATTACH  ?Y ↵
SPOOL   ?Y ↵
```



- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Carrega um arquivo na memória executável
Converte um arquivo '.HEX' em um arquivo de comando '.COM' executável
(LOAD.COM)

FORMATO:

LOAD *filename*

ARGUMENTO:

filename O nome do arquivo com uma extensão '.HEX': a extensão não precisa ser especificada.

DESCRIÇÃO:

O programa LOAD pega um programa que está no 'formato hexadecimal' Intel válido e o converte em um arquivo de comando que pode ser executado (arquivo com uma extensão '.COM'). O arquivo de comando se torna *filename*.COM (o arquivo hexadecimal é *filename*.HEX)

COMO UTILIZÁ-LO:

Para executar LOAD.COM, é necessário tê-lo em um disco acessível. Pode-se executar LOAD.COM a partir de um disco alternativo, especificando uma letra prefixo do *drive* antes do comando. Usa-se o arquivo '.COM' criado por LOAD como um comando transiente, a ser carregado e executado no TPA sempre que se digitar apenas o *filename* do *filename*.COM.

EXEMPLO:

A > LOAD SAMPLE ↵

(*SAMPLE.HEX é convertido em SAMPLE.COM.*)

A > SAMPLE ↵

(*Agora você pode executar SAMPLE.COM.*)

- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Reconfigura uma versão do CP/M para se adaptar a outras exigências de memória.
(MOVCPM.COM)

FORMATO:

MOVCPM { * / *bb* } *

ARGUMENTOS:

bb Tamanho opcional da memória, em dígitos decimais que representam o número de kilobytes (por exemplo, "32" para um sistema 32K). Se substituído por um asterisco (*) o MOVCPM irá calcular a quantidade total de RAM (memória de acesso randômico) do computador a ser usado, e construir um CP/M para esse tamanho.

* Um segundo asterisco (*) depois de ou bb ou do primeiro asterisco (*) indica ao MOVCPM que deixe o novo sistema na memória, preparando-se para uma operação SYSGEN ou SAVE. Este argumento também é opcional. Se não for fornecido, o MOVCPM executa o novo sistema sem gravá-lo no disquete (disco).

DESCRIÇÃO:

O programa MOVCPM cria uma imagem de memória do sistema e o reconfigura para combinar com um tamanho dado em bb ou o tamanho máximo do sistema principal. Se um segundo asterisco (*) não for fornecido, como em 'MOVCPM**', ou bb não for seguido de um asterisco como em 'MOVCPM 32*', o MOVCPM deixa o novo sistema adaptado na TPA para preparar-se para um novo SYSGEN ou um SAVE que grava a versão no disquete. Se a pessoa não fornecer o asterisco, então o novo sistema será executado, mas não gravado permanentemente.

COMO UTILIZÁ-LO:

Pode-se executar o MOVCPM.COM se ele existir em qualquer disquete ou disco acessível. É mais freqüentemente utilizado para preparar um novo sistema para as alterações para um outro hardware.

EXEMPLOS:

A > MOVCPM 48 ↵

(Este comando constrói uma versão 48K do CP/M e a executa sem armazená-la no disquete.)

A > MOVCPM 32* ↵

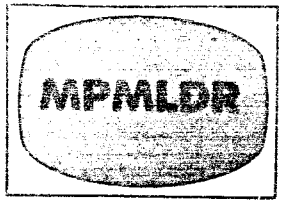
(Este comando cria um CP/M 32K e o deixa na memória, aguardando uma operação SYSGEN ou SAVE.)

READY FOR "SYSGEN" OR
"SAVE 32 CPM32.COM"

A > MOVCPM ** ↵

(Este comando constrói uma versão de memória máxima do CP/M e a deixa na memória, pronta para um SYSGEN ou SAVE).

OBSERVAÇÃO: Consulte o Capítulo 5, "Instalando e alternando o CP/M" se desejar informações adicionais.



- ○ CP/M versão 1.4
- ○ CP/M versão 2.2
- ● MP/M versão 1.0

Carregador MP/M
Carrega, recoloca e executa o sistema MP/M
(MPMLDR.COM)

FORMATO:

MPMLDR

DESCRIÇÃO:

O programa MPMLDR carrega o arquivo MPM.SYS que contém o sistema gerado, recoloca-o na memória, e depois executa-o para "trazer" o MP/M. O MPMLDR também oferece um *display* dos parâmetros do sistema – o número de console, o ponto de interrupção, o topo da memória, e uma tabela de segmentos de memória.

COMO UTILIZÁ-LO:

Pode-se executar o MPMLDR.COM a partir do MP/M ou CP/M. Pode-se também trazê-lo para o sistema a partir das duas primeiras trilhas de um disquete do sistema utilizando um programa carregador de partida "a frio". Para maiores informações, consulte o Capítulo 5, "Instalando e alterando o MP/M".

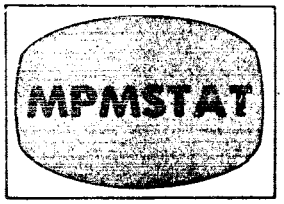
EXEMPLO:

A > MPMLDR ↵

MP/M 1.0 Loader

Number of consoles = 2
Breakpoint RST # = 5
Top of memory = C0FFH
Memory Segment Table:

SYSTEM DAT C000H 0100
⋮



- ○ CP/M versão 1.4
- ○ CP/M versão 2.2
- • MP/M versão 1.0

Apresenta o estado do sistema MP/M
(Processos residentes no MPMSTAT.PRL)

FORMATO:

MPMSTAT

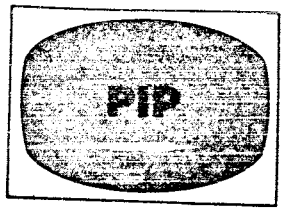
DESCRIÇÃO:

O MPMSTAT apresenta os nomes dos processos esperando por tempo de CPU, processos aguardando mensagens de filas e processos aguardando para enviar mensagens. Também apresenta processos retardados e de *polling,* sinalizadores aguardando, sinalizadores posicionados, filas em operação, processos aguardando consoles, processos vinculados a consoles, assim como a alocação de memória para todo o sistema.

EXEMPLO:

1A > MPMSTAT ↵

OBSERVAÇÃO: A saída mostra o estado dos vários processos. Veja a página 104 para um exemplo mais detalhado.



- • CP/M versão 1.4
- • CP/M versão 2.2
- • MP/M versão 1.0

***Peripheral Interchange Program* (*Programa de intercâmbio periférico*)**
Executa uma ou mais operações de cópia
(PIP.COM)

FORMATO:

1. PIP { d:newcopy / d: } =d:oldcopy[p]

2. PIP
*d:newcopy=d:oldcopy[p]
*...
*...
* ↵

3. PIP { dev: / d:filename } = { dev: / d:filename } [p], { dev: / d:filename } ,... [p]

4. PIP d:=d:filematch[p]

ARGUMENTOS:

{ d: newcopy / d: } O usuário deve escolher, nos formatos 1 e 2, se ele deseja que a nova cópia tenha o nome novo *newcopy,* ou se a nova cópia deve ter o mesmo nome mas estar em um *drive* diferente, o *drive* d:

d:oldcopy Tanto nas formas 1 como 2, o nome do arquivo a ser copiado é exigido, mas não o especificador do *drive.*

[p] Em todos os formatos, o usuário pode utilizar um parâmetro PIP depois do arquivo a ser afetado por esse parâmetro (opcional).

dev:
d:filename — No formato 3, o usuário deve escolher entre um *dev*: (nome do dispositivo, por exemplo, CON:) ou um *filename* com um especificador opcional do *drive* d. Esta forma pode ser utilizada para enviar um arquivo para um dispositivo, receber dados de um dispositivo e colocá-los no arquivo, ou enviar códigos especiais de dispositivo para um dispositivo.

d: = *d*:filematch — Tanto d: para destino, e *filematch* para fonte, são essenciais: o d: no *filematch* é opcional. O formato 4 é empregado para copiar vários arquivos em um outro disquete, utilizando os mesmos nomes para os arquivos.

DESCRIÇÃO:

Formato 1: Se for fornecido apenas d: e não *newcopy*, a nova cópia terá o mesmo nome que a cópia anterior, mas o d: à esquerda deve ser diferente do prefixo d: opcional à direita. Se d: for omitido, assume-se que *oldcopy* está no *drive* atual do disco. Se o usuário fornece *newcopy*, a nova cópia terá um novo nome, e o usuário pode copiar *oldcopy* para *newcopy* sem especificadores de *drive* (sem d: ou sem *d*:).

Formato 2: As expressões PIP seguem as mesmas regras; não obstante, se o usuário desejar executar várias operações, ele pode executar o PIP e deixá-lo na memória enquanto o usuário fornece expressões ao *prompt* do PIP (asterisco) (*). O PIP pode ser finalizado teclando-se RETURN. Com alguns parâmetros ocorrem ações diferentes quando o PIP é executado como um comando (formato 1) ao invés de um programa (formato 2).

Formato 3: Nas expressões ou comandos PIP, *dev*: (nomes de dispositivos) também podem ser usados, assim como d: *filename* (nomes de arquivo). O usuário não pode copiar a partir de um dispositivo que só recebe, e também não pode enviar a um dispositivo que só envia.

A parte esquerda da expressão é sempre o destino (ou seja, um dispositivo receptor ou um arquivo), e a parte direita sempre é a fonte (ou seja, o dispositivo que envia ou o arquivo a ser copiado). O usuário pode concatenar (juntar) arquivos-fonte em um arquivo ou dispositivo de destino. Também poderá usar os nomes especiais dos dispositivos listados no apêndice F.

Formato 4: O usuário pode copiar vários arquivos em outro disquete utilizando em *filematch* (*filename match*) com um especificador de *drive* opcional *d*:. Observe que o especificador do *drive*, à esquerda, é essencial (d:), e que as novas cópias têm o mesmo nome que as antigas, mas estão em outro disquete. O usuário deve lembrar-se de que não pode ter dois arquivos com o mesmo nome no mesmo disquete (na mesma área do usuário).

O apêndice F oferece uma listagem dos nomes dos dispositivos permitidos nas expressões PIP. As palavras-chave para executar funções especiais estão relacionadas no apêndice G.

COMO UTILIZÁ-LO:

Pode-se executar o PIP.COM a partir de qualquer *drive* alternativo especificando a letra do *drive* como um prefixo do comando (por exemplo, "B:PIP ↵"). Pode-se também executar o PIP, deixá-lo na memória enquanto se retira o disquete do sistema, depois de inserir um a ser copiado, e voltar para o sistema depois de reinserir o disquete do sistema e terminar o PIP, com um simples RETURN. O Capítulo 3 apresenta descrições completas de praticamente todas as aplicações do PIP.

EXEMPLOS:

```
A > PIP B:=*.* ↵
```

(Este comando copia todos os arquivos no drive atual para o drive B.)

```
A > PIP ↵
*FILE2 = TEST2 ↵
```

(Esta expressão copia TEST2 e dá à cópia o nome de FILE2.)

```
*LST:=FILE2 ↵
```

(Esta expressão envia uma cópia de FILE2 para o dispositivo LST:)

```
*PUN:=NUL:,PROG.ASM,EOF: ↵
```

(Esta expressão envia 40 caracteres nulos para o dispositivo PUN:, junto com o arquivo PROG.ASM, e o caractere de fim de arquivo.)

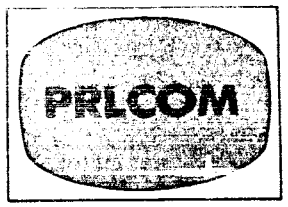
```
*B:=PROG.ASM ↵
```

(Cria uma cópia de PROG.ASM no drive B com o mesmo nome.)

```
A > PIP FILE2=FILE1[G2] ↵
```

(A cópia do arquivo FILE1 na área 2 do usuário para o arquivo FILE1 na área atual do usuário, no mesmo disco, só acontece na versão 2.2 do CP/M).

NOTA: Para informações adicionais veja o Capítulo 3 e os apêndices F (nomes dos dispositivos PIP), G (palavras-chave PIP) e H (parâmetros PIP).



- ○ CP/M versão 1.4
- ○ CP/M versão 2.2
- • MP/M versão 1.0

Transforma um arquivo PRL em um arquivo COM
(PRLCOM.COM ou PRLCOM.PRL)

FORMATO:

PRLCOM programname1. PRL programname2. COM

ARGUMENTOS:

programname1 O nome do programa-fonte (pode ser precedido de um indicador de unidade de disco)

programname2 O nome do programa destino (pode ser precedido de um indicador de unidade de disco)

DESCRIÇÃO:

Este comando transforma um programa do tipo PRL em um programa do tipo COM. Se o nome do programa COM resultante já tiver sido usado, o usuário é avisado e tem a opção de cancelar o comando.

COMO UTILIZÁ-LO:

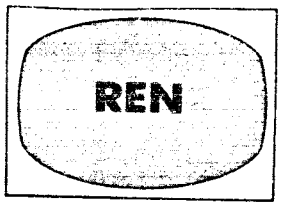
Sempre que o usuário quiser que um programa resida na TPA ao invés de num segmento de memória relocável, consegue-se isto com PRLCOM, transformando o programa em um arquivo COM ao invés de um arquivo PRL.

EXEMPLOS:

1A > PRLCOM DOIT.PRL DOIT.COM ↵

ou:

2A > PRLCOM DOIT.PRL B:NUNAME.COM ↵



- • CP/M versão 1.4
- • CP/M versão 2.2
- • MP/M versão 1.0

Dá um novo nome a um arquivo
(Comando embutido no CP/M, REN.COM ou REN.PRL em MP/M)

FORMATO:

REN *newname* = *oldname*

ARGUMENTOS:

newname e *oldname* Argumentos essenciais para nomes de arquivos incluindo extensões; não se permitem letras de *drive* como prefixos.

DESCRIÇÃO:

O comando REN (ou programa REN) altera *oldname* para *newname*; o usuário deve incluir o *filenamc* completo, com as extensões.

COMO UTILIZÁ-LO:

No sistema CP/M, REN é um comando embutido que pode ser executado em qualquer ocasião. No MP/M, tem-se que ter REN.CON ou REN.PRL em um disco acessível.

EXEMPLO:

A > REN NEWTRIC.HEX = OLDDOG.HEX ↵

(Este comando dá ao arquivo OLDDOG.HEX o novo nome NEWTRIC.HEX.)

- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

(Salva o conteúdo da memória em um arquivo de disco)
(Comando Embutido)

FORMATO:

SAVE p *filename*

ARGUMENTOS:

p O número de "páginas" (segmentos de 256 bytes) exigido para ser reservado, em decimal.

filename O nome (é essencial que seja definido) para o novo arquivo de disco, com extensão.

DESCRIÇÃO:

O comando SAVE salva o conteúdo da TPA começando na posição 100H até p páginas (segmentos de 256 bytes) em *filename*, que pode ser subseqüentemente depurado ou executado (se o conteúdo da TPA tiver sido um programa executável antes de ser reservado).

No MP/M, esta operação é implementada dentro dos programas de depuração revistos, DDT ou SID.

COMO UTILIZÁ-LO:

Para calcular p, necessita-se, em primeiro lugar, usar DDT para carregar o programa original na memória e usar o valor NEXT. O endereço NEXT será 1 a mais que o endereço final do programa; não obstante, para calcular o número de páginas, use este algoritmo simples:

Se os dois últimos dígitos de NEXT (hexadecimais) são 00, subtraia 1H (por exemplo, 1000H – 1H = 1BFFH). Se os dois últimos dígitos não forem 00, deixe o número como está. Agora, tome os dois primeiros dígitos, ou os "bits de ordem superior" (por exemplo, '1B do valor 1BFFH) e faça a conversão desse valor hexadecimal para um valor decimal, e assim obterá o número de páginas (p).

Como SAVE é um comando embutido, pode-se executá-lo quando se desejar.

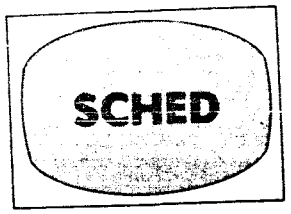
OBSERVAÇÃO: na versão 1.4, o usuário não pode executar dois SAVEs consecutivos no mesmo conteúdo da TPA, porque o primeiro SAVE ocasiona uma operação de diretório que altera várias áreas da TPA. Na versão 2.2, contudo, este problema foi resolvido – o usuário pode executar dois SAVEs consecutivos na mesma TPA.

EXEMPLO:

```
A > DDT SAMPLE.COM ↵
NEXT  PC
1D00  00
– G0 ↵
```

(O valor embaixo de NEXT é 1D00H. Subtraia 1H, para obter 1CFFH. Tome o número 1CH e o converta para a notação decimal para obter 28).

```
A > SAVE 28 COPY.COM ↵
```



- ○ CP/M versão 1.4
- ○ CP/M versão 2.2
- • MP/M versão 1.0

Escalona um programa para ser executado em uma data e hora posterior
(Processo residente ou SCHED.PRL)

FORMATO:

SCHEDmm/dd/yy hh:mm program

ARGUMENTOS:

mm/dd/yy O argumento necessário para fixar a data, onde mm representa o mês (1 a 12), dd é o dia (01 a 31) e yy representa os dois últimos algarismos do ano.

program O nome de arquivo para um arquivo com uma extensão '.COM' ou '.PRL', a extensão não precisa ser fornecida.

DESCRIÇÃO:

O programa SCHED, quando executado, permanece na memória aguardando a hora e a data que combinem com o que foi especificado no argumento. Ao encontrar essa hora e data específicas, SCHED automaticamente executa o programa que especificou.

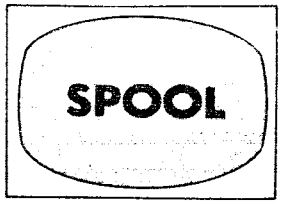
COMO UTILIZÁ-LO:

O programa SCHED pode existir como um arquivo '.PRL' ou como um processo residente no sistema, e a pessoa deve fornecer os argumentos adequados. Observe que qualquer pessoa pode alterar a hora e a data com uma operação de comando TOD, o que implica uma certa dose de espírito de cooperação para que se possa confiar no SCHED.

EXEMPLOS:

0A > SCHED 12/31/80 23:59 EIGHTY ↵

(Este comando escalona o programa EIGHTY.COM (ou EIGHTY.PRL) para execução no dia 31 de dezembro de 1980, às 11 horas e 59 minutos da noite, ou seja às 23 horas e 59 minutos.)



- ○ CP/M versão 1.4
- ○ CP/M versão 2.2
- • MP/M versão 1.0

Envia um ou mais arquivos para a fila "spool", geralmente para a impressora de linhas.
(Processo residente ou SPOOL.PRL)

FORMATO.

SPOOL *filename,* filename...

ARGUMENTOS:

filename e *filename* O primeiro *filename* é essencial, e os seguintes são nomes opcionais de arquivos a serem enviados para a fila do "spool". O usuário também deve especificar as extensões dos nomes de arquivo.

DESCRIÇÃO:

O comando SPOOL envia os arquivos um a um para a fila, onde esperam, em ordem, até que possam ser utilizados adequadamente pelo dispositivo de listagem LST: (geralmente a impressora de linhas, embora também possamos alocar outros dispositivos ao dispositivo LST: utilizando STAT). Os arquivos devem ser arquivos-texto ASCII (arquivos-fonte, arquivos editados, listagens etc.)

COMO UTILIZÁ-LOS:

O programa SPOOL deve ou ser um arquivo '.PRL' no *drive* atual do disco ou um processo residente no sistema, que se pode executar como um comando, fornecendo pelo menos um nome de arquivo. Use o comando STOPSPLR para cancelar a operação de formação de fila.

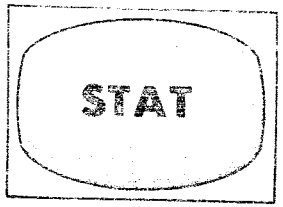
OBSERVAÇÃO: Se SPOOL.PRL for utilizado ao invés de SPOOL.RSP, o usuário poderá abortar a partir de outro terminal. Por exemplo, se SPOOL.PRL tiver sido ativado a partir do terminal 3, poderá ser interrompido digitando-se:

2B > STOPSPLR 3 ↵

EXEMPLO:

```
0A > SPOOL PROG.PRN,SAMPLE.TXT,NOVEL.JNC /
```

(Este comando envia PROG.PRN.SAMPLE.TXT e NOVEL.JNC para o dispositivo LST:, que geralmente é uma impressora de linhas. Os arquivos esperam na fila até serem manipulados pelo dispositivo.)



- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Apresenta informação a respeito do estado, e aloca dispositivos.
(STAT.COM ou STAT.PRL)

FORMATOS:

1. STAT { DEV: / VAL: }

2. STAT gen:=dev:,gen:=dev:,...

3. STAT *d*:=R/O

4. STAT *d*: { filename / filematch }

5. STAT *d*: { filename / filematch } { $S / $R/O / $R/W / $SYS / $DIR }

6. STAT *d*: { DSK: / USR: }

ARGUMENTOS:

DEV: / VAL: — No formato 1, DEV: produz o *display* das atribuições atuais dos dispositivos, ao passo que VAL: produz o *display* das atribuições potenciais (na versão 2.2. do CP/M, VAL: também faz uma listagem dos comandos possíveis do STAT).

gen:=dev; No formato 2, *gen*: representa um dispositivo genérico (CON:, PUN:, RDR:, ou LST:), ao passo que *dev*: representa qualquer dispositivo físico que é apropriado para uma atribuição de dispositivo a um dispositivo genérico.

d:=R/O No formato 3, d:=R/O (onde d: é uma letra do *drive*) faz com que o *drive* d: adquira o estado "read-only"; somente com d: O STAT mostra o estado do *drive*; sem argumentos, o STAT mostra o estado do *drive* atual (isto é, "ready-only" ou "read-write").

d: { filename / filematch } No formato 4, o usuário pode fornecer d: *filename* (d: é opcional) para apresentar o estado (tamanho em registros e bytes, número de domínio etc.) de um arquivo especificado, ou o usuário pode fornecer d:*filematch* para apresentar vários arquivos ao mesmo tempo.

$S
$R/O
$SYS
$R/W
$DIR

Ao usar os argumentos para o formato 4, o formato 5 permite aos usuários da versão 2.2 do CP/M e de MP/M empregar o parâmetro $S para apresentar mais informações a respeito do tamanho para um arquivo ou grupo de arquivos. e utilizar os parâmetros $R/O $R/W, $SYS e $DIR para fixar atributos dos arquivos. O atributo $R/O evita sobreposição de gravações ou deleções de um arquivo; ele é cancelado pelo atributo $R/W. O atributo $SYS (sistema) "esconde" o arquivo do comando DIR; ele é cancelado pelo atributo $DIR (diretório).

d:DSK:
USR:

Na versão 2.2 do CP/M e no MP/M, o formato 6 apresenta as características do disco dom DSK: (o disco atual ou o d alternativo opcional, o *drive*). Para apresentar as áreas atuais e ativas do usuário, use USR:.

DESCRIÇÃO:

O STAT fornece informações estatísticas a respeito de arquivos e discos (disquetes) e atribui dispositivos físicos e nomes genéricos de dispositivos (a serem usados com o PIP). O STAT também torna um disco "read-only" (formato 3), um arquivo "read-only" (formato 5), na versão 2.2 do CP/M e no MP/M, e apresenta a área atual do usuário, assim como as áreas ativas dos usuários.

COMO UTILIZÁ-LO:

Para apresentar informações estatísticas, simplesmente execute o STAT.COM (ou STAT.PRL) em um dos seus vários formatos. Para atribuir dispositivos, deve-se usar o formato 2. Os nomes genéricos dos dispositivos são CON: (dispositivo de console), RDR: (dispositivo de leitura), PUN: (dispositivo de perfuração), e LST: (dispositivo de listagem), ao passo que os dispositivos físicos foram listados com o PIP no Capítulo 3.

EXEMPLOS:

Usando a versão 2.2. do CP/M

```
A> STAT PIP.COM $S
Size      Recs      Bytes     Ext     Acc
  55        55        12K       1     R/O     A:PIP.COM
A> STAT SAMPLE.COM $R/O
A> STAT B:
BYTES REMAINING ON B:  192K
B:      R/O
A>
```

- ○ CP/M versão 1.4
- ○ CP/M versão 2.2
- • MP/M versão 1.0

Cancela uma operação de fila "spool" e esvazia essa fila
(Processo residente ou STOPSPLR.PRL)

FORMATO:

STOPSPLR

DESCRIÇÃO:

O comando STOPSPLR interrompe SPOOL em execução e esvazia a fila do "spool". (Veja o comando SPOOL.)

EXEMPLO:

0A > STOPSPLR /

- • CP/M versão 1.4
- • CP/M versão 2.2
- • MP/M versão 1.0

Executa um lote de comando
(SUBMIT.COM ou SUBMIT.PRL)

FORMATO:

SUBMIT *filename v1 v2 v3.*

ARGUMENTOS:

filename O nome necessário de um arquivo-texto com linhas de comando, que deve ter uma extensão '.SUB'. '.SUB' não é fornecido no argumento *filename.*

v1 v2 v3 Os valores opcionais a serem conectados às variáveis no arquivo submit. As variáveis assumem a forma $1, $2, $3 etc., e v1 representa $1, v2 representa $2 etc.

DESCRIÇÃO:

O programa SUBMITaceita o arquivo *filename.*SUB e constrói o arquivo $$$.SUB, que é executado depois de uma partida "a quente" (depois que termina o programa SUBMIT). As linhas de comando do arquivo $$$.SUB são executadas até esgotar o arquivo. Para construir o arquivo $$$.SUB, SUBMIT substitui v1 no lugar de 1$ no arquivo '.SUB', v2 no lugar de $2 etc. Arquivos submetidos só podem aceitar operações se estiverem no *drive* A.

COMO UTILIZÁ-LO:

Crie um arquivo '.SUB', usando o ED, que contenha linhas de comando com argumentos expressos como variáveis ($1, $2 etc.). Realize o lote de comandos executando o SUBMIT.COM no arquivo '.SUB'.

EXEMPLO:

Vamos supor que o arquivo SMALL.SUB contenha as seguintes linhas de texto:

```
DIR $1.*
PIP $2: = $1.BAK
ERA$1.BAK
```

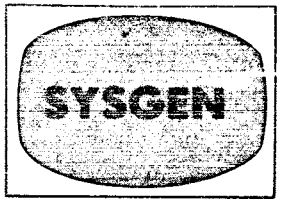
Se o usuário submetesse este arquivo utilizando o comando SUBMIT:

```
A > SUBMIT SMALL PROG B ↵
```

obteria as seguintes linhas de comando em $$$SUB:

```
DIR PROG.*
PIP B:=PROG.BAK
ERA PROG.BAK
```

Quando SUBMIT terminasse de substituir para construir $$$SUB, o sistema executaria os conteúdos de $$$.SUB.



- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

(Gera uma cópia do CP/M)

Traz o sistema para dentro da memória e/ou faz uma cópia do disquete do sistema.

FORMATO:

```
SYSGEN
```

DESCRIÇÃO:

O programa SYSGEN inicializa um disquete do sistema (grava o sistema nas primeiras duas trilhas do disquete). O SYSGEN também traz o sistema para a memória e o executa.

COMO UTILIZÁ-LO:

Execute o programa SYSGEN.COM:

```
A > SYSGEN ↵
SYSGEN VERSION xx.xx
SOURCE DRIVE NAME (OR RETURN TO SKIP) A
```

(Responda com a letra do drive em que o sistema está localizado, a não ser que deseje pular a operação de leitura do sistema se este já estiver na memória em virtude de uma operação MOVCPM)

```
SOURCE ON A, THEN TYPE RETURN ↵
FUNCTION COMPLETE
```

(A operação de leitura do sistema foi completada. O sistema agora se encontra na memória principal.)

```
DESTINATION DRIVE NAME (OR RETURN TO REBOOT) B
```

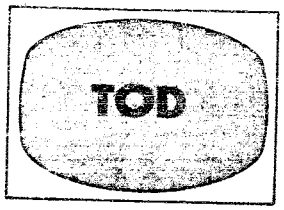
Responda com a letra do drive que contém o novo disquete do sistema a ser inicializado, ou tecle RETURN para executar o sistema na memória.

DESTINATION ON B, THEN TYPE RETURN ↵
FUNCTION COMPLETE

(O sistema agora está gravado no novo disquete que agora pode ser empregado, como disquete do sistema.)

DESTINATION DRIVE NAME (OR RETURN TO REBOOT) ↵
A>

(Tecle RETURN para finalizar SYSGEN.)



- ○ CP/M versão 1.4
- ○ CP/M versão 2.2
- ● MP/M versão 1.0

Hora do dia
Apresenta ou fixa a hora e a data
(Processo residente ou TOD.PRL)

FORMATO:

TOD *mm/dd/yy hh:mm:ss*

ARGUMENTOS:

mm/dd/yy Isto só é necessário quando se fixa a data, mm representando o mês, dd o dia, e yy se refere aos dois últimos algarismos do ano.

hh:mm:ss Isto é essencial quando se fixa a hora ou a data, onde hh representa a hora (0 a 24), mm o minuto, (00 a 59) e ss o segundo (00 a 59).

DESCRIÇÃO:

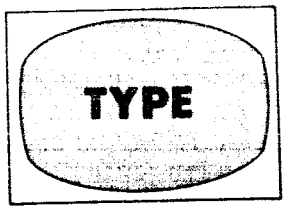
Em um sistema MP/M, o usuário pode apresentar a hora e a data (incluindo o dia) executando TOD sem quaisquer argumentos. Se a data e a hora são fornecidos como argumentos, TOD irá enviar uma mensagem *prompt* ao usuário e este poderá então pressionar qualquer tecla quando estiver pronto a fazê-lo.

COMO UTILIZÁ-LO:

O programa TOD pode existir em seu *drive* atual de disco como um arquivo com uma extensão '.PRL', ou pode ser um processo residente no sistema (comando) carregado com os outros usuários num sistema de múltiplos usuários, já que o computador só conhece a hora como o operador a determina, e outros usuários já poderiam ter escalonado programas para serem processados usando SCHED.

EXEMPLOS:

0A > TOD ↵
Sat 12/29/79 02:37:21
0A > TOD 12/29/79 02:38:00 ↵
Strike a key to set time
Sat 12/29/79 02:38:00



- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Mostra o conteúdo de um arquivo na tela do console (terminal).
(Comando embutido no CP/M, TYPE.COM ou TYPE.PRL em MP/M)

FORMATO:

TYPE { filename / filematch }

ARGUMENTOS:

filename O usuário deve fornecer um *filename* específico com sua extensão, ou um
filematch *filename match* (*filematch*) para mostrar vários arquivos.

DESCRIÇÃO:

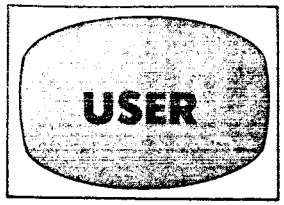
O comando TYPE mostra o conteúdo de qualquer arquivo, mas o usuário só será capaz de ler o conteúdo de um arquivo-texto ASCII (arquivo-fonte, arquivo '.PRN', ou arquivo de listagem).

COMO UTILIZÁ-LO:

No CP/M, TYPE é um comando que o operador pode executar em qualquer ocasião. No MP/M é fornecido como TYPE.COM (comando transiente) ou como TYPE.PRL (programa relocável).

EXEMPLOS:

```
A > TYPE SMALL.SUB ↵
A > TYPE *.TXT ↵
```

- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Altera a área atual do usuário, ou apresenta a área do usuário em um sistema MP/M
(USER.COM ou USER.PRL)

FORMATO:

1. USER *n* *(somente em MP/M)*

2. USER *n*

ARGUMENTOS:

n Argumento essencial na versão 2.2 do CP/M e argumento opcional no MP/M, que representa o número da área do usuário (zero a quinze).

DESCRIÇÃO:

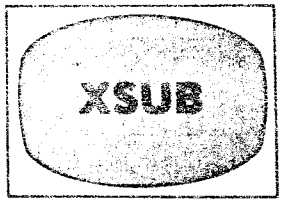
O formato 1 apresenta o número do usuário atual se não for fornecido *n*; se *n* for fornecido, USER altera a área do usuário para a área *n* do usuário. O formato 2 só permite alterar (e não apresenta) a área do usuário. O formato 2 altera a área do usuário para a área do usuário representada por n.

COMO UTILIZÁ-LO:

O programa USER é fornecido ou como USER.COM ou USER.PRL.

EXEMPLO:

```
A > USER 3 ↵
A >
```



- CP/M versão 1.4
- CP/M versão 2.2
- MP/M versão 1.0

Uma facilidade SUBMIT ampliada para fornecer entradas para programas executadas no arquivo submit.
(XSUB.COM)

FORMATO:

XSUB

DESCRIÇÃO:

Quando XSUB é colocado no início de um arquivo '.SUB' (ou quando XSUB é executado como um comando), ele reloca para a área diretamente abaixo do CCP para processar as linhas de comando no arquivo '.SUB', e desta forma fornece entradas do "buffer" de console aos programas executados dentro da operação submit. Os programas que lêem entradas do "buffer" do console obtêm a sua entrada diretamente do arquivo '.SUB'.

COMO UTILIZÁ-LO:

Insira 'XSUB' como primeira linha de comando do arquivo '.SUB' e submeta o arquivo ao programa SUBMIT. O XSUB permanece ativo até a próxima partida 'a frio", e o arquivo '.SUB' é processado até que se esgote.

EXEMPLO:

file NEW.SUB:
```
XSUB
DDT
I$1.HEX
R
G0
SAVE 1 $2.COM
```

A> SUBMIT NEW THIS THAT ↵

('THIS' fica no lugar de $1, e 'THAT' fica no lugar de $2; o programa XSUB fornece o DDT com as linhas de comando DDT 'ITHIS.HEX', 'R' e 'G0', e o XSUB fornece a CCP "SAVE 1 THAT.COM". XSUB permanece ativo até uma partida "a frio".)

Dicas Práticas

INTRODUÇÃO:

Agora que o leitor adquiriu os conhecimentos básicos necessários para utilizar o CP/M e seus recursos, apresentamos algumas "dicas práticas" para que possa vir a usar o sistema ainda mais eficientemente. Este capítulo sugere formas de se evitar alguns dos problemas que podem ocorrer na utilização de um sistema CP/M. Oferecemos uma grande variedade de recomendações e soluções.

Ao usar um sistema de computador, evitar problemas é de importância primordial. Equívocos muito simples, óbvios, que o usuário comete, podem causar erros graves, muito sérios. Portanto, é essencial muita disciplina por parte do usuário.

DISCIPLINA DO USUÁRIO

Faça sempre uma cópia de qualquer programa ou disquete novo que esteja usando pela primeira vez, para ter um backup no caso de dano acidental ao original. Ao inserir um novo disquete no qual deseja gravar, execute um CTRL-C para registrá-lo no sistema. De vez em quando, execute um comando STAT ou DIR para verificar o seu diretório e o espaço que ainda resta no disquete.

No final de uma sessão, onde um arquivo muito extenso foi manipulado, recomenda-se que o usuário crie uma cópia desse arquivo em um novo disquete. Os setores serão copiados seqüencialmente, e a nova cópia será carregada muito mais depressa do que o arquivo original por qualquer programa de aplicação. O arquivo original deve ser conservado como backup.

O ambiente do computador deve ser mantido organizado para encorajar disciplina por parte do usuário (o apêndice deste livro apresenta uma lista de materiais essenciais). Deve-se ter sempre à mão a documentação do usuário, disquetes virgens, e programas dos usuários. Deve-se também registrar em local acessível o procedimento para dar partida ao sistema, além das precauções a serem obedecidas.

MANIPULANDO DISQUETES

Respeite a integridade magnética e física dos disquetes. Ao manipulá-los:

- Não coloque nenhum objeto magnético perto ou em contato com um disquete. Exemplos de objetos magnéticos incluem transformadores (telefones), chaves de fenda (a maioria delas fica magnetizada depois de algum tempo) e outros objetos metálicos que possam se tornar magnetizados.

- Armazene sempre os disquetes em suas capas protetoras. Não os exponha à contaminação pela poeira, fumaça de cigarros ou outras partículas. Nunca se esqueça, especialmente, de *não* tocar, arranhar ou tentar limpar a superfície do disco.
- Rotule sempre adequadamente os discos, indicando a data e o seu conteúdo. Uma "dica prática" é gerar uma listagem do diretório para cada disco, recortá-la e colá-la na capa do disco, de modo que um simples olhar seja suficiente para verificar os conteúdos do disco.
- Guarde uma cópia de quaisquer disquetes importantes em um lugar separado, de modo a não correr o risco de perder toda a informação de uma só vez.
- Não dobre, não puxe com força e nem curve ou mutile os disquetes.
- Não exponha os disquetes ao calor nem à luz direta do sol.
- Escreva sempre no disquete com uma caneta hidrográfica. Não use caneta com ponta muito fina, pois isto danifica o seu interior.
- Assegure-se de que o disquete não esteja completamente inserido no *drive* do disco quando ligar ou desligar a força.
- Faça regularmente cópias de backup de todas as informações essenciais.

Os disquetes devem ser cuidadosamente protegidos de contaminação pela poeira. Isto acontece principalmente em um ambiente seco, no qual se devem tomar precauções especiais. A poeira acumula uma carga elétrica estática, fazendo com que a poeira e outras partículas adiram à superfície do disquete.

Eis algumas precauções importantes:

- Nunca coloque os disquetes perto de uma fonte de poeira. Isto se aplica especialmente a armazéns, locais de depósito, áreas em que se executam trabalhos odontológicos (locais de próteses dentárias), em salas de aula (com poeira de giz) e ambientes industriais.
- Mantenha o ambiente razoavelmente limpo de poeira, usando com freqüência um aspirador de pó e mantendo limpas as partes de cima das mesas e dos arquivos. Se necessário, coloque um filtro no sistema de aquecimento ou instale um limpador de ar eletrostático na sala.
- Tente evitar que se forme eletricidade estática. Pode-se utilizar um "spray" especial antiestático nos tapetes. A solução mais eficaz, porém, é utilizar um umidificador.
- Obviamente, também é importante sempre conservar os disquetes dentro de suas capas.

A IMPRESSORA

A maioria das impressoras de boa qualidade poderá funcionar, confiavelmente, por meses ou anos a fio, sem apresentar defeitos. Não obstante, devem ser tratadas com carinho, ou seja, de uma maneira muito organizada. Todos os ajustamentos mecânicos devem ser perfeitos, sem nenhuma exceção. Se o usuário não aprender a função de cada alavanca e não fizer os ajustamentos corretos, a impressora pode operar incorretamente.

Um exemplo de uma alavanca que deve ser verificada é a da "largura do papel" (rotulada de A até E nas máquinas de escrever da IBM). Se um papel fino for utilizado na impressora e a alavanca for deixada (acidentalmente) na posição D ou E (destinadas a papéis mais pesados) a impressora poderá funcionar mal, de maneira imprevisível. Pode parecer que se trata de um problema de "interface" de software, quando, de fato, seria suficiente fazer um ajustamento para a largura do papel. Colocar a alavanca na posição "A" irá resolver o problema.

LISTAGENS

As listagens podem exigir uma quantidade significativa de tempo, já que a impressora geralmente é o dispositivo mais lento de entrada/saída ligado ao computador. Muitas vezes é conveniente que o operador saia da sala enquanto a listagem está sendo processada. Portanto, recomendamos ao operador nunca deixar de verificar periodicamente a operação correta da impressora. Isto é especialmente importante no início de uma listagem, quando o papel pode ficar preso na impressora (principalmente se forem usados rótulos auto-adesivos). Se forem utilizadas "fitas de seda", elas pararão abruptamente no fim do carretel, ao invés de fazer com que os caracteres fiquem cada vez mais claros. O resultado disso é que uma parte da listagem do arquivo pode sair totalmente em branco. A não ser que se exija uma impressão da mais alta qualidade, recomendamos o uso de fitas comuns.

Ao imprimir rótulos, é importante que o operador esteja sempre presente. Isto acontece porque qualquer funcionamento inadequado – incluindo um problema mecânico da impressora – pode resultar em um pulo de linha ou que os rótulos fiquem presos. Qualquer uma destas ocorrências requer uma nova partida da operação e pode também ocasionar algum dano físico à impressora (se o rótulo ficar preso).

Sempre que ocorrer um problema durante a listagem, esta deve ser recomeçada. O caso mais freqüente é tentar dar nova partida no meio de um arquivo. Se o arquivo for extenso, o procedimento mais seguro é usar o PIP ou ED (descritos nos Capítulos 3 e 4) para escolher a parte que necessita ser impressa. Se o arquivo for de tamanho moderado, uma solução rápida e conveniente consiste em listar o arquivo rapidamente na tela, e depois ligar a impressora no lugar exato com um CTRL-P, se o programa de impressão permitir que o usuário o faça. TYPE permite-o, mas outros programas especializados de impressão poderão não permiti-lo.

Se a impressora não funcionar quando o sistema estiver ligado, verifique todos os posicionamentos físicos da mesma. (Ela pode, por exemplo, estar no modo local.) Se várias versões do CP/M estiverem sendo usadas em sua instalação, verifique se o seu disco CP/M corresponde ao tipo de impressora que está sendo utilizado – um erro muito comum.

Ao fazer listagens longas, é possível que o operador especifique listagens de múltiplos arquivos e deixe a impressora funcionando sozinha durante um período extenso de tempo (utilizando, por exemplo, uma atribuição PRN).

Finalmente, para obter listagens mais rapidamente, utilize o PIP ao invés de TYPE (use CON: = para uma listagem no CRT e LST: = para uma listagem na impressora).

ARQUIVOS

Estouro da capacidade exigida pelos arquivos (Overflow)

Se o tamanho do arquivo exceder a capacidade de um disquete, ele terá que residir em dois ou mais disquetes. Em geral não se deve simplesmente alternar para outro disquete quando a capacidade do primeiro estoura. Tente ordenar o seu arquivo no primeiro disquete de acordo com algum critério útil. Por exemplo, se o seu arquivo for uma lista de nomes e endereços, ordene-o alfabeticamente ou por meio de um código postal. Ao usar uma ordenação alfabética, o primeiro disquete conterá os nomes de A até L, e o segundo será utilizado para armazenar os nomes de M a Z. Cada disquete ficará parcialmente vazio.

Esta é uma divisão conveniente do disquete. Com isso, ainda seria possível, usando um programa apropriado, obter uma listagem por códigos de endereçamento postal; porém são necessárias duas escolhas, uma para cada disquete. Não podem ser fundidas a não ser que esteja disponível um disco rígido, isto é, um disco de maior capacidade.

Outra possibilidade, uma vez que um arquivo se torna longo demais, é separá-lo em subgrupos. Por exemplo, se o arquivo contiver nomes e endereços de fabricantes e compradores, estes podem ser separados e colocados em disquetes diferentes.

Outro problema ainda poderá ocorrer. Imaginemos que o leitor tenha um arquivo muito grande, digamos de 170K, dentro do qual queira fazer uma classificação. O seu programa de classificação está em A, e A está razoavelmente cheio: tem 120K de arquivos. Será impossível fazer a classificação, já que muitos programas de classificação exigem pelo menos 170K de "área de trabalho" no disco para se poder fazer a classificação de arquivo de 170K.

A solução é simples: crie um novo disco do sistema contendo apenas o programa SORT e utilize-o. Ele terá espaço suficiente para fazer a classificação. Se o programa de classificação ainda assim não funcionar, separe o seu arquivo original em dois menores, usando um editor ou outro programa que possa extrair parte do seu arquivo (como o PIP).

Fusão de Arquivos

Lembre-se de que o PIP pode ser utilizado para fundir dois ou mais arquivos.

Palavra Errada

Pode ser necessário alterar uma palavra ou um código em todo o arquivo. O programa-editor fa-lo-á adequadamente. Consulte o Capítulo 4 para maiores informações a respeito.

Arquivo danificado

Um arquivo pode sofrer danos devido a um engano do operador ou a algum tipo de funcionamento errôneo do sistema. Isto pode ocorrer se o operador digitar caracteres de controle não permitidos no programa, ou se acontecer algum tipo de falha de funcionamento. Contudo, como conseqüência o arquivo já não poderá mais ser carregado ou executado. Em muitos casos, se o aprendiz estiver familiarizado com a estrutura do arquivo, isto é, a aparência do mesmo, e o arquivo contiver texto, é possível recuperar o arquivo com uma intervenção adequada feita por meio do editor. Os detalhes específicos dependem do arquivo a ser operado. Muitas vezes, arquivos-texto podem ser restaurados e recuperados, mas se o arquivo for pequeno, geralmente é melhor redigitá-lo do que tentar recuperá-lo, a não ser que o usuário já esteja familiarizado com este tipo de operação.

Quando um arquivo que funcionava corretamente de súbito parece estar danificado ou se comporta mal, podemos suspeitar de um erro do operador. Porém, outra ocorrência freqüente é a do dano no disquete do sistema ou do programa. Se o disquete do sistema for danificado ou contaminado de alguma forma, a informação nele contida foi alterada e o comportamento do sistema será errôneo. Os programas usuais podem parecer estar sendo executados normalmente, mas de fato alguns dos comandos não funcionarão corretamente e poderão danificar arquivos existentes. Infelizmente, isto só costuma ser descoberto quando um arquivo já sofreu um dano de maior importância. Neste caso, mude para um novo disquete do sistema ou um novo disquete do programa. Crie um arquivo novo ou use a cópia de backup do arquivo antigo. Se isto funcionar, o aprendiz deve suspeitar que um dos disquetes anteriores foi danificado e tratar de descartá-lo (e/ou descartar todos os que forem danificados.)

PROGRAMAS ÚTEIS

Editor

O Capítulo 4 demonstrou que um editor é uma facilidade muito poderosa para operar com arquivos e modificá-los. O ED oferece tal facilidade. Outros editores comerciais disponíveis poderão ser mais poderosos ou mais adequados.

Copiadora de trilha para trilha

Um certo número de programas utilitários geralmente é fornecido pelo fabricante do sistema de computador ou do controlador do disco. Um programa que copia discos em geral copia um disco de outro, uma trilha de cada vez. É muito mais rápido do que o PIP quando se está copiando um disquete inteiro. Um editor direto de disco também pode estar disponível para fazer alterações no disquete e examiná-lo.

Apagando um disquete

Pode ser necessário apagar um disquete por uma das seguintes razões: ou o disquete apenas necessita ser apagado e não foi danificado, ou o diretório do disquete foi alterado, tornando inutilizável o disquete (pressupondo que o disquete ainda está fisicamente intacto).

Apagar um disquete ainda "bom" pode ser executado com o emprego do comando ERA**. Caso contrário, se o diretório tiver sido danificado, o disquete pode ser apagado com um programa apropriado de inicialização. Tal programa, geralmente denominado INIT, é fornecido pelo fabricante do computador ou do controlador de disco, e diverge de sistema para sistema. O INIT é uma facilidade útil para se apagar disquetes rapidamente.

Seqüência de comandos

Ao executar uma seqüência de comandos freqüentemente, o arquivo comando SUBMIT pode ser criado. O arquivo consiste de uma seqüência de comandos que devem ser executados no computador e que seriam criados com um editor, do tipo ED. Depois, digitando-se SUBMIT, seguido do nome do programa, a seqüência será automaticamente executada. Por exemplo, se um determinado programa é utilizado com freqüência e exige a digitação de um certo número de parâmetros-padrão antes de poder ser empregado, a seqüência pode ser facilmente digitada em um arquivo denominado STAR.SUB. Neste caso, o operador só teria que digitar a ordem seguinte para poder executar esse programa:

```
SUBMIT START /
```

PARADA

Quando alguma coisa der errado, pode-se querer parar tudo. Não retire o fio elétrico da tomada. Primeiro, tente o CTRL-C. Se isto não funcionar, use RESET. Lembre-se, contudo, de que irá perder qualquer informação que esteja na memória do computador. O aprendiz não irá danificar arquivos. Para parar a impressão, basta desligá-la fisicamente.

DICAS VARIADAS

Lembre-se de que o próprio CP/M "não ocupa espaço" no seu disquete, duas trilhas sempre "se perdem" em um disquete de 400 mm, quer a pessoa instale o CP/M ou não o faça. Às vezes vale a pena colocar o CP/M na maioria de seus disquetes quando ainda estão virgens. Desta forma, a pessoa pode dar a partida desde qualquer disquete. Desta maneira, também não corre o risco de fazer dano aos arquivos ao alternar discos com O PIP, se fizer um movimento errado.

Se for necessário fazer duplicações, o PIP pode lhe ajudar muito

OS SETE MANDAMENTOS DEPOIS QUE O SISTEMA FALHA

Em primeiro lugar, suspeite do operador:

1. Verifique os dispositivos mecânicos:
   - Todas as chaves estão nas posições corretas? (Faça uma verificação sistemática, não omita nada.)
   - Os fusíveis estão intactos?
   - Todos os fios estão ligados, sem conexões frouxas ou quebradas?
2. Você deu o comando correto?
   - Desligue tudo. Agora, ligue o sistema novamente.
   - Repita o comando.

Suspeite do disquete:

3. Use um novo disquete. (Às vezes o disquete em uso foi danificado por ter sido incorretamente manipulado, e com isto causa um comportamento errôneo ao sistema.)
   - Use um disquete de backup. Não use qualquer programa do disquete atual.
   - Se não houver nenhum disquete completo, backup, reserve tempo para gerar um.

Suspeite do software:

4. Certifique-se de estar usando os programas corretos:
   - A versão correta do CP/M se tiver várias.
   - O interpretador correto para o seu programa de aplicação. Por exemplo, pode ser necessária a versão CBASIC 2.
   - O programa correto de aplicação.

   Muitos dos programas de aplicação, principalmente os processadores de palavras, precisam ser adaptados ao seu terminal e à sua impressora. Se isto não for feito, algumas teclas do terminal podem não funcionar, e a pessoa pode não ser capaz de imprimir.

Ao final de tudo, suspeite do hardware:

5. Faça nova verificação rigorosa da parte mecânica.
   - Em especial, retire as placas, limpe as conexões se necessário e coloque-as de volta fixando-as bem.
   - Se a fonte errônea de funcionamento puder ser atribuída a uma placa, retire os componentes de suas posições, limpe as conexões e recoloque-os.
6. Tente sempre identificar o dispositivo suspeito de estar funcionando mal, trocando por um que se sabe estar funcionando perfeitamente: troque de placa, troque as impressoras etc. Isto lhe dará uma prova positiva e irá evitar perda de tempo. Nunca acuse um dispositivo antes de tentar trocá-lo por um que esteja funcionando bem. De outra forma poderá fazer muito esforço inútil.
7. Doravante, utilize as técnicas preventivas corretas, como já explicamos neste livro. Em resumo:
   - Sempre seja disciplinado.
   - Não tente usar atalhos. Não use nunca exceções; siga as regras.

Finalmente, o aprendiz encontrará, na seção de apêndices desta obra, um certo número de listas úteis de verificação. Lembre-se de que é mais útil prevenir do que remediar. A disciplina do usuário é a chave essencial para uma utilização bem-sucedida do computador.

Capítulo 9

O Futuro

A HISTÓRIA DO CP/M

O sistema operacional CP/M foi criado por Gary Kildall. Na época em que era assessor-consultor da Intel Corporation, Gary Kildall escreveu o primeiro compilador de linguagem de alto nível produzido pela Intel, o PL/M. Em seguida, em 1974, ele criou a primeira versão do sistema de arquivos CP/M que na época foi construído para apoiar um compilador PL/M residente.

O primeiro aparecimento do CP/M no mercado, em 1975, época em que foram emitidos os primeiros contratos de licenciamento, passou relativamente despercebido durante cerca de um ano pelo menos. Nessa ocasião foram elaboradas as versões primitivas do editor (ED), do assembler (ASM) e do depurador (DDT). O primeiro usuário comercial, em larga escala, deste sistema operacional, foi a IMSAI (firma que já se retirou do mercado), que obtivera licença para distribuir a versão 1.3 do CP/M, que evoluiu para o que a IMSAI denominou IMDOS. Agora, o CP/M evoluiu para versão 2.2 (e outras sucessivas), destinada a aproveitar a maior capacidade de armazenamento dos discos rígidos disponíveis. O MP/M foi elaborado para oferecer um ambiente de múltiplos usuários e tempo compartilhado para sistemas de multiprogramação.

No presente momento, o CP/M é, provavelmente, um dos sistemas operacionais utilizados com maior freqüência em microcomputadores. Apesar de sofrer críticas por usuários e projetistas de sistemas operacionais que estão familiarizados com máquinas de tempo compartilhado mais poderosas, atende bem a seus objetivos, e de fato se tornou um padrão para muitos usuários de microcomputadores.

CP/M E OUTROS SISTEMAS OPERACIONAIS

O desenvolvimento de um *bom* sistema operacional sempre representou um investimento econômico relevante, e por isso sistemas operacionais de tempo compartilhado potentes só foram elaborados para alguns grandes computadores. O tipo de sistema operacional mais complexo é um sistema de tempo compartilhado com estratégias poderosas de escalonamento e proteção. Embora não haja nenhum consenso quanto ao melhor sistema de tempo compartilhado, o sistema operacional UNIX tem obtido muita popularidade no campo dos minicomputadores de 16 bits. Foram feitas algumas tentativas no sentido de implementar o UNIX nos microcomputadores de 16 bits, mas o investimento exigido para tornar esses microcomputadores completamente compatíveis com um sistema como o UNIX é grande, e a probabilidade de pleno êxito é limitada.

Uma das maiores vantagens do CP/M (além da sua conveniência) é o fato de que todo o software e arquivos compatíveis com o CP/M podem agora ser compartilhados pelos usuários. O CP/M possui as virtudes de qualquer sistema operacional padrão: a compatibilidade. Por este motivo, o CP/M será provavelmente usado ainda por muito tempo – enquanto os processadores nos quais usualmente reside forem sendo construídos.

EVOLUÇÃO

Já que sempre se pode fazer aperfeiçoamentos (e correções) em qualquer programa grande já existente, o CP/M e o MP/M continuarão a evoluir. Não obstante, as novas versões destes dois sistemas geralmente serão compatíveis com as anteriores. Isto significa, na prática, que a maior parte dos conhecimentos que o leitor possa ter adquirido neste livro deve ser aplicável a quaisquer futuras versões do CP/M ou MP/M que forem lançadas no mercado. Além disso, ao usar e compreender o CP/M, irá entender as funções de um sistema operacional "padrão". Uma vez compreendidas tais funções, deverá ser capaz de se adaptar facilmente a qualquer outro sistema operacional.

CONCLUSÃO

Depois de ler este livro, o aprendiz deve ter adquirido proficiência no uso do seu computador equipado com CP/M. O "*The CP/M Handbook (With MP/M)* (*Manual do CP/M com o MP/M*)" foi elaborado para ensinar-lhe a utilizar o seu sistema e ajudá-lo a compreender como ele opera.

Ao aprender a usar qualquer programa como o CP/M, lembre-se de que a disciplina é a chave para a operação, livre de problemas, de um computador. Seguindo os procedimentos adequados, serão evitados erros e problemas. Especialmente no início, siga todas as regras apresentadas no texto, estritamente e sem exceções. À medida que for adquirindo mais experiência, o aprendiz será capaz de modificar ou ignorar algumas das regras. A utilização correta de um computador de pequeno porte e dos periféricos é assunto de outro livro do mesmo autor.

Ao usar este manual, pode-se consultar qualquer capítulo para melhorar a compreensão de um tópico específico. Com exceção do Capítulo 1, não é necessário decorar o conteúdo completo de qualquer capítulo, sendo suficiente aprender aquelas características do seu interesse. Além do mais, os resumos na seção de apêndices servem para consultas rápidas.

À medida que o aprendiz continuar a utilizar o seu sistema microcomputador, deverá aprender a usar todos os seus recursos. Por exemplo, mesmo que não planeje usar agora o editor, deve tentar fazê-lo. Desta forma será capaz de avançar facilmente para a utilização de um programa de processamento de palavras ou para a adaptação ou avaliação de um novo programa na área comercial.

Após ter-se familiarizado com todos os conceitos e técnicas apresentados neste livro, o leitor será um usuário competente de computadores. Também estará capacitado a adaptar-se rapidamente a outros programas e/ou sistemas operacionais similares.

Apêndice A

Mensagens de Erro Comuns em CP/M

Existem três condições de erro que são comuns ao sistema. Tais condições são veiculadas pela seguinte mensagem geral:

BDOS ERR ON *d: error*

onde *d* é uma letra que indica o *drive* do disco no qual o erro ocorreu, e *error* é uma das seguintes mensagens de erro:

BAD SECTOR
SELECT
READ ONLY

Existe ainda uma quarta condição de erro na versão 2.2 do CP/M na qual *error* seria a mensagem de erro:

FILE R/O

(Consulte a documentação fornecida com o programa em questão para descrição dos erros dos programas ASM, DDT, ED e outros.)

BAD SECTOR:

Um erro "área ruim" (bad sector) irá ocorrer se o controlador do disco não puder retirar a informação do disquete. Isto acontece quando o disquete está gasto (possui uma "área ruim"), ou se o controlador do *drive* do disco não estiver funcionando bem. Outra causa seria o fato de o disquete não estar no *drive* quando se tenta obter acesso a ele. Pode-se obter essa mensagem de erro, também, tentando ler arquivos colocados no disquete por um controlador diferente daquele que a pessoa estiver usando. Embora se diga que os controladores de discos são "compatíveis com IBM", podem ocorrer pequenas diferenças nos formatos dos registros. Por exemplo, arquivos gravados em disquetes com o emprego do controlador Intel MDS.800 podem ser lidos por outro controlador, mas arquivos escritos por um terceiro controlador podem produzir o erro "BAD SECTOR" quando lidos por um controlador MDS.800. Este tipo de erro pode também ocorrer se a informação em um arquivo tiver sido danificada pelo fato de se manipular inadequadamente com um disco, ou se um programa tiver sido danificado ou contiver erro.

Para tentar recuperar-se deste erro, a pessoa pode fazer um ↑C (CTRL-C para dar nova partida ao sistema), o que aborta o processamento do programa ou do arquivo e possibilita o retorno ao sistema, ou pode ignorar o erro e continuar a execução do programa e o processamento do arquivo teclando RETURN, indicando assim ao sistema que ignore o "setor ruim".

Talvez não seja seguro ignorar o erro. Se o seu programa ou a operação do arquivo envolver uma operação de gravação no diretório, pode-se destruir a integridade do seu disquete ignorando o erro. Certifique-se de possuir cópias adequadas de backup.

SELECT:

Este erro ocorre quando o aprendiz seleciona um *drive* de disco inexistente. O valor para *d* é o *drive* selecionado de forma errada. O sistema retorna automaticamente se o operador pressionar qualquer tecla no terminal.

READ ONLY:

Este erro acontece ao se tentar gravar num disquete designado como "read only" (só leitura) por meio do emprego do COMANDO STAT (ou por programa usando a função BDOS). Também pode ocorrer se for inserido um novo disquete sem executar um ↑C para dar nova partida ao sistema (e mudado o mapa para o disquete); é necessário executar um ↑C em qualquer novo disquete recém-inserido para se poder gravar nele (gravar sobre arquivos, deletar arquivos, criar arquivos ou atualizá-los).

Pressionando qualquer tecla no seu terminal, o operador pode recuperar-se dessa condição de erro e executar automaticamente uma nova partida do sistema (↑C), que *também* altera o disquete para um "read-write" (isto é, um disquete do qual a pessoa pode ler e no qual também pode gravar).

FILE R/O:

Este erro ocorre apenas nas novas versões do CP/M (versão 2.2 ou posteriores), quando se tenta gravar (gravar em cima, atualizar ou deletar) um arquivo que possui o atributo $R/O – read-only – (atribuído pelo comando STAT ou por um programa de usuário). O atributo $R/O é descrito com todos os detalhes no Capítulo 2, na seção referente à versão 2.2 do CP/M e MP/M.

Para recuperar-se desse erro, o operador pode pressionar qualquer tecla no seu terminal. A operação envolvendo o arquivo read-only é abortada, devendo-se alterar o atributo $R/O para $R/W se gravar no arquivo. Utilize STAT para alterar os atributos do arquivo.

Apêndice B

Tabela de Conversão Hexadecimal

HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	00	000
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	0
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	256	4096
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	512	8192
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	768	12288
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	1024	16384
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	1280	20480
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	1536	24576
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	1792	28672
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	2048	32768
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	2304	36864
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	2560	40960
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	2816	45056
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	3072	49152
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	3328	53248
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	3584	57344
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	3840	61440

Apêndice C

Conjunto de Caracteres ASCII

CÓDIGO	CAR	CÓDIGO	CAR	CÓDIGO	CAR	CÓDIGO	CAR
00	NUL	20[1]		40	@	60[5]	
01	SOH	21	!	41	A	61	a
02	STX	22	"	42	B	62	b
03	ETX	23	#	43	C	63	c
04	EOT	24	$	44	D	64	d
05	ENQ	25	%	45	E	65	e
06	ACK	26	&	46	F	66	f
07	BEL	27[2]	'	47	G	67	g
08	BS	28	(	48	H	68	h
09	TAB	29	)	49	I	69	i
0A	LF	2A	*	4A	J	6A	j
0B	VT	2B	+	4B	K	6B	k
0C	FF	2C[3]	,	4C	L	6C	l
0D	CR	2D	-	4D	M	6D	m
0E	SO	2E	.	4E	N	6E	n
0F	SI	2F	/	4F	O	6F	o
10	DLE	30	0	50	P	70	p
11	DC1	31	1	51	Q	71	q
12	DC2	32	2	52	R	72	r
13	DC3	33	3	53	S	73	s
14	DC4	34	4	54	T	74	t
15	NAK	35	5	55	U	75	u
16	SYN	36	6	56	V	76	v
17	ETB	37	7	57	W	77	w
18	CAN	38	8	58	X	78	x
19	EM	39	9	59	Y	79	y
1A	SUB	3A	:	5A	Z	7A	z
1B	ESC	3B	;	5B	[	7B	{
1C	FS	3C	<	5C	\	7C	|
1D	GS	3D	=	5D	]	7D[6]	}
1E	RS	3E	>	5E	↑	7E	~
1F	US	3F	?	5F[4]	←	7F[7]	RUBOUT

[1] espaço
[2] apóstrofo
[3] vírgula
[4] ou sub-linha
[5] acento
[6] ou Alt Mode
[7] ou Del

Apêndice D

Caracteres de Controle ED

Chaves*	Significado
CTRL-C (↑C)	Nova partida do sistema ("warm boot"), restaura o *prompt* do sistema.
CTRL-E (↑E)	Move o cursor para próxima linha para continuar a linha de comando (sem executar ou transmitir a linha).
*CTRL-H (↑H)	*Faz voltar para trás o cursor para apagar o último caractere datilografado.
CTRL-I (↑I)	Move o cursor em um espaço de tabulação (sete colunas).
*CTRL-J (↑J)	*Executa um RETURN
*CTRL-M (↑M)	*Executa um RETURN
CTRL-L (↑L)	Substituição para a seqüência retorno do cursor gerada por RETURN em strings utilizados com os comandos "search" e "substitute".
*CTRL-R (↑R)	Reescreve uma linha atual (escreve uma linha limpa).
CTRL-U (↑U)	Deleta a linha atual.
*CTRL-X (↑X)	*Volta para trás para o início da linha atual e apaga a linha.
**CTRL-D (↑D)	**Desvincula o programa atual do terminal.
RETURN (↵)	Transmite (executa) a linha atual, ou gera um retorno do cursor para separar linhas do arquivo-texto.
RUBOUT or DELETE	Deleta o último caractere digitado (ecoa o caractere).
CTRL-Z (↑Z)	Termina a inserção realizada pelo comando, ou separa "strings" de texto na operação "search" e "substitute", podendo ser também colocado como um marcador no fim do arquivo-texto.
CTRL-P (↑P)	"Ecoa" tudo que foi digitado ou mostrado na impressora de linhas.
CTRL-S (↑S)	Suspende temporariamente um *display* longo (para continuar o *display*, é suficiente pressionar qualquer tecla).
BREAK	Interrompe a execução do comando ED que está sendo executado.

*CP/M 2.2
**MP/M

Apêndice E

Comandos ED

Comandos	Significado
nA	Acrescenta *n* linhas (ou 1 linha se não houver n) do arquivo-fonte ao "buffer" do editor. Um "#" para n acrescentará 65535 linhas (preencherá o "buffer"), e um zero para n irá acrescentar o número de linhas necessário para preencher metade do "buffer" (o número de linhas depende do tamanho do seu "buffer" e do seu sistema).
+ / − nB	Move o CP para o início do "buffer" se + B ou para o fim se −B.
+ / − nC	Move o CP (+) n caracteres para a frente ou (−) n caracteres para trás. O CP conta uma seqüência de "retorno do cursor" como se fossem dois caracteres (RETURN e LINE FEED).
+ / − nD	Deleta n caracteres à frente (+), incluindo CP, ou deleta n caracteres atrás (−), não incluindo o CP. Se não houver, n, deleta apenas o caractere para o qual o CP aponta.
E	Termina normalmente a sessão ED. O comando E salva o texto que ficou no "buffer" e o resto do arquivo-texto-fonte em um arquivo temporário de saída, depois renomeia o arquivo de saída com o nome do arquivo-fonte (copiando o arquivo-fonte em um arquivo de reserva ".BAK" para preservar o arquivo-fonte original). ED termina, então, trazendo de volta o sistema.
nFstring {↑Z, ↵}	Acha o "string" de caracteres n vezes (se *n* não for especificado, acha-o uma só vez). F procura o CP no "buffer" e move este para o fim do "string" localizado. Coloca um terminador Z após o "string" se o operador for acrescentar mais comandos ED: caso contrário, utiliza RETURN para terminar o "string".
H	Termina a sessão ED, executa um comando E, e a seguir executa ED novamente no novo arquivo-fonte (salva as suas atualizações de arquivo e volta de novo à edição).
OBSERVAÇÃO: Pode-se substituir *n* por um caractere "#" em qualquer um dos comandos ED; isto dará a n o maior valor que é possível assumir − 65.535. *CP é o indicador de caracteres.	
I↵	Insere uma nova linha de texto depois do CP, movendo-o para o fim da última linha inserida. No CP/M 2.2: — Se forem utilizadas letras MAIÚSCULAS no comando, todo o texto inserido estará apenas em letras MAIÚSCULAS. — Se forem utilizadas letras minúsculas, o texto será inserido em MAIÚSCULAS e minúsculas.

itext ↵ Z	Insere caracteres depois do CP, movendo-o para o fim do último caractere inserido.
nJstring1 ↵ Zstring2 ↵ Zstring3 { ↵ Z / ↵ / J }	Justapõe "strings" de texto localizando o "string1", insere o "string2" no final do primeiro, e deleta todos os caracteres até o (sem incluí-lo) "string3" (justapõe todos os três "strings" do texto). O CP é movido para o início do 3º "string".
+ / – nK	"Kill" deleta as seguintes (+) n linhas, incluindo o CP e os caracteres seguintes da linha atual, ou deleta (–) as n linhas anteriores incluindo os caracteres antes do CP na linha atual.
+ / – nL	Move o CP para o início da linha atual se n for zero; caso contrário, move o CP para o início da linha atual e movimenta-o para a frente (+) ou para trás (–) equivalente a n linhas.
nMstring { ↵ Z / ↵ / J }	Repete a execução do "string" de comandos ED n vezes, se n for maior do que 1. Se n for zero ou um, M irá executar o "string" de comandos repetidamente até ocorrer um erro.
nNtext { ↵ Z / ↵ / J }	Procura a *enésima* ocorrência do texto no "buffer" e no arquivo-fonte (termina o texto com RETURN ou Z para acrescentar mais comandos). N move o CP para o fim do texto encontrado. N irá acrescentar linhas-fonte até encontrar o texto.
O	Omite a sessão ED e volta ao arquivo-fonte original.
+ / – *nP*	Move o CP, mostra e imprime as *páginas* do texto existente no "buffer". O n representa o número de páginas (24 linhas por página) impressas, onde + n imprime n páginas depois do CP, e –n imprime n páginas antes do CP.OP (zero para n) irá imprimir a linha e página atual (as primeiras 23 linhas depois da linha atual). O CP é movimentado para o início da página impressa.
Q	Sai *sem* alterações no arquivo (deixa o arquivo temporário, o arquivo-fonte e o arquivo do "buffer" intactos). Q faz o operador retornar ao sistema. *Não* é criado o arquivo "backup" (".BAK") para o arquivo-fonte, mas se havia um arquivo ".BAK" prévio com o mesmo nome, ele é *deletado* (Atenção para isto!)
R	Lê a partir do arquivo X$$$$$$$$.LIB e insere as linhas que se seguem ao CP, movendo-o para o fim das linhas inseridas (não esvazia o arquivo ".LIB").
Rfilename	Lê desde o *filename*.LIB e insere as linhas que se seguem ao CP, movendo-o para o fim das linhas inseridas (não esvazia o arquivo ".LIB").
nSoldtext ↵ Znewtex { ↵ Z / ↵ / J }	Acha oldtext no "buffer" após CP e coloca em seu lugar newtext; repete a seqüência n vezes se n for maior do que um.
+ / – nT	Se n não for especificado ou for igual a 1, mostra (apresenta) os caracteres que se seguem ao CP até o fim da linha. Se n for zero, mostra os caracteres na linha atual até o CP, sem incluí-lo. Se n for positivo (+), apresenta as linhas seguintes, incluindo a atual. Se n for negativo (–), apresenta as n linhas anteriores não incluindo a atual e apresenta os caracteres na linha atual até o CP, sem incluí-lo. A seqüência de comando "B # T" apresentará o "buffer" por inteiro.

+ / – U	Traduz todas as entradas (digitadas ou inseridas) em caracteres minúsculos para MAIÚSCULOS se + U, ou interrompe a tradução executando – U.
V	Liga a apresentação (display) do número da linha (os números das linhas não estão atualmente no arquivo) para as linhas do "buffer".
0V	Apresenta o número de bytes remanescentes no "buffer" e o tamanho total da memória do "buffer" (em números decimais). Por exemplo, no display "27648/28832", "27648" é o número de bytes e "28832" é o número total de bytes no "buffer" atual.
nW	Grava no arquivo temporário de saída com a extensão ".$$$" as seguintes n linhas do CP (incluindo a atual). Se não houver n, grava apenas a linha atual.
nX	Copia as linhas seguintes do texto para o arquivo X$$$$$$$$.LIB (não suprime as linhas originais). Recupera as linhas utilizando o comando R. Se n for zero, este comando irá *deletar* o arquivo X$$$$$$$$.LIB.
**nZ	Suspende o programa ED para n sinais do relógio (aproximadamente *n* segundos).
+ / –n	Executa uma seqüência de comando "+ / – nLT".
n:	Move o CP para o início da linha número n.
n1: : n2	Especifica uma faixa de números de linhas começando com n1 e terminado com n2. Se n1 ou n2 não constarem desse comando, coloca em seu lugar "a linha atual".
*CP é o indicador de caracteres.	

Apêndice F

Nome dos Dispositivos PIP

DISPOSITIVOS LÓGICOS

CON:	representa "console" ou terminal, incluindo o teclado e o display (Entrada/Saída).
RDR:	representa fita de papel ou leitora de cartões (só entrada).
PUN:	representa fita de papel ou perfuradora de cartões (só saída)
LST:	representa um dispositivo de "listagem" como uma impressora de linhas, por exemplo (só saída)

DISPOSITIVOS FÍSICOS

TTY:	representa um console ou um terminal, um dispositivo de listagem, uma leitora de cartões ou uma perfuradora. (pode se referir a um teletipo).
CRT:	para um console ou terminal, ou um dispositivo de listagem (terminal de vídeo).
PTR:	para uma fita de papel de ou um dispositivo de leitora de cartões.
PTP:	para uma fita de papel ou um dispositivo de perfuração de cartões.
LPT:	para um dispositivo de listagem (impressora de linhas)
UC1:	para um console ou terminal definido pelo usuário
UR1:	para uma leitora definida pelo usuário
UR2:	para uma segunda leitora definida pelo usuário
UP1:	para um dispositivo de saída (perfuração) definido pelo usuário
UP2:	para um segundo dispositivo de saída (perfuração) definido pelo usuário.
UL1:	para um dispositivo de listagem definido pelo usuário.

OBSERVAÇÃO: BAT: não foi incluído, porque apenas reatribui os valores para RDR: e LST: (Consulte "Dispositivos de atribuição")

Apêndice G

Palavras-Chaves PIP

NULL:	Envia 40 "nulls" (ASCII código 0) para o dispositivo, geralmente um dispositivo de perfuração para a saída. Por exemplo, envia PROG.HEX para a perfuradora: *PUN:=PROG.HEX,NULL: ↵
EOF:	envia um fim-de-arquivo (ASCII ↑Z) para o dispositivo (enviado automaticamente pelo PIP durante transferências de arquivo-texto ASCII e necessário somente para casos especiais.) Exemplo: *PUN:=NULL:,X.ASM,EOF:,NULL: ↵ Este exemplo envia 40 "nulls" para o dispositivo de perfuração, seguido de uma cópia do arquivo X.ASM, seguido de um caractere de fim-de-arquivo (↑Z e mais 40 "nulls).
PRN:	O mesmo que LST: (envia para a impressora, exceto que as tabulações são expandidas, em cada oitavo caractere, as linhas são numeradas (como no programa ED), e "page ejects" (alimentação de formulários são inseridos em cada uma das 60 linhas (para fazer avançar o papel da impressora para a página seguinte), com um "page eject" inicial. Exemplo: *PRN:=SAMPLE.TXT ↵
INP:	código especial de dispositivo de entrada que pode ser "remendado" no próprio programa PIP (deve-se escrever a correção na linguagem assembler e acrescentá-la ao PIP). O PIP recebe a entrada caractere por caractere chamando uma posição na memória (103H) e armazenando os dados começado na posição 109H (o bit de paridade deve ser zero, utilize o parâmetro Z).
OUT:	código especial de saída para o dispositivo que pode ser corrigido programa PIP, como IPN: descrito acima. O PIP chama a posição 106H e envia os dados para o registrador C (cada caractere). Observação para programadores assembler: as posições 109H até 1FFH da memória de imagem do PIP não são usadas e podem ser substituídas por códigos para mecanismos impulsores de dispositivos de objetivos especiais (use o DDT — o Depurador CP/M fornecido pela Digital Research com CP/M ou MP/M). Exemplos: *GIZMO.CLK=INP: ↵ *(entrada de um dispositivo especial é armazenada no arquivo GIZMO.CLK)* :OUT:=GIZMO.CLK ↵ *(cópia de GIZMO.CLK é enviada para o dispositivo especial).*

Apêndice J

Parâmetros PIP

B	Transferência de modo de bloco. O PIP coloca os dados em um "buffer" até que leia um caractere "x-off" (^S) ASCII do dispositivo. O PIP então limpa o "buffer" do disco e volta para receber mais dados. O tamanho do "buffer" depende do tamanho do seu sistema (consulte a documentação fornecida com o seu sistema). Utilize este parâmetro para transferir dados de um dispositivo de leitura contínua como um gravador de cassete ou uma leitora. Exemplo: *ENUFF.TXT = RDR:[B] ↵
Dn	O PIP irá deletar caracteres que vão além da enésima coluna (colunas verticais em seu terminal) ao copiar arquivos-texto. Use este parâmetro para truncar linhas extensas se estiver enviando um arquivo para um dispositivo estreito. Exemplo: *PRN: = LONG.TXT[D52] ↵
E	Ecoe (apresente novamente) todas as operações de cópia na tela do terminal à medida que estão sendo executadas. Exemplo: *COPY.TXT = SOURCE.TXT,S2.TXT,S3.TXT,S4.TXT[E] ↵
F	O PIP irá "filtrar alimentadores de formulários" do arquivo (isto é, os removerá). Você também pode usar o parâmetro P para inserir novos alimentadores de formulários.
Gn	Obtenha o arquivo da área do usuário n (versão 2.2 do CP/M e MP/M)
H	Transferência de dados hexadecimais: O PIP verifica todos os dados para "checar" se estão de acordo com o formato hexadecimal Intel adequado (consulte: "Notas a respeito de cópias para código de máquina (arquivos (HEX)")
I	Ignora "os registros "00" na transferência de arquivos em formato Intel hex (automaticamente posiciona o parâmetro H).
L	Traduz todos os caracteres em maiúsculas para minúsculas.
N	Acrescenta números de linhas para cada linha copiada para o arquivo novo (começando na linha 1). Cada número de linha é seguido de dois pontos. Zeros à esquerda (por exemplo, 003) são suprimidos, a não ser que se especifique o parâmetro "N2". "N2" não remove os zeros à esquerda e insere um espaço de tabulação depois dos números. Pode-se expandir tais espaços usando o parâmetro T.
O	Transferência de arquivo-objeto (para arquivos não ASCII) O PIP ignora o fim físico do arquivo durante a concatenação (consulte "Concatenando arquivos" no Capítulo 2).
Pn	O PIP irá incluir "page ejects" em cada enésima linha (com um "page eject" inicial). Se n for 1 (ou você não especificar n) isto ocorrerá em cada uma de 60 linhas. Se você também usar o parâmetro F, o PIP remove os alimentadores de formulários antes de inserir os "page ejects".
Qstring ↵ Z	O PIP irá parar de copiar do dispositivo ou arquivo ao encontrar o string de caracteres especificados (um string é um grupo de caracteres; por exemplo, STRING 105%). Termina-se o "string" com um ↵ Z (CTRL e Z simultaneamente). Consulte "Copiando partes de arquivos" no Capítulo 2.
R	Lê (copia) arquivos de sistema ($SYS); também executa uma operação correspondente ao parâmetro "W" (versão 2.2 no CP/M e MP/M.)
Sstring ↵ Z	O PIP irá começar a copiar do dispositivo ou arquivo ao encontrar o "string" de caracteres especificado. Termine seu "string" com um ↵ Z. Consulte "Copiando partes de arquivos" no Capítulo 2.
Tn	Expande os espaços de tabulação para cada enésima coluna durante a transferência de arquivos-texto. Cria-se um espaço de tabulação em um arquivo-texto usando ↵ I; este parâmetro irá expandir o espaço de tabulação além do seu limite de colunas geralmente fixado.
U	Traduz todos os caracteres em minúsculas para maiúsculas durante a cópia de arquivos-texto.
V	O PIP irá verificar se os dados foram copiados corretamente, relendo o novo arquivo-cópia a seguir (o arquivo-cópia não pode ser um dispositivo) e apresentando uma mensagem se a cópia foi bem-sucedida.
W	Deleta arquivos "ready-only" (ignora o atributo $R/O). (Versão 2.2. do CP/M e versão MP/M somente).
Z	Transforma o bit de paridade para zero na entrada de caracteres em ASCII. Utilize este parâmetro, em especial, se estiver dando entrada a partir do dispositivo de correção INP.

Aqui estão alguns exemplos de expressões PIP com parâmetros.

*LST: = SAMPLE.TXT[NT8P60] ↵

Esta expressão envia o arquivo SAMPLE.TXT para o dispositivo listagem (LST:), com números de linhas, tabulações expandidas para cada oitava coluna de caracteres, e "page ejects" a cada 60 linhas.
OBSERVAÇÃO: O dispositivo PRN: assume estes parâmetros; se o dispositivo de listagem fosse atribuído a PRN: o exemplo acima poderia ser reescrito:

*PRN: = SAMPLE.TXT ↵

Apêndice U

Resumo dos Comandos CP/M (e MP/M)

COMANDO	CP/M VERSÃO 1.4	CP/M VERSÃO 2.2	MP/M VERSÃO 1
ABORT			X
ASM	X	X	X
ATTACH			X
CONSOLE			X
DDT	X	X	X
DIR	X	X	X
DSKRESET			X
DUMP	X	X	X
ED	X	X	X
ERA	X	X	X
ERAQ			X
GENHEX			X
GENMOD			X
GENSYS	X	X	X
LOAD	X	X	X
MOVCPM	X	X	X
MPMLDR			X
MPMSTAT			X
PIP	X	X	X
PRLCOM			X
REN	X	X	X
SAVE	X	X	X
SCHED			X
SPOLL			X
STAT	X	X	X
STOPSPLR			X
SUBMIT	X	X	X
SYSGEN	X	X	X
TOD			X
TYPE	X	X	X
USER			X
XSUB		X	X

Apêndice J

Controles de Comandos de Edição

CONTROLES COMUNS

rubout/delete	deleta e repete o último caractere
CTRL-U ou CTRL-X	deleta linha
CTRL-R	reescreve linha
CTRL-E	continua na linha seguinte
CTRL-C	dá nova partida ao CP/M

OUTROS

CTRL-D	desvincula o console (MP/M)
CTRL-H	volta para trás
CTRL-J – (line feed)	termina a entrada (input)
CTRL-M (carriage return)	termina o comando
CTRL-P	liga/desliga impressora
CTRL-Q	"prende" a impressora (MP/M)
CTRL-S	pára/dá nova partida à saída (output) do console

Apêndice K

Tipos de Extensão CP/M

Extensão	Tipo	Exemplo
COM	Requerido para arquivos de comando transientes ou (programa)	PIP.COM LOAD.COM
ASM	Essencial para arquivos (texto) fonte em linguagem assembler usados com um comando ASM.	PROG1.ASM PATCH.ASM
PRN	Essencial para um arquivo de listagem do programa em linguagem assembler	PROG1.PRN PATCH.PRN
HEX	Essencial para um arquivo-programa em formato "hex" (linguagem de máquina) que está pronto para ser LOADed (carregado)	PROG1.HEX PATCH.HEX
PRL	Essencial para programas relocáveis MP/M	RDT.PRL
RSP	Essencial para "programas residentes no sistema", em MP/M.	SPOOL.RSP
BAS	Essencial para arquivos-fonte (texto) programas BASIC.	PROGBAS.BAS
INT	Essencial para arquivos intermediários de programas BASIC para execução (já compilados).	PROGBAS.INT
BAK	Criado pelo ED (editor de texto) como cópia backup de arquivo antes que este seja alterado.	LETTER.BAK
$$$	Arquivos temporários (a serem apagados) criados e normalmente apagados pelo ED e outros programas.	LETTER.$$$
SUB	Arquivo-texto com comandos ou programas CP/M embutidos ou transientes; a serem executados em "batch" pelo programa SUBMIT.	TRANSFORM.SUB

Apêndice L

Acessórios (Lista de Verificação)

- ☐ Disquetes virgens
- ☐ Fitas
- ☐ Papel para a impressora
- ☐ Manual do computador
- ☐ Manual da impressora
- ☐ Manual do terminal CRT
- ☐ Documentação do CP/M
- ☐ Documentação dos programas de aplicação
- ☐ Disquete do sistema
- ☐ Disquete dos programas de aplicação

Apêndice M

Organização do Ambiente do Computador (Lista de Verificação)

- ☐ Ventilação suficiente
- ☐ Nenhum objeto nas saídas de ventilação do computador
- ☐ Pastas não metálicas para arquivos de disquetes
- ☐ Acessórios suficientes para o computador (veja lista de verificação separada)
- ☐ Todos os manuais essenciais
- ☐ Registro de posicionamentos corretos do terminal
- ☐ Registro de posicionamentos corretos da impressora
- ☐ Registro de manutenção
- ☐ Números de telefones para manutenção e assistência técnica
- ☐ Nenhum telefone perto da área de trabalho (um telefone que toca em cima de um disquete ou de um *drive* do disquete limpa o disquete)
- ☐ Nenhuma chave de fenda perto da área de trabalho (magnético)
- ☐ Nenhum líquido na sala do computador
- ☐ Não fumar muito perto dos *drives* de discos
- ☐ Não mover nem sacudir os *drives* de discos
- ☐ Procedimento de ligação afixado
- ☐ Nenhum tapete ou carpete que tenha predisposição para estática.

Apêndice U

Lista de Verificação de Falhas

NADA FUNCIONA

- ☐ Verifique as conexões mecânicas
- ☐ Fios de força
- ☐ Cabos
- ☐ Chaves "ligadas"
- ☐ Fusíveis

A IMPRESSORA NÃO OPERA

- ☐ Tente a impressora em "local"
- ☐ Execute CTRL-P a partir do console
- ☐ Verifique todos os posicionamentos
- ☐ Reinsira o papel adequadamente
- ☐ Verifique o fusível

A IMPRESSORA NÃO PÁRA

- ☐ Pressione CTRL P
- ☐ Pressione CTRL C
- ☐ Desligue a impressora

O SISTEMA NÃO OPERA

- ☐ Faça um "reboot" (CTRL C)
- ☐ Pare o sistema e execute uma nova partida completa

O DRIVE DO DISCO ESTÁ CONTINUAMENTE LIGADO

- ☐ Não há disquete no *drive*. Insira um disquete
- ☐ Retire o disquete, recomece o processo

COMPORTAMENTO GROSSEIRAMENTE ANÔMALO

- ☐ Suspeite de um erro do operador. Tente de novo. Verifique se o disquete do sistema está correto e se os posicionamentos da impressora estão corretos.
- ☐ Suspeite de um disquete danificado. Substitua por outra cópia.
- ☐ Suspeite de um programa de aplicação danificado. Substitua por outra cópia.
- ☐ Desligue tudo. Tente de novo.
- ☐ Suspeite de uma falha do hardware.

Apêndice V

Regras Básicas para Verificar que tipo de Problema Está Acontecendo

- Nesta ordem:
  1. Suspeite de um erro do operador
  2. Suspeite de um disquete danificado
  3. Suspeite do software
  4. Suspeite do hardware
- Mantenha uma documentação detalhada da falha ocorrida.

- Tente de novo a partir do "nada". Utilize disquetes novos. Verifique o estado de todos os posicionamentos e conexões mecânicas.

Glossário

Apresentamos um glossário de palavras comumente usadas em inglês, mas que acreditamos merecerem uma tradução adequada ao dia-a-dia do usuário de CP/M.

Esperamos com isto contribuir para uma padronização definitiva desta terminologia.

O grupo de trabalho foi composto de tradutores-professores de inglês e analistas de sistemas, que procuraram dar a este Manual um conteúdo técnico objetivo, atualizado e de fácil acesso.

BACKUP	Reserva – cópia de segurança.
BATCH	Lote – grupo de registros que, para efeito de processamento, são considerados como uma unidade.
CARRIAGE RETURN	Retorno do cursor (vídeo) em alguns casos "Retorno do Carro de Impressão".
COLD BOOT	Partida "a frio" o mesmo que "cold start".
CP/M (CONTROL PROGRAM FOR MICROPROCESSORS)	Programa de Controle para Microprocessadores.
DEBBUG	Depuração (de um programa, por exemplo) ou seja, detectar, localizar e eliminar erros numa rotina.
DEFAULT	Atributo ou opção alternativos presumidos como corretos.
DUMP	Despejar – é o ato de mostrar o conteúdo da memória em algum dispositivo.
FILE MATCH (FILENAMEMATCH)	Uma chave (nome de arquivo) que se associa a vários outros nomes de arquivos, com o objetivo de mostrá-los em um terminal, imprimi-los etc.
MENUS	Palavra usada para identificar as modalidades de programas ou rotinas utilitárias presentes no sistema.
MP/M (MULTIPROGRAMMING CONTROL PROGRAM FOR MICROPROCESSORS)	Programa de Controle de Multiprogramação para Microprocessadores. Obs.: Tanto o CP/M quanto o MP/M são sistemas operacionais.
PATCH	Correção.
POLLING	Sondagem – técnica de linhas de transmissão. Cada terminal é chamado, verificando se o usuário necessita utilizar a linha.
PROMPT	Sinal mostrado no vídeo, pelo sistema, que autoriza o usuário a digitar um novo comando.

READ/ONLY (R/O)	Só leitura – tipo de especificação de disquete, que permite a leitura e a gravação do mesmo.
READ/WRITE (R/W)	Ler/gravar – tipo de especificação do disquete, que permite a leitura e a gravação do mesmo.
SCRATCH PAD	Auxiliar.
SLUFT LOCK	Tecla especial que impede que o cursor se movimente no vídeo.
SPOOL	Nome dado às operações de R/W.
TURNKEY	Sistema "sob encomenda". Já vem pronto para utilizar.
WARM BOOT	Partida "a quente" – o mesmo que "warm start".

Índice Analítico

O MANUAL DE CP/M

Incluindo MP/M

O CP/M foi elaborado para facilitar o uso dos microcomputadores. Este manual, um dos mais vendidos nos Estados Unidos, foi escrito para tornar simples o emprego do CP/M e seus recursos. Não é necessário que você já tenha algum conhecimento anterior, mas é recomendável que tenha acesso a algum sistema equipado com CP/M.

O CP/M tornou-se um sistema padrão de operação para microcomputadores, e praticamente todos os usuários destes equipamentos acabarão por empregá-lo, pois ele está disponível em quase todos os computadores que empregam os microprocessadores 8080, 8085 ou Z80 e ainda em alguns dos que empregam o microprocessador 6502. Dependendo do nível de sofisticação dos programas de aplicação que se venha a executar, serão usados todos ou apenas alguns dos recursos oferecidos pelo CP/M. Assim, este texto foi estruturado para satisfazer uma ampla variedade de interesses.

Cobrindo o CP/M e suas inúmeras versões, inclusive CP/M 1.4 e CP/M 2.2 e o novo sistema operacional de múltiplos usuários denominado MP/M, é também aplicável a sistemas operacionais compatíveis com CP/M, como, por exemplo, o CDOS da CROMEMCO.

Muitas tabelas úteis são apresentadas nos apêndices e devem ser consultadas depois da leitura do livro. Elas incluem os códigos binários usuais, mensagens de erros, símbolos e comandos oferecidos pelo CP/M, ED e PIP.

Leia também:

- **FERNANDEZ & ASHLEY. Usando CP/M; Um Guia em Ensino Programado. 1984.**

ISBN 85-7001-188-1
(Edição original: 0-89588-048-2 Sybex Inc., USA.)

Capa/Otavio Studart