SOX

CONCEITOS BÁSICOS

MARCOS C. SAMPAIO
JACQUES P. SAUVÉ
J. ANTÃO B. MOURA

McGraw-Hill/INFOCON

Cobra
Computadores e Sistemas Brasileiros S.A.

SOX

Conceitos Básicos

SOX

Conceitos Básicos

MARCUS C. SAMPAIO
Mestre em Ciência da Computação pela Universidade Federal da Paraíba (UFPb)
Professor Adjunto do Departamento de Sistemas e Computação da UFPb
Coordenador da Coleção de Informática da McGraw-Hill/INFOCON

JACQUES P. SAUVÉ
PhD em Engenharia Elétrica pela University of Waterloo, Canadá
Professor Adjunto do Departamento de Sistemas e Computação da UFPb
Diretor da INFOCON Software

J. ANTÃO B. MOURA
PhD em Engenharia Elétrica pela University of Waterloo, Canadá,
Assessor para Ciência da Computação junto ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq)
Professor Adjunto do Departamento de Sistemas e Computação da UFPb
Diretor da INFOCON Software

Cobra Computadores e Sistemas Brasileiros S.A.
Rio de Janeiro
Av. Comandante Guaranys, 447
CEP 22700
(021) 342-9393

McGraw-Hill
São Paulo
Rua Tabapuã, 1105, Itaim-Bibi
CEP 04533
(011) 881-8604 e (011) 881-8528

Rio de Janeiro • Lisboa • Porto • Bogotá • Buenos Aires • Guatemala • Madrid • México • New York • Panamá • San Juan • Santiago

Auckland • Hamburg • Kuala Lumpur • London • Milan • Montreal • New Delhi • Paris • Singapore • Sidney • Tokyo • Toronto

SOX – CONCEITOS BÁSICOS

Editor: Milton Mira de Assumpção Filho
Coordenadora de Revisão: Daisy Pereira Daniel
Supervisor de Produção: José Rodrigues
Capa: Informática Graphico Visual
Rua Dr. Emilio Ribas, 136

Dados de Catalogação na Publicação (CIP) Internacional
(Câmara Brasileira do Livro, SP, Brasil)

Sampaio, Marcus C.
S184s SOX : conceitos básicos / Marcus C. Sampaio, J. Antão B. Moura, Jacques P. Sauvé. -- São Paulo : McGraw-Hill, 1987.

1. SOX (Sistema operacional de computador) I. Moura, J. Antão B. II. Sauvé, Jacques P. III. Título.

87-1723 CDD-001.6425

Índices para catálogo sistemático
1. SOX: Sistema operacional : Computadores : Processamento de dados 001.6425

AGRADECIMENTOS

Os autores agradecem a Tadeu Beltrão e Lucicleide Guimarães da INFOCON Software pelos seus trabalhos de digitação, revisão e produção das laudas originais deste trabalho.

Agradecemos à Diretoria Técnica da COBRA e, em especial, a Manoel Lage e Júlio Laufer, pela atenção a nós dispensada durante a confecção de cada capítulo.

Os Autores

APRESENTAÇÃO

A Indústria Brasileira de Informática é exemplo de esforço incessante, em que, apesar das dificuldades, vamos avançando com firmeza e criando para o País uma riqueza fundamental.

Esta Indústria significa domínio tecnológico, que, por sua vez, significa, nas vésperas do século XXI, desenvolvimento em todas as frentes, bem estar para todos e soberania – o direito de escolher o próprio caminho.

O progresso e a criatividade nas áreas de vanguarda não são, nem podem ser, privilégio exclusivo de alguns países que tiveram a ventura de chegar antes aos conhecimentos científicos e tecnológicos.

O Brasil não abre mão de participar ativamente da corrida aos novos horizontes da ciência e da técnica, que mudarão a face do planeta.

Foi desta determinação que nasceu o SOX – um sistema inteiramente desenvolvido no Brasil por técnicos brasileiros.

Compatível com o sistema operacional UNIX AT&T, que vem ao longo dos últimos anos firmando-se como padrão internacional de fato, o SOX foi projetado pela COBRA para converter-se em padrão nacional.

Sua tecnologia está agora à disposição de todas as empresas nacionais, o que garantirá seu amadurecimento e crescimento acelerados.

Esta, pois, é uma contribuição efetiva à capacitação do País.

Parabéns aos seus autores pela sensibilidade e pela coragem da idéia.

Parabéns à COBRA pelo estímulo e pelo empenho a uma causa que é de todos os brasileiros.

Renato Archer
Ministro da Ciência e Tecnologia

SUMÁRIO

FIGURAS

TABELAS

PREFÁCIO

A história da afirmação de padrões de fato, no cenário tecnológico internacional, tem sido marcada por uma continuada atenção por parte dos profissionais envolvidos com as respectivas tecnologias.

Não tem sido diferente com a Informática - no entanto, neste ramo da atividade humana, que vem conseguindo atingir índices dominantes de aplicabilidade e, portanto, de popularidade, o próprio usuário final se vê envolvido com a questão da emergência de padrões.

A razão é simples - padronização implica disponibilidade garantida e crescente, multiplicidade de opções, conectividade, independência de fornecedores.

Não há hoje qualquer dúvida de que o sistema operacional UNIX, originalmente apresentado pela empresa AT&T norte-americana, tenha se tornado uma base para a emergência de padrões de nível internacional.

A crescente popularidade de tais sistemas deriva certamente de vários fatores, mas, indubitavelmente, desponta como principal o fato de a própria AT&T ter permitido a disseminação do sistema UNIX, em suas versões preliminares, pelo ambiente universitário norte-americano - principalmente na Universidade da Califórnia, em Berkeley, que introduziu aperfeiçoamentos importantes, posteriormente incorporados pela própria AT&T.

Destarte, o ambiente brasileiro de informática não poderia ficar imune à ascenção dessa tendência tecnológica. A COBRA Computadores, em particular, começou, desde 1982, a capacitar-se no domínio da tecnologia em sistemas tipo UNIX.

Naquele ano, a maior parte da equipe que havia desenvolvido o SOD - sistema operacional multiusuário e multitarefa que equipa a linha Cobra/500 de minicomputadores - passou a dedicar-se integralmente ao estudo e à especificação de um sistema operacional compatível com o UNIX.

Árdua tarefa. Pois, sem contar com qualquer apoio de empresas que já dominassem tal tecnologia, e baseada unicamente no conjunto de especificações de interface contidas no manual "System Five Interface Definition" - "SVID" - publicado pela própria AT&T, e na literatura eventualmente disponível, a equipe da COBRA lançou-se ao desafio.

Não se tratava, no entanto, de empreitada meramente sonhadora e incoseqüente - a equipe original, de 10 profissionais altamente qualificados, detinha à época um total de cerca de 1200 homens/mês de experiência concreta no desenvolvimento de sistemas operacionais.

O resultado aí está. E se multiplica. No empenho de colocar a tecnologia desenvolvida à disposição de toda a indústria nacional de informática, e propor o SOX como padrão nacional, a COBRA tem contado com o inestimável apoio de importantes empresas fabricantes, de inúmeras empresas produtoras de "sofware", e de outras instituições.

Mas apenas o SOX não basta. É preciso que haja compiladores das linguagens mais importantes, que haja bancos de dados, que haja pacotes para comunicação de dados, que haja utilitários. Tais programas para o ambiente SOX também vêm sendo desenvolvidos pela COBRA, e por outras empresas associadas neste esforço comum – toda esta tecnologia está sendo efetivamente colocada à disposição do País.

É nossa crença mais profunda que estamos vivendo o limiar de uma era de realizações tecnológicas vitais para o Brasil, em que esforços cooperativos como este envolvendo o SOX representam nossa afirmação no desenvolvimento de "software" moderno, competitivo e de qualidade internacional.

OS AUTORES

Marcus C. Sampaio, Jacques P. Suavée e J. Antão B. Moura vêm desempenhando, através de sua empresa, INFOCON, de Campina Grande, Paraíba, um sério esforço de contribuição à autonomia tecnológica nacional em software, em particular no que se refere a sistemas operacionais do tipo UNIX.

Sua colaboração com a COBRA data de muitos anos, e atinge agora um de seus melhores momentos com a publicação deste livro, tornado possível em grande parte pela sensibilidade dos autores em acreditar na importância que terá o SOX como peça-chave no cenário brasileiro de software.

O principal objetivo pretendido para a obra foi o de estabelecer uma base de disseminação da tecnologia representada pelo SOX, o que, à vista da alta qualidade do material, será plenamente atingido.

Todos da COBRA sentimo-nos orgulhosos pela publicação deste SOX-Conceitos Básicos, e esperamos que possa vir a contribuir para a consolidação do SOX como um sistema operacional à altura das necessidades do mercado brasileiro.

MANOEL LAGE P. DA SILVA
Diretor de Desenvolvimento
COBRA S.A.

CAPÍTULO 1

APRESENTAÇÃO

SOX: CONCEITOS BÁSICOS apresenta, às vezes sem esconder um sentimento de entusiasmo, o Sistema Operacional SOX, desenvolvido pela COBRA – COMPUTADORES E SISTEMAS BRASILEIROS. Torcemos para que o SOX venha a se tornar um padrão, no campo de sistemas operacionais, para a indústria brasileira de computadores. Este livro empunha, com muita convicção, esta bandeira e aspira dar sua contribuição, por mínima que seja, para a disseminação da cultura SOX – para que o sistema da COBRA se torne o padrão nacional, de fato.

As razões para isto são várias e deverão se esclarecer à medida que você, leitor, for lendo este livro. No momento, podemos adiantar que o SOX foi desenvolvido à imagem e semelhança do UNIX, um sistema operacional da AT&T, empresa americana, que vem se firmando cada vez mais no mercado americano, a caminho de se tornar um padrão em todo o mundo. Deste modo, investindo no SOX, a indústria brasileira de computadores ocupará também seu lugar na linha de frente dos esforços mundiais por tecnologias de sistemas operacionais que venham a superar, ou reduzir, consideravelmente, os grandes obstáculos ainda existentes ao desenvolvimento mais racional da informática, também neste campo específico.

1.1 A QUEM ESTE LIVRO SE DESTINA

SOX: CONCEITOS BÁSICOS é um livro didático, voltado para qualquer tipo de leitor – mesmo o leigo em informática. Não se exige, portanto, conhecimentos prévios de computadores e de informática, muito menos do SOX.

1.2 OBJETIVOS DO LIVRO

Concebido de forma a interessar a uma vasta gama de leitores, foi necessário, no entanto, definir o perfil do usuário-alvo do livro, sem perder de vista, entretanto, os interesses dos demais tipos de leitores. Tal estratégia se impôs, pois, do contrário, este livro estaria desprovido de propósitos bem estabelecidos, terminando por desagradar a todos.

O leitor que mais se encaixa nas intenções deste livro é aquele que, não sendo um especialista em informática, deseja, contudo, manter-se bem informado a respeito das possibilidades das novas tecnologias da informática, particularmente sobre o SOX; se juntarmos a este interesse a existência do SOX no local de trabalho, então este livro cairá como uma luva – ele se esforça por levar o leitor a um bom uso do SOX, corrigindo-lhe os defeitos de um treinamento puramente pragmático (do tipo "fulano não enxerga um palmo adiante do nariz"), por dar, até onde não possa comprometer a clareza da exposição, um certo embasamento teórico.

Leitores não-leigos, e aqueles já com conhecimentos do SOX ou similares, podem tirar também bom proveito deste livro, principalmente porque a exposição dos assuntos se dá com uma dosagem razoável de rigor, dispensando, muitas vezes, a consulta a herméticos e tediosos manuais de documentação do produto.

Este livro pode ser também uma boa referência elucidativa para cursos e seminários sobre o SOX, nas salas de treinamento pelo país afora.

1.3 A ESTRUTURA DO LIVRO

O livro é composto de treze capítulos e dois apêndices.

O Capítulo 1 é esta apresentação.

O Capítulo 2 apresenta conceitos básicos de computadores, imprescindíveis à compreensão dos demais capítulos. São eles: a organização básica de qualquer computador, o software, os sistemas operacionais, famílias de computadores e interconexão de equipamentos.

Pelo exposto, este capítulo é leitura obrigatória ao leitor leigo. O leitor versado nesses assuntos pode passar para o Capítulo 3.

O Capítulo 3 apresenta o Sistema Operacional SOX, iniciando com um painel da história da filosofia deste sistema, que é a história do sistema norte-americano UNIX, no qual o SOX se inspirou, em sua totalidade (dizemos que o SOX é *compatível* com o UNIX). O capítulo termina com uma ilustração de algumas importantes aplicações do SOX.

O Capítulo 4 introduz o leitor na terminologia do SOX. Deste modo, são apresentados conceitos como cursor, arquivo, edição e formatação de arquivos, diretório de arquivos, sistema de arquivos, comandos, teclas especiais (teclado) e outras convenções adotadas.

O Capítulo 5 mostra como o SOX funciona, de vários pontos de vista. A *arquitetura estruturada* do SOX confere-lhe enormes vantagens sobre os competidores como, por exemplo, um alto índice de produtividade no desenvolvimento de programas aplicativos e a possibilidade, sem qualquer necessidade de ajuste, de um programa ser executado em diferentes máquinas, de diferentes fabricantes.

O Capítulo 6 ensina ao leitor como se conectar (e como se desconectar) ao SOX. Apresenta também o Interpretador de Comandos do SOX, que é o meio de comunicação entre o usuário e o SOX.

Os Capítulos 7 e 8 tratam de ensinar como o leitor pode preparar seus documentos (cartas, memorandos, ofícios, artigos etc.) com o auxílio do SOX.

O Capítulo 9 trata de como localizar informações dentro de um texto, como ordenar um texto, além de outras operações básicas com textos.

O Capítulo 10 apresenta o sistema de arquivos do SOX. Simples e elegante, é outro trunfo do SOX na competição com seus concorrentes.

O Capítulo 11 mostra uma série de recursos de que o SOX dispõe para tornar mais rápido e seguro o seu trabalho.

O Capítulo 12 discute o ambiente de um escritório automatizado com o auxílio do SOX.

Finalmente, o Capítulo 13 é dedicado aos recursos do SOX para a impressão de textos dos usuários.

Seguem-se dois apêndices, o Índice Alfabético dos Comandos e o Índice Analítico.

CAPÍTULO 2

INTRODUÇÃO À INFORMÁTICA

A aprendizagem rápida e o uso eficiente de qualquer recurso da informática são facilitados quando se conhece um pouco sobre o sistema de computação empregado e quando se tem noção da *Informática* como um todo. Este capítulo visa familiarizá-lo com a terminologia e conceitos referentes às funções genéricas de computadores, seus componentes e suas potencialidades de aplicação. Isto não só facilitará a absorção do conteúdo do restante do livro, como também proporcionará esclarecimentos introdutórios sobre sistemas de computação, visando preparar o leitor para aproveitar melhor a leitura de manuais técnicos e outras publicações que circulam no efervescente mercado da informática.

O capítulo existe, portanto, para atender ao leitor leigo em Informática. Se você já trabalhou com computadores ou se já está familiarizado com seus conceitos genéricos, sugerimos que salte para o capítulo seguinte e comece o exame do sistema operacional SOX.

2.1 ORGANIZAÇÃO BÁSICA DE COMPUTADORES

Um computador é um tipo de máquina que assiste ao homem em tarefas como processamento de cálculos, armazenamento, recuperação e análise de informações, controle de processos em empresas e instituições diversas (p. ex., fábricas, escritórios, hospitais), emissão de relatórios, diagnósticos etc. As aplicações de computadores são inúmeras, nos mais variados campos de atividade humana.

Existem computadores de diversos tipos, tamanhos e preços. As diferenças entre eles advêm do próprio fabricante, das aplicações pretendidas, da capacidade e da velocidade de processamento, da maneira de manipular informações, do modo como atendem múltiplos usuários (ou apenas um único usuário) e da realização (ou não) de múltiplas tarefas simultaneamente, e do tipo e quantidade de equipamentos acessórios (ou "periféricos", no linguajar técnico da informática).

A similaridade de características e da forma de executar tarefas são usadas para classificar máquinas distintas em famílias. É assim que falamos, por exemplo, de *computadores de grande porte* ("mainframes"), de *minicomputadores*, de *superminicomputadores*, de *supermicrocomputadores* e de *microcomputadores*. Neste capítulo, traçaremos as principais características de cada uma destas classificações. No momento, basta saber que existem muitos computadores diferentes. O SOX pode estar disponível em muitos deles.

Esta variedade de máquinas, contudo, dispõe dos mesmos componentes básicos. Os componentes básicos são classificados genericamente em:

– *Hardware*
– *Software*

Como *Hardware* entendem-se todos os componentes físicos. O *Software* roda no Hardware e faz com que este último seja programado para desempenhar funções distintas. Você pode preferir executar um software que possibilite a sua máquina (hardware) processar textos. Seu filho pode achar mais interessante rodar o software que joga xadrez. Após estas breves observações, examinaremos primeiro os componentes de Hardware e, em seguida, os de Software.

2.2 HARDWARE

Numa configuração típica de um sistema de computação, encontraremos os seguintes componentes de Hardware:

– o computador (propriamente dito)
– dispositivos para armazenamento de dados
– dispositivos de entrada de dados
– dispositivos de saída de dados

Discutiremos brevemente cada um desses componentes, num nível de detalhes apenas suficiente para o enfoque introdutório que pretendemos. Um tratamento mais aprofundado requer cursos ou livros especializados.

COMPUTADOR

Um computador compõe-se de três partes básicas, segundo o esquema da Figura 2.1.

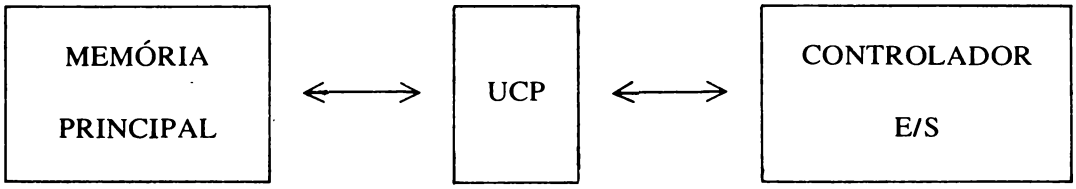


Figura 2.1 Organização Básica de um Computador.

A *Unidade Central de Processamento (UCP)* é o cérebro do computador. É ela que interpreta e executa as instruções em um programa. A UCP pode ser construída com uma simples pastilha de circuito integrado (*CI*), com um conjunto de CIs, ou com várias "placas" contendo CIs interligados por fios e conectores. A construção da UCP é também usada para classificar o computador.

A *Memória Principal* é usada para armazenar os programas e os dados a serem processados pela UCP.

O *Controlador de Entrada e Saída (E/S)* é responsável pela comunicação da UCP com o mundo exterior.

ARMAZENAMENTO DE DADOS

A memória principal do computador é limitada no sentido de que não comporta todos os programas e informações (*dados*) dos seus usuários. Além disso, ela é geralmente volátil, apagando o seu conteúdo quando a energia é desligada. Daí a necessidade de dispositivos de *armazenamento externo* (ou dispositivos de *memória secundária*) de maior capacidade (a memória principal é mais cara) e de natureza não-volátil. Estes dispositivos são os *disquetes* (ou *discos flexíveis*), *discos rígidos removíveis* ou *selados* (*discos Winchester*), e *unidades de fita magnética* (*cassete* ou *cartucho* – este último também conhecido por fita *streamer*). Estes dispositivos armazenam os programas, textos e dados dos usuários.

A UCP transfere informações dos dispositivos de memória secundária para a memória principal quando necessita executá-las ou processá-las. Os softwares que você irá processar com o SOX geralmente são armazenados em discos.

DISPOSITIVOS DE ENTRADA E SAÍDA DE DADOS

A troca de informações entre o computador e seus usuários se faz através dos *dispositivos de entrada e saída*. O mais comum dos dispositivos de entrada de dados é o *teclado*. Quase todo teclado usado em Informática tem o mesmo arranjo de teclas de uma máquina de escrever, com teclas adicionais para realizar funções especiais. Ao pressionar as teclas, você envia caracteres para o computador que os armazena na memória principal (ou secundária). Um *caractere* equivale a uma (qualquer) letra do alfabeto, numeral, sinal de acentuação ou pontuação etc. Freqüentemente, o computador ecoa o caractere na *tela* do *monitor de vídeo* (semelhante à tela de um aparelho de televisão) para que você veja o que digitou. O monitor de vídeo é um exemplo de um dispositivo de saída de dados. O conjunto monitor de vídeo mais o teclado pode ser chamado de *terminal*.

Um outro dispositivo de saída é a *impressora*. É na impressora que o computador lista (de forma permanente) os resultados do processamento realizado. No monitor de vídeo, você vê os resultados; na impressora, você tem uma cópia (que pode arquivar, ou enviar para alguém).

Em capítulos adiante, você aprenderá como utilizar o seu terminal e a(s) impressora(s) na sua máquina SOX.

A qualidade e a velocidade da impressão dependem do tipo de impressora. Para qualidade igual à de uma máquina de escrever, você precisa de uma impressora com *Qualidade de Carta* (*Letter Quality - LQ*). Essas impressoras são mais caras e mais lentas que as impressoras de *Matriz de Pontos*. Estas últimas formam os caracteres através da composição de vários pontos. É possível conseguir qualidade de impressão *Quase-Qualidade de Carta* (*Near Letter Quality - NLQ*) com matriz de pontos, fazendo-se a sobreimpressão de cada caractere. A velocidade de impressão, neste caso, é reduzida. Bom mesmo é ter uma impressora a laser. Estas impressoras produzem textos e documentos de alta qualidade mas são caras e poucos fabricantes brasileiros se interessam em ofertá-las.

Mesmo com impressoras matriz de pontos, sem a facilidade NLQ, você pode lançar mão de recursos para reforçar a impressão de cada caractere (diminuindo a separação entre os pontos da matriz) possibilintando um acabamento mais nítico.

Verifique as características da impressora que você tem disponível e leia seu manual para aprender como operá-la.

No caso de microcomputadores de *8* e *16 bits* (como os micros que utilizam *MSX, CP/M, MS-DOS/SISNE* e o próprio SOX) todo o hardware cabe numa mesa. Sistemas com máquinas de maior porte exigem uma sala (talvez um prédio). Num microcomputador pessoal (*Personnal Computer - PC*), a UCP e a memória principal geralmente se alojam numa placa, e o controlador de E/S em outra, dentro do gabinete (veja Figura 2.2). Você pode rodar o SOX em um microcomputador do tipo *IBM-PC*. Neste caso, você adquiriria uma placa adicional, chamada pela *COBRA* de *SOX-PC*, que também se encaixaria dentro do gabinete.

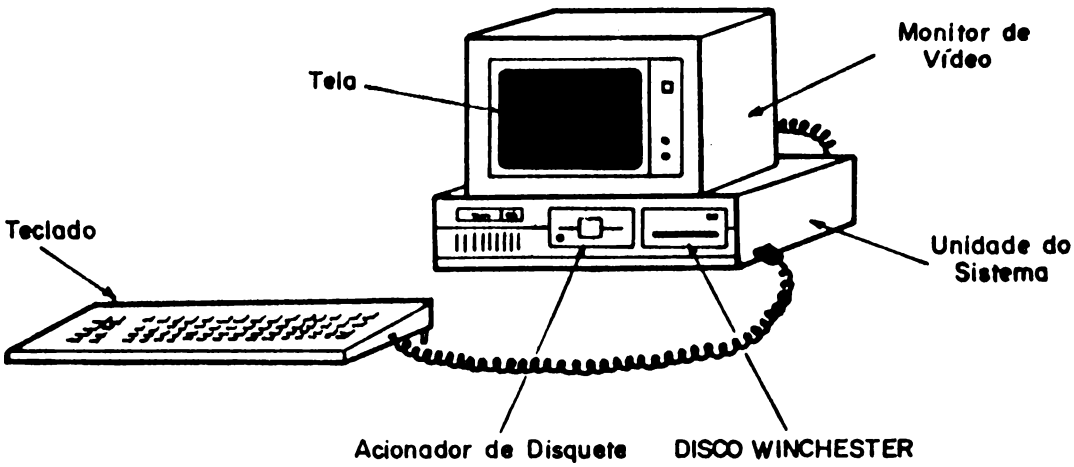


Figura 2.2 Ilustração do Hardware de um Microcomputador tipo PC.

No gabinete também estão fixados (na parte frontal) as unidades de disquete e/ou o Winchester. Os dispositivos de entrada e saída de dados (teclado e monitor de vídeo) comumente são destacáveis para conforto do usuário.

Os micros domésticos (tipo MSX e até os compatíveis com o *Apple*), via de regra, têm o teclado moldado no gabinete. O monitor de vídeo pode ser uma simples TV (com tela comportando 24 linhas e 40 colunas, ao contrário dos monitores profissionais com 24 linha e 80 colunas). Por questões de custo, os acionadores de disquete, nestes micros, são separados do gabinete. Muitos dispõem de um gravador cassete como dispositivo de memória secundária.

Se você trabalha com equipamento maior (p.ex., supermicro SOX ou "mainframe"), o hardware à sua disposição consistirá provavelmente de um terminal de vídeo (teclado + monitor de vídeo), o qual se conecta ao computador remoto. Seria instrutivo visitar o *Centro de Processamento de Dados (CPD)* da sua empresa para conhecer a "sua" máquina. Isto servirá para ilustrá-lo um pouco mais nos conceitos básicos da Informática.

2.3 SOFTWARE

O conceito de *software* é mais evasivo à nossa percepção, possivelmente porque, ao contrário de hardware, não é palpável. Você não vê o software, apercebe-se apenas das conseqüências de sua execução. Software pode ser comparado a Música. Você compra um disco (que seria equivalente ao disquete) ou fita, põe no equipamento de som (equivalente ao computador) e ouve. A música está escondida no disco; o software está escondido no disquete.

Um sinônimo para software é *programa*. Um programa é composto de *instruções* que, em conjunto, realizam tarefas específicas (p. ex., processamento de textos). As instruções são fornecidas numa *linguagem de programação*, inteligível ao computador. As instruções são transmitidas da memória secundária para a principal e, então, interpretadas pela UCP.

Os conceitos acima podem ser melhor fixados fazendo-se paralelo com um ser humano ao preparar uma feijoada. O humano corresponderia ao computador e seu cérebro à UCP. Nosso humano cozinheiro segue as instruções constantes numa receita de um livro de culinária ou que já se encontre na sua memória. A receita equivale ao software (que realiza uma tarefa específica – fazer feijoada) e o livro de receita, à memória secundária, onde o "software" é armazenado permanentemente e de onde é requisitado para reutilização. Note que a receita deve ser fornecida numa linguagem inteligível ao cozinheiro, digamos português, caso contrário, esqueça a feijoada. Talvez agora a coisa faça mais sentido. Examinemos, pois, a questão do software.

Numa primeira classificação, o componente "software" divide-se em:

– *Software Básico*
– *Software Aplicativo*

Software Aplicativo realiza tarefas específicas (a receita da feijoada seria um software aplicativo) como contabilidade, controle de estoque e processamento de textos.

Software Básico faz a interface entre o hardware e os softwares aplicativos. É o software básico que controla o hardware e executa as rotinas para sua manutenção/supervisão. Em software básico, incluem-se as facilidades das *linguagens de programação* e do *sistema operacional*. O SOX é um sistema operacional.

LINGUAGENS DE PROGRAMAÇÃO

Algumas facilidades relativas à linguagem de programação acompanham a sua máquina SOX.

As linguagens de programação são usadas por *programadores* para escreverem programas a serem executados por computadores. Todo computador "entende" apenas a sua linguagem nativa, de baixo nível, chamada *linguagem de máquina*. Assim, para que o computador execute um programa escrito numa linguagem de programação, é necessário traduzi-lo para sua linguagem de máquina. Softwares (básicos) especiais, chamados de *Compiladores* e *Interpretadores* são responsáveis pela tradução.

As linguagens de programação diferem ente si em termos de facilidade de uso, abrangência e elegância na solução de problemas diversos, disponibilidade de compiladores para várias máquinas etc. Existem várias linguagens de programação disponíveis (da mesma forma que existem vários idiomas no mundo). *BASIC, FORTRAN, COBOL, PASCAL* e *C* são nomes de algumas linguagens existentes.

A linguagem de programação C é uma das mais populares hoje em dia, dispondo de compiladores para uma vasta gama de equipamentos. Um software escrito em C pode pois ser transportado para qualquer máquina com um compilador C. O SOX foi escrito em C. Isto explica porque ele pode executar desde em micros pessoais até em máquinas de grande porte.

As linguagens de programação são importantes somente para programadores. Usuários de software não precisam se preocupar com seus detalhes. As explicações acima lhe servem para entender melhor o funcionamento de computadores e saber por que o uso do SOX possibilita independência de máquina.

SISTEMA OPERACIONAL

O sistema operacional é o intermediário entre o hardware do computador e os programas aplicativos. É o sistema operacional que escalona e controla os acessos aos componentes de hardware do computador. O sistema operacional atua em favor do(s)

programa(s) de aplicação, recebendo pedidos de acesso a dispositivos ou a outros componentes de hardware. Os pedidos são traduzidos em comandos ou diretivas a serem executados pelo hardware do computador. Como já foi dito, o SOX é um sistema operacional.

Um exemplo com um pedido simples para o SOX ilustra as ações do sistema operacional. Suponha que você dissesse ao computador "Quero saber a data e hora corrente" (a maneira correta de dizer isto é discutida no Capítulo 6). O SOX examinaria o pedido e enviaria comandos que acionariam a unidade de disco em busca do programa chamado *date*, que atende ao pedido feito. Ao encontrá-lo, o programa é trazido para a memória principal e, na tela de seu terminal, apresentará a informação pedida. Todas estas atividades são realizadas com supervisão do sistema operacional.

Os sistemas operacionais classificam-se em:

– *tipo lote*
– *tipo interativo*

Em um sistema *tipo lote*, o usuário não interage com seu programa (ou tarefa), quando em execução. Uma vez submetido, o programa executa, fora de contato com o usuário, até ser concluído, quando o usuário recebe os resultados. Alguns sistemas tipo lote são também *multitarefa*, executando vários programas "simultaneamente". Na verdade, a UCP devota um pouco de atenção (durante um intervalo de tempo pré-especificado) a um programa e passa a atender ao próximo etc. Os primeiros sistemas operacionais eram do tipo lote e ainda hoje prestam serviços em algumas instituições de pesquisa e nos CPDs de grandes empresas.

Os sistemas operacionais do *tipo interativo* respondem quase que instantaneamente aos comandos/estímulos dos usuários. Os sistemas operacionais mais novos, inclusive o SOX, são invariavelmente deste último tipo.

Os sistemas operacionais podem ainda ser classificados em:

– *monousuário*
– *multiusuário*

Um sistema operacional *monousuário*, como a própria classificação implica, admite apenas um usuário por vez. Em geral, um sistema monousuário é também *monotarefa*, não permitindo que mais de uma tarefa (p.ex., processar texto e fazer contabilidade) seja executada simultaneamente. A maioria dos microcomputadores de 8 e 16 bits disponíveis tem sistema operacional interativo, monousuário e monotarefa. O *SISNE* e o *MS–DOS* são sistemas operacionais monousuário, monotarefa que executam em computadores de 16 bits do tipo IBM-PC.

Um sistema operacional multiusuário permite que vários usuários, desde seus terminais, utilizem o computador para realizar tarefas distintas, simultaneamente. Por exemplo, um terminal pode estar sendo usado para fazer processamento de textos, um outro para atualizar as fichas no cadastro de funcionário de uma empresa, um outro para rodar e verificar uma planilha de custos etc. Sistemas operacionais multiusuários comumente estão disponíveis em máquinas maiores, a partir dos microcomputadores de 16 bits e são empregados principalmente em ambientes comerciais.

O SOX é um sistema operacional multiusuário e multitarefa que está disponível para equipamentos de 16 e 32 bits. Esta disponibilidade cobre uma vasta gama de equipamentos, o que constitui em grande vantagem para o usuário, pois reduz a necessidade de retreinamento quando ele troca de máquina. Este livro inicia o treinamento no SOX na maioria de suas atividades com o seu computador. O capítulo seguinte traz mais informações sobre este sistema operacional, inclusive um breve histórico do seu desenvolvimento, que foi genuinamente brasileiro.

2.4 FAMÍLIAS DE COMPUTADORES

Expusemos os conceitos básicos de qualquer computador. Isto basta para iniciarmos o aprendizado do uso do SOX. Entretanto, como o propósito do capítulo é esclarecer termos básicos da Informática, no tocante a computadores, restam algumas informações sobre as várias famílias de máquinas. O SOX está disponível para algumas delas. Além disto, menção a essas famílias é feita com freqüência em jornais, revistas e palestras especializadas. O conhecimento das características das principais famílias possibilitará uma melhor compreensão da literatura da informática e um enquadramento de sua máquina de trabalho no universo das famílias. Este enquadramento pode abrir-lhe perspectivas a considerar quando da necessidade de troca ou aquisições futuras.

As famílias que discutiremos são:

– *microcomputadores*
– *supermicrocomputadores*
– *minicomputadores*
– *superminicomputadores*
– *computadores de grande porte*

Até o início da década de 80, *microcomputador* era o nome genérico para máquinas de 8 ou 16 bits com sistema operacional monousuário; *máquina de grande porte*, ou "mainframe", em inglês, referia-se (e ainda se refere) a computadores de grande tamanho

físico e de alta capacidade de processamento (medida geralmente em *instruções por segundo* pela UCP – *ips*). Falava-se também (e ainda se fala) em *minicomputadores* e *superminicomputadores*, para indicar máquinas de 16 ou 32 bits multiusuário e máquinas de 32 bits mais possantes que os minis e menos possantes que os mainframes.

Com o avanço tecnológico, essas definições (informais) tornaram-se nebulosas e foram revistas, com novas famílias sendo introduzidas para acobertar equipamentos antes inexistentes. Os micros ficaram muito possantes e surgiram os chamados *supermicros*. Hoje, a diferenciação entre as famílias de micros e de minis é feita em termos exclusivos da tecnologia empregada na sua construção. Um *micro* é uma máquina cuja UCP é construída com microprocessadores (uma ou duas pastilhas de circuito de integração em escala muito grande – *VLSI*). A UCP de um *mini*, em contraste, emprega lógica "discreta", ou seja, é construída com algumas dezenas de pastilhas de circuito com integração de média escala (*MSI*). O prefixo "super" (supermicro, supermini) é atribuído às máquinas com desempenho superior, mais memória (principal), e com sistema operacional multiusuário.

Hoje em dia, adotam-se as definições seguintes:

Microcomputador – computador com UCP baseada em microprocessador VLSI de 8 ou 16 bits, com até 1 milhão de octetos (1 *Megaocteto* ou 1*Mo* onde 1 *octeto* corresponde a 8 bits e 1 *bit* é a unidade básica de informação) de memória, sistema operacional monousuário (ou multiusuário, atendendo a apenas 2 ou 3 usuários) e com capacidade de processamento na faixa de 100 a 500 mil ips (100 a 500 *Kips*).

Exemplos de microcomputadores no mercado nacional incluem o *HOTBIT* da *Epcom/Sharp*, *Expert* da *Gradiente*, *Cobra C-210*, os compatíveis com o *Apple* e os compatíveis com o *IBM-PC*.

Supermicrocomputador – tem UCP baseada em microprocessador(es) *VLSI* de 16 ou 32 bits, mínimo de 1 Mo de memória, sistema multiusuário e com capacidade de processamento entre 500 Kips e alguns *Mips* (milhões de ips). Um supermicro pode ser tão possante quanto um mini, supermini ou até mainframe. O supermicro imprensa, assim, os minis e os superminis, havendo quem prenuncie, por causa disso, o desvanecimento destas duas últimas famílias. Os computadores *ED-680*, *DGR-8000*, *EBC série 32020*, *COBRA XM* são exemplos de supermicros no mercado nacional.

Minicomputador – usa tecnologia discreta MSI/LSI na UCP de 16 ou 32 bits, tem tamanho de memória na faixa de 250ko a dezenas de Mb, dispõe de sistema operacional multiusuário (suportando cerca de 15 usuários) e capacidade de processamento entre 500 kips e 1 Mip. Os computadores da *série 500 da Cobra* são minis.

Superminicomputadores – A UCP usa tecnologia MSI/LSI de 32 bits, tem, no mínimo, 1mo de memória, sistema operacional multiusuário e capacidade de vários Mips. O computador *MX 850 da Elebra* é um supermini.

Mainframe – computador com UCP baseada em tecnologia MSI/LSI de 32 bits a 60 bits, vários Mo de memória, sistema multiusuário, com capacidade de vários Mips. Os computadores *IBM 3090* e *4341* são exemplos de mainframes no mercado brasileiro. O 4341 seria classificado como um mainframe de "Pequeno Porte" (com definição semelhante à de supermini).

A classificação acima, com suas várias nuances de comparação entre máquinas de várias famílias, é valiosa apenas quando serve para indicar a habilidade de um certo equipamento em atender aplicações específicas. O avanço tecnológico e o conseqüente barateamento de computadores pequenos (micros) expandiu o horizonte de aplicações, tornando difusa a antiga separação entre sistemas. Hoje, é comum encontrar equipamentos de várias famílias numa mesma empresa. Hoje, mais do que nunca, é importante dispor de software multifamília, para facilidades dos usuários. O SOX foi desenvolvido para rodar em equipamentos de todas as famílias de computadores discutidas. Uma preocupação a menos para você.

2.5 COMUNICAÇÃO ENTRE COMPUTADORES

Num ambiente onde convivem muitos computadores, como talvez seja o caso na sua empresa, logo surge a necessidade de se estabelecer comunicação entre eles. Para tanto, os equipamentos devem ser, primeiro, ligados uns aos outros. A solução que sua empresa adotará para ligar seus vários equipamentos dependerá da extensão da área na qual eles se dispersam, dos recursos de comunicação usáveis (p. ex., rede telefônica) para interconectá-los, das famílias de computadores e dos sistemas operacionais envolvidos, e, finalmente, do tipo e freqüência da comunicação pretendida. Consideremos algumas situações hipotéticas para ilustrar o assunto.

Situação 1: sua empresa tem na matriz, no Rio de Janeiro, um computador de grande porte (*IBM* ou *BURROUGHS*, por exemplo) e na filial de Salvador um supermicro SOX. O supermicro é usado na automação da filial (fazendo seu controle de estoque, contas a receber/pagar, contabilidade etc.). Mensalmente, o supermicro prepara um relatório sumarizando o movimento da filial que deve ser encaminhado à matriz para composição do relatório geral sobre a empresa, a ser apreciado pela diretoria. Esta é uma situação de comunicação infreqüente que pode ser atendida pela rede telefônica, pelo uso de equipamentos especiais chamados de *MODENS*. O supermicro é ligado ao "mainframe" via

uma chamada telefônica normal, com o supermicro SOX fazendo o papel de um terminal remoto do mainframe (vide Figura 2.3). Esta solução, contudo, pode não ser atraente para comunicação freqüente, devido ao custo da chamada telefônica a longa distância.

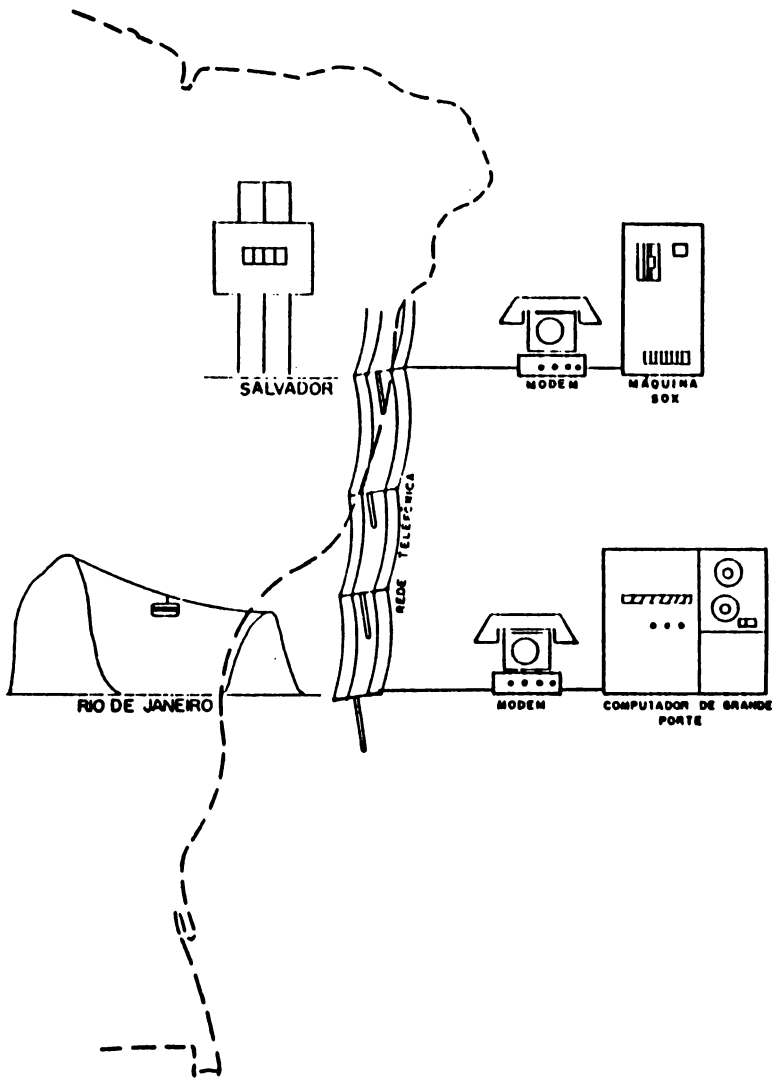


Figura 2.3 Conexão Remota Supermicro SOX – Mainframe via Rede Telefônica.

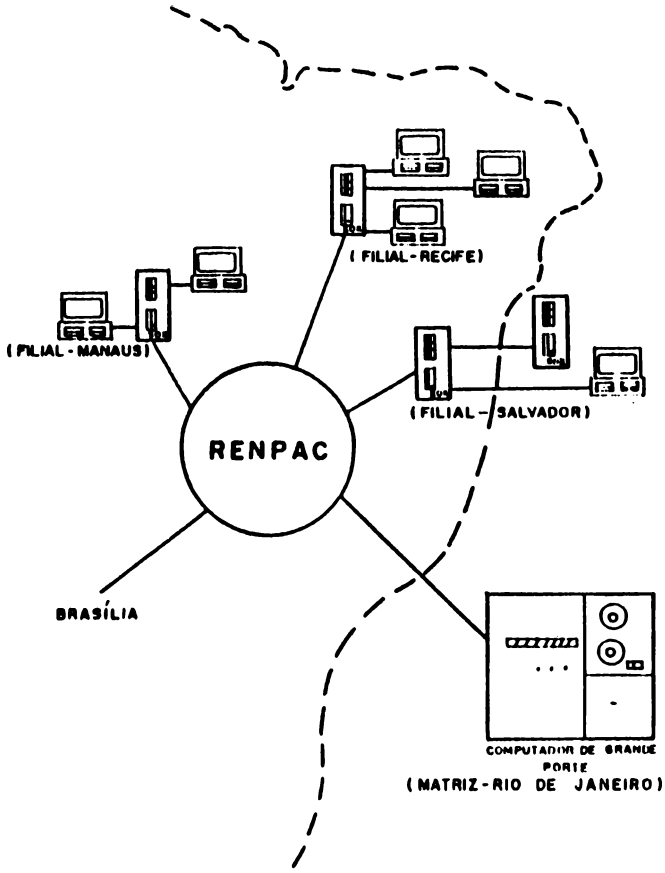


Figura 2.4 Conexão Supermicro SOX – Mainframe via RENPAC

Situação 2: sua empresa tem filiais espalhadas por todo o território nacional, cada uma com seus próprios computadores (talvez em grande quantidade), que fazem acessos freqüentes e rápidos ao mainframe na matriz no Rio de Janeiro. Os acessos servem para verificar a validade dos cartões de crédito que a empresa distribui a seus inúmeros clientes. Agora, a solução mais racional será utilizar a *Rede Nacional de Comutação de Pacotes (RENPAC)* da *EMBRATEL*. Um supermicro SOX poderia concentrar o tráfego de dados gerado (ou distribuído) pelos vários equipamentos em cada filial e enviar (ou receber) o tráfego concentrado, via RENPAC, para o mainframe. O supermicro atuaria como um intermediário junto à RENPAC – que é a rede especializada em comunicação entre computadores – simplificando a conexão dos vários equipamentos (vide Figura 2.4).

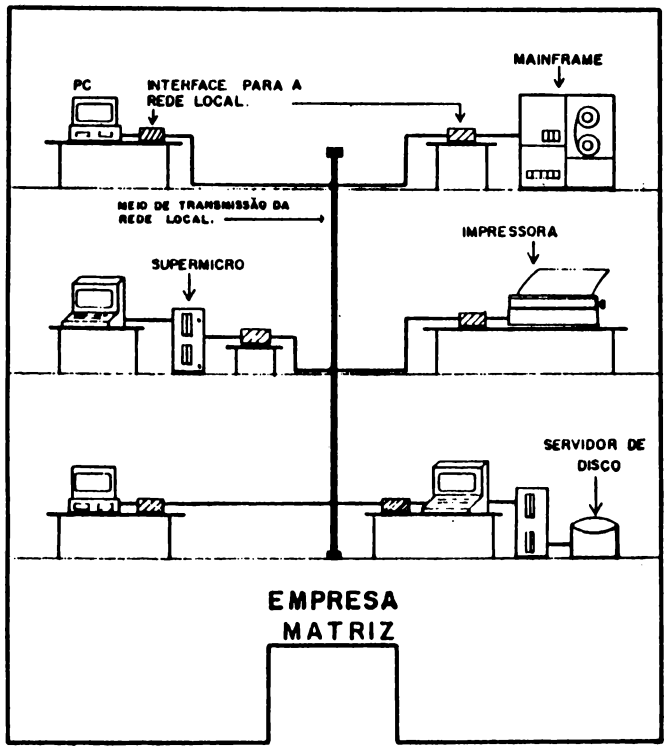


Figura 2.5 Ilustração de uma Rede Local em um Prédio de Escritórios

Situação 3: sua empresa deseja automatizar os escritórios espalhados pelos 12 andares do prédio da matriz, isto é, quer fazer *Automação de Escritório*. A automação de escritórios possibilitará a preparação e a troca de memorandos e documentos via computadores e terminais (reduzindo o fluxo de papel entre os funcionários), o arquivamento e recuperação mais racionais de informações (através de um *servidor de arquivos* ou de disco), o compartilhamento mais eficiente de periféricos (por exemplo, uma impressora qualidade de carta sendo compartilhada entre os escritórios do décimo primeiro e décimo segundo andares) etc. Na matriz existem, além do mainframe da situação 1, micros IBM-PC e alguns supermicros com o SOX. O suporte de comunicação nesta situação pode ser oferecido por uma *Rede Local* de computadores – uma mini-rede para comunicação de dados, como se fosse uma mini RENPAC de abrangência limitada (ao prédio) e de configuração simples (vide Figura 2.5).

Poderíamos continuar a identificar situações e soluções por muito tempo. Mas não é o objetivo deste livro. A discussão nesta seção tem três objetivos principais:

- Aumentar sua percepção da Informática introduzindo a área de comunicação de dados e, mais especificamente, as *Redes de Computadores*.
- Discutir brevemente as potencialidades do *Teleprocessamento* e serviços da *Telemática*.
- Apontar as capacidades do SOX na parte de comunicação.

Teceremos agora alguns comentários acerca de cada um dos três objetivos.

A possibilidade de interconectar computadores dá origem a uma vasta gama de aplicações antes inviáveis (como o *correio eletrônico* entre usuários em máquinas distintas e distantes) e leva a uma maior eficiência no uso de hardware, software e informações (compartilhamento de recursos caros ou informações em uma determinada instalação). Para interconectar equipamentos, podemos utilizar a rede telefônica, canais para comunicação privada de dados (rede TRANSDATA da EMBRATEL) e, mais recentemente, as *Redes de Computadores*. Uma rede de computadores que cobre grandes distâncias é chamada de *Rede a Longa Distância* (ex: RENPAC); uma rede restrita a uma pequena área é chamada de *Rede Local*. Para você operar uma rede de computadores necessita de hardware e software de comunicação (ou *Protocolos de Comunicação*). Revistas e jornais em informática trazem freqüentes artigos sobre o assunto; existem também alguns livros, em português, que descrevem as redes nos mínimos detalhes. Você deve procurá-los para melhor conhecimento. A EMBRATEL é uma ótima fonte de informação sobre as redes a longa distância TRANSDATA e RENPAC, com equipes técnicas que podem aconselhar sua empresa no projeto de interconexão. Alguns fabricantes nacionais como a CETUS, AMPLUS, EDEN etc. têm redes locais como principal produto e, certamente, enviarão brochuras e panfletos se você os requisitar. Ciente do que a Informática oferece como recursos de interconexão, explore-os quando conveniente ou quando sua empresa ventilar planos para projetos de rede ou de teleprocessamento. Passemos ao objetivo dois.

Uma vez interconectados os equipamentos, o usuário, a partir de um terminal, pode ter acesso a recursos remotos para listar um relatório em uma impressora rápida, de qualidade (da qual você não dispõe), ou para executar um software aplicativo (processar a folha de pagamento, por exemplo). Em outras palavras, você pode fazer processamento à distância (teleprocessamento). Existem algumas aplicações em teleprocessamento que são bastante comuns, como transferir informações ou arquivos, submeter uma tarefa para execução remota *(Remote Job Entry - RJE)*, conectar um computador a outro, como se o primeiro fosse um terminal do último *(Emulação de Terminal)*. Estes são serviços básicos

que muitas empresas necessitam e que, se oferecidos no mercado, facilitarão a vida de programadores e usuários. Outras aplicações mais recentes são os serviços de telemática. Incluem-se aí o *correio eletrônico,* o *teletexto* (uma espécie de telex mais elaborado), o *vídeotexto* (troca de informações incluindo imagem) etc. A concessionária da TELEBRÁS de seu estado talvez possa fornecer-lhe mais detalhes; a telemática freqüenta, também, as principais publicações em informática no país.

Por fim, o objetivo três. Como já foi dito, o SOX é um sistema operacional multiusuário, multitarefa e de propósito geral – podendo atender às necessidades de comunicação com outras máquinas. Como tal, uma máquina SOX tem facilidades embutidas que permitem a comunicação entre os seus usuários, o compartilhamento de discos ou impressoras e comandos para transferir arquivos e para submissão remota de tarefas em outros equipamentos. O uso de alguns destes recursos – aqueles que requerem o suporte de meio de transmissão à distância (p. ex., as redes) – escapa da abrangência deste livro. O exame de tais recursos é facilitado pelo conhecimento dos conceitos básicos de operação e uso do SOX em uma única máquina e são justamente estes conceitos que abordaremos nos capítulos adiante. A leitura do material neste livro é um preparatório para assuntos mais avançados, como as facilidades de comunicação remota do SOX. Você verá, contudo, como utilizar o SOX em certas tarefas de comunicação entre usuários e de compartilhamento de periféricos num ambiente local, em torno de uma única máquina SOX. Ambientes deste tipo são muito comuns no mercado, principalmente em empresas que iniciam a automação de suas atividades. O ambiente a que nos referimos é ilustrado na Figura 2.6.

Observe a Figura 2.6. Como você vê, à máquina SOX se conectam terminais de vídeo, computadores de menor porte (como o IBM-PC), impressora(s) e unidades de disco e de fita. Possivelmente, a máquina SOX poderia estar ligada também a computadores de grande porte ou a redes de computadores. Estas últimas ligações não são ilustradas porque não são objeto de estudo neste livro. Como ilustrado, a máquina SOX serve de elemento centralizador entre alguns equipamentos *locais* e assim pode ser esteio de comunicação para a automação de alguns escritórios! O SOX permite que mais de um usuário utilize a impressora ou o disco e ainda que os usuários conversem ou troquem mensagens através de seus terminais/micros. Os microcomputadores podem ser integrados ao ambiente fazendo-os emular terminais SOX. No modo de operação isolada (*stand-alone*), os micros tipo IBM-PC podem rodar o sistema operacional MS-DOS ou SISNE, executar os softwares aplicativos específicos a estes sistemas e, quando ligarem-se às máquinas SOX, transferirem os resultados para outros PCs. Para as empresas que já possuem PCs, existem no mercado aplicativos como planilhas de cálculo, processadores de textos, sistemas de entradas de dados etc. que rodam de forma idêntica sob o MS-DOS e o SOX. Estes

aplicativos são muito vantajosos na automação de escritórios em um ambiente que integra PCs e máquinas SOX.

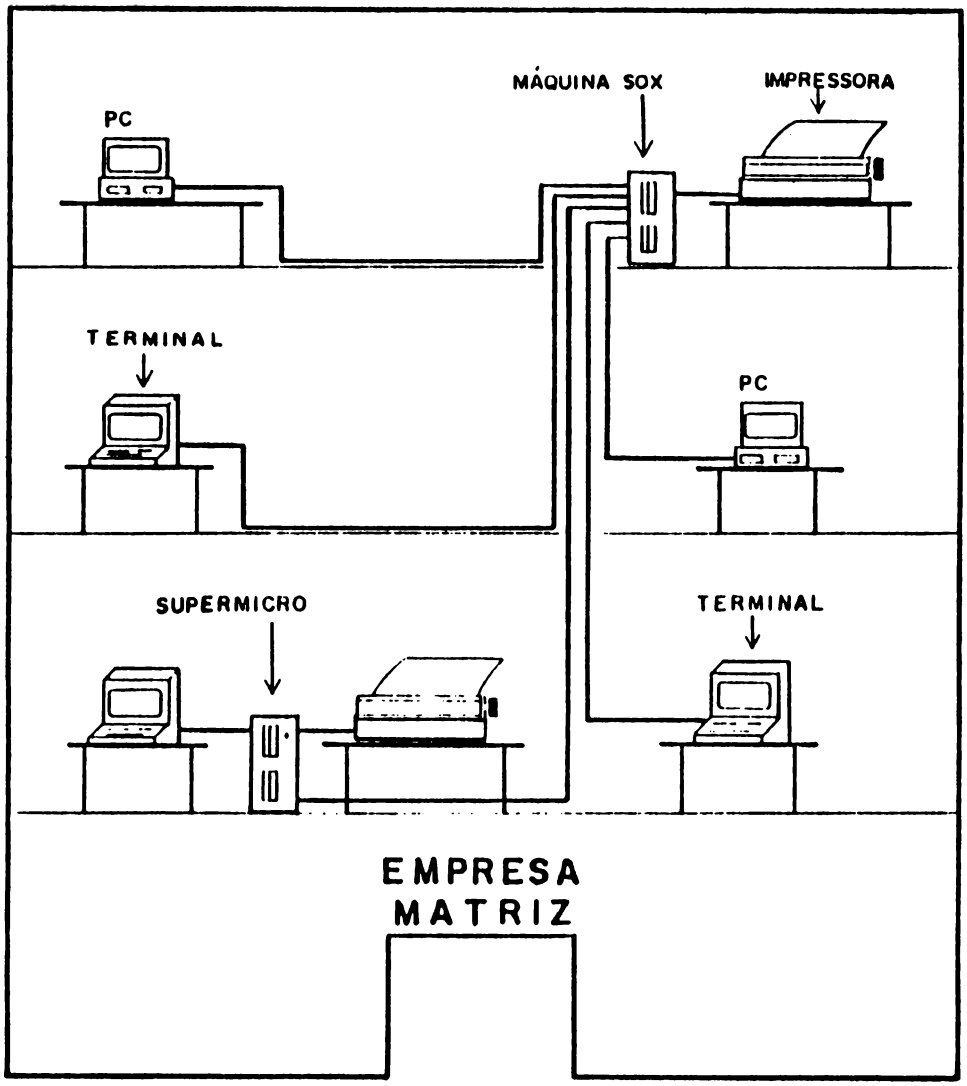


Figura 2.6 Máquina SOX Centralizando Comunicações em um Prédio de Escritórios.

Após estas observações, você talvez já tenha concluído que a máquina SOX, como mostrada na Figura 2.6, serve de rede local entre seus usuários, oferecendo os recursos de software para troca de mensagens e compartilhamento de periféricos. Estes recursos são estudados neste livro, com capítulos dedicados às facilidades de comunicação (Capítulo 12) e ao uso de impressoras (Capítulo 13). O uso do disco é tratado de forma implícita nos vários capítulos que abordam o *Sistema de Arquivos SOX* (introduzido no Capítulo 4).

Esperamos que as cinco seções deste capítulo tenham lhe motivado à leitura de material sobre os vários aspectos da informática e de seu casamento com as telecomunicações, a chamada telemática. Cabe-lhe agora a tarefa de aprofundamento nos assuntos de seu interesse. Um destes é sobre sistemas operacionais – a prova é a sua paciência conosco neste livro – mais especificamente, o SOX. Os capítulos restantes trazem material que deverá satisfazer a suas necessidades introdutórias e de uso cotidiano do sistema operacional SOX. O capítulo seguinte traça um breve histórico do desenvolvimento do sistema, bem como descreve sua compatibilidade com o sistema operacional Unix, padrão internacional para uma vasta gama de equipamentos.

CAPÍTULO 3

APRESENTANDO O SISTEMA OPERACIONAL SOX

O capítulo anterior se propôs a familiarizar o leitor com os principais conceitos e termos da Informática, a fim de simplificar a leitura de capítulos subseqüentes. O material do Capítulo 2 basta para você prosseguir com o treinamento para uso do sistema operacional SOX. Talvez lhe interesse saber um pouco da história do SOX e sua relação com o sistema operacional UNIX, considerado padrão de fato internacional numa vasta gama de equipamentos. Não que este conhecimento seja imprescindível para sua maestria no trato com o SOX. Se você desejar, pode passar imediatamente para o Capítulo 4 e iniciar contato com o SOX. Acreditamos, contudo, que o conhecimento sobre o histórico e o potencial do SOX – UNIX o tornará um usuário SOX diferenciado, com aquele "algo mais" particular de pessoas que detêm não apenas o "know-how" mas também o "know-why' (o "saber por quê"). Além disso, o capítulo é curto, podendo ser lido em alguns minutos.

Para que você possa vislumbrar de forma mais coerente as razões para o surgimento do SOX, as vantagens de sua utilização e o seu potencial em aplicações, discutiremos primeiro o histórico e as razões para a popularidade do sistema operacional Unix, a nível internacional e no Brasil. Esta discussão é relevante porque o SOX é compatível com este poderoso sistema operacional.

3.1 O SISTEMA OPERACIONAL UNIX

O Unix tem suas origens no sistema operacional Multics, projetado em meados da década de 60. Foi o Multics um dos primeiros sistemas operacionais multiusuários e multitarefas. Relembrando nossa discussão do capítulo anterior, isto equivale a dizer que o Multics interagia com seus vários usuários de forma conversacional (em contraste com os antigos sistemas tipo batch) e permitia-lhes executar vários programas simultaneamente (ao contrário de sistemas tipo monousuário que executam apenas uma tarefa do único usuário).

O projeto do Multics foi realizado pelo Massachusets Institute of Technology (MIT), pela General Eletric (GE) e pelos Laboratórios Bell (Bell Labs) da American Telephone and Telegraph (AT&T). Pretendia-se que o Multics fosse a última palavra em sistemas de tempo compartilhado. Em 1969, uma versão primitiva do sistema executava em um computador GE645. O sucesso inicial, contudo, não bastou para a coesão das instituições engajadas no projeto.

Duas razões principais foram citadas para explicar a saída dos Bell Labs do projeto. Primeiro, três instituições com objetivos díspares dificilmente alcançariam uma solução satisfatória para cada uma delas (a MIT fazia pesquisa, a AT&T monopolizava os serviços de telefonia americanos e a GE queria vender computadores). A segunda razão é que os participantes sofriam a "síndrome do segundo projeto" e, por isto, queriam incluir no Multics tudo que tinha sido excluído dos sistemas experimentais até então desenvolvidos.

O cientista Ken Thompson estudava o Multics quando o Bell Labs, onde trabalhava, retirou-se do projeto. Thompson começou então a desenvolver um novo sistema operacional, parcialmente baseado nas idéias do Multics, porém, muito menor. Nascia o UNIX, assim batizado por Brian Kernighan (outro pesquisador do Bell Labs), em referência ao ambiente unificado de programação e desenvolvimento que o sistema deveria proporcionar.

Surgia o Unix, baseado no Multics e originalmente escrito em linguagem de montagem do computador PDP-7.

A EVOLUÇÃO DO UNIX

Em pouco tempo, o Unix despertava a atenção de outros pesquisadores dos Bell Labs, dos quais merece destaque Dennis Ritchie por sua grande influência na evolução do sistema. Um marco importante foi estabelecido em 1973, quando Ritchie reescreveu o Unix na linguagem de alto nível C (desenvolvida por ele), para um computador PDP-11, o

minicomputador mais popular já comercializado. Pela primeira vez, um sistema operacional era escrito numa linguagem de alto nível e isto é, possivelmente, a principal razão para a rápida aceitação do Unix por usuários fora dos Bell Labs.

Na época, a AT&T detinha o monopólio de exploração do serviço telefônico nos Estados Unidos. Por isto, era-lhe vetada a comercialização de produtos de Informática. A AT&T não podia anunciar, vender ou suportar o Unix no mercado. A empresa podia, contudo, licenciar o sistema para universidades, órgãos de governo e centros de pesquisa para fins instrucionais. A AT&T, como não podia oferecer suporte, entregava os próprios programas-fontes do sistema, escritos na linguagem C.

Imagine, o leitor, o interesse de pesquisadores e alunos de computação: eles recebiam um sistema operacional pequeno, escrito numa linguagem de alto nível e com código-fonte disponível. Era uma maravilha instrucional e prática. Muitos usaram o sistema, tiveram que estudar seus programas-fontes (pois não havia assistência da AT&T, em caso de problemas) e eventualmente partiram das instituições de pesquisa e ensino para o mercado de trabalho. As empresas que acolheram muitos ex-alunos treinados no Unix logo começaram, por pressão dos seus recém-contratados, a requerer o Unix em computadores que compravam para automatizar suas atividades.

Em 1977, a AT&T começou a fornecer o Unix para instituições comerciais. A abertura do mercado comercial para o Unix deve muito a Peter Weiner – cientista de Yale e fundador da Interactive Systems Corporation. Weiner conseguiu da AT&T então já desnudada de seu monopólio nas comunicações e liberada para atuação no mercado de Informática, licença para transportar e comercializar o Unix para o computador Interdata 8/32 para ambiente de automação de escritório. O Unix saía da linha das máquinas PDP da Digital Equipment Corporation (DEC), demonstrando sua característica, emprestada pela linguagem C, de fácil migração (transporte) para outros computadores.

O sucesso da Interactive de Wiener com seu produto provou que o Unix era vendável e encorajou outros fabricantes a seguirem o mesmo curso. Iniciava-se a abertura do chamado "mercado Unix".

A EXPANSÃO DO MERCADO UNIX

Com a crescente oferta de microcomputadores, outras empresas transportaram o Unix para novas máquinas. Devido à disponibilidade das fontes do Unix e à sua simplicidade, muitos fabricantes alteraram o sistema, gerando variantes "personalizadas" a partir do Unix básico licenciado pela AT&T.

De 1977 a 1981 a AT&T integrou muitas variantes no primeiro sistema Unix comercial chamado de System III. Em 1983, após acrescentar vários melhoramentos ao System III, a AT&T apresentava o novo Unix comercial, agora chamado de System V. Hoje, o Unix System V é o padrão internacional de fato no mercado Unix, constando das licitações de compra de equipamentos de grandes clientes na América, Europa e Ásia.

A popularidade do Unix, todavia, não é restrita aos círculos de fabricantes e, caso fosse, não teríamos interesse nesta discussão. O Unix traz muitas vantagens para os usuários: um poderoso ambiente para desenvolvimento e execução de software; facilidade de agrupar seus comandos para gerar soluções de software (sem que o usuário tenha que desenvolvê-los), e a independência de fabricante – permitindo ao usuário mudar de fornecedor sem maiores transtornos, pois o Unix está disponível em vários outros equipamentos.

Em 1985, o número de equipamentos Unix no mundo atingia cerca de 200.000. Estas máquinas eram de todos os tipos, desde microcomputadores até computadores de grande porte, produzidas por inúmeros fabricantes ao redor do globo. Fabricantes de peso oferecem Unix nas suas linhas de produtos. Incluem-se aí empresas como a IBM, AT&T, Hewlett-Packard, DEC, Data General, Cray Research, Phillips, Nixdorf, Olivetti etc. Nenhum outro sistema operacional tem ou teve a popularidade do Unix nem suas excelentes perspectivas de expansão de mercado. É a "onda Unix" que se faz sentir em todo o mundo.

3.2 O SISTEMA OPERACIONAL X – SOX

A onda Unix logo chegou também ao Brasil. Empresas brasileiras de Informática, apercebendo-se das vantagens decorrentes da adoção do Unix nos computadores nacionais, buscaram licenciar o sistema junto à AT&T. Apesar do interesse do lado brasileiro – representado em conjunto, por alguns fabricantes e software houses nacionais – e da AT&T em negociar o licenciamento dos fontes do Unix System V, nenhum negócio concreto foi realizado até a data em que escrevíamos este capítulo (junho de 1987).

Um dos fabricantes nacionais interessados no licenciamento era a COBRA – Computadores e Sistemas Brasileiros S.A. – que desejava incorporar o Unix nos seus supermicrocomputadores que comporiam a nova linha de produtos da empresa (denominada de linha X). As delongas nas negociações com a AT&T levaram a COBRA a decidir pelo desenvolvimento do seu próprio sistema Unix. Concebia-se o Sistema Operacional X, ou SOX como passou a ser conhecido nacionalmente, no início de 1984.

SOX: COMPATIBILIDADE COM O UNIX SYSTEM V

O SOX é um sistema operacional totalmente desenvolvido com tecnologia brasileira de software. É compatível com o Unix System V da AT&T, última palavra em sistemas operacionais, padrão de fato da indústria de informática internacional. Como o Unix System V, o SOX é um sistema operacional multiusuário e multitarefa.

Escrito na linguagem de programação C, o SOX está disponível para uma vasta gama de computadores, podendo se tornar o sistema operacional padrão brasileiro – a julgar pela sua aceitação por outros fabricantes nacionais, a exemplo da SCOPUS e ITAUTEC.

Para tanto, a COBRA licencia os programas-fontes do SOX a qualquer fabricante interessado e está aberta para receber empresas nacionais que queiram contribuir, com inovações tecnológicas, para a evolução do sistema.

Além das características inerentes ao Unix System V, o SOX tem características adicionais que facilitam o desenvolvimento e o suporte de aplicações comerciais de propósito geral.

APLICAÇÕES DO SOX

O SOX permite a execução de pacotes de softwares aplicativos para apoio às diversas atividades empresariais. Dentre estes pacotes destacam-se:

– geradores gráficos
– planilhas eletrônicas
– processadores de textos
– geradores de aplicações
– linguagens de 4ª geração
– bancos de dados

O SOX possui recursos de apoio à comunicação de dados que proporcionam sua integração com outros sistemas SOX ou UNIX e até com outros sistemas operacionais distintos. A integração com sistemas heterogêneos é feita segundo o padrão internacional especificado pelo Modelo de Referência para Interconexão de Sistemas Abertos (RM/OSI – *Reference Model for Open Systems Interconnection*) da Organização Internacional de Padronização (ISO – International Standards Organization). A aderência ao RM/OSI simplifica a ligação de equipamentos baseados no SOX a *rede de computadores a longa distância* e a *redes locais*, provendo as seguintes facilidades:

– compartilhamento de recursos e informações

– transferência de informações

– comunicação entre usuários remotos

– submissão de programas para serem executados em computadores remotos

– utilização dos terminais de uma máquina SOX em terminais de outras máquinas remotas, mesmo com sistemas operacionais distintos.

Vale lembrar ainda o suporte do SOX à aplicação importante de Automação de Escritórios. Para esta última, o SOX oferece um ambiente integrado e amigável, voltado para a gestão automatizada de escritório, com serviços que atenderão às seguintes áreas:

– arquivamento eletrônico de informações

– processador de documentos

– agenda / calendário

– calculadora

– correio eletrônico

Saliente-se que as aplicações mencionadas são aquelas que julgamos mais importantes para a automação das atividades de uma empresa. O SOX, contudo, sendo um sistema operacional de propósito geral, pode atender a muitas outras aplicações. É provável que o SOX atenda, por muitos anos, ainda, às aplicações que você imaginar ou que sua empresa exigir.

VOCÊ E O SOX

O SOX tem muito que oferecer a você e à empresa ou instituição onde você trabalha. Com o SOX, você poderá aumentar sua produtividade nas suas tarefas, realizando-as mais rapidamente e melhor. Além disso, você estará lidando com software de primeira linha, desenvolvido no Brasil, visando usuários brasileiros, sem, contudo, perder nada em compatibilidade a nível internacional. Para alcançar isto, exige-se de você um mínimo de esforço: ler os breves capítulos restantes deste livro.

O seu treinamento com o SOX possivelmente será de valia também no futuro. Haja visto a popularidade do Unix, que é um fato, a sua ampla disponibilidade numa vasta gama de máquinas e a compatibilidade do SOX com este sistema, você, ao iniciar-se no trato com o SOX, estará fazendo um investimento a longo prazo, minimizando a necessidade de retreinamento no caso de mudança de emprego, migração interna na sua empresa atual ou no caso de aquisições futuras de equipamentos da informática. Muito provavelmente, você e sua empresa evoluirão junto com o SOX.

Este livro o guiará nos seus primeiros passos com o SOX. Ao concluir sua leitura, você poderá valer-se do SOX nas tarefas cotidianas do seu trabalho.

CAPÍTULO 4

DICAS E CONCEITOS PRELIMINARES

Após o breve histórico e resumo das principais características e facilidades do SOX no capítulo anterior, nada mais natural que seu interesse e sua possível ansiedade em lançar-se ao seu uso. Antes disso, porém, apresentaremos mais alguns conceitos e informações sobre procedimentos simples, mas valiosos, para o trato de seu equipamento. O pouco tempo gasto aqui fará com que você prossiga de forma mais rápida e eficiente nos próximos capítulos. Você usuará, também, o seu equipamento de forma mais segura.

Ao ler o capítulo anterior você teve chance de apreciar o benefício em avançar no estudo de um capítulo com a bagagem conceitual apropriada. Essa leitura deve ter sido facilitada pelas informações acerca do computador e de recursos a ele atrelados que foram assunto do Capítulo 2. O benefício é mais duradouro, servindo-lhe também para leitura de revistas e publicações em Informática e dos manuais que acompanham sua máquina SOX.

Este capítulo começa apresentando dicas de uso de seu terminal (ou microcomputador que servirá de terminal) o qual é o equipamento básico para comunicação com sua máquina SOX.

Conhecer bem suas principais características físicas e de operação do SOX é uma decisão sábia. Em seguida, o capítulo acrescentará alguns conceitos úteis visando maior suavidade na exposição do conteúdo dos próximos capítulos. Esses conceitos referem-se à organização de suas informações no **SOX**. Saber como organizá-las adequadamente o tornará mais eficiente e causará boa impressão em quem examinar seu trabalho com o SOX.

CONVENÇÕES ADOTADAS

Para simplificar a apresentação do material neste e em outros capítulos e a fim de se evitar dúvidas, adotaremos, no restanto do livro, as seguintes convenções:

1. Os nomes das teclas de seu terminal serão indicados entre colchetes angulares ("<" e ">"). Por exemplo:

   <CR>

   representa a tecla de retorno do carro ("Carriage Return"). **<A>** representa a tecla "A".

2. Os comandos do SOX serão mencionados pelos seus nomes, de maneira idêntica à forma de invocação, com destaque em **negrito.**

3. As opções de comandos serão fornecidas entre "[" e "]". Por exemplo:

   ls [-l]

   indica que o comando **ls** tem opção **-l** (você pode ou não usá-la).

4. As instruções que você deve digitar, para exercitar o uso de algum comando, serão fornecidas com destaque em **negrito.** Você deve digitá-las tal e qual aparecem no texto.

5. As mensagens e/ou informações emitidas pelo SOX serão apresentadas sem destaque, mas em áreas devidamente destacadas com uma moldura.

4.1 SEU TERMINAL SOX

O sistema operacional SOX pode ser executado numa vasta gama de equipamentos. Na maioria das vezes, a comunicação entre você e sua máquina SOX é feita através de um terminal. A comunicação também pode ser realizada através de um microcomputador tipo PC, por exemplo. Neste último caso, o micro deve se comportar como um terminal, e, para tanto, você deve fazer executar no PC um software emulador de terminal (isto é, um programa que transforma um PC num terminal). Do ponto de vista do SOX, um terminal ou um PC emulando um terminal são idênticos. O uso desses dois equipamentos, porém, apresenta algumas diferenças. As diferenças são oriundas das características físicas (por exemplo, teclado) entre os dois equipamentos e do procedimento para carregar o emulador no PC.

Você pode dispor do SOX no seu PC com sistema operacional SISNE ou qualquer outro compatível com o MS-DOS, diretamente. Basta adquirir uma placa SOX e a inserir em um dos "slots" do gabinete com a CPU. Como o SOX é multiusuário, você teria o teclado e o vídeo do PC com a placa funcionando como um terminal e mais outros terminais remotos (ou PCs emulando terminais) que poderiam ser conectados ao PC com a placa SOX-PC. As informações apresentadas aqui valem tanto para o cenário do SOX rodando numa máquina de maior porte – como um supermicro ou minicomputador – quanto no cenário de um PC com placa SOX-PC.

A apresentação desta seção é feita em duas partes: a primeira trata de familiarizá-lo com o seu terminal propriamente dito e a segunda, com o trato do seu PC, emulando um terminal SOX. Nós aconselhamos ler ambas já que, com freqüência, terminais e PCs coexistem em muitas empresas. Alguns fabricantes, inclusive, oferecem configuração SOX onde os terminais são PCs. Isto porque o preço de PCs vem caindo continuadamente e os usuários podem, com o PC, dispor de capaçidade de processamento isolada da CPU SOX, o que não ocorre com um terminal comum. Fora isto, a discussão relativa ao PC assume conhecimentos das informações acerca dos terminais, fornecidas a seguir.

FAMILIARIZANDO-SE COM O TERMINAL

Vários são os tipos de terminais no mercado. Hoje em dia, quase todos (ou pelo menos os mais comuns) apresentam-se em dois módulos: o monitor de vídeo (onde se aloja a tela) e o teclado. Para sua comodidade e facilidade de arrumação de sua mesa de trabalho, os terminais mais novos têm teclado destacável do monitor (vide Figura 4.1).

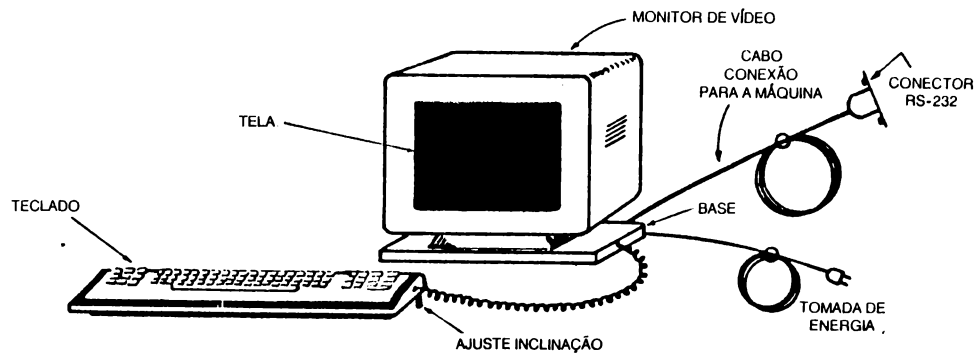


Figura 4.1 Ilustração de um Terminal Típico.

O teclado, por baixo, às vezes dispõe de duas linguetas retráteis para ajuste de inclinação. Tal ajuste é para deixar seu punho confortável em sessões de trabalho mais longas. Pelo mesmo motivo, o monitor de vídeo apóia-se numa coluna giratória que se encaixa na base. Você pode girar o monitor para a esquerda ou para a direita e incliná-lo para cima ou para baixo de forma a posicionar a tela à altura de seus olhos. Estas cômodas características são, freqüentemente, mencionadas como "ergométricas", ou seja, visam diminuir o esforço de seus músculos quando você trabalha com o terminal. Não se preocupe se o terminal que lhe foi alocado não tem estas facilidades; você pode obter os mesmos resultados com uma cadeira confortável (ajustável) e uma mesa com altura certa.

Sugerimos que, neste ponto, examine o seu terminal com cuidado e busque e leia seus manuais de operação. Converse com a pessoa responsável pela administração do seu sistema SOX e peça-lhe esclarecimentos sobre o seu uso, como ligá-lo e limpá-lo. Verifique a voltagem do terminal (220V ou 110V) e evite ligá-lo à tomada de energia em caso de dúvidas. O outro cabo que sai do terminal com uma tomada estranha (chamada de conector RS-232) é para ser conectado ao computador (na placa de entrada/saída, mais especificamente). Este cabo é o "cordão umbilical" do seu terminal e para operação correta deve estar sempre bem seguro (com os parafusos do conector). Consulte o administrador ou manual para mais detalhes.

Como dissemos, existem vários tipos de terminais. Eles diferem nas características do teclado e na forma de apresentar a informação na tela. Alguns teclados não dispõem das teclas de acentuação, nem do "ç". Terminais com este tipo de teclado geralmente não apresentam letras acentuadas na tela. Terminais de fabricantes distintos freqüentemente apresentam diferenças – nem que seja no arranjo de teclas ou na omissão de teclas especiais. Mesmo os terminais de um mesmo fabricante podem diferir, devido à evolução de sua linha de produtos. É possível que você se depare com diferentes tipos de terminais mesmo em configurações de equipamentos de um único fabricante SOX.

O SOX permite sua operação com vasta gama de terminais. Em alguns casos é necessário consultar o fabricante e pedir que ele faça certos ajustes nos terminais. Em outros, o próprio administrador do seu sistema resolverá a questão, registrando no SOX os vários terminais em uso na sua instalação. O SOX passa a conhecê-los e eles operarão sem maiores problemas. Agora, um pouco de prática.

Primeiro, descubra a chave de "liga-desliga" do terminal. Geralmente ela está atrás do monitor de vídeo, ou na lateral da base ou abaixo da tela. Ligue o terminal. Deve surgir alguma coisa na tela – o que, exatamente, não importa por enquanto. Agora vamos ajustar a intensidade da coisa na tela. Procure o botão de ajuste de luminosidade, junto da chave "liga-desliga". Às vezes existem dois botões um para brilho e outro para contraste. Gire os botões até obter uma intensidade confortável para seus olhos.

Aqui vale um aviso. Uma intensidade muito forte causará o envelhecimento precoce do fósforo que reveste a parte interna da tela. Se você puder enxergar informações com menor intensidade, melhor. Prossigamos esclarecendo alguns termos referentes ao seu terminal.

Caractere

Um **caractere** é uma letra, numeral, espaço em branco, sinal de pontuação (?, ! etc.) ou qualquer um dos símbolos presentes no seu teclado. Você gera um caractere pressionando uma tecla. Vários caracteres juntos – o que geralmente se chama de **cadeia** ou **seqüência** – podem formar uma palavra, frase ou comando para o SOX. Você usa o teclado para falar com o SOX, combinando caracteres para dizer o que deseja.

Em geral, os caracteres que você digita são apresentados na tela (ou são "ecoados"). O SOX também envia caracteres para a tela de seu terminal em resposta às suas ordens ou perguntas ou para pedir-lhe algo (p.ex., o seu nome, conforme veremos adiante).

Tamanho da Tela

Em monitores de vídeo profissionais (ou monitores que não sejam uma TV), a largura da tela pode acomodar até 80 (ou mais) caracteres; na altura cabem 24 (ou 25) linhas com informação (uma frase, comando etc.). A respeito desta tela dizemos ser de 80 colunas x 24 (ou 25) linhas. É incomum um terminal SOX que não seja profissional.

Cursor

Um outro item para o qual queremos chamar sua atenção ainda se refere à tela. Ao ligar o terminal, poderão surgir na tela algumas informações como nome do fabricante, número de referência do modelo do terminal etc. Estas informações geralmente são precedidas de um sinal sonoro, para chamar sua atenção. Caso o terminal esteja conectado ao SOX e este esteja executando (ou, no jargão de informática, o SOX esteja "no ar"), você possivelmente escutará outro apito ou "bip", a tela será limpa e surgirá o pedido do SOX para que você se identifique.O procedimento de identificação será discutido no Capítulo 6. No momento queremos apresentar-lhe o **cursor** que estará na tela junto ao pedido de identificação.

O cursor é geralmente indicado por um traço de sublinhamento, "_" ou um retângulo cheio. Ele pode se apresentar de forma intermitente. Você pode mover o cursor por toda a tela, com as teclas de seta $<\leftarrow>$, $<\rightarrow>$, $<\uparrow>$, $<\downarrow>$ ou equivalentes (veremos isto em seguida). O cursor indica onde, na tela, será mostrado o próximo carac-

tere a ser digitado por você ou emitido pelo SOX.

Teclado

Inspecione agora o teclado de seu terminal. Um desenho estilizado de um teclado está na Figura 4.2.

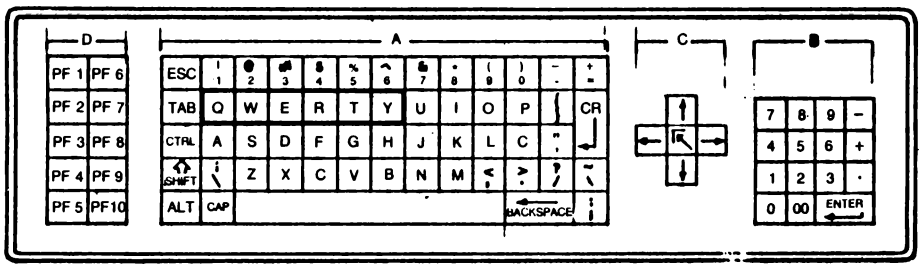


Figura 4.2 Ilustração da Arrumação das Teclas no Teclado de um Terminal.

A arrumação das teclas ilustradas na Figura 4.2 serve apenas aos nossos propósitos de discussão. A arrumação no seu teclado pode diferir e, inclusive, não apresentar todas as teclas indicadas. Isto não compromete o seu uso com o SOX. Tudo o que você tem a fazer é consultar o manual de operação ou o administrador do seu sistema para descobrir como gerar os caracteres correspondentes às teclas não existentes. Caracteres ausentes em alguns terminais são o "abre-chave" e "fecha-chave" ("{" e "}"), por exemplo. Você pode gerá-los com uma combinação de duas ou mais teclas, pressionadas em seqüência.

Devido às diferenças entre os tipos de terminais, as teclas que faltam em um tipo podem estar presentes em outro e vice-versa. A geração das teclas ausentes também difere. Você deve consultar os manuais do seu equipamento para saber como gerar os caracteres que não se encontram marcados no seu teclado. Nosso objetivo aqui é apresentar-lhe algumas teclas muito empregadas com o SOX.

Observe a Figura 4.2. A parte "A" é um conjunto de teclas alfanuméricas e não difere muito do teclado de uma máquina de escrever comum. Este é um teclado "QWERTY" – o mais comum em informática – o qual tem este nome devido à arrumação das teclas correspondentes (vide destaque na figura). Nesta parte você poderá encontrar as teclas:

< CTRL > – tecla de controle;
< CR > – retorno de carro;
< ESPAÇO > – barra de espacejamento.

como você pode se lembrar, usamos a notação < **nome** > para indicar o nome de uma tecla no seu equipamento.

A tecla < CTRL > é importante para o SOX pois em combinação com as teclas < H > , < D > , < S > , e < Q > é utilizada para:

< **CTRL** > < **H** > – retroceder o cursor de uma posição;
< **CTRL** > < **D** > – sinalizar o fim das entradas de dados, a partir do teclado, para muitos programas no SOX;
< **CTRL** > < **S** > – parar a listagem de informação na tela;
< **CTRL** > < **Q** > – recomeçar a listagem de informação na tela.

Ambas as teclas, < CTRL > < H > por exemplo, devem ser pressionadas simultaneamente. O uso da combinação acima será abordado em capítulos subseqüentes quando você terá tempo para entender melhor a funcionalidade de cada uma delas. Não se afobe; por enquanto descubra apenas onde se encontram as teclas que apontaremos.

Outra tecla importante é < **CR** > . Em alguns teclados ela é marcada com o rótulo TRANS (transmite), ENTRA, RET, (retorno do carro) ou simplesmente <↵>. Ela faz com que o cursor salte de linha, passando para o início da linha imediatamente abaixo da corrente. Ela serve, por exemplo, para sinalizar a conclusão de um comando ou o fim de uma informação para o SOX.

A barra de espacejamento, < ESPAÇO > , é a tecla mais longa de seu teclado, é freqüentemente acionada pelo polegar e, quando pressionada, gera um caractere em branco (espaço em branco). Ela é muito usada para separar trechos de um comando para o SOX.

Ainda na parte alfanumérica você pode encontrar a tecla < **DEL** > (em alguns teclados ela é marcada com ELIM (elimina), RUBOUT, ...). Esta tecla serve para cancelar um programa ou comando que você pediu para o SOX executar. Na Figura 3.2 ela é mostrada na parte C inferior.

A parte B (vide Figura 4.2) é o teclado numérico ou **Ilha Numérica** e repete as teclas numéricas mais as teclas < - > , < + > , < . > e < CR > da parte A (alfanumérica). Ela é fornecida para comodidade de digitadores que têm de inserir grandes volumes de dados numéricos (p.ex.: conta de luz, água, imposto de renda etc.) no com-

putador. Em alguns terminais, a parte B inexiste, sem perda de funcionalidade (talvez com perda de comodidade para o usuário).

A parte C contém as teclas de setas que servem para movimentar o cursor pela tela. Ela é freqüentemente chamada de **teclado de controle** e pode estar ausente em alguns terminais. Neste caso, busque a tecla de retrocesso, também chamada de Backspace no teclado alfanumérico ou por <←>. Você deve encontrar também as teclas <→>, <↓> e <↑>. Resumindo:

<→> – move o cursor uma posição para a direita;
<←> – move o cursor uma posição para a esquerda (equivale a <CTRL> <H> ou <BACKSPACE>);
<↓> – desce o cursor para a linha abaixo;
<↑> – sobe o cursor para a linha acima.

Finalmente, a parte D da Figura 4.2 ilustra as **teclas de função.** Na figura são mostradas apenas 10 teclas, mas podem existir 12, 16 ou mais em seu equipamento. Uma tecla de função pode ser programável (ou configurada dinamicamente) pelo usuário ou pode não ter esta facilidade. No primeiro caso ela é indicada pelas letras "PF", no último simplesmente por "F".

As teclas de função podem ser localizadas no lado esquerdo do teclado, como mostra a Figura 4.2, ou distribuídas na parte superior, acima das teclas numéricas do teclado alfanumérico. Em alguns teclados, as teclas de função concentram-se no canto superior direito ou ainda são associadas a cada uma das teclas numéricas da parte A. Neste último caso, para que possam ser utilizadas, elas devem ser pressionadas em conjunto com uma outra tecla de controle, p.ex., <ALT>. Verifique os manuais fornecidos pelo fabricante se seu terminal for deste último tipo.

Uma tecla de função serve o propósito de gerar seqüências de caracteres que representam certos comandos para o SOX ou para certos softwares que executam no SOX. Isto torna sua comunicação com o SOX mais simples, pois em vez de memorizar certas seqüências de teclas a pressionar, basta lembrar que, por exemplo, "a tecla PF1 fornece explicações sobre tal ou tal assunto".

Você só tem a ganhar com o uso das PFs. Informe-se como programá-las examinando os manuais do seu terminal e do SOX. Bata um papo com o administrador sobre o assunto. Desta vez, convide-o para um cafezinho.

FAMILIARIZANDO-SE COM O PC

A maior parte das informações acima para o terminal aplica-se ao PC, funcionando como terminal SOX. Antes de explicitarmos as correspondências, examine a Figura 4.3.

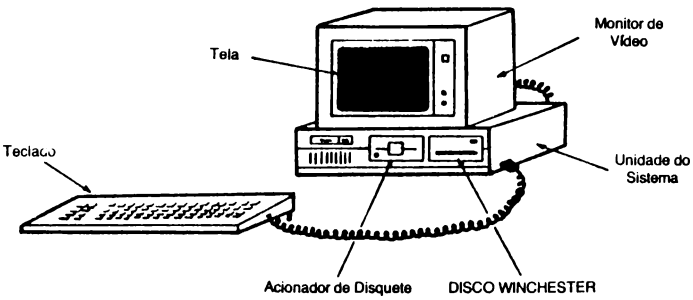


Figura 4.3 Ilustração dos Módulos de um Microcomputador Tipo PC (MS-DOS).

Se não fosse pela unidade do sistema, o PC seria muito semelhante, fisicamente, aos terminais SOX discutidos pouco atrás.

Tome as mesmas medidas para ligar o seu PC à tomada de energia e ajuste a luminosidade das informações na tela. No PC, geralmente existem uma chave "liga-desliga" para o monitor/teclado (examine a parte traseira ou frontal do monitor) e outra chave para a unidade do sistema (atrás do gabinete onde alojam-se os acionadores de disco). É possível que seu PC seja diferente, mas um dos que usamos "prefere" que liguemos primeiro a unidade do sistema e depois o monitor de vídeo.

O ajuste de brilho e contraste pode ser encontrado na parte frontal ou traseira do monitor. Vale a mesma dica relativa aos efeitos de luminosidade intensa no fósforo da tela: mantenha a luminosidade não muito intensa.

O cursor também apresenta-se como um traço de sublinhamento ou retângulo cheio, intermitente ou não. A tela geralmente é de tamanho 80 colunas x 24 (ou 25) linhas.

O teclado é separado da unidade do sistema e do monitor de vídeo. São três módulos soltos. Para sua comodidade, experimente vários ângulos e posições do teclado. Estenda as linguetas de ajuste de inclinação. A maioria das pessoas prefere colocar a unidade do sistema atrás do teclado e o monitor de vídeo em cima da unidade. Quem vai dizer qual a melhor arrumação dos módulos são o seu pescoço e os músculos que o ligam aos seus ombros.

O teclado do PC também tem características semelhantes à que já discutimos para terminais. Não dispense, contudo, a leitura dos manuais correspondentes.

Uma vez que você familiarizou-se com os módulos do seu PC, é chegada a hora de fazê-lo emular um terminal SOX. Perceba que o SOX pode rodar no próprio PC, com a placa SOX, mas não é este nosso enfoque aqui. Aqui, queremos discutir como fazer o seu PC, executando um sistema operacional compatível com MS-DOS (o SISNE, por exemplo), funcionar como um **terminal** ligado ao SOX. Para tanto, o procedimento requer que você primeiro carregue o MS-DOS e em seguida execute o software emulador de terminal SOX.

Carga do MS-DOS

Nossa discussão supõe o sistema operacional MS-DOS, mas é válida para outros sistemas operacionais compatíveis como o PC-DOS, o SISNE da Scopus etc.

Não há um procedimento padronizado para "colocar no ar" (ou carregar o) MS-DOS no seu micro. O procedimento difere ligeiramente de um fabricante para outro, e até entre equipamentos de um mesmo fabricante, dependendo da configuração: se a sua configuração dispõe de disco rígido (Winchester) o procedimento é trivial; numa configuração apenas com unidades de disquetes (geralmente duas, denominadas de "A" e "B"), o procedimento de carga é um pouquinho mais elaborado.

Ligue seu equipamento.

Carga a partir de Winchester

Se você dispõe de um winchester (disco rígido) na sua configuração, tudo que tem a fazer é ligar o equipamento. O MS-DOS será lido a partir do winchester, automaticamente. Você saberá que o sistema operacional está sendo carregado (isto é, está sendo colocado na memória principal e começando a rodar) se aparecer na tela do seu equipamento um pedido para você fornecer a data corrente.

Forneça a data, no formato mês (um ou dois algarismos), dia (um ou dois algarismos) e ano (dois algarismos), separados por hífens (-) ou barras (/). Por exemplo, a data 6 de janeiro de 1987 pode ser fornecida como:

01-06-87 (alternativamente: 1-6-87)

ou

01/06/87 (alternativamente: 1/6/87)

Erros de digitação podem ser corrigidos com o uso da tecla de retrocesso (<BACKSPACE> ou <←>) para você colocar o cursor na posição a ser corrigida. Redigite a data corretamente e, ao terminar, tecle <CR> .

Após ler a data, o sistema pedirá a hora. Aqui você deve seguir o formato:

horas:minutos:segundos

Observe a separação dos campos (a qual é feita por dois pontos " : "). Utilize um relógio de 24 horas. Assim, 11 horas, 37 minutos e 5 segundos da noite, seriam:

23:37:05

Opcionalmente, você pode descartar os segundos ou fornecer centésimos de segundos, colocando um ponto decimal no último campo, seguido dos centésimos (um ou dois números). Por exemplo, meio segundo após as 9 horas da manhã:

9:0:0.5

Observação: A ausência dos pedidos de data e hora não é necessariamente sinal de anomalia do sistema operacional. Isto pode ocorrer devido a dois motivos:

1. o MS-DOS não se encontra no Winchester;
2. o arquivo AUTOEXEC.BAT – executado quando o sistema entra no ar – não inclui esses pedidos.

No primeiro caso, você (ou alguém) deve copiar o disquete que contém MS-DOS (chamado de "disquete do sistema") para o winchester (veja as instruções no manual do seu equipamento). Uma vez que o MS-DOS esteja instalado no winchester, é só seguir o procedimento aqui descrito.

No segundo caso, o MS-DOS entrará no ar sem que você forneça data/hora. Os comandos **DATE** e **TIME** do MS-DOS podem ser usados subseqüentemente para fazer o ajuste desejado (ou então corrigir informações erradas que, porventura, você tenha fornecido). O sinal de que o MS-DOS foi carregado e se encontra ao seu dispor é o surgimento na tela da mensagem de inicialização do MS-DOS no seu sistema, seguida do sinal ou caractere de prontidão (*prompt*) do interpretador de comandos do MS-DOS:

A>

Em alguns sistemas, o sinal de prontidão é

A:

Observe que o sinal de prontidão pode não ser "A >". Na verdade, a letra será aquela correspondente à partição do winchester onde se encontra o procedimento de carga. Por exemplo, poderá ser "C >", "E >" etc.

O sinal de prontidão do interpretador de comandos significa que o MS-DOS está executando, na espera de um comando seu.

Carga a partir do disquete do sistema

Antes de ligar seu equipamento, insira o disquete que contém o MS-DOS (chamado de Disquete do Sistema) num dos acionadores de disquetes. Ligue agora seu microcomputador.

Após algum tempo, o computador pedirá pela data (e depois pela hora). Siga o procedimento para *Carga a partir de winchester*, descrito anteriormente.

O surgimento na tela do sinal de prontidão do interpretador de comando do MS-DOS.

A>

ou

A:

indicará que o sistema está no ar e à espera de seus comandos.

Observação: se você inserir o disquete no acionador **B**, o sinal será: " B > " ou " B: ", em vez de " A > " ou " A: ".

Muito bem. Vejamos agora o segundo passo do procedimento: carregar o software emulador de terminal SOX.

Carga do emulador de terminal SOX

Seu PC encontra-se operacional, rodando o MS-DOS, de forma dissociada do SOX. Vamos transformar o PC em um terminal SOX.

A primeira providência é ligar o PC à máquina SOX através do cabo RS-232 (vide a discussão anterior sobre terminais). Descubra onde fica a saída serial assíncrona do seu PC (examine a parte traseira da unidade do sistema) consultando seus manuais ou um

usuário afeiçoado a este tipo de equipamento. É lá que você vai inserir um dos conectores do cabo RS-232 (procure o administrador para arranjar-lhe um). O conector na outra extremidade deve ser acoplado a uma porta (livre) serial assíncrona da máquina SOX. O administrador é indispensável para esta operação. Tomara que você já lhe tenha pago um cafezinho.

Temos duas situações a contemplar. A primeira, mais simples, é aquela em que seu PC encontra-se a uma distância inferior a uns 35 metros da máquina SOX. Tudo que você precisa é de um cabo RS-232 deste comprimento e um sorriso amarelo no rosto ao entrar na sala de seus ex-amigos para rebentar furos nas paredes, na sua tentativa epopéica de lançar o cabo até o SOX. Seja determinado, use a marreta na cabeça dos ex-amigos mais chatos – o SOX retribuirá com eficiência. Você vai ligar o PC à máquina SOX diretamente, como ilustrado na Figura 4.4.

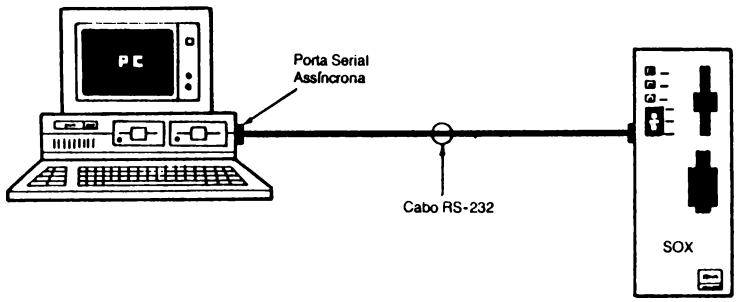


Figura 4.4 Conexão Direta PC-SOX.

Se seu PC está em outro prédio, distante do prédio onde reside a máquina SOX, ou em outra cidade etc., você precisará usar **MODEMS**. Fale com o administrador, a companhia concessionária de comunicações do seu estado (ex. TELERJ, TELESP, TELPE, TELMA etc.) ou com a EMBRATEL. Estas companhias alugam MODEMS junto à prestação de serviço de comunicação de dados. Se a conexão for restrita a seu prédio, use os ramais telefônicos de seu PABX e compre MODEMS. Peça sugestões ao administrador. No final, você terá uma conexão como mostra a Figura 4.5.

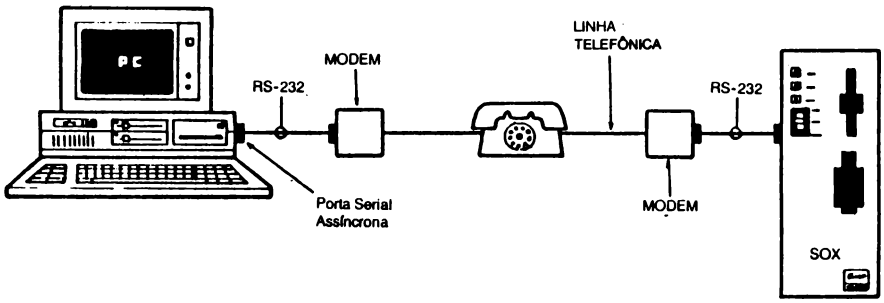


Figura 4.5 Conexão Remota PC-SOX via MODEM.

Um pouco mais de paciência. Está tudo pronto. Só falta o software emulador de terminal SOX que rode no PC com o MS-DOS. Contacte o fabricante do seu equipamento SOX e peça-lhe informações sobre tal produto. Existem alguns no mercado. Antes de comprar verifique a documentação e examine os manuais. Em caso de dúvidas, peça esclarecimentos **antes** de comprar. Certifique-se de que ficará satisfeito. Compre um. Seguindo as instruções do manual do emulador, insira o disquete com o software emulador de terminal SOX em um dos acionadores de disquetes do PC; trave o acionador e examine a tela.

Encontre o sinal "A>" (ou "B>") na tela. O sinal de prontidão "A>" (ou "B>") ou outra letra (C, D, E) indica que o interpretador de comandos MS-DOS está aguardando suas instruções. É o interpretador que identifica os comandos ou programas que você deseja executar (no caso, o programa emulador recém-adquirido), busca-os na memória ou disco (no caso, o disquete recém-inserido) e os coloca em execução.

O nome do programa a executar é digitado por você em seguida ao sinal de prontidão e terminado com um < CR > . A tecla < CR > é identificada nos teclados de alguns PCs por <RETURN> , em outros por <ENTER> e ainda pelo símbolo <↵>. Supondo que para carregar o emulador você tenha que chamar **emula** (verifique a chamada correta nos manuais do software que adquiriu), você digitaria:

```
A>emula <CR>
```

O que fazer em casos de erros

Se você cometer erros ao digitar suas instruções na linha de comandos (linha do sinal de prontidão), poderá se valer da tecla de retrocesso <BACKSPACE> ou (<—>) para apagar um caractere errado e em seguida redigitar o caractere correto.

Exemplifiquemos. Se você, por acaso, digitou "emlua" em vez de "emula" e ainda não teclou <CR> , sua linha de comando estará como abaixo (supondo o disquete com o emulador no acionador "A"):

```
A>emlua_
```

onde o sinal de sublinhar representa o cursor. Para corrigir, retroceda o cursor para o caractere "l", teclando (←) ou <BACKSPACE> três vezes. Após a primeira vez, a linha de comando deve mudar para:

```
A>çmlu_
```

ou seja, ao retroceder o cursor, o caractere "a" foi apagado. Retroceda até o caractere "l". Você terá então:

```
a>em_
```

redigite <u> <l> <a> e então tecle <CR> , assim:

```
A>em<u><l><a><CR>
```

Se você deseja cancelar toda a linha de comando tecle <ESC> . Retomando o exemplo acima, poderemos fazer:

```
A>emlua<ESC>
```

Como resultado, o cursor saltará para a próxima linha, abandonando o que existia na anterior, e ficará à espera do nosso comando. Alguns sistemas nem reapresentam o sinal de prontidão "A>" na linha para onde o cursor saltou. Simplesmente redigite o comando

e tecle <CR> (vide abaixo):

```
A>emlua \
   emula<CR>
```

Observe que para indicar que a linha anterior foi cancelada, o sistema inseriu uma contra-barra, "\", no seu final. Estas medidas corretivas só podem ser efetivas antes de você digitar <CR> . Após você concluir a linha de comandos com <CR> , o sistema tentará encontrar o comando ou programa pedido para colocá-lo em execução. Se o sistema não encontrar o que você pediu, ele exibirá uma mensagem de erro dizendo que o comando ou o arquivo com o programa é inválido (ou alguma outra mensagem de erro semelhante).

Existem duas possibilidades para a mensagem de erro acima:

1. você cometeu erro de digitação na linha de comando;
2. o programa não se encontra no disco que está sendo acessado pelo acionador identificado no sinal de prontidão.

Há ainda a possibilidade de 1 e 2 ocorrerem em conjunto. As soluções a serem apontadas englobam também esta combinação.

Para remediar 1, simplesmente redigite a linha de comando.

Vejamos como sair da situação 2 . Se você, por exemplo, não identificar o acionador de disco na linha de comando, o MS-DOS tomará como referência o acionador indicado no sinal de prontidão do interpretador. Assim, se o sinal de prontidão for "A>", a linha de comando.

```
A>emula<CR>
```

fará com que o programa emulador de terminal SOX seja procurado no acionador A. Se você acabou de colocar o MS-DOS no ar, o disquete que se encontra no acionador A é o do sistema. Ele não contém o emulador. Problema semelhante aconteceria com o uso do winchester, o qual tem o disco particionado em vários *discos lógicos*, digamos C, D, E etc.

Suponha que o emulador esteja no disco B (um outro disquete que você coloca no acionador B – se seu micro tem dois acionadores, ou um disco lógico B do winchester – para onde foi copiado a partir do disquete original). Você poderá acessar o emulador com esta suposição, mudando o acionador no sinal de prontidão (acionador "padrão" ou por omissão) para "B", com as linhas de comando:

```
A>b: <CR>
B>emula<CR>
```

A primeira linha muda o acionador padrão para "B"; a segunda pede para executar o emulador.

Uma outra maneira é não mudar o acionador para **B** e invocar o programa **emula** a partir do acionador **A,** indicando que o programa se encontra no disquete inserido no acionador **B.** O comando é:

```
A>b:emula<CR>
```

O "b:" indica para o interpretador que deve buscar o programa **emula** no acionador **B.** Após executar este programa, o sinal de prontidão continuará sendo "A >".

Se você vai utilizar muito o conteúdo do disco em "B", a primeira maneira é mais indicada, pois lhe torna mais eficiente.

O emulador entrará no ar e apresentará algumas mensagens dizendo-lhe como proceder. Consulte o manual fornecido para maiores esclarecimentos. Correndo tudo bem, o SOX apresentará o pedido de identificação (ou de *login*) na tela do seu PC.

Parabéns. Você acaba de conectar o PC como um terminal do SOX. Daqui para frente, todo procedimento no trato com o SOX será idêntico àquele seguido num terminal propriamente dito.

4.2 CONCEITOS SOBRE O SISTEMA DE ARQUIVOS

Apresentamos, nesta seção, alguns conceitos e informações sobre o Sistema de Arquivos no SOX. A maestria no uso do SOX, que você alcançará rapidamente com os próximos capítulos, passa necessariamente pela devida absorção do material nas páginas restantes deste capítulo. Comecemos com a noção de *arquivo*, seguida da de *diretório* e da de *sistemas de arquivos em árvore*.

ARQUIVO

A quantidade de informação gerada pela humanidade é astronômica. Para piorar as coisas, a **taxa** de geração vem crescendo muito rapidamente ao longo dos anos. Graças ao volume de informações preservadas e disponíveis a qualquer pessoa ou grupo interessados em algum assunto, é possível absorver os pontos importantes e rapidamente gerar novo conhecimento. De posse deste conhecimento recém-gerado, abrem-se outros horizontes de possibilidades para aplicações e pesquisa. É um processo em cadeia, em aceleração permanente. Só para ilustrar, na área da medicina foram publicados mais artigos sobre os diversos aspectos e especialidades médicas no período entre 1945 - 1970 do que em todo o período anterior da história humana.

Todo o conhecimento científico da humanidade (em todos os campos da matemática, física, química, medicina etc.) até o início do século XIX dobrava a cada 50 anos. Já em 1950 dobrava a cada 10 anos. Na década de 70 dobrava a cada 5 anos e em 1990 estima-se que estará duplicada a cada 2 anos. Paciência. É informação demais.

Graças a Deus, talvez você não se depare com uma calamidade destas na sua empresa. De qualquer forma, você deve se encontrar às voltas com um volume **suficiente** de informações: são memorandos, cartas, contratos, agendas de endereços/telefones, cadastros de funcionários, formulários de impostos, contra-cheques, lembretes... A lista é infindável. Para sermos eficientes no desempenho de nossas atribuições na empresa, devemos armazenar (ou seja, guardar) tais informações de maneira a permitir recuperá-las rapidamente quando necessário.

Existem várias maneiras de armazenar informações. Endereços e números de telefones podem ser – e são – escritos em agendas, cartas são depositadas em pastas-arquivos suspensas e estas penduradas em gavetas de gabinetes-fichário etc. Uma outra maneira é armazenar as informações do seu escritório no computador. E é esta última maneira que nos interessa aqui.

O computador guarda suas informações geralmente em dispositivos de memória

secundária: disquetes, disco e fita. Qualquer informação que possa ser expressa em um conjunto de caracteres inteligíveis ao SOX pode ser armazenada. Um conjunto de caracteres, na terminologia da Informática, recebe o nome de **arquivo**. Com o SOX, você não apenas pode armazenar como também gerar e, obviamente, recuperar arquivos.

Arquivo refere-se pois à informação (p.ex., texto) de seu interesse. A conotação mais freqüente, contudo, é a localização (indicada por um nome) onde a informação desejada se encontra armazenada no disco (ou disquete). Assim, o arquivo de nome "receita_feijoada" possivelmente conteria as informações (ou o texto) para se fazer uma feijoada. Os arquivos tratados pelo SOX podem conter textos simples (como cartas, poesia), programas escritos em alguma linguagem de programação (código-fonte) ou conter código executável referente a algum programa que já foi compilado. Os comandos do SOX que você encontrará nos capítulos à frente são armazenados em arquivos executáveis. Eles são executados quando você fornece os seus nomes (i.é, os nomes dos arquivos com os comandos) para o SOX. O procedimento é semelhante à execução do programa que chamamos de **emula** na seção anterior.

PROTEÇÃO DE ARQUIVOS

Cartas de amor têm endereço certo e, espera-se, não devem ser lidas por qualquer pessoa. Tais cartas geralmente contêm informações *delicadas* e, não poucas vezes, ridículas. O certo é que deseja-se protegê-las contra leituras indesejáveis. Faz-se isto trancando-as em gavetas, caixas de sapatos. Limita-se o acesso a certas pessoas autorizadas a saber do seu conteúdo. O mesmo fato ocorre com folhas de pagamento ou relatórios de vendas na sua empresa. Se você vai guardar suas informações no SOX, é melhor que ele tenha como resguardá-las contra intrujões.

O SOX facilita a utilização de seus arquivos por outros usuários. Ao mesmo tempo, tem como impedir que outras (certas) pessoas possam ler o conteúdo, escrever (alterar) ou executar seus arquivos. Convém observar que o serviço de proteção não se aplica somente à leitura. Você pode permitir que alguém leia o valor que receberá de salário no final do mês e ao mesmo tempo desejar que a pessoa não altere (p.ex., aumentar) o valor. Em uma outra situação, você deseja impedir certos usuários de executarem alguns arquivos. Por exemplo, não é boa idéia permitir a execução indiscriminada da ficha cadastral dos funcionários da empresa. Apenas determinadas pessoas, no Departamento de Pessoal, devem ter tal autorização.

Para atender a estas necessidades, você pode pedir ao SOX para estabelecer os seguintes **modos de permissão** nos seus arquivos, no que se refere aos usuários do sistema:

1. leitura (**r** – do inglês **r**ead);
2. gravação (**w** – do inglês **w**rite);
3. execução (**x**).

Você não pode ler arquivos para os quais não tem permissão de leitura. De modo semelhante, você só pode gravar (alterar o conteúdo) ou executar arquivos para os quais tem permissão de gravação ou execução, respectivamente.

Os modos de permissão de um arquivo são atribuíveis ao usuário que o criou (**dono do arquivo**), a grupos de usuários (todos envolvidos em atividades correlatas, por exemplo), aos demais usuários. Os modos de permissão de um arquivo só podem ser atribuídos ou alterados pelo seu dono ou por um usuário especialíssimo do sistema – o **super usuário** (o administrador). Não se preocupe com estes últimos detalhes. Eles serão abordados no Capítulo 6.

MANIPULAÇÃO DE ARQUIVOS

Desde que você tenha as permissões apropriadas, pode manipular ou trabalhar num arquivo SOX da forma que desejar.

Ao iniciar um trabalho pela primeira vez num arquivo, usa-se a expressão **criar um arquivo** que significa escolher o nome e as permissões de leitura/gravação por outras pessoas. Pode-se fazer analogia com a expressão "abrir uma conta no banco". Ao abri-la você a identifica e enumera as pessoas com acesso permitido (conta conjunta).

Na manipulação ou reutilização de um arquivo já aberto, usam-se os termos:

– **carregar** (ou **ler**)

– **editar**

– **gravar** (ou **salvar**)

Arquivos geralmente são mantidos em dispositivos de memória secundária (discos). **Carregar um arquivo** significa então trazer o conteúdo (ou parte dele) do arquivo referenciado (pelo nome) do disco para a memória principal. A CPU pode agora *ler* as informações e colocá-las na tela para seu exame. Uma vez carregado, o arquivo pode ter seu conteúdo (ou parte dele) criado, alterado ou removido. Estas atividades são referenciadas pelo termo **editar**. Assim, editar um arquivo significa poder alterar seu conteúdo. Finalmente, **gravar** um arquivo significa guardar, de forma permanente, o conteúdo edi-

tado, de volta no disco.

Cuide em gravar seu trabalho de edição com freqüência. Este procedimento pode significar tranqüilidade (e estabilidade emocional) no caso de falhas de energia ou de seu computador. Quando ocorre uma falha dessas, você perde todo o trabalho de edição feito desde a última gravação. Assim, grave com freqüência (a cada meia tela de texto digitado, digamos). O editor do SOX tem comandos simples para gravação e você deve usá-los sem parcimônia (*vide* Capítulo 7).

DIRETÓRIO

Fazendo-se analogia com o mobiliário de um escritório, um diretório corresponderia a uma gaveta do armário-fichário usado para guardar arquivos (cada arquivo numa pasta suspensa). No SOX um diretório contém um ou mais arquivos. Pode também conter um ou mais diretórios (chamados de subdiretórios ou diretórios descendentes), além de arquivos. Um diretório é identificado por um nome (conjunto de letras, algarismos ou outros símbolos).

Quando você chegou na empresa em que trabalha, deram-lhe provavelmente uma mesa, cadeira, estante e, se você manipula muitos documentos/informações, um armário-fichário. Inicialmente o fichário estava vazio, você foi colocando pastas nele à medida que trabalhava. O mesmo procedimento acontece no SOX.

Quando você pede uma conta no SOX ao administrador – se ainda não fez isto, faça-o agora – ele lhe fornece um diretório. No início, o diretório nada contém (exceto algum arquivo especial, chamado de ".profile" – cortesia do administrador – para facilitar-lhe a vida, como mostraremos mais adiante). Você, com o uso do SOX, começará a criar arquivos e/ou diretórios a partir do diretório recebido – também chamado de **diretório-casa**. Como criar arquivos e diretórios é assunto para o Capítulo 10. O problema com o qual você logo se deparará – pense neste estágio – é o que fazer para organizar todos os seus arquivos e diretórios de forma cômoda e eficiente no *único* diretório que recebeu. Muito bem pensado. Daremos a solução do SOX mais abaixo. No momento, contudo, permita-nos um breve aparte para tratarmos das permissões em diretório.

PROTEÇÃO DE DIRETÓRIOS

Continuando com a analogia da gaveta-fichário com um diretório, em muitos casos, você deseja *trancar* uma gaveta do fichário (pois foi lá que guardou as cartas de amor ou a folha de pagamento). Só as pessoas que possuem a chave podem abri-la. Ou, se

desejar, pode colocar um aviso na frente, permitindo leitura dos documentos nela guardados mas instruindo que ninguém deposite qualquer coisa nela. Pois bem, o SOX permite o mesmo tipo de proteção em seus diretórios.

O SOX pode estabelecer os seguintes modos de permissão nos seus diretórios, relativamente aos usuários do sistema:

1. leitura (**r** – do inglês **r**ead);
2. gravação (**w** – do inglês **w**rite);
3. execução (**x**).

Você já ouviu esta cantiga antes. Agora, porém, existem algumas poucas diferenças.

Só quem tem permissão de leitura em um diretório consegue obter a lista dos arquivos e outros diretórios nele contido. Isto não impede, contudo, a leitura, gravação ou execução dos arquivos (desde que as permissões no arquivo permitam) ou a transferência para um (sub-) diretório. Permissão de gravação é exigida para que você possa criar novos arquivos neste diretório, ou remover arquivos deste diretório. Finalmente, a permissão de execução, quando removida, impossibilita qualquer trabalho ou manipulação do conteúdo do diretório. Convém notar a infelicidade do nome deste último modo de permissão: execução. Não faz sentido *executar* um diretório, apenas um arquivo. Um diretório é apenas uma lista.

Novamente, os modos de permissão de um diretório aplicam-se ao usuário-dono, grupo de usuários e demais usuários do sistema. Os modos são atribuídos ou alterados apenas pelo dono ou pelo superusuário. Mais detalhes no Capítulo 6.

SISTEMAS DE ARQUIVOS NO SOX

O uso de arquivos e diretórios torna eficiente a organização e manipulação de informações na memória secundária. Não desanime, esta é apenas a introdução à resposta do problema que você identificou acima. Vamos ao resto da resposta.

Você tem um punhado de arquivos e a idéia é organizá-los de forma a encontrar rapidamente as informações que deseja. Este é o problema. O SOX é reconhecidamente um sistema operacional que resolve este problema de forma simples e elegante.

Imagine a quantidade de arquivos que você criará no seu trabalho. Os conteúdos destes arquivos provavelmente serão bastante distintos: uns arquivos conterão cartas, outros menus, outros contratos, outros relatórios, outros programas (como o **emula** acima), outros dados a serem processados por algum pacote aplicativo etc. Se estes arqui-

vos estiverem todos residentes num único diretório você terá uma bela bagunça que vai atribular suas atividades no dia-a-dia. A lista talvez seja tão grande que nem caiba na tela de seu terminal de uma só vez. Cada arquivo terá um nome é claro, e, se possível um nome sugestivo. Por exemplo, *carta_concorrência*. Até aqui, tudo bem. E se existirem duas ou mais cartas de concorrência? Você teria então *carta_concorrência_1*, *carta_concorrência_2*,... Depois de 2 meses, você saberia exatamente o que um destes dois arquivos, dentro de um emaranhado de 300 ou mais arquivos, contém? É provável que não.

Para organizar esta bagunça, podemos criar um diretório com um determinado propósito e com nome sugestivo e nele então criar (ou transferir) os arquivos e (sub-) diretórios relacionados. Se todos os diretórios assim criados forem mantidos "separados" o uso do sistema se tornará bem mais simples. Não é assim na maioria das empresas? Isto é, não são elas organizadas em departamentos, divisões etc., onde são colocados pessoal e outros recursos? Por exemplo, uma empresa pode ser dividida em departamentos de vendas, marketing, financeiro, de produção e administrativo. As pessoas são distribuídas pelos vários departamentos, listas de ramais telefônicos são elaboradas baseadas nesta divisão etc. A gerência da organização fica mais eficiente. Da mesma forma que cada pessoa ou departamento tem um nome que o identifica, cada arquivo ou diretório criado no SOX é assim identificado. Você fala: o "João do departamento de vendas"; no SOX você diria: "o arquivo 1987 do diretório balanços". Você termina com uma organização hierárquica de arquivos e diretórios – todos saindo do diretório-casa (aquele que você recebeu na abertura de sua conta). É igual à empresa, onde existe uma hierarquia de departamentos/divisões, todos saindo da diretoria, que por sua vez sai da presidência.

Agora que você já tem idéia da solução de como organizar eficientemente suas informações no SOX, vamos examinar os pormenores e conceitos envolvidos na solução como ela realmente é.

Os sistemas operacionais mais modernos como o SOX suportam uma estrutura de organização de arquivos e diretórios (ou, simplesmente, **sistema de arquivos**) em árvore hierarquizada. Sistemas de arquivos em árvore são particularmente interessantes para os usuários que geram arquivos com freqüência – ou que dispõem de computadores com unidades de disco. Como os discos geralmente contêm vários arquivos, a estrutura em árvore é de grande valia na manipulação e manutenção de arquivos.

Um sistema de arquivos em árvore é organizado como uma hierarquia de diretórios, na forma de uma árvore invertida, com a raiz em cima. O início do sistema fica no diretório principal – adequadamente chamado de **raiz**. Todos os demais arquivos e/ou diretórios do sistema derivam-se da raiz. Perceba que cada diretório pode conter qualquer número de diretórios ou arquivos.

Podemos dizer que um diretório é uma lista de diretórios ou de arquivos que podem ser alcançados a partir dele (para "baixo" na estrutura, ou, equivalentemente, em direção às folhas da árvore). Continuando com a analogia da árvore, vemos que um diretório corresponde a um *galho* (do qual podem sair ramificações ou outros galhos) e um arquivo corresponde a uma *folha* (presa a um galho).

O sistema de arquivos resultante é visto como a coleção de todos os diversos diretórios e arquivos onde se destaca a interdependência lógica entre os elementos. Antes que a coisa fique confusa, permita-nos apresentar uma figura.

Na Figura 4.6, apresentamos um pequeno sistema de arquivos em árvore. Nesta figura, os retângulos denotam diretórios e os círculos, arquivos. A figura talvez ilustre a organização de um escritório que se ocupe da correspondência enviada aos clientes e da preparação de documentos referentes (no caso, IAPAS) e confecção/armazenamento dos balanços da empresa. Na figura, o diretório no nível mais alto da hierarquia é o diretório-raiz que contém dois outros diretórios: *cartas* e *documentos*. O diretório *cartas* contém apenas dois arquivos: *cobrança* e *parabéns*, cujos conteúdos podem ser inferidos dos seus nomes.

O diretório *documentos* se desdobra em dois outros diretórios: *iapas* e *balanços*. O primeiro tem os arquivos *jan* e *fev*; o segundo tem os arquivos *1985, 1986* e *1987*. Veja a organização facilitada pelo sistema hierárquico em árvore quando comparado a uma simples lista de arquivos em ordem alfabética, por exemplo. Uma simples olhadela na estrutura da Figura 4.6 fornece, de imediato, a funcionalidade de cada elemento e o relacionamento lógico entre eles e as atividades do escritório, já que reflete toda a estrutura operacional do ambiente de trabalho ao qual ela objetiva suportar.

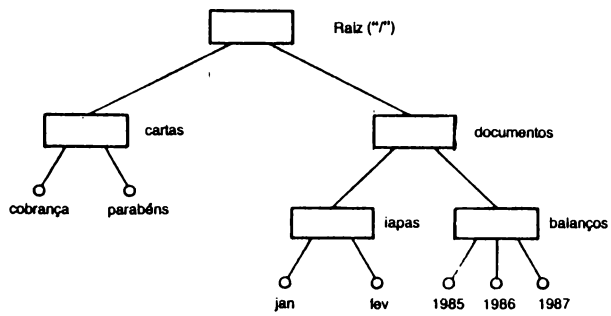


Figura 4.6 Ilustração de um Sistema de Arquivos Hierarquizado.

O Capítulo 10 traz maiores detalhes sobre a criação, manutenção e uso do seu sistema de arquivos no SOX.

CAPÍTULO 5

A ORGANIZAÇÃO DO SOX

O SOX é um sistema operacional que se impõe no mercado, mercê de características fundamentais, algumas já mencionadas em capítulos anteriores. Enumeremo-las:

- pode ser instalado em qualquer tipo de máquina, de microcomputadores a mainframes;
- é fácil de ler, entender, mudar e transportar para outras máquinas;
- comunica-se com o usuário de maneira simples;
- permite que programas complexos sejam construídos de programas mais simples;
- usa um Sistema de Arquivos bastante funcional;
- é multiusuário e multitarefa;
- "esconde" a máquina do usuário.

Qual a explicação para tantos méritos? Diríamos que grande parte do "segredo" está em sua estrutura organizacional: ela é o principal assunto deste capítulo, mas os outros fatores são também ventilados.

5.1 A ESTRUTURA DE ORGANIZAÇÃO DO SOX

A Figura 5.1, a seguir, sintetiza os vários compartimentos em que o SOX se compõe: cada compartimento é representado por uma camada circular e o aspecto fundamental é que camadas vizinhas se comunicam entre si, mas uma não conhece os detalhes da outra – tal fato tem importantes implicações e será o mérito deste capítulo convencê-lo disto.

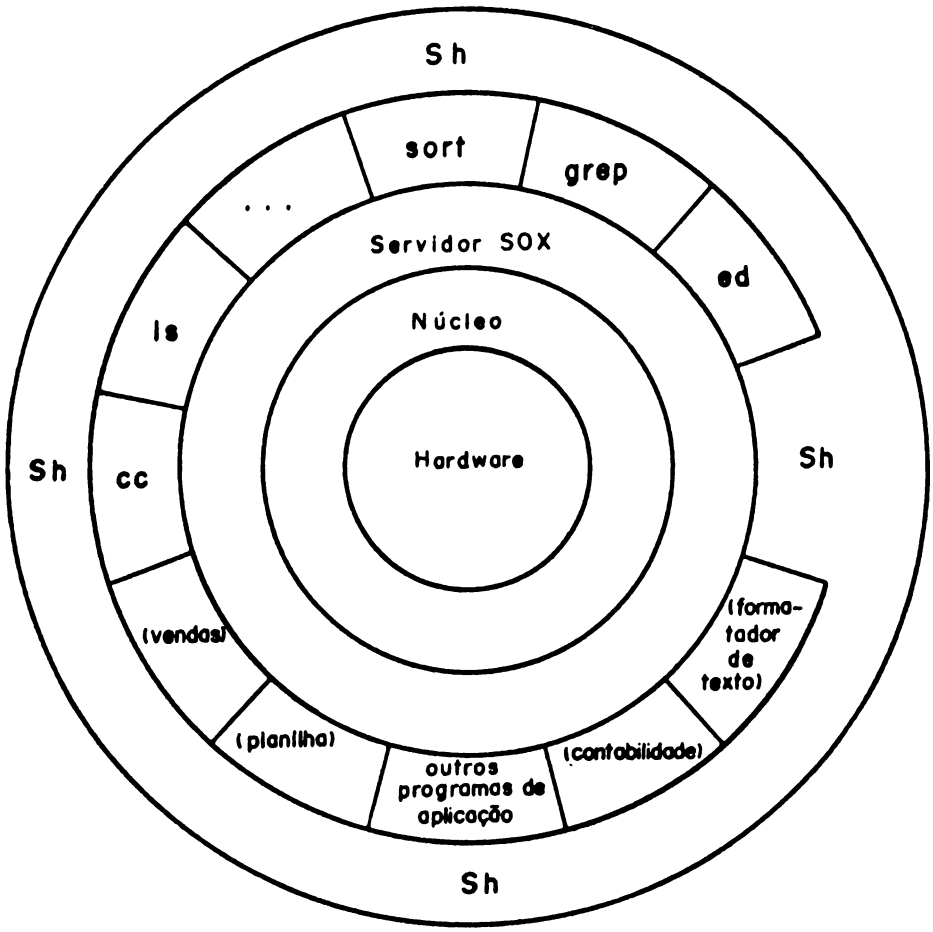


Figura 5.1 A Estrutura de Funcionamento de um Sistema SOX.

Quando você está trabalhando com o SOX, o que realmente está fazendo é utilizar-se de um ou vários programas nele existentes, que vão executar as tarefas apropriadas. Vamos dar exemplos:

- Você está precisando do calendário do mês corrente. Você deve saber que existe no SOX um programa chamado **cal** que fornece o calendário para o mês que você especificar (ou até para vários meses, o ano todo etc.). Você deve então digitar o comando **cal**, que é um pedido ao SOX para que execute o programa de mesmo nome (você aprenderá a se conectar ao SOX e como introduzir seus comandos no próximo capítulo): fazendo isto, aparecerá na tela de seu terminal (ou em outro lugar se você assim o desejar) o calendário pedido.

- Você tem sua lista de telefones e verificou que ela está fora de ordem: para ordená-la, você pode utilizar o comando **sort** (daqui por diante confundiremos as noções de comando e de programa, porque, de fato, significam a mesma coisa) que faz exatamente a tarefa de ordenar um texto da maneira que você indicar.

- Quer processar a folha de pagamento? Deve existir um programa com tal finalidade.
- Quer conhecer seus arquivos? Existe um comando para tal fim.

- Quer se desconectar do SOX (sair do ar)? Existe um comando para fazer isso.

A lista de programas (ou comandos) não tem fim. Quando você adquire o SOX, uma quantidade bastante grande de programas já vem com ele. Programas com **sh, ed, grep, ls** e outros (veja a Figura 5.1), se confundem com o próprio SOX. Esses programas congênitos do SOX têm nomes ingleses (como **sort**) ou siglas de nomes ingleses (por exemplo, **sh** é um mnemônico da palavra inglesa **shell**) – a razão é a necessidade de compatibilização do SOX com o seu inspirador, o UNIX.

Cada novo programa que é instalado sobre o SOX, torna-se um novo comando que é adicionado a ele. Um novo programa pode ser desenvolvido por uma equipe de desenvolvimento local a uma instalação do SOX, ou pode ser adquirido externamente a ela, de uma indústria de programas para o SOX, por exemplo. Assim, programas como folha de pagamento, um formatador de textos, uma versão mais agradável (e até melhor) de um dos programas nativos do SOX etc., tudo isso poderá ser adicionado à coleção de programas do seu SOX particular. O fato de, além da COBRA, outros fabricantes nacionais de computadores (como a SCOPUS e a ITAUTEC) usarem também o SOX com seus equipamentos forçará, mais e mais, a dinamização do mercado brasileiro de programas para o

SOX e outros similares – isto implica que breve você poderá ter sua biblioteca de programas para o SOX cada vez mais variada e seletiva. É importante lembrar, embora a Figura 5.1 não possa sugerir, que um novo programa pode ser desenvolvido com a ajuda de qualquer comando nativo do SOX (ou mesmo de um comando adicionado depois).

Permeando você e os diversos programas aplicativos do SOX está o **sh**, que também é, como já dissemos, um programa aplicativo, só que ele é um programa especial: funciona primordialmente como uma comunicação entre você e o SOX – é ele que aciona o comando que você mandou executar, que lhe avisa que o comando já foi executado e lhe diz que o SOX está pronto para receber seu próximo comando.

Reexamine a Figura 5.5. Um comando não se comunica diretamente com o hardware, mas com o *servidor SOX*, que fala com o *núcleo* do sistema operacional (que, por sua vez, controla o hardware). A comunicação entre um comando e o servidor SOX se dá através de *chamadas ao sistema*. Uma chamada ao sistema instrui o servidor SOX a fazer várias operações para o comando, além de trocar dados entre ele e o servidor.

Deste modo, para desenvolver um programa aplicativo para sua instalação, a equipe de desenvolvimento não precisa entender como o servidor SOX e o núcleo funcionam, mas necessita, tão somente, conhecer as chamadas ao sistema – seus significados e como funcionam; elas são em número de mais de 60, mas, em verdade, somente metade delas é usada com freqüência.

É fácil deduzir que o não-envolvimento com o servidor SOX e o núcleo do SOX (e muito menos com o hardware) vai permitir um enorme ganho de produtividade no desenvolvimento de aplicações sobre o SOX e esta é uma das razões capitais para a aceitação do SOX como paradigma de sistemas operacionais. Tudo isto porque o servidor SOX e o núcleo são módulos de grande complexidade, como lhe daremos esta idéia adiante – entendê-los bem não é uma tarefa trivial.

Agora, pensemos no seguinte: em que difere o SOX de um PC de um fabricante X do SOX de um supermini de um fabricante Y? A diferença está nos respectivos núcleos: o núcleo do SOX para um PC tem que conhecer o hardware desse PC, enquanto o núcleo do SOX para um supermini tem que conhecer o hardware desse supermini. Isso permite o importantíssimo detalhe de que todos os programas de aplicação são exatamente os mesmos em uma e outra máquina – assim, com o SOX, migrar de uma máquina pequena para uma grande (e vice-versa) deixou de ser um problema, e este fato confere ao SOX um grande trunfo em relação a outros sistemas operacionais. A razão para tal facilidade é que, qualquer que seja o ambiente computacional, as chamadas ao sistema são exatamente as mesmas.

Observando a Figura 5.1, você pode constatar que o **sh** sendo, no fundo, um comando como qualquer outro, também tem acesso ao servidor SOX.

Resta-nos passar a vista sobre as principais tarefas do servidor e do núcleo do SOX. Enumerêmo-las:

- Controlar a execução dos programas aplicativos, podendo também suspender sua execução, estabelecer uma comunicação entre dois programas etc. (Lembre-se: o SOX é um sistema multitarefa e multiusuário.)
- Organizar a lista dos programas que serão executados pela CPU. Os programas compartilham o tempo da CPU. (Lembre-se: o SOX é um sistema de tempo compartilhado.)
- Gerenciar a memória principal para a execução dos mais diferentes programas.
- Gerenciar a memória secundária para que o armazenamento e a recuperação dos dados dos usuários sejam feitos eficientemente.
- Controlar os acessos dos programas aos periféricos tais como terminais, discos magnéticos, fitas magnéticas etc.

Esperamos ter-lhe dado uma idéia do funcionamento do SOX como um todo. Na próxima seção, apresentamos quatro comandos muito representativos do SOX.

5.2 APRESENTAÇÃO DE ALGUNS COMANDOS DO SOX

O INTERPRETADOR DE COMANDOS

O Interpretador de Comandos do SOX (popularmente conhecido como **shell**), é um programa (comando **sh**) responsável pela interface entre o usuário (você) e o sistema operacional. É o Shell que interage junto ao SOX em seu benefício. É o shell que interpreta os seus comandos e proporciona as diretrizes apropriadas ao sistema para atendê-lo. Ele busca, na biblioteca de programas, o programa associado ao comando que você emite, traz o programa para a memória principal e, quando termina a execução do comando, apresenta o símbolo $ na tela, indicando que está pronto para atendê-lo no seu próximo comando.

OS COMANDOS PARA O SISTEMA DE ARQUIVOS

O Sistema de Arquivos é a estrutura de organização dos dados dos usuários. Ele tem uma representação em árvore hierarquizada, é simples e eficiente e é um outro grande fator de propaganda do SOX. Para criar e acessar essa árvore, há uma série de comandos nativos do SOX como **pwd, mkdir, cd, ls** e outros, todos de fácil compreensão e utilização.

O EDITOR DE LINHAS

O Editor de Linhas **ed** do SOX é um programa interativo que permite a criação de seus documentos e, muito importante, proporciona-lhe um diálogo com ele para corrigir erros, fazer modificações no texto, adicionar-lhe coisas e outras operações.

O COMPILADOR DE LINGUAGEM C

A Linguagem C vem-se tornando uma das mais populares no mundo todo. O próprio SOX foi quase todo escrito com ela (assim como o seu inspirador, o UNIX). Deste modo, é muito natural que os comandos nativos do SOX tenham sido escritos também em C, bem como muitos dos aplicativos que vão sendo adicionados a ele. O compilador desta linguagem, **cc**, também se confunde com o SOX, embora muitos outros compiladores da Linguagem C estejam disponíveis nos mais diversos sistemas operacionais. Isto é também muito bom para o SOX: a popularidade da Linguagem C facilita o transporte, para o SOX, de programas escritos em outros ambientes computacionais (a recíproca também é verdadeira); mais ainda, facilita também o transporte do próprio SOX para outras máquinas.

O uso do compilador **cc** está fora do escopo deste livro. Os demais comandos apresentados nesta seção, bem como muitos outros comandos nativos do SOX, serão estudados, com os detalhes que forem necessários, ao longo deste livro.

CAPÍTULO 6

BEM-VINDO AO SOX

Com exceção de alguns minutos no estudo do Capítulo 4 – quando você mexeu com seu terminal SOX – a leitura deste livro até aqui poderia ter sido – e muito provavelmente o foi – feita na praia, à beira da piscina ou à sombra, no acolhimento de sua espreguiçadeira preferida. Como acompanhamento, você não dispensou seu aperitivo favorito. Nada a ser reprovado nesta sua atitude, muito pelo contrário. Agora, porém, a coisa muda.

Este capítulo inicia o seu verdadeiro aprendizado do SOX. É aqui que você começa a interagir com os vários recursos e comandos do SOX. Portanto, levante-se, arregace as mangas, pegue seu aperitivo junto com este livro e vá direto para o seu terminal SOX. Não há maneira mais direta e rápida de aprender como utilizar os recursos do SOX do que praticando no seu terminal, com o sistema no ar, ao seu dispor.

Este capítulo põe você em contato com o SOX e discute as questões básicas para começar a usá-lo. Primeiro, conversaremos sobre a identificação de usuários no sistema e apresentaremos rapidamente alguns tipos de usuários especiais. Em seguida, mostraremos o procedimento para você se conectar (ou ter acesso) ao sistema e como se desconectar do SOX. O próximo passo é apresentar-lhe o interpretador de comandos do SOX. Finalmente, introduzimos alguns comandos simples, mas úteis, do sistema. Aproveitamos para ensinar-lhe como corrigir erros de digitação nos comandos ou nas informações que você passa para o SOX.

6.1 IDENTIFICAÇÃO DE USUÁRIOS

Na sociedade em que vivemos, as pessoas são identificadas por seus nomes, números de carteira de identidade ou de CPF etc. Tal identificação é necessária para a burocracia do sistema social, por comodidade em nossas conversas e contatos e imprescindível para as fofocas. De forma semelhante, cada usuário do Sistema SOX é identificado por um nome de usuário, o qual deve ser empregado por ocasião da conexão ao sistema. A identificação dos usuários no SOX é necessária para que eles possam enviar e receber mensagens de outros usuários, evitar uso indevido dos recursos do sistema a eles alocados e para o próprio bem da organização da instalação.

Diferentemente do que ocorre na sociedade, você pode escolher o nome pelo qual deseja ser identificado junto ao SOX. Sugerimos, porém, que você seja comedido com esta liberdade. O melhor é ser identificado pelo SOX com o mesmo nome que as pessoas empregam para dirigir-se a você no ambiente de trabalho. Isto simplificará a vida de seus colegas de empresa quando desejarem contactar-lhe por intermédio do SOX, ou quando perguntarem ao SOX quais usuários encontram-se conectados (através de um comando apresentado ainda neste capítulo). Assim, se seu nome é "José Silva" e o pessoal o chama de Silva, use "silva" como nome de usuário SOX.

Em algumas instalações, a identificação dos usuários SOX serve para fins de contabilização do uso de recursos do sistema. É por esta razão que a identificação de um usuário SOX é também chamada de *conta*. Fala-se, então, do *nome de conta* do usuário quando se refere à sua identificação no SOX. Nós, em particular, preferimos o termo *conta* e é o que adotamos neste livro.

Muito bem, você já deve ter escolhido o nome de sua conta no SOX. Digamos, "silva", para efeito de ilustração. Falta informar ao SOX sobre sua escolha. Você faz isto contactando o administrador do sistema e pedindo-lhe para cadastrar sua conta junto ao SOX, fornecendo-lhe o nome escolhido. Em outras palavras, você pede para "abrir sua conta no SOX". Enquanto ele faz isto, discutamos mais um pouco a questão de contas no SOX.

Em muitas empresas, um único usuário do SOX pode utilizar muitas contas. Isto acontece porque o usuário desempenha diversas atividades na empresa e prefere mantê-las separadas já que é uma pessoa sábia, associando uma conta a cada atividade. Por exemplo, numa certa empresa, a pessoa responsável pela produção de livros e manuais chama-se Tadeu. Ele tem uma conta com este nome – sua conta *pessoal* – e também acessa outra conta, de nome *manuais*, quando prepara os manuais e livros da empresa.

TIPOS DE USUÁRIOS SOX

O SOX, sendo um sistema operacional multiusuário, pode atender a vários usuários, cada um deles com pelo menos uma conta de acesso – a qual geralmente tem o nome da pessoa como ela é conhecida na empresa. A maioria das pessoas que utilizam o SOX é usuário *normal* do sistema, como secretárias, gerentes ou mesmo programadores. Um usuário normal vale-se dos recursos do SOX, associados à sua conta, para desempenhar suas atividades corriqueiras. Por exemplo, a secretária usa o SOX para editar correspondência, arquivar documentos, enviar e receber memorandos etc. Um gerente de vendas é atendido pelo SOX na preparação de relatórios das vendas mensais, na confecção de planos estratégicos etc. O programador, com as ferramentas do SOX, desenvolve novos programas de contabilidade, controle de estoque ou contas a pagar. Comumente, estes usuários normais não são responsáveis pela boa operação e administração da instalação SOX. Eles meramente a usam e apreciam ser bem servidos. Você, possivelmente, começará seu contato com o SOX como um usuário normal.

O SOX também admite o agrupamento de vários usuários sob uma única identificação ou conta. Esta facilidade reflete bem a estrutura organizacional da sua empresa. Ilustremos: vários funcionários, de vários departamentos, podem ser atribuídos à tarefa de fazerem o planejamento financeiro para a empresa, para o ano 1988. É claro que estes funcionários estarão trabalhando, por algum tempo, em assuntos relacionados. Para uma melhor organização e comunicação interna, a empresa decide chamá-los coletivamente de *grupo do planejamento*. Para tanto, o administrador do sistema cadastra um grupo chamado *planejamento* e indica que, por exemplo, os usuários com nomes *joão*, *alexandre* e *otávio* pertencem a este grupo. Observe que *planejamento* não é um nome de usuário. Agora, quando o usuário joão entrar no ar, ele adquirirá todos os direitos e responsabilidades de um usuário do grupo de planejamento. O SOX usa o conceito de grupo para decidir quem pode acessar determinados arquivos do sistema. Explicaremos esse ponto com mais detalhes no Capítulo 10.

Além dos usuários normais e dos grupos de usuários numa instalação SOX, queremos apresentar-lhe uma outra classe de usuários. Estes últimos têm identificação especial junto ao SOX. São os chamados superusuários. Na sua instalação deve existir pelo menos um, cujo nome de conta é *root*. O superusuário *root* é seu amigo administrador. Ele tem poderes ilimitados junto ao SOX, abrindo contas, identificando usuários, cuidando para que o SOX execute de forma devida. Ele dispõe de acesso irrestrito a todos os comandos e recursos do SOX, mesmo àqueles indisponíveis a outros usuários. Ele é um usuário *anormal*, senhor absoluto do SOX, seu amigo de cafezinho. Cobre-lhe a abertura de sua conta.

Talvez existam outros superusuários na sua instalação, com poderes menos abrangentes que os do *root*. Não é nosso objetivo discuti-los neste livro. Só para satisfazer sua curiosidade, um deles chama-se *bin*. O superusuário *bin* cuida dos comandos do SOX alojados nos diretórios /bin e /usr/bin e das bibliotecas residentes nos diretórios /lib e /usr/lib. Nada impede que todos os superusuários SOX estejam encarnados na única pessoa do administrador ou em alguns dos seus auxiliares (caso existam).

Não admire quando um dia você se encontrar no papel de *root* numa instalação SOX. Com a popularização deste sistema operacional e a disponibilidade do SOX em supermicros, há instalações SOX em pequenas empresas e até para uso em escritórios domésticos. Nestes casos, você, dono do sistema, será o próprio administrador. Não se apavore com isto. Este livro é um bom começo para iniciar sua convivência com o SOX e, dentro em breve, com prática e um pouco mais de boa leitura, você se terá tornado um superusuário SOX, um usuário anormal.

SENHAS

Se você usou seu nome para sua conta SOX, como evitar que outras pessoas conectem-se ao sistema como se fossem você? Como todos conhecem seu nome, você não terá privacidade na sua conta nem impedirá intrusos. O SOX resolve este problema de segurança no acesso à sua conta com o uso de senhas.

O SOX permite que você associe uma senha à sua conta. Só quem conhece a senha poderá ter acesso a sua conta no SOX. A senha é uma cadeia de caracteres (p.ex., "S! O!X") que você fornece ao SOX no procedimento de conexão, após digitar seu nome de conta. Se a senha fornecida não for a que você escolheu, a conexão não será estabelecida. Para segurança, escolha a senha de sua conta sigilosamente e não a forneça a ninguém, nem mesmo ao administrador (observe que o usuário root pode acessar seus arquivos sem restrição e sem saber sua senha. Mesmo assim, não forneça a senha a **ninguém**).

Você pode alterar sua senha a qualquer instante (veremos como, mais adiante).

Pronto, o administrador já deve ter aberto sua conta no SOX. Vejamos como começar a utilizá-la.

6.2 COMO SE CONECTAR AO SOX

Ligue seu terminal; o cursor deve aparecer na tela. Caso ele não se apresente, pressione a tecla < CR > . Talvez seja necessário pressionar < ESC > , < BREAK > etc., dependendo do seu equipamento. Se não aparecer nada na tela, o sistema está fora do ar e o administrador deve ser avisado para colocá-lo em execução. Se o que aparecer na tela for incompreensível (*lixo*), há incompatibilidade entre a velocidade de seu terminal e o computador. O ajuste de velocidade é feito automaticamente em alguns equipamentos (pressione < CR > ou < BREAK > mais algumas vezes) e, em outros, manualmente. Neste último caso, consulte os manuais de operação do terminal ou o administrador.

Estando tudo OK (o SOX rodando e o terminal com velocidade adequada), aparecerá na tela a identificação do sistema SOX (particularizada pelo fornecedor da máquina) seguida do pedido de *login* (conexão). A tela terá informações arranjadas de forma semelhante à mostrada abaixo:

```
** IDENTIFICAÇÃO DO SEU SISTEMA SOX**
login: _
```

Como vê, o cursor (aqui representado pelo traço de sublinhamento, "–") fica posicionado imediatamente após a mensagem de login. Você deve, então, digitar o nome de sua conta SOX, seguido de < CR > . Suponha que o nome de sua conta é "silva"; teríamos então:

```
** IDENTIFICAÇÃO DO SEU SISTEMA SOX**
login: silva<CR>
```

Use apenas letras minúsculas, caso contrário o sistema poderá usar maiúsculas até que você se desconecte e se conecte novamente. Não se preocupe se você cometeu algum erro na digitação de seu nome. Você verá como corrigi-lo adiante.

Após ler o seu nome de conta, o SOX pedirá sua senha. A tela será então:

```
**IDENTIFICAÇÃO DO SEU SISTEMA SOX**
login: silva
senha: _
```

Tecle agora a sua senha (se já tiver uma) e pressione <CR> . Por questão de segurança, a senha, ao contrário do nome de conta, não aparece na tela. Isto evita que sua senha seja revelada a quem observa a tela de seu terminal.

Se você ainda não tem uma senha na sua conta SOX (veremos como estabelecê-la dentro em breve), simplesmente tecle <CR> . Caso você tenha esquecido qual é a sua senha, fale com o administrador.

Se a associação nome-de-conta/senha for válida, você estará conectado ao SOX. Este poderá então lhe fornecer algumas informações como notificações de chegada de mensagens que foram depositadas na sua conta enquanto você esteve desconectado, avisos de períodos de indisponibilidade do SOX para manutenção preventiva pelo administrador, data e hora correntes etc. Estas informações são controladas pelo pessoal responsável pela operação do seu sistema SOX (p.ex., o administrador). Não há nada errado com seu sistema se elas não aparecem. Logo depois, deve surgir na tela o sinal "$" (ou outro símbolo típico de sua instalação). "$" indica que a conexão foi bem-sucedida e o SOX está à disposição para acatar seus comandos. Vide a seguir:

```
**IDENTIFICAÇÃO DO SEU SISTEMA SOX**
login: silva
senha:
**MENSAGENS DO SISTEMA (DIA, HORA, ...)
$ _
```

Se você cometeu algum engano na digitação do nome de sua conta ou da sua senha, o SOX dará um aviso de *login incorreto* e não estabelecerá a conexão. Você então terá a chance de repetir o procedimento (vide abaixo).

```
**IDENTIFICAÇÃO DO SEU SISTEMA SOX**
login: silvva
senha:
login incorreto
login: _
```

No caso acima, houve erro no nome da conta (o "v" foi duplicado). Redigite o nome de sua conta e senha; você deverá obter êxito. Em caso de insucesso – e você tem certeza de

que não cometeu erros (principalmente na senha, que não é vista na tela) – fale com o administrador. Ele pode ter esquecido de abrir a sua conta.

Agora que você já sabe como se conectar ao SOX, talvez deseje saber como se desconectar. É o que veremos a seguir.

6.3 COMO SE DESCONECTAR DO SOX

Quando você encerra uma sessão de trabalho com o SOX, pode se desconectar do sistema, digitando

<CTRL> <D>

quando aparecer o sinal "$". Supondo que você acaba de se conectar e deseja se desconectar imediatamente; digite a parte em negrito na tela abaixo.

```
**IDENTIFICAÇÃO DO SEU SISTEMA SOX**
login: silva
senha:

**MENSAGENS DO SISTEMA (EX: DIA, HORA, ...)
$ <CTRL> <D>
```

Ao receber o <CTRL> <D> , o SOX apresentará novamente o pedido de login (confirmando a desconexão):

```
**IDENTIFICAÇÃO DO SEU SISTEMA SOX**
login: _
```

Aqui cabem três observações. Primeiro, o procedimento de desconexão é também chamado de *logout* ou *logoff*. Segundo, se seu terminal está conectado à máquina SOX via MODEMS e linha telefônica, o logoff provavelmente desligará também o telefone, o que obviamente impedirá a obtenção do novo pedido de login. A terceira e última observação é uma dica prática. Não se acostume a fazer desconexão do SOX simplesmente desligando o seu terminal. Se você assim proceder (em alguns sistemas

SOX) poderá dar margem para qualquer outro usuário – na sua ausência – religar o terminal e ter acesso à sua conta. Sempre encerre uma sessão de trabalho no SOX com o comando "^D" (emitido pressionando-se as teclas <CTRL> e <D> , simultaneamente). Você terá certeza da desconexão, se o sistema apresentar-lhe a mensagem de login na tela. Agora sim, desligue o terminal e durma tranqüilamente.

6.4 O INTERPRETADOR DE COMANDOS SOX

O Interpretador de Comandos do SOX (popularmente conhecido como *Shell*) é um programa responsável pela interface entre o usuário (você) e o sistema operacional. É o Shell que interage junto ao SOX em seu benefício. Ele interpreta os seus comandos e proporciona as diretrizes apropriadas ao sistema para atendê-lo. O Shell busca, no disco, o programa associado ao comando que você emite (ao ver $), traz o programa para a memória principal e, quando termina a execução do comando, apresenta o símbolo "$" na tela, indicando que está pronto para atendê-lo no seu próximo comando.

O sinal $ que aparece na tela após as mensagens do sistema no procedimento de conexão discutidos na Seção 6.2 é o **sinal de prontidão** do Shell (ou do sistema). Os comandos para o SOX são passados na linha da tela iniciada pelo $. Por isto, tal linha é denominada de linha de comando. Para ilustrar a operação do Shell, lançaremos mão de três comandos do SOX:

- **passwd**
- **date**
- **who**

O comando de nome **passwd** serve para instituir ou alterar a senha de sua conta. O segundo, **date**, informa a data e hora correntes e o terceiro, **who**, descobre quais usuários estão presentemente conectados ao sistema. Com estes três comandos, você já poderá ter uma breve sessão de trabalho com o SOX. Vamos ao primeiro.

MUDANÇA DE SENHA

Conecte-se ao sistema (usando sua senha atual) da forma indicada na seção anterior. Quando aparecer o sinal de prontidão do Shell ($), digite **passwd<CR>** , assim:

```
$ passwd<CR>
```

O sistema pedirá que você forneça a sua senha antiga (atual), com a mensagem do tipo:

```
$ passwd
senha atual: _
```

Forneça a senha atual (observe que ela não é mostrada na tela, por questão de segurança) e tecle <CR> . Se você ainda não tiver uma senha, digite apenas <CR> . O programa **passwd** pedirá agora a sua nova senha com uma mensagem semelhante a:

```
$ passwd
senha atual:
senha nova: _
```

Digite sua nova senha seguida de <CR> . Senhas mais longas são mais difíceis de advinhar. Lembre-se disto para aumentar a segurança de sua conta. O sistema SOX admite senhas com até 8 caracteres. Você pode usar qualquer caractere na composição de sua senha.

Certos caracteres – os chamados *metacaracteres* – têm significado especial para o Shell. Os metacaracteres incluem ?, @ , / ,* . Evite utilizar tais caracteres no nome de sua conta SOX e na senha a ele associada, pois você poderá ter problemas no procedimento de conexão (login).

Quando tiver sua nova senha seguida de <CR> , o sistema pedirá para você redigitá-la, com a tela:

```
$ passwd
senha atual:
senha nova:
Redigite senha nova: _
```

Isto minimiza a chance de ocorrência de um erro de digitação, pois você não vê a senha que digita. Digite a nova senha mais uma vez e tecle <CR> .

A alteração (ou instituição) da senha será efetivada apenas quando ocorrer correspondência entre as suas duas digitações da nova senha. Se ocorrer de você esquecer sua senha, peça ao administrador do sistema para que ele tome as providências que lhe permitam usar a sua conta.

DATA E HORA CORRENTES

Freqüentemente, enquanto você trabalha com o SOX, é necessário saber a data e a hora atuais. Isto é possível com o comando **date** do SOX, o qual você emite após o sinal de prontidão do Shell "$" e conclui com um <CR> . A resposta do SOX será algo parecido com o mostrado na tela abaixo:

```
$ date<CR>
Sex Abr 24 08:10:05 1987
$ _
```

Lembre-se de que você deve digitar a parte em negrito. A resposta do SOX indica que hoje é sexta-feira, 24 de abril de 1987 e 8 horas, 10 minutos e 5 segundos da manhã. Se você perceber imprecisão na informação do SOX, contacte o administrador. Ele, como superusuário, poderá acertar o *relógio* do seu sistema. Esta é uma das tarefas que só o superusuário pode desempenhar. A propósito, ele usará o próprio comando **date** com uma ligeira alteração de formato (uma opção especial do comando **date**).

Em caso de cometer um erro de digitação no comando, digamos você digitou **dste** em vez do correto **date**, nada de muito grave ocorrerá após você concluir com o <CR> . Neste caso, o SOX informará que o comando **dste** não existe com uma mensagem semelhante a:

```
$ dste<CR>
dste: comando não encontrado
$_
```

Veremos como corrigir erros como estes logo mais.

USUÁRIOS NO SISTEMA

Freqüentemente, você precisará saber quais usuários estão conectados ao SOX, por mera curiosidade ou para se comunicar com um determinado usuário. O comando do SOX que lhe informa quais usuários estão no sistema é **who.**

O comando **who** deve ser emitido na linha de comando, após o sinal de prontidão do Shell "$". Após digitar o comando, tecle <CR> .O <CR> serve para indicar ao Shell que você concluiu a linha de comando e que o SOX pode iniciar o atendimento do seu pedido ou comando.

```
$ who<CR>
```

Observe a saída na tela do seu terminal. O SOX informa o nome de conta de cada usuário conectado ao sistema, o terminal que está sendo utilizado (tty0, tty1, ... etc.) e a hora do estabelecimento da conexão (vide exemplo abaixo).

```
$ who
silva     tty0        abr 26        08:05
tadeu     tty1        abr 26        08:20
manuais   tty5        abr 25        22:12
adm_ltd   tty7        abr 26        07:44
$ _
```

Caso você cometa algum erro na digitação do comando **who** e tecle <CR> , o SOX indicará que não foi possível encontrar o comando – a não ser que o erro tenha sido tal que o tenha transformado noutro comando SOX. Portanto, tome cuidado, pois o seu erro pode trazer resultados indesejados ou perigosos para sua conta. A próxima seção mostra como corrigir erros de digitação.

6.5 CORREÇÃO DE ERROS DE DIGITAÇÃO

Não só é possível, como também provável, que você cometa alguns enganos ou erros ao digitar um comando para o Shell. Antes de teclar <CR> , é possível remediar estes erros. Mesmo quando você teclar <CR> nada de grave ocorrerá (salvo em situações especialíssimas). Shell simplesmente indicará que não encontra o comando que você emitiu e aparecerá outro "$" ficando à espera de suas novas ordens.

Por exemplo, suponha que você digitou **whho** (com dois agás) no último comando estudado na seção anterior. O SOX teria agido da maneira abaixo.

```
$ whho <CR>
whho: comando não encontrado
$ _
```

Se você tivesse notado o erro de digitação antes do <CR> na linha de comando, poderia remediar a situação.

Para corrigir caracteres errados numa linha de comandos (que você escreve após o "$" e termina com <CR>), use as teclas de retrocesso (<←> ou < CTRL > < H >) para posicionar-se no lugar da correção e então tecle o caractere correto. O novo caractere substituirá o anterior. Você terá que refazer a porção da linha de comando (caso exista) à direita do caractere corrigido.

As teclas de retrocesso, <←> , <CTRL> <H> , são usadas para apagar o último caractere digitado.

Para cancelar toda a linha de comando digitado, use a tecla <@> , antes do <CR> . Ao digitar <@> , o cursor saltará para a próxima linha na tela e esperará que você digite a nova linha de comando, seguida de <CR> .

Convém observar que as teclas de retrocesso e de apagamento de linha são configuráveis, ou seja, você ou o administrador pode atribuir ou alterar sua funcionalide. Converse com o administrador sobre o assunto.

Vamos praticar um pouco o procedimento de correção, usando o comando (errado) **whho**. Digite **whho** após o sinal de prontidão do Shell, "$", mas não tecle <CR> .Você terá então:

```
$ whho_
```

Agora, corrijamos o erro ("h" duplo). Retroceda o cursor para o segundo "h", teclando <←> ou <CTRL><H> , duas vezes. A primeira vez retrocede o cursor para o caractere "o", apagando-o; a segunda para o "h" em excesso, apagando-o também. Tecle agora <o> – você terá então **who**, ou seja, o nome correto do comando – e tecle <CR> . O SOX executará o comando corretamente, com resultado já descrito no final da seção passada.

CANCELAMENTO DA LINHA DE COMANDO

Ao aparecer o "$", depois de concluída a execução acima, introduza **whho** mais uma vez, mas não tecle <CR> . Desta feita, vamos cancelar toda a linha de comando, com a tecla <@> . Ao teclar <@> , Shell cancela a linha de comando atual (mas não a apaga da tela) e posiciona o cursor na próxima linha (sem, contudo, apresentar o sinal de prontidão "$") à espera de novo comando. Veja abaixo.

```
$ whho <@>
_
```

Digite agora o comando corretamente, seguido de <CR> .

```
$ whho@
who<CR>
```

O SOX fornecerá agora o resultado desejado como indicado no final da Seção 6.4.

É importante notar que você pode corrigir erros de digitação no comando emitido na linha seguinte ao cancelamento com <@> , da mesma forma descrita mais acima (desde que antes do <CR>). Basta utilizar as teclas de retrocesso <←> ou < CTRL ><H> . Suponha que você tenha digitado:

```
$ whho @
whho
```

Como vê, você repetiu o mesmo erro. Use a tecla <←> duas vezes (para anular o "o" e o segundo "h"), digite "o" e tecle < CR > .

O resultado será o esperado.

Você poderia também cancelar a segunda linha de comando com <@> e, então, digitar o comando na linha seguinte. Experimente.

6.6 INTERRUPÇÃO DE COMANDO

Pode acontecer de você emitir um comando para o SOX e então desistir do resultado. Neste caso, você digitou o comando e teclou < CR > . O Shell passa então a tomar as providências para executar o seu pedido. Aí você desiste. Pode acontecer também do comando já estar sendo executado e listando a saída na tela. Só que a saída é muito longa e você não quer esperar pelo final. Em ambos os casos você pode interromper ou abandonar a execução do comando teclando <BREAK> , <RUBOUT> , < DEL> ou <CANCELA> (verifique seu teclado). O comando será interrompido e o SOX apresentará o sinal de prontidão "$".

Ilustremos o segundo caso, com o comando **who** e a tecla <DEL > (ou a tecla equivalente no seu teclado). Lembre-se de que, agora, o Shell já está mostrando a saída do comando.

```
$/ who<CR>
silva        tty0        abr 26        8:05
tadeu        tty1        ab<DEL>
$_
```

Quando o comando **who** completava a segunda linha da saída, você interrompeu o comando com a tecla <DEL> . O SOX abandonou o restante da saída e o Shell apresentou o sinal de prontidão, à espera do seu novo comando.

CAPÍTULO 7

O SEU PRIMEIRO ARQUIVO

No Capítulo 4, "Dicas e Conceitos Preliminares", você aprendeu alguns conceitos básicos fundamentais como *arquivo, edição de arquivos, diretórios* e *sistemas de arquivos, comandos* etc. Chegou a hora de começar a praticar estes conceitos, criando e organizando logicamente seus arquivos, verificando seus conteúdos etc.

Como foi visto, um **Sistema de Arquivos** no SOX compõe-se de diretórios e arquivos estruturados hierarquicamente. Imaginemos que seu **diretório-casa** (lembra-se deste conceito? se não, reveja-o no Capítulo 4) é **voce** e que você tem vários arquivos classificados como **memos, cartas** e **miscelanea.** Em **miscelanea,** você tem **poemas** e **outros.** Você aprenderá a montar esta estrutura e incluir um trecho de **Folhas de relva,** de **Walt Whitman** (século XIX), com o nome de **folhas** em **poemas.** Isto implicará o uso de alguns comandos de manipulação de diretórios que serão revistos detalhadamente no Capítulo 10.

A Figura 7.1 expõe um Sistema de Arquivos incluindo o diretório **voce.**

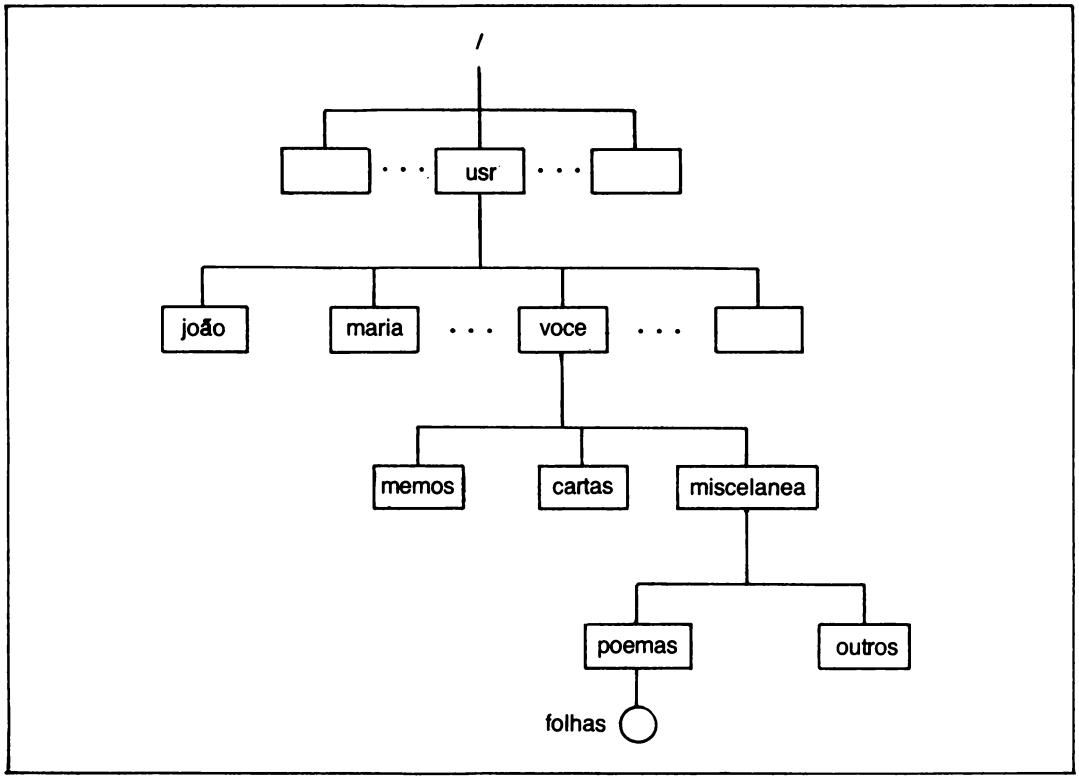


Figura 7.1 Um Sistema de Arquivos.

Como você pode observar, o diretório-casa **voce** tem três subdiretórios: **memos, cartas** e **miscelanea. Miscelanea** , por sua vez, tem dois subdiretórios: **poemas** e **outros. Poemas** contém o texto **folhas. Poemas, folhas** etc. são todos **arquivos**, de tipos diferentes: **poemas** é um arquivo tipo **diretório**, enquanto **folhas** é um arquivo tipo **texto.**

Embora nosso exemplo não mostre, um diretório pode ter como conteúdo tanto arquivos de texto como outros diretórios (subdiretórios). Você verá todos os detalhes sobre sistemas de arquivos, repetimos, no Capítulo 10. Nossa intenção, no momento, é apenas colocar o texto-exemplo **folhas** no seu lugar apropriado, ou seja, **poemas.** Com isto queremos, desde o início, chamar sua atenção para a boa metodologia de guardar seus documentos devidamente localizáveis por assunto.

7.1 CRIANDO E LISTANDO DIRETÓRIOS: OS COMANDOS *pwd, ls, mkdir* E *cd*

Quando você entra no SOX, é automaticamente colocado no seu **diretório-casa** (veja o Capítulo 4). Este diretório foi criado pelo SOX e recebeu o nome de sua conta, **voce**, no nosso exemplo. **joao, maria** etc. são outros diretórios-casa que, por sua vez, são subdiretórios do diretório **usr** do SOX. Este último está ligado ao **diretório-raiz /** do SOX.

Para se certificar de que você está em seu diretório-casa, use o comando **pwd** (do inglês **p**rint **w**orking **d**irectory: imprima diretório de trabalho – lembre-se: o **SOX** é um Sistema Operacional **compatível** com o norte-americano UNIX e um dos requisitos para a compatibilidade é a interface comum).

```
$ pwd<CR>
/usr/voce
$
```

Para indicar o término de qualquer comando, é obrigatório pressionar a tecla <CR> (veja o Capítulo 4). O SOX responde dizendo que seu diretório corrente é **voce**. Ele também lhe dá a posição de **voce** na hierarquia: **/** é o diretório-raiz, depois vem **usr** e, finalmente, **voce**. O segundo "/" é apenas um *separador* (mais detalhes no Capítulo 10).

A partir de seu diretório-casa **voce** (no sentido descendente) você poderá criar seus diretórios e seus textos. Neste momento, o diretório **voce** está *vazio*. Para obter a confirmação disto digite o comando **ls** (do inglês **l**ist **s**tatus: liste status desse arquivo), que é o comando do SOX para listar conteúdos de diretórios.

```
$ ls <CR>
$
```

Qual a resposta do SOX ao comando **ls?** Nada, evidentemente. Ele está pronto para receber um novo comando (caractere de prontidão $).

Para criar subdiretórios de **voce**, use o comando **mkdir** (do inglês **make dir**ectory: construa diretório). O argumento para este comando é simplesmente o nome do diretório que você quer criar.

```
$ mkdir memos <CR>
$ ls <CR>
memos
$ mkdir cartas <CR>
$ ls <CR>
cartas
memos
$ mkdir miscelanea <CR>
$ ls <CR>
cartas
memos
miscelanea
$
```

Observe que o SOX lista os arquivos em ordem alfabética. Observe também que, apesar de ter criado estes novos diretórios, você continua no diretório **voce** e a listagem via **ls** reflete exatamente isto.

O passo seguinte é criar os subdiretórios **poemas** e **outros**, de **miscelanea**. Para isto, é conveniente se posicionar neste último diretório. O comando correspondente é **cd** (do inglês change directory: mude diretório), seguido do novo diretório corrente.

```
$ cd miscelanea <CR>
$ pwd <CR>
/usr/voce/miscelanea
$ ls <CR>
$
```

Vamos então criar **poemas** e **outros**.

```
$ mkdir poemas <CR>
$ ls <CR>
poemas
$ mkdir outros <CR>
$ ls <CR>
outros
poemas
$ pwd <CR>
/usr/voce/miscelanea
$
```

Para relembrar todos esses conceitos, saia do SOX (<CTRL>-<D>), entre nele novamente e posicione-se no diretório **miscelanea.**

Feito isso, posicione-se agora em **poemas.**

```
$ cd poemas <CR>
$ pwd <CR>
/usr/voce/miscelanea/poemas
$ ls <CR>
$
```

Pronto! Podemos agora pensar em como incluir neste último diretório o texto poético **folhas.**

7.2 EDIÇÃO DE TEXTOS NO SOX

O SOX tem dois editores de texto: o **ed** e o **vi.**

Um **editor** é um programa de serviço que você usa para digitar o conteúdo de um arquivo de texto ou para fazer modificações nele. Um editor é sempre **interativo,** ou seja, você pode ver o que já fez antes de decidir fazer mudanças.

Podemos classificar editores em duas categorias: editores **de linha** e editores **de tela.** O **ed** é um editor de linha, enquanto o **vi** é um editor de tela.

Por um editor de linha, entendemos aquele em que a unidade básica é uma **linha** (uma cadeia de caracteres terminada por um caractere indicador de uma nova linha). Os comandos desse editor executam operações em linhas: você pode adicionar linhas, imprimir, mudar seu conteúdo, eliminar, inserir, mover ou copiar linhas para um outro local do seu texto, localizar e/ou substituir um grupo de caracteres em várias linhas, e assim por diante.

Editores de tela (também chamados editores *visuais*) têm como unidade básica a *tela* de seu terminal: um cursor se movimenta sobre ela para indicar qualquer ponto da tela de onde você deseja realizar alguma operação.

Editores visuais são mais charmosos que editores de linha. No entanto, requerem mais prática de uso que os últimos. Assim, veremos somente o editor de linhas do SOX – além de sua simplicidade, seu aprendizado facilitará um posterior uso de um editor visual.

Ambos, editores de linha e editores visuais, têm um ponto em comum – as operações não são realizadas diretamente sobre os originais do texto: em vez disto, o texto (se existir) é copiado para uma área de trabalho da memória principal chamada **buffer** e as operações são realizadas nesse buffer. Cabe ao usuário, através de um comando apropriado, **salvar** o arquivo, ou seja, gravar o arquivo modificado no buffer sobre o arquivo original. O usuário pode salvar seu arquivo sempre que achar conveniente. A última operação de salvamento reflete a versão final do texto no buffer.

Uma vantagem desta estratégia é que, se você cometer algumas "besteiras", estas poderão não comprometer seu arquivo original.A desvantagem é que,em caso de interrupção anormal de seu sistema (por exemplo, uma queda de energia), você poderá perder grande parte do trabalho já feito. Uma maneira de minimizar este problema é salvar o arquivo a cada intervalo não muito grande de tempo (por exemplo, a cada cinco minutos).

7.3 INTRODUÇÃO AO EDITOR DE LINHAS ed

Nesta seção, veremos como criar e salvar arquivos de texto através do editor de linhas **ed**. Veremos também como inspecionar o conteúdo de um texto. O tratamento dado ao **ed** neste capítulo será, quase sempre, informal. Um tratamento mais rigoroso será observado no capítulo seguinte, que também trata do editor **ed**.

Para colocar o **ed** em execução, você usa o comando homônimo **ed** do SOX. Uma vez acionado, você *sai* do ambiente do SOX e *entra* no ambiente do **ed**. Uma característica reveladora do ambiente do SOX é a emissão por ele mesmo do caractere de prontidão **$**, quando espera um comando do usuário. No ambiente do **ed**, ele normalmente não emite qualquer caractere de prontidão (o **ed** é naturalmente econômico com palavras!). Entretanto, você pode definir para ele um caractere de prontidão de sua escolha (pode ser também um *grupo* de caracteres) que ele passará a usar como tal.

Uma vez no **ed**, você pode usar uma série de comandos, para editar seus textos, salvá-los, além de outros. Somente após sair do **ed** é que você volta ao ambiente do SOX (caractere de prontidão $, comandos próprios do SOX etc.).

A Figura 7.2 ilustra a idéia de entrar e sair do editor.

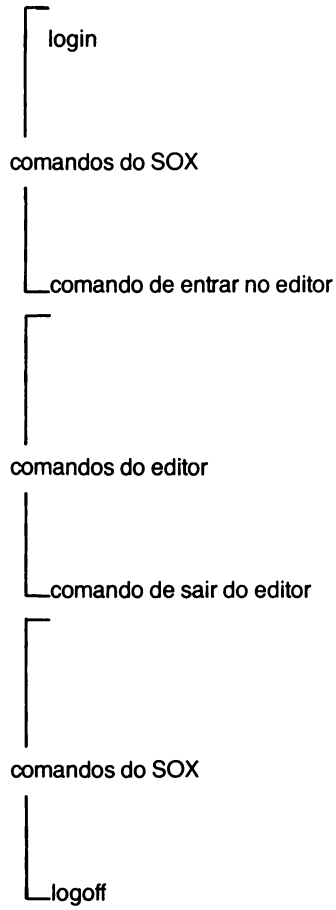


Figura 7.2 Uma Sessão do SOX com Uso do Editor de Linhas.

Vamos então criar, com o editor **ed**, o arquivo **folhas**, contendo o poema *Folhas de relva*. Posicione-se no diretório **poemas**. Assim, um arquivo criado com o **ed** estará automaticamente ligado ao diretório **poemas**.

A forma geral do comando do SOX para colocar o **ed** em execução é:

```
ed [ opções ] [ nome do arquivo ]
```

Uma das opções que pode ser usada é para indicar que você deseja que o **ed** imprima caractere(s) de prontidão e qual(ais) é(são) ele(s). O nome do arquivo também é opcional. Quando você vai criar um arquivo, ele ainda não existe para o **ed**, logo, é mais natural você informar o nome do seu arquivo somente quando da primeira vez que você for salvar as linhas digitadas. Quando o arquivo já existe, o melhor é dar o seu nome logo na chamada do **ed.**

```
$ ed -p - <CR>
-
```

A opção **-p** indica ao **ed** que o(s) caractere(s) seguinte(s) até um espaço ou um <CR> é (são) de prontidão. No exemplo, o caractere de prontidão que passará a ser adotado pelo **ed** é – . O **ed** está pronto, portanto, para receber seus comandos.

Vejamos como seria se chamássemos o **ed** sem a opção **-p.**

```
$ ed <CR>
```

Observe que nada acontece! Na verdade, o **ed** está esperando um comando seu. Sempre que ele estiver parado desconfie que ele está esperando um comando! De forma geral, tudo está certo se o **ed** não conversar. Ele conversa apenas quando você faz algo que ele não entendeu.

Se, usando o **ed** sem caractere de prontidão, você quiser passar a trabalhar com caractere de prontidão, não há necessidade de sair do editor e entrar nele novamente com a opção **-p**: o comando **P** (maiúsculo) faz com que o editor lhe dê o caractere de prontidão* (observe: com **-p** você pode escolher um caractere (ou um grupo de caracteres) de prontidão de seu gosto, ao contrário deste caso, em que você é compulsoriamente servido com o caractere *). O editor continuará imprimindo este caractere de prontidão até que você dê um outro comando **P** – trata-se de um *comando alternante* (liga/desliga). Todo comando do **ed** deve ter obrigatoriamente um <CR> no seu final.

```
$   ed <CR>
P <CR>
*
```

Se você cometer algum engano quando estiver digitando um comando do **ed**, poderá redigitá-lo, desde que perceba o engano **antes** de pressionar a tecla <CR> . Para isto, você pode utilizar a tecla <← > , ou equivalente, de seu teclado, que faz o cursor retroceder na linha, apagando caracteres digitados, permitindo que você introduza novos caracteres. Caso o engano não seja percebido a tempo, o **ed** poderá ajudá-lo a descobrir a causa de seu erro (veremos isso logo mais).

É importante entender desde já o conceito de *número de linha*. Já vimos que a edição de um texto se dá num buffer para ele alocado. Sabemos também que o **ed** interpreta seu texto como um conjunto de linhas. Para facilitar sua manipulação dessas linhas, o **ed** as numera com números inteiros seqüenciais. Tais números de linha não fazem parte do texto. No entanto, você pode identificar uma linha dentro de um comando através de seu número de linha, o **ed** pode exibir as linhas com seus devidos números etc. Quando você salvar o texto, evidentemente tais números não serão gravados: o texto salvo consistirá única e exclusivamente nos caracteres digitados por você.

Comandos do **ed** são compostos de uma única letra, opcionalmente precedida por um número ou um conjunto de números de linha.

O **ed** tem dois *comandos de ajuda,* **h** e **H**. Quando você digita um comando que não existe, ou em qualquer outra situação de erro ou exceção, o **ed** emite um **?**. Para que você saiba do que se trata, sem ter que recorrer a um Manual, poderá utilizar os comandos **h** ou **H**, que lhe explicarão, de forma sintética, o que estará ocorrendo. Se você usar o comando **h** (minúsculo), o **ed** explicará o último **?** emitido. Assim, para cada **?** você terá que usar o comando **h** novamente. Já o comando **H** (maiúsculo) obriga o **ed** a explicar todos os "?\s" subseqüentes *até um outro H* – trata-se de mais um *comando alternante*.

```
$ ed -p -<CR>
-z<CR>
?
-h<CR>
comando inexistente
-
```

```
-H<CR>
-z<CR>
comando inexistente
-
```

Para introduzir os versos de nosso poema use o comando **a** (do inglês **a**ppend:

acrescente, adicione). O cursor imediatamente se posiciona no início da linha seguinte, à espera de seus caracteres. Cada vez que você pressionar a tecla <CR> , o cursor se deslocará para o início da linha seguinte. Para indicar o término das linhas que estão sendo acrescentadas, você deve digitar o caractere especial . *na primeira posição da linha*, seguido de <CR> .

```
-a < CR >
          Estes são em verdade os pensamentos de todos os  <CR>
            homens em todas as épocas e países: não são  <CR>
                                          [ originais meus.  <CR>
          Se não são tão teus quantos meus, nada são os  <CR>
                                          [ quase nada,  <CR>
          Se não são o enigma e a solução do enigma, nada  <CR>
                                          [ são,  <CR>
          Se não são tão próximos quanto distantes,     nada  <CR>
                                          [ são.  <CR>
          < CR >
          Esta é a relva que cresce onde há terra e água,  <CR>
          Este é o ar comum que banha o planeta.  <CR>
. < CR >
-
```

Algumas observações sobre a digitação:

- Controle o espacejamento antes dos versos utilizando a tecla de tabulação <TAB> (ou equivalente). Esta tecla, quando acionada, desloca o cursor até a próxima *parada de tabulação*. As paradas de tabulação são *invisíveis* e são colocadas a intervalos de *oito colunas* (primeira parada: coluna nove, segunda parada: coluna dezessete, e assim por diante). No nosso exemplo, se o cursor estiver em qualquer coluna anterior à nove, pressionando <TAB> o cursor se posiciona automaticamente na coluna nove. Assim, <TAB> pode ser de muita utilidade: a alternativa, bem mais ineficiente e deselegante, seria você teclar a barra de espacejamento tantas vezes quantas fossem necessárias.

- Uma linha em branco é obtida simplesmente pressionando a tecla <CR> .

- Consulte o administrador de sua instalação para verificar se você pode acentuar palavras, e como.

- Você pode corrigir uma linha que está sendo digitada da mesma forma que corrige um comando que está sendo digitado.

- Não tente corrigir agora linhas já digitadas: no capítulo seguinte você conhe-

cerá as ferramentas do **ed** para a correção de seus textos.

Como já explicamos anteriormente, tudo o que você digitou está em um buffer da memória principal: para gravar em memória permanente você usa o comando **w**, que grava o conteúdo do buffer na sua memória permanente e no seu diretório (no caso, **poemas**). Ao fazê-lo, o **ed** também informa o número de caracteres transferidos para a memória permanente: neste número estão incluídos caracteres *invisíveis*, correspondentes a cada <CR> digitado, os quais servem, internamente, para o **ed** descobrir onde começa uma nova linha.

```
- w<CR>
faltando nome do arquivo
- w folhas<CR>
571
-
```

Observe que, no primeiro **w**, o **ed** ainda não sabia qual o nome do arquivo. Assim, o procedimento correto é igual ao do segundo **w**, em que você digita o comando **w** seguido do nome do arquivo.

Para sair do editor e retornar ao SOX, você deve usar o comando **q** (do inglês: **q**uit: desista, pare). Ao fazê-lo, ressurge o caractere de prontidão **$**, indicando que o próximo comando deve ser do SOX.

```
- q<CR>
$
```

Se você tivesse digitado o comando **q** sem antes ter salvo pelo menos uma vez o arquivo que está sendo editado, o **ed** daria uma mensagem comunicando esse fato. Somente com a repetição do comando **q** é que você sairia do **ed**.

Examine agora a seção a seguir, que também cria o arquivo **folhas**.

```
$ ed -p - folhas <CR>
?folhas
- h<CR>
arquivo folhas ainda não foi criado
- a<CR>

        conteúdo

.<CR>
- q<CR>
?
- h<CR>
não considero as alterações efetuadas
- w<CR>
571
- q<CR>
$
```

Neste caso, não é necessário o nome do arquivo para o comando **w**, uma vez que o mesmo apareceu no comando **ed.**

Para ter certeza de que seu arquivo **folhas** está no diretório **poemas**, digite **ls:**

```
$ ls<CR>
folhas
```

Agora vamos entrar novamente no editor e verificar o conteúdo de **folhas.** O comando **p** imprime uma linha ou um conjunto de linhas do texto.

```
$ ed -p - folhas <CR>
571
- 1p<CR>
            Estes são em verdade os pensamentos de todos os
- 4p<CR>
            Se não são tão teus quanto meus, nada são ou
- 2,3p<CR>
                  homens em todas as épocas e países: não são
                                                [ originais meus.
- 7,20p<CR>
?
- h<CR>
linha fora do limite
-
```

Quando se digita o comando **ed,** como o arquivo **folhas** agora existe, ele é trazido da memória permanente para o buffer para ser verificado, editado etc. Temos aqui uma operação chamada *carregar* o arquivo, inversa àquela do comando **w** – logo, os caracteres transferidos aqui devem ser em mesma quantidade (571) que quando do último **w** (mesmos caracteres, transferidos no sentido inverso).

Para exibir até a última linha de um texto, você pode usar o caractere especial **$** que, no contexto do comando **p**, refere-se sempre à última linha de um texto.

```
- 10,$p<CR>
        Esta é a relva que cresce onde há terra e água,
        Este é o ar comum que banha o planeta.
-
```

Há vários outros comandos do **ed.** Os que acabamos de mostrar, de maneira informal, são aqueles estritamente necessários para que você possa criar, verificar e salvar o conteúdo de seus textos, bem como sair do editor. A tabela que segue contém um sumário dos comandos apresentados. No próximo capítulo veremos, entre outras coisas, como você pode modificar seus textos.

Tabela 7.1 Comandos Básicos do ed.

Comando	Função
a	adiciona linhas de texto (para terminar use . como primeiro e único caractere da linha).
h	explica o último erro ou exceção.
H	explica todos os erros subseqüentes, até um outro H.
p	imprime linhas.
P	imprime o caractere de prontidão *.
w	grava arquivo na memória permanente.
q	sai do editor.

7.4 OUTROS RECURSOS PARA PROCESSAMENTO DE TEXTOS

Um processador de textos deve conter, a rigor, recursos para *criação de textos,* recursos para *modificação de textos* (a *edição* de textos propriamente dita) e recursos para *formatação de textos*. Estes últimos compreendem, entre outras coisas:

- formatação automática de texto na tela;
- reformatação de parágrafos;
- transporte de palavras;
- alinhamento de margens.

Como você já deve estar pressentindo, o editor de textos **ed** do SOX não oferece nenhum recurso de formatação: ele é unicamente um editor. A ênfase que estamos dando a ele é devida a que, além de sua simplicidade, trata-se de um serviço básico do SOX, ou seja, ele faz parte do conjunto de serviços que acompanham o SOX quando você o adquire.

Você está percebendo, no entanto, que poderá precisar de processadores de texto muito mais completos que o **ed** (e que o **vi**, o editor visual), como de resto muitos outros serviços, de diferentes finalidades, muitos deles ainda não existindo, certamente.

Em termos de Brasil, uma parte da indústria de software já está consciente da demanda cada vez maior de bons produtos para ambiente UNIX (conseqüentemente para todos os seus compatíveis, como o SOX). Levando em conta o excelente ambiente de desenvolvimento de software proporcionado por tais sistemas e a grande possibilidade de o UNIX vir a se tornar um padrão internacional para sistemas operacionais, é de se prever que haverá oferta para todos os gostos, num futuro não muito distante.

CAPÍTULO 8

O EDITOR ed REVISITADO

No capítulo anterior você aprendeu a entrar no editor, criar um texto, verificar o conteúdo do texto, gravá-lo em um arquivo e sair do editor. Para tornar o ambiente mais confortável, você também aprendeu a obrigar o **ed** a emitir caracteres de prontidão para seus comandos, bem como a utilizar os comandos de ajuda do **ed** caso você cometa um erro ou em uma situação de exceção. Tudo isto são coisas bastante básicas, mas há outras coisas igualmente básicas que você precisa aprender e que veremos neste capítulo.

Que outras operações são necessárias? Imagine que, quando você criou um texto, pode ter cometido vários erros, não percebidos na ocasião da digitação (já vimos como a tecla <←> pode ajudar a corrigir alguns erros perceptíveis). Tentemos enumerar essas situações:

- Você digitou uma ou várias linhas com erros. Em alguns casos, é preferível *trocar* toda uma linha por uma outra, ou então um grupo qualquer de linhas por um outro grupo qualquer de linhas. Talvez, na maioria dos casos, o necessário seja *substituir* alguns caracteres.

- Você esqueceu de digitar uma ou várias linhas de seu texto. Neste caso, é necessário *inserir* ou *adicionar* ao texto as linhas esquecidas.

- Você pode ter digitado linhas estranhas a seu texto. Neste caso, você necessita simplesmente *remover* linhas.

- Você pode ter digitado linhas fora do lugar. Neste caso, você terá que *mover* linhas de um lugar para outro.

- Você pode conscientemente desejar duplicar linhas. Neste caso, você terá de *copiar* linhas.

Deste modo, você aprenderá a usar comandos para trocar linhas, substituir caracteres em linhas, inserir ou adicionar novas linhas, remover, mover e copiar linhas. Você aprenderá ainda a usar outros comandos muito úteis, principalmente quando trabalhar com vários arquivos de texto. Com mais este elenco de comandos você estará apto a bem usar o **ed**. Deve ficar claro, no entanto, que este capítulo não abrange todos os recursos do **ed**.

Intimamente relacionado com todas essas operações está o conceito de *endereçamento de linhas*: para trocar ou substituir linhas é necessário saber *quais* linhas endereçar para trocá-las ou substituí-las; para inserir ou adicionar linhas é necessário saber *onde* (antes ou após determinada linha endereçada) inseri-las ou adicioná-las; igualmente para remover linhas é necessário primeiro endereçá-las. É necessário ainda que o endereçamento de linhas possa ser rápido e eficiente pois seu texto pode ser longo (ou até muito longo). Assim, antes de passarmos à descrição dos comandos, vejamos a questão de como endereçar linhas.

8.1 ENDEREÇAMENTO DE LINHAS

Há dois tipos de endereçamento: endereçamento *por número* e endereçamento *por conteúdo*. No primeiro caso, as linhas são identificadas pelos seus números; no segundo caso, você deve fornecer alguns caracteres de uma linha para que o **ed** possa localizar uma linha que contenha esses caracteres. Como sempre, cada estratégia de endereçamento tem suas vantagens e desvantagens.

No endereçamento por número, o mesmo é feito *sem ambigüidade*, ou seja, dado um número de linha, só existe uma linha com este número ou não existe tal linha – no entanto, é necessário ter conhecimento prévio do número da linha.

No endereçamento por conteúdo, este pode ser *ambígüo*: dado um conteúdo, podem existir várias linhas com esse conteúdo (ou não existir qualquer linha). A vantagem é não ter que conhecer previamente o número da linha.

Para lidar principalmente com as desvantagens de cada estratégia de endereçamento de linhas, o **ed** oferece vários recursos que passaremos a ver.

ENDEREÇAMENTO POR NÚMERO DE LINHA

Já vimos no capítulo anterior (comando **p**) que a forma **[endereço1][,endereço2]** indica uma faixa de números de linha: se somente **endereço1** é especificado, trata-se de

uma única linha, cujo número de linha é **endereço1**. Se é especificado **endereço1,endereço2**, trata-se de um conjunto de linhas, desde **endereço1** até **endereço2.**

As linhas são sempre numeradas pelo **ed** com números inteiros, de um em um. Se você remover a linha 4 (logo aprenderá como fazê-lo), por exemplo, a linha seguinte existente, que era a 5, passará a ser a linha 4 e assim por diante, ou seja, todas as linhas após a linha removida terão seus números de linha diminuídos de 1 pelo **ed** para preservar a seqüência de inteiros, de um em um. Da mesma forma, se você adicionar duas linhas após a linha 1 (logo aprenderá como fazê-lo), essas novas linhas receberão os números 2 e 3, respectivamente, e todas as seguintes serão renumeradas de um em um, a partir do número quatro.

No capítulo anterior você utilizou os exemplos **4p** e **2,3p**, entre outros. No primeiro caso, tratou-se do comando para o **ed** exibir na tela o conteúdo da linha número 4 de **folhas**; no segundo caso, tratou-se da exibição das linhas 2 e 3 do mesmo arquivo. Como **folhas** tem somente 12 linhas fica fácil descobrir o número de cada linha. E se seu texto for longo (ou mesmo muito longo)?

A Última Linha do Texto

No capítulo anterior, você já começou a utilizar uma facilidade do **ed** para trabalhar com textos longos: trata-se do **metacaractere $**, que significa sempre que você está endereçando a **última linha do texto.** Isto é muito útil: imagine que você digitou o comando **ed folhas**, ou seja, o comando que põe o **ed** em execução e já o faz carregar o texto de **folhas** para o buffer, e que você quer que o **ed** exiba todo o texto na tela. Como advinhar que **folhas** tem 12 linhas se você porventura tiver se esquecido disto? Por aproximação sucessiva, como no exemplo abaixo?

```
$ ed -p-folhas <CR>
571
-H<CR>
-1,20p<CR>
linha fora do limite
-1,15p<CR>
linha fora do limite
-1,14p<CR>
linha fora do limite
-1,13p<CR>
linha fora do limite
-1,12p<CR>
                    conteúdo
-
```

Convenhamos que tal método não é razoável. Observe que ainda partimos do pressuposto de que você tinha uma noção do tamanho do seu texto. Em vez disso, você deve usar **1,$p<CR>** , tal como o fez no capítulo anterior. Como o **ed** interpreta este último comando? Alguma coisa como "devo exibir da linha 1 até a última linha de **folhas**".

A Linha Corrente do Texto

Um outro metacaractere muito útil é o . . Quando você o usa, está se referindo à **linha corrente** do texto. Qual é essa linha? é a *última* linha afetada pelo *último* comando usado. Por exemplo, seu último comando foi **1,12p<CR>** , então a linha corrente é a última do texto, ou seja, a linha em que o **ed** fica posicionado após este comando. Um lembrete importante: quando o **ed** carrega seu texto, a linha corrente é a última do texto.

```
-.,$p<CR>
          Este é o ar comum que banha o planeta.
-1p<CR>
          Estes são em verdade os pensamentos de todos os
-.,2p<CR>
          Estes são em verdade os pensamentos de todos os
            homens em todas as épocas e países: não são
-.,3<CR>
            homens em todas as épocas e países: não são
-                                          [originais meus,
```

Observe que a omissão de **p** no último comando é intencional: é isto mesmo, o **p** em verdade é opcional – quando o omitimos, o **ed** entende que *implicitamente* trata-se do comando **p**.

O Comando "vazio"

Uma operação muito comum é você endereçar uma linha de seu texto e, a partir desta linha, exibir uma seqüência de linhas, uma a uma. Para fazer isso, não é necessário usar o comando **p** repetidas vezes – em vez disso você utiliza unicamente a tecla <CR> . Fazendo isto você está digitando um comando "vazio", sem a identificação das linhas e sem especificar o comando: o **ed** então exibe a próxima linha, ou seja, entende que você implicitamente especificou a linha seguinte à linha corrente e o comando **p**.

```
-10p<CR>
-<CR>
        Esta é a relva que cresce onde há terra e água,
-<CR>
        Este é o ar comum que banha o planeta.
-<CR>
linha fora do limite
-
```

ENDEREÇAMENTO POR CONTEÚDO

Em vez de dar o número da linha (ou um conjunto de números de linhas) podemos endereçar linhas através de seu conteúdo. Por exemplo, no texto de **folhas**, podemos querer localizar a linha que contém a expressão **quanto**. Como fazer? Para indicar a busca pela expressão você pode indicar

/expressão/ ou **?expressão?**

No primeiro caso, o **ed** pesquisa o texto *da linha corrente para a frente* e pára na primeira linha que contiver a expressão (nova linha corrente). No segundo caso, o **ed** pesquisa o texto *da linha corrente para trás* e pára na primeira linha que contiver a expressão. Em ambos os casos, a busca só é encerrada ao alcançar a *linha origem* da mesma: isto introduz o conceito de *busca circular*. A Figura 8.1 ilustra este conceito.

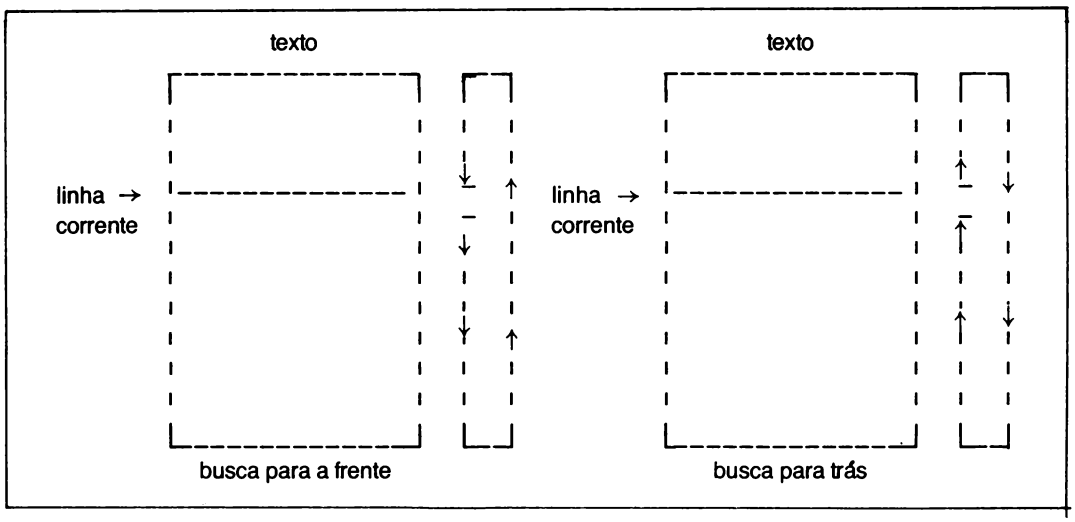


Figura 8.1 O Conceito de Busca Circular.

Geralmente usa-se a busca para trás se a linha a ser localizada estiver *antes* da linha corrente (você pode ter alguma certeza disto); se a linha a ser pesquisada estiver *depois* da linha corrente, geralmente o melhor será usar a busca para a frente.

Deve estar claro que, se não existir uma linha com a expressão de busca, o **ed** permanecerá na linha origem (em outras palavras, a linha origem da busca continua sendo a linha corrente).

```
-8p <CR>
            Se não são tão próximos quanto distantes, nada
-?quanto? <CR>
            Se não são tão teus quanto meus, nada são ou
-/AAA/ <CR>
expressão não localizada
- . p <CR>
            Se não são tão teus quanto meus, nada são ou
-/distantes/p <CR>
            Se não são tão próximos quanto distantes, nada
-
```

Observe que usamos o comando **p** (implícita ou explicitamente) endereçando as linhas por seus conteúdos. Nenhuma novidade, portanto.

Como resolver o problema da ambigüidade comentado anteriormente? Recordemos: quando você der ao **ed** uma expressão de busca poderão, na verdade, existir várias linhas com esta mesma expressão. O **ed** pára na primeira linha encontrada que contém a expressão, que pode não ser a que você está querendo. Neste caso, você pode ordenar ao **ed** que busque a próxima linha, usando

/ / ou **? ?**

e assim sucessivamente. Observe que, para o **ed**, a repetição não tem fim!!! Entendeu o por quê?

```
-/são/ <CR>
                                                [são.
-/ / <CR>
                Estes são em verdade os pensamentos de todos os
-? ? <CR>
                                                [são.
-
```

EXIBINDO LINHAS COM SEUS ENDEREÇOS

Existe um outro comando do **ed**, além de **p**, para mostrar o conteúdo de um texto – trata-se do comando **n** (do inglês **n**umber: numere). Qual a diferença então deste último para o comando **p**? É que o comando **n** exibe também os números das linhas. Os números são exibidos a partir da primeira coluna, enquanto o conteúdo da linha é exibido a partir da primeira parada de tabulação.

A exibição de linhas numeradas pode ser extremamente útil pois as mudanças em um texto podem implicar a remuneração das linhas – através do comando **n** você pode descobrir o novo número de uma linha.

Também quando você identifica uma linha por seu conteúdo, pode fazer o **ed** exibir o número dessa linha nas formas seguintes:

/expressão/= ou **?expressão?=**

```
-6,7n<CR>
6                Se não são o enigma e a solução do enigma, nada
7                                                          [ são,
-/enigma/=<CR>
6
-
```

EXPRESSÕES DE ENDEREÇAMENTO

O **ed** oferece também a possibilidade de *endereçamento relativo*, ou seja, podemos querer a terceira linha *antes* da linha X, ou, então, a décima linha *depois* da linha X, e assim por diante. Podemos expressar tais coisas através de *expressões de endereçamento*. A seguir, apresentamos alguns exemplos de expressões de endereçamento válidas:

$–2	endereça a antepenúltima linha do texto;
. +4	endereça a quarta linha depois da linha corrente;
/distante/+	endereça a linha depois de uma que contenha a expressão "distante". O endereço relativo 1 pode ser omitido da expressão de endereçamento;
/distante/–	endereça a linha antes de uma linha que contenha a expressão "distante";

1,/quantos/+ faixa de números de linhas, da linha 1 até a linha seguinte à primeira com a expressão "quantos";

?aaa?,/bbb/ faixa de números de linhas, da primeira linha com a expressão "aaa", antes da linha corrente, até a primeira linha com a expressão "bbb", depois da linha corrente.

8.2 FORMA GERAL DE UM COMANDO DO EDITOR ed

A forma geral de um comando do editor **ed** é a seguinte:

[identificação das linhas][comando][parâmetros para o comando]

onde:

identificação das linhas para a maioria dos comandos, se omitida, refere-se à linha corrente. Para o comando "vazio", refere-se à linha seguinte à linha corrente. Para certos comandos, deve simplesmente ser ignorada;

comando formado por uma letra. Se omitido, refere-se ao comando **p**, que imprime conteúdos de linhas;

parâmetros variam para cada comando. Para o comando **w** o parâmetro é o nome do arquivo a ser gravado. Já os comandos **p** e **n** não têm parâmetros.

8.3 MODIFICANDO LINHAS DO TEXTO

Já comentamos que pode haver dois tipos de modificação de linhas:

- troca total de algumas linhas por outras (não necessariamente pelo mesmo número de linhas);
- substituição de caracteres em linhas.

No primeiro caso, o comando apropriado é **c** (do inglês **c**hange: troque, mude); no segundo caso, o comando apropriado é **s** (do inglês **s**ubstitute: substitua).

O comando **c** troca linhas inteiras do texto. As linhas a serem trocadas são pri-

meiramente eliminadas e, a seguir, o **ed** fica esperando suas novas linhas até que você dê um . (tal como no comando **a**). Você está percebendo que o comando **c** vem em favor de sua comodidade: ele é, na verdade, a fusão de dois outros comandos, o de remoção e o de inserção, os quais serão vistos nas próximas seções.

```
-q <CR>
não considero as alterações efetuadas
-q <CR>
$ed -p -<CR>
-a<CR>
primeira linha <CR>
terceira linha <CR>
segunda linha <CR>
quinta linha <CR>
.<CR>
-2,3c<CR>
segunda linha <CR>
terceira linha <CR>
quarta linha <CR>
.<CR>
-1,$ <CR>
primeira linha
segunda linha
terceira linha
quarta linha
quinta linha
-
```

Vamos agora colocar um numeral nosso em cada linha. Podemos fazer assim: substituir a primeira letra de cada linha por um numeral seguido dessa primeira letra.

Vejamos os parâmetros para o comando **s:**

/[caracteres a substituir]/caracteres de substituição/[outros parâmetros]

- Quando uma linha é identificada por conteúdo, *caracteres a substituir* pode ser omitido: neste caso o **ed** *relembra* que os caracteres a substituir são os mesmos da expressão do conteúdo.

- Outros parâmetros podem ser: um comando para exibir a linha, um *indicador de pesquisa global* (veremos esse conceito logo mais) etc.

```
–/p/s/p/1– p/p<CR>
1– primeira linha
–/s/s//2– s/p<CR>
2– segunda linha
–/s/+1s/t/3– t/p<CR>
3– terceira linha
–/q/s//4– q/p<CR>
4– quarta linha
–//s//5– q/p<CR>
5– quinta linha
–1,$<CR>
1– primeira linha
2– segunda linha
3– terceira linha
4– quarta linha
5– quinta linha
–1,$s/–/–)/p<CR>
1–) primeira linha
2–) segunda linha
3–) terceira linha
4–) quarta linha
5–) quinta linha
–
```

Quando você manda substituir um grupo de caracteres de uma linha, caso existam vários desses grupos na mesma linha, normalmente o comando **s** só substitui o primeiro desses grupos. Assim, você teria que repetir sucessivamente o comando **s** até substituir todos os grupos de caracteres. Para evitar isso você pode usar o parâmetro indicador de pesquisa global **g** (do inglês **global**: global, total); neste caso, a repetição é automática.

```
–. s /i /l /p<CR>
5–) qulnta linha
– s /i /l /p<CR>
5–) qulnta llnha
– s /l /i /gp<CR>
5–) quinta linha
–q<CR>
não considero as alterações efetuadas
–q<CR>
$
```

8.4 INSERÇÃO DE LINHAS

Você pode inserir um número qualquer de novas linhas no seu texto *antes* ou *depois* da linha corrente. Para inserir antes da linha corrente o comando é **i** (do inglês **i**nsert: insira). Para inserir depois da linha corrente o comando é **a**, o mesmo do capítulo anterior. A operacionalização do comando **i** é idêntica à de **a**, ou seja, você deve introduzir suas linhas até uma linha que contenha unicamente o caractere **.**.

Aqui cabe uma observação bem interessante: você usou o comando **a** no capítulo anterior para criar um texto, ou seja, não existia um texto original onde você pudesse inserir aquelas linhas. Assim, a dúvida existencial é: *depois* de qual linha inseri meu texto *Folhas de relva*?! Pois bem, tal linha existe: ela é a *linha zero*!

A LINHA ZERO DO TEXTO

A linha zero é uma linha *imaginária*, no sentido de que não há texto associado a ela. Seu nome advém de que ela é anterior à linha 1. Ela pode ser endereçada normalmente, podendo ser usada em comandos que não impliquem operações dentro dela. Deste modo, na criação de *Folhas de relva* sua linha zero já existia: todas as linhas do texto foram incluídas *depois* desta linha pelo comando **a** (implicitamente **0 a**) – a linha zero era, portanto, a linha corrente. Após a execução do comando, a linha corrente é a última linha inserida.

```
$ed -p -<CR>
-H<CR>
-2i<CR>
a linha 2 não existe
-0a<CR>
linha 1<CR>
linha 2<CR>
.<CR>
-1a<CR>
primeira linha depois de linha 1<CR>
.<CR>
-1i<CR>
primeira linha antes de linha 1<CR>
.<CR>
-1,$n<CR>
1       primeira linha antes de linha 1
2       linha 1
3       primeira linha depois de linha 1
4       linha 2
-
```

8.5 REMOÇÃO DE LINHAS

O comando para remover linhas de seu texto é **d** (do inglês **d**elete: remova). **d** remove todas as linhas endereçadas no texto. Se você usar **dp** ou **dn, ed** imprime a próxima linha não-removida, que é a nova linha corrente. Se a linha removida for a última do texto, a nova linha corrente será a nova última linha do texto (penúltima, anteriormente). Isto é muito útil para que você saiba onde ficou depois das remoções feitas.

```
–/antes/dn<CR>
1           linha 1
–/depois/dn<CR>
2           linha 2
–
```

Vamos agora abrir um parênteses e reproduzir o exemplo que ilustrou o uso do comando **c**, agora em termos dos comandos **d** e **a.**

```
$ed  – p  –<CR>
–a<CR>
primeira linha<CR>
terceira linha<CR>
segunda linha<CR>
quinta linha<CR>
.<CR>
–/ter/d<CR>
–a<CR>
terceira linha<CR>
quarta linha<CR>
.<CR>
–1,$<CR>
primeira linha
segunda linha
terceira linha
quarta linha
quinta linha
–
```

8.6 MOVIMENTANDO LINHAS DO TEXTO

O **ed** permite que você movimente linhas para *depois* de uma linha especificada como parâmetro. A movimentação implica a remoção automática das linhas de suas posições originais e a inserção automática dessas linhas depois da linha especificada como parâmetro. O comando para essa operação é **m** (do inglês **m**ove: movimente). Voltando ao exemplo da seção 8.5, podemos fazer:

```
-1i<CR>
linha 3<CR>
.<CR>
-1,$<CR>
linha 3
linha 1
linha 2
-1m3<CR>
-1,$n<CR>
1       linha 1
2       linha 2
3       linha 3
-
```

Se você quiser movimentar suas linhas sem removê-las de suas posições originais poderá usar o comando **t** (do inglês **t**ranspose: transponha):

```
-0a<CR>
**********<CR>
.<CR>
-/*/t$<CR>
-1,$<CR>
**********
linha 1
linha 2
linha 3
**********
-
```

8.7 DESFAZENDO UMA OPERAÇÃO

Se você acabou de realizar uma operação (uma remoção, por exemplo) e percebeu que cometeu um engano, ainda assim o **ed** pode ajudá-lo a voltar à situação anterior ao engano: é através do comando **u** (do inglês **u**ndo: desfaça), que deve ser digitado como pura e simplesmente **u**. Por situação anterior entenda-se no seu sentido rigoroso: a linha corrente também é reposicionada.

É fundamental que você use o **u** imediatamente após o erro cometido, pois não é possível usar uma *cascata* **de u**'s para desfazer todo um histórico de operações. Você pode usar **u** no máximo duas vezes: a primeira vez tem o efeito que explicamos; usado a segunda vez, o comando **u** *desfaz* o primeiro comando **u!**

```
-$dp<CR>
linha 3
-u<CR>
-1,$<CR>
**********
linha 1
linha 2
linha 3
**********
-1dp<CR>
linha 1
-u<CR>
-u<CR>
-u<CR>
comando ignorado
-1,$<CR>
linha 1
linha 2
linha 3
**********
-
```

8.8 GRAVANDO EM OUTRO ARQUIVO

É comum você ter um arquivo, modificá-lo e gravar o novo texto em um outro arquivo, preservando assim o arquivo original.

O comando **w** permite também realizar esta operação: basta especificar o novo nome do arquivo como o novo parâmetro.

```
-w versão1<CR>
35
-q<CR>
$ed -p - versão1<CR>
35
-a<CR>
nova linha 1<CR>
nova linha 2<CR>
.<CR>
-w versão2<CR>
61
-
```

8.9 MUDANDO DE ARQUIVO SEM SAIR DO EDITOR

Muitas vezes você está dentro do editor, editando um texto, e quer passar a editar um outro texto. Felizmente você não tem que sair do editor e novamente entrar nele para fazer com que ele carregue o novo arquivo. Para evitar este caminho tortuoso, você pode usar o comando **e** (do inglês enter: entre), cuja forma geral é

e nome-do-arquivo

que lê o conteúdo de **nome-do-arquivo** para dentro do buffer, apagando tudo o que havia nele antes (o arquivo anterior).

```
-1,$<CR>
linha 1
linha 2
linha 3
**********
nova linha 1
nova linha 2
-e versão1<CR>
35
-1,$<CR>
linha 1
linha 2
linha 3
*********
-
```

8.10 LENDO UM ARQUIVO EXTERNO AO EDITOR

Vamos recordar a criação de **versão2**: para fazê-lo, você já tinha o conteúdo de **versão1** no seu buffer, acrescentou as novas linhas e gravou esse novo conteúdo com o **nome de versão2.** Vamos ver agora uma nova maneira de criar este último arquivo, a qual, dependendo da situação, poderá ser mais aconselhável. Desta vez, você entra no editor sem fazer qualquer alusão à **versão1** e cria as novas linhas. Feito isso você ordena ao **ed** que leia para seu buffer, no local apropriado, o conteúdo de **versão1.** Com isso você obtém o conteúdo de **versão2.**

O comando que lê um arquivo para o buffer sem destruir seu conteúdo anterior (ao contrário do comando **e**) é **r** (do inglês **r**ead: leia), cujo formato geral é

linha r nome–do–arquivo

onde *linha* é o número da linha *depois* da qual o conteúdo de **nome–do–arquivo** é colocado no buffer.

Vamos reproduzir o exercício da seção 8.8 para recriar o arquivo **versão2.**

```
-q<CR>
$ed -p - <CR>
-a<CR>
nova linha 1<CR>
nova linha 2<CR>
.<CR>
-0r versão1<CR>
-w versão2<CR>
61
-
```

8.11 RELEMBRANDO O NOME DE SEU ARQUIVO

Quando você está transferindo texto de um arquivo para outro (através de **r** e **w**) ou mudando constantemente de arquivo (através de **e**), é fácil ficar perdido, sem ter mais idéia de qual é seu texto corrente. Para relembrá-lo, o **ed** permite que você use o comando **f** (do inglês **f**ile: arquivo), cujo formato geral é

f [nome–do–arquivo]

Se você não digitar **nome–do–arquivo** (provavelmente porque já não tem a menor idéia de quem poderia ser) o **ed** lhe relembrará, exibindo o nome do arquivo que está sendo concorrentemente editado.

Se você digitar **nome–do–arquivo** (certamente porque tem uma razoável certeza de onde você está) o **ed** lhe confirma exibindo o nome do arquivo que está sendo editado.

```
–f folhas<CR>
arquivo folhas não está sendo editado
–f<CR>
versão2
–f versão2<CR>
versão2
–
```

8.12 EXECUTANDO UM COMANDO DO SOX DE DENTRO DO EDITOR

Muitas vezes você está dentro do editor e necessita executar um comando do SOX (por exemplo, listar o conteúdo de seu diretório, verificar quem está, no momento, usando o SOX etc.). Seria extremamente incômodo se você tivesse que sair do **ed** (conseqüentemente entrar no SOX), executar o comando do SOX desejado e depois voltar ao **ed.** Devemos lembrar também que, antes de sair do **ed,** você provavelmente teria que salvar o texto que estaria sendo editado. Para evitar tudo isso, o **ed** permite que você execute um comando do SOX de dentro dele. Para o **ed** reconhecer que se trata de um comando do SOX você deve colocar o caractere especial **!** antes do comando.

Por exemplo, para relembrar qual é seu diretório corrente:

```
–!pwd<CR>
/usr/voce/miscelanea/poemas
–
```

8.13 SUMÁRIO DOS COMANDOS DO ed APRESENTADOS

A Tabela 8.1 apresenta uma descrição resumida dos comandos do **ed** apresentados neste capítulo.

Tabela 8.1 Comandos Básicos do ed.

Comando	Função
c	troca linhas do texto
d	remove linhas do texto
e	introduz um novo texto
f	exibe ou confirma o texto corrente
i	insere novas linhas do texto
m	movimenta linhas do texto
r	lê um texto para dentro do buffer
s	substitui caracteres em linhas
t	copia linhas do texto
u	desfaz o último comando usado

Encerramos aqui nossa introdução ao editor de linhas **ed** do SOX. Você agora está mais do que habilitado a corrigir os possíveis erros cometidos por ocasião da digitação do poema *Folhas de relva*, bem como de quaisquer outros textos, certamente muito mais longos.

CAPÍTULO 9

OPERAÇÕES DO SOX COM TEXTOS

Vimos nos dois capítulos anteriores os comandos do editor de linhas **ed** do SOX para criar ou modificar um texto, ou seja, tarefas que chamamos de edição de textos. Há várias outras operações com textos que não são, a rigor, tarefas de edição de textos. Entre estas operações podemos citar as tarefas de *ordenar* um texto, *recuperar* informações de um texto, e outras. Tais operações, que não dizem respeito à edição propriamente dita de textos, serão vistas neste capítulo: para cada uma delas existe um comando específico do SOX.

Já que este capítulo trata de comandos do SOX, vamos inicialmente discutir as características gerais de seus comandos.

9.1 FORMATO GERAL DOS COMANDOS DO SOX

Os comandos do SOX são muitos e variados. No entanto, eles guardam entre si algumas características comuns, no seguinte sentido: um comando, de um modo geral, é composto de sua sigla, seguida de *argumentos*. Tais argumentos são classificados como: *opções, expressão, arquivos*. Assim, um comando pode vir seguido de uma ou mais opções, uma expressão e uma ou mais identificações de arquivo, nesta ordem.

Com estas idéias em mente, podemos dizer que um comando do SOX é um conjunto de *palavras* (uma palavra é uma cadeia de caracteres diferentes de espaço ou branco, ou, caso tenha que conter brancos, uma cadeia de caracteres delimitada por aspas) separadas (ou precedidas) de um ou vários espaços, terminando obrigatoriamente com o caractere especial <CR> . Na maioria dos comandos do SOX esse conjunto de palavras tem o seguinte significado:

comando [opções] [expressão] [arquivos] <CR>

em que a primeira palavra é sempre a sigla do comando, a segunda (se existir) representa as opções do comando, a próxima palavra (se existir) representa uma expressão e, finalmente, podem aparecer ainda uma ou mais palavras, cada uma identificando um arquivo. Observe que nem todos os comandos do SOX seguem este gabarito. Ele é, entretanto, bastante comum e se aplica, entre outros, a todos os comandos deste capítulo.

Como você pode ver, apenas a sigla do comando e o caractere <CR> aparecem em todos os comandos.

OPÇÕES

Em geral, as opções vêm precedidas do caractere – (sinal menos) e cada caractere seguinte dentro da palavra representa uma opção. No entanto, a regra para o primeiro caractere não é geral: pode vir inicialmente o caractere + (sinal mais), ou mesmo inexistir um caractere especial como o primeiro da palavra de opções (neste caso, o primeiro caractere da palavra já seria uma opção). E ainda: nem mesmo é rigorosamente verdadeiro afirmarmos a existência de uma única palavra de opções – pode, na verdade, existir mais de uma.

Você já deve estar se perguntando pelo significado desses caracteres especiais (+ e –) para opções. Pois bem, não procure tal significado porque ele não existe! É uma das falhas do original Unix: escrito por várias pessoas, que não tiveram a preocupação de padronizar suas preferências particulares, o resultado final virou esta "salada"! Neste caso, a compatibilização com o Unix prejudicou oSOX!

É importante salientar que uma opção só tem um significado quando associada a um comando. Assim, é possível que um caractere, por exemplo, **x**, represente uma opção para um comando **y**, e esse mesmo caractere represente uma outra opção para um outro comando **z**.

Exemplo: **–fr**, indicando duas opções, **f** e **r** (o significado, como vimos, depende do comando associado).

EXPRESSÃO

Para facilitar a exposição dos conceitos deste capítulo, inventaremos o conceito de *expressão*. A rigor, este conceito não existe como tal no SOX, mas nos será de valia ao longo deste capítulo. Uma expressão no contexto de um comando do SOX é qualquer

conjunto de caracteres (inclusive espaços ou brancos) que aparece entre aspas duplas" ou entre aspas simples '.

Exemplos de expressões:

"Livro sobre o SOX"
'Maria Bonita'
"1234"

Aqui caberia uma interrogação: se uma expressão pode conter qualquer caractere, poderá logicamente conter tanto o caractere " como o caractere '. Deve ficar claro que tal fato pode complicar a vida do SOX, como neste exemplo:

' f 's significa vários efes'

Neste caso, o SOX (na realidade, o Shell) não compreenderia que o segundo ' faria parte da expressão, enquanto o primeiro ' e o terceiro ' a delimitariam. Isto porque seu "raciocínio" seria mais ou menos o seguinte: "encontrei o primeiro ' (suponha que ele estaria lendo caractere por caractere, da esquerda para a direita), logo, tudo a partir de agora são caracteres de uma expressão, até o próximo '"; portanto, para o SOX, somente o primeiro espaço e o primeiro f constituiriam a expressão. O restante

s significa vários efes'

passaria a significar para ele quatro palavras (s, significa, vários e efes') e, ao encontrar o terceiro ', acharia que uma nova expressão estaria começando, mas não terminando (faltaria um quarto ').

Assim, a reação final do SOX seria uma mensagem de erro e você teria de reintroduzir sua linha de comando da maneira correta. De que forma?

A solução para qualquer problema do tipo mostrado anteriormente é usar convenientemente os caracteres " e '. Se a expressão contiver um ', ela deve ser delimitada somente por ", e vice-versa, se a expressão contiver ", delimite-a com '. Deste modo, o exemplo anterior correto é:

" f's significa vários efes"

ARQUIVOS

O nome de um arquivo no SOX é uma palavra de um a quatorze caracteres quaisquer. Na prática, no entanto, o mais comum é usar letras minúsculas (a-z) sem acen-

tuação, números (0-9) e alguns outros caracteres como o *grifo* (_), o *ponto* (.) e o *sinal menos* (-).

Exemplos de nomes de arquivo:

folhas
capitulo_1
capitulo-2
folhas.poema

9.2 EXIBINDO TEXTOS COM O COMANDO cat

Antes de mais nada, vamos satisfazer sua curiosidade: afinal acabamos de estudar o editor de linhas **ed** nos dois capítulos anteriores e lá aprendemos a operação de exibir na tela do terminal o conteúdo de um texto. É verdade. Agora, pense bem: suponha que seu texto esteja convenientemente editado, pronto, portanto. Você, de vez em quando, precisa verificar seu conteúdo. Para usar o **ed** você teria que, primeiro, entrar nele (isto é, executar o editor), segundo, verificar o conteúdo de seu texto (provavelmente tendo que escolher trechos do texto) e, finalmente, sair do **ed**. Convenhamos que é um processo cansativo, principalmente se você precisar fazer isso várias vezes, num curto espaço de tempo. Que tal deixarmos o **ed** exclusivamente para as tarefas de edição de textos e usarmos comandos próprios do SOX para tarefas não especificamente de edição?

Para a exibição de um texto, tal comando existe: é o comando **cat** (do inglês concatenate: concatene, reúna). Este comando concatena ou reúne vários arquivos de texto existentes (os arquivos *de entrada* para o comando) e exibe na tela de seu terminal (a *saída* do comando) o conteúdo dos vários arquivos, obedecendo à lista dos arquivos de entrada, ou seja, começando com o conteúdo do primeiro arquivo da lista, depois com o conteúdo do segundo arquivo, e assim por diante. Se sua lista de arquivos contiver um único arquivo, somente o conteúdo desse arquivo será exibido.

Comparando com o formato geral dos comandos do SOX, o comando **cat** é um dos mais simples: não tem opções relevantes para o nosso contexto, não tem expressão e uma lista de arquivos de entrada deve ser fornecida; quando discutirmos o conceito de *redirecionamento* de entrada/saída (Capítulo 11) veremos como o comando **cat** pode ser usado sem uma lista de arquivos de entrada.

Para exemplificar, crie, através do editor de linhas **ed**, dois arquivos, *gatos* e *cachorros*, e exiba seus conteúdos por meio do comando **cat**.

```
$ cat gatos<CR>
gato angorá
gato siamês
gato vira-lata
$
```

```
$ cat cachorros<CR>
pastor alemão
pastor belga
fila brasileiro
vira-lata
$
```

```
$ cat gatos cachorros<CR>
gato angorá
gato siamês
gato vira-lata
pastor alemão
pastor belga
fila brasileiro
vira-lata
$
```

É comum o conteúdo de seus textos ultrapassar o tamanho de sua tela. Nestes casos, você pode utilizar as teclas < CTRL–S > para parar a exibição, e as teclas < CTRL–Q > para continuar a exibição, sempre de acordo com suas conveniências.

9.3 ORDENANDO SEUS TEXTOS

Muitas informações precisam estar ordenadas, ou, dito com outras palavras, certas informações não farão sentido se não estiverem ordenadas. O exemplo mais sonoro do que estamos falando é o das listas telefônicas: o que seria da operação de encontrar o número do telefone de um assinante se a lista não estivesse apropriadamente ordenada?

Deste modo, no seu dia a dia, você terá que ordenar vários de seus textos. Os

exemplos são diversos: sua lista telefônica particular, uma lista de projetos por orçamento, eventos por data etc.

COMANDO PARA ORDENAÇÃO: sort

Para tornar possível a operação de ordenar um texto, você dispõe do comando **sort** (do inglês **sort**: ordene, classifique) do SOX.

Você pode ordenar seus textos de duas formas: por ordem alfabética e por ordem numérica. Qualquer que seja a ordem escolhida, você terá que tomar antes algumas decisões:

- qual a *estratégia* da ordenação (por um *campo* da linha, por vários campos, pela linha inteira)?
- que *opções* de ordenação usar? Entre outras: a ordem será *ascendente* ou *descendente*? O que fazer com linhas duplicadas?

Em resumo, suas decisões terão que ser de dois tipos: a primeira, diz respeito à escolha dos campos de ordenação, enquanto a segunda trata dos critérios que deverão ser observados pelo comando **sort** por ocasião da ordenação.

Vamos então mostrar como especificar seus campos de ordenação e o significado das opções do comando **sort** (não de todas, mas somente das mais essenciais).

Começaremos com exemplos simples e paulatinamente introduziremos exemplos que usem os vários recursos do comando **sort.**

Comecemos com uma pequena lista de telefones que você mesmo criou utilizando o editor de linhas **ed.** Quer exibi-la?

```
$ cat lista <CR>
Alceu Aboim                              - 3311100
Djenane Regina Machado                   -2211341
Frederico Sidnei César                   -4010081
João José da Silva Nicolletti            -2222014
Jussara Marques Viana                    -3211384
Maria Alzira Dornelles Vargas            -1681012
Maria Creuza Dantas                      -3221514
Orlando Abid Jateme                      -4011984
Zacarias Rolim de Moura                  -4011305
Maria Clara Machado                      -3221515
$
```

Observe que você colocou Maria Clara Machado fora de seu devido lugar na lista. É claro que você poderia fazer as modificações necessárias através do **ed** (por exemplo, usando o comando **t**, mover a última linha para seu devido lugar), mas não é isto que nos interessa neste momento. Interessa-nos agora usar os recursos do SOX para resolver problemas de ordenação de textos. Pensando bem, o **ed** não se prestaria para resolver a maioria desses problemas. Imagine um texto de mil linhas, desordenado (ou até parcialmente ordenado) e a tarefa cavalar que seria ordená-lo por meio do **ed** – isto não vem em detrimento do **ed**: acontece simplesmente que ele não foi criado para tal tipo de operação.

ORDENAÇÃO ALFABÉTICA

Aplique o comando **sort** ao arquivo de texto **lista:**

```
$ sort lista<CR>
Alceu Aboim                          – 3311100
Djenane Regina Machado               –2211341
Frederico Sidnei César               –4010081
João José da Silva Nicolletti        –2222014
Jussara Marques Viana                –3211384
Maria Alzira Dornelles Vargas        –1681012
Maria Clara Machado                  –3221515
Maria Creuza Dantas                  –3221514
Orlando Abid Jateme                  –4011984
Zacarias Rolim de Moura              –4011305
$
```

Entendamos bem o que se passou. Primeiro: você não especificou campos para a ordenação – nestes casos, a linha inteira é o campo de ordenação. Dito de outra forma, quando não especificamos campos de ordenação, o comando **sort** entende que ele deve reorganizar um texto de forma tal que cada caractere de uma linha esteja ordenado em relação aos caracteres da posição correspondente nas demais linhas. Segundo: você não usou qualquer opção – nestes casos o comando **sort** entende que se trata de uma ordenação alfabética, em ordem ascendente.

Sabemos, por intuição, que uma ordenação é feita comparando-se caracteres correspondentes em cada linha. Toda comparação introduz a noção de grandeza (valor quantitativo): assim, aceitamos que a letra **a** é menor que a letra **c** ou então que a letra **c** é maior que a letra **a**, por exemplo. Mas como comparar uma letra com um número ou com um sinal menos, por exemplo?

Para ordenar, o comando **sort** obedece a uma lógica que é a de atribuir um valor a cada caractere, seja ele letra, número ou qualquer outro caractere. Exemplificando para letras, números e os caracteres + (sinal mais), – (sinal menos) e o . (ponto ou ponto decimal), além do espaço, temos a seguinte ordem de valores:

espaço < **+** < **.** < **0** < . . . < **9** < **A** < . . . < **Z** < **–** < **a** < . . . < **z**

Exemplo de ordem alfabética ascendente:

123
+123
Maria
–12.3
maria

Resta ainda uma observação sobre a operação realizada: você não especificou como o comando **sort** deveria salvar o texto agora ordenado. Nestes casos, ele entende que você deseja que ele exiba o texto ordenado na sua tela (para interromper ou continuar a exibição do texto você deve proceder tal como para o comando **cat**). Deve estar claro que seu arquivo *lista* não sofreu qualquer alteração (comprove-o): em verdade, *lista* foi a *entrada* para o **sort**, enquanto seu conteúdo ordenado foi a *saída* para o **sort**. Adiante veremos também como criar uma saída que não seja a tela de seu terminal.

ORDENAÇÃO NUMÉRICA

A ordenação numérica seqüencia *números* (em ordem ascendente ou descendente) pelos seus valores aritméticos, levando em conta o sinal positivo ou negativo (a ausência de sinal é interpretada como sendo um número positivo). Vejamos um exemplo de ordenação numérica:

–123
–12.3
123 (ou + 123)

Observe que, quando se tratar somente de números, o fator de diferenciação entre uma ordenação numérica e uma ordenação alfabética é a existência ou não de números negativos. Se tomarmos somente números positivos, não importa se a ordenação for alfabética ou numérica, a ordem final será

12.3
123
+123

O próximo exemplo trata da ordenação de *lista* por números de telefone. Qualquer que seja a forma da ordenação, precisamos especificar que o campo de ordenação agora é número de telefone. Vamos então fazer uma pausa e falar sobre campos de ordenação de um modo geral, bem como o formato geral do comando **sort.**

CAMPO DE ORDENAÇÃO

Um campo para o comando **sort** é um trecho de uma linha que termina, normalmente, ao encontrar espaço ou <CR> (a pesquisa na linha sendo sempre feita da esquerda para a direita). Para terminar por um espaço é preciso que o caractere anterior seja diferente de espaço. Dizemos então que espaço é um *separador de campos*, não fazendo parte, portanto, de nenhum campo. O próximo campo começa imediatamente após o separador, mesmo que o próximo caractere seja também um espaço. Tomemos, como exemplo, a primeira linha de *lista*:

Para indicar ao comando **sort** que você quer ordenar seu texto *a partir* do campo 2 até o final da linha, você deve especificar **+1** como opção, que significa: *salte* o campo 1, portanto, o sinal + pode sempre ser traduzido assim neste contexto.

Para indicar ao comando **sort** que você quer ordenar somente pelo campo 2 você deve especificar o par **+1 -2** que significa: salte o campo 1 (como já sabemos) e ordene *até* o fim do campo 2, portanto, o sinal – pode sempre ser traduzido assim neste contexto.

Escolhido um campo, você pode ainda determinar que a ordenação não seja feita desde o início desse campo, mas a partir de um determinado caractere desse campo. Por exemplo, se você digitar **+1.3** estará dizendo ao comando **sort**: salte o primeiro campo e salte os três primeiros caracteres do campo 2.

Você pode também especificar coisas do tipo **-2.4**, que significa: ordene até o fim do campo 2 e até o quarto caractere do campo 3 (campo seguinte).

Outros exemplos:

+0 -1 ordenamento pelo campo 1

+0 –2	ordenamento pelos campos 1 e 2
+0 –1 +2 –3	ordenamento pelos campos 1 e 3
+0 –1 +2	ordenamento pelos campos 1, 3 e seguintes
+1.5 –4.1	ordenamento pelo campo 2, a partir de seu sexto caractere, até o primeiro caractere do campo 5

FORMA GERAL DO COMANDO sort

A forma geral do comando **sort** é:

sort [opções globais] [especificação dos campos] [–o arquivo_de_saída] [arquivos de entrada]

As *opções globais* são aquelas que valem para todos os campos simultaneamente. Elas compõem uma palavra de opções, conforme explicado na seção 9.1.

Podem também ser especificadas *opções locais* a cada campo (elas formam um subconjunto das opções globais) e devem aparecer junto com as especificações dos campos. Se, na especificação de um campo, não forem especificadas opções locais, valem as opções globais.

Algumas das opções serão discutidas ao longo dos vários exemplos que seguem. No final desta seção apresentaremos um sumário das opções vistas com seus respectivos significados.

Podemos verificar que poderão existir vários arquivos de entrada para o **sort**. Veremos algumas situações em que isso pode acontecer.

Se desejarmos uma saída num arquivo poderemos, como indicado, dar um nome de um arquivo de saída, precedido da opção **–o**. Se o nome do arquivo de saída for o mesmo do arquivo de entrada, o arquivo original será destruído, ficando em seu lugar e com o mesmo nome o novo texto ordenado. Se não aparecer a opção **–o** a saída será no terminal.

Ajustando-nos ao formato geral da maioria dos comandos do SOX, podemos dizer que a parte das opções compreende:

- opções globais;
- opções locais;

– opção de saída em arquivo.

Deste modo, já estamos lidando com um comando pouco ortodoxo! Não importa: a importância do comando **sort** anima-nos a apresentá-lo desde logo.

A OPÇÃO t

Tratemos agora do exemplo de ordenar *lista* por número de telefone. Já sabemos que devemos especificar o campo correspondente aos números de telefone. Como fazê-lo? Pelo que aprendemos, na primeira linha ele seria o campo 3, na segunda linha, o campo 4, na quarta linha, o campo 6 – paremos por aqui. E agora? Problema insolúvel? Não, felizmente. O que temos que fazer é informar ao comando **sort** que o separador de campos não é mais o padrão, espaço, mas o caractere – (sua colocação antes dos números foi, como você vê, intencional). Assim, usaremos a opção **t**, seguida de –, que indicará o novo separador de campos –. De uma maneira geral, você deve sempre especificar na forma **tx**, onde **x** representa qualquer separador de sua escolha. Esta opção é *sempre* global.

```
$ sort -t- +1 lista < CR >
Alceu Aboim                          3311100
Maria Alzira Dornelles Vargas      - 3311100
Djenane Regina Machado             - 1681012
João José da Silva Nicolletti      - 2211341
Jussara Marques Viana              - 2222014
Maria Creuza Dantas                - 3211384
Maria Clara Machado                - 3221514
Frederido Sidnei César             - 3221515
Zacarias Rolim de Moura            - 4010081
Orlando Abid Jateme                - 4011305
$                                  - 4011984
```

A OPÇÃO n

A opção **n** (do inglês **n**umeric: numérico) pode ser global ou local e indica ordenação numérica. Se for global, a ordenação será numérica por todos os campos de ordenação; se for local, significa que, naquele campo específico, a ordenação será numérica. Se existir somente um campo de ordenação tanto faz especificar **n** como global ou local.

```
$ sort -t- +1n lista < CR >
Maria Alzira Dornelles Vargas          - 1681012
Djenane Regina Machado                 - 2211341
João José da Silva Nicolletti          - 2222014
Jussara Marques Viana                  - 3211384
Maria Creuza Dantas                    - 3221514
Maria Clara Machado                    - 3221515
Alceu Aboim                              3311100
Frederico Sidnei César                 - 3311100
Zacarias Rolim de Moura                - 4010081
Orlando Abid Jateme                    - 4011305
$                                      - 4011984
```

Como você pode observar, a ordenação numérica despreza os espaços não-significativos e considera tão somente os números e seus valores.

A OPÇÃO b

A opção **b** (do inglês **b**lank: branco, espaço), quando aplicada à ordenação alfabética, faz o comando **sort** considerar que os espaços iniciais no campo de ordenação não são significativos para a ordenação. A opção **b** pode ser global ou local.

Reproduza a mesma ordenação numérica anterior usando a opção **b**.

```
$ sort -t- +1b lista <CR >
Maria Alzira Dornelles Vargas          -1681012
Djenane Regina Machado                 -2211341
João José da Silva Nicolletti          -2222014
Jussara Marques Viana                  -3211384
Maria Creuza Dantas                    -3221514
Maria Clara Machado                    -3221515
Alceu Aboim                            - 3311100
Frederico Sidnei César                 -4010081
Zacarias Rolim de Moura                -4011305
Orlando Abid Jateme                    -4011984
$
```

Relembremos mais uma vez: embora os resultados dos dois últimos exemplos tenham sido os mesmos, a maneira de proceder do comando **sort** foi diferente em cada caso.

Ordene agora por números de telefone, sem considerar os prefixos (três primeiros números).

```
$ sort -t- +1.3b lista <CR>
Frederico Sidnei César                    - 4010081
Maria Alzira Dornelles Vargas             - 1681012
Alceu Aboim                                 3311100
Zacarias Rolim de Moura                   - 3311100
Djenane Regina Machado                    - 4011305
Jussara Marques Viana                     - 2211341
Maria Creuza Dantas                       - 3211384
Maria Clara Machado                       - 3221514
Orlando Abid Jateme                       - 3221515
João José da Silva Nicolletti             - 4011984
$                                         - 2222014
```

A OPÇÃO r

A opção **r** (do inglês **r**everse: contrário), global ou local, indica para o comando **sort** que a ordenação é *descendente* (maior primeiro, menor por último).

Ordene **lista** por ordem descendente de números de telefone.

```
$ sort -t- +1br lista <CR>
Orlando Abid Jateme                       - 4011984
Zacarias Rolim de Moura                   - 4011305
Frederico Sidnei César                    - 4010001
Alceu Aboim                                 3311100
Maria Clara Machado                       - 3311100
Maria Creuza Dantas                       - 3221515
Jussara Marques Viana                     - 3221514
João José da Silva Nicolletti             - 3211384
Djenane Regina Machado                    - 2222014
Maria Alzira Dornelles Vargas             - 2211341
$                                         - 1681012
```

ORDENAÇÃO POR VÁRIOS CAMPOS

Vamos agora criar um outro arquivo para ilustrar a necessidade de termos vários campos para a ordenação. Criaremos tal arquivo diretamente pelo comando **sort**. De que

forma? Quando não especificamos arquivos de entrada (reveja a forma geral do comando **sort**: arquivos de entrada são opcionais), o comando **sort** inicialmente recebe as linhas de seu texto via terminal, até que você dê <CTRL–D> , que indicará para ele o fim de seu texto. A partir daí ele fará a ordenação do texto digitado.

Cada linha de nosso texto constará dos seguintes campos (o separador é o normal, espaço): *departamento, nome, divisão, área.* A ordenação será por *área, divisão, departamento* e *nome.*

```
$ sort +3 +2 -3 +0 -1 +1 -2<CR>
d3 nome1 div3 a1<CR>
d2 nome2 div4 a2<CR>
d1 nome3 div4 a2<CR>
d1 nome4 div2 a2<CR>
d2 nome5 div3 a1<CR>
d3 nome6 div3 a1<CR>
<CTRL-D>
d2 nome5 div3 a1
d3 nome1 div3 a1
d3 nome6 div3 a1
d1 nome4 div2 a2
d1 nome3 div4 a2
d2 nome2 div4 a2
$
```

Para entender como as opções foram escolhidas, observe que as linhas devem ser ordenadas primeiro pelo campo 4. Usamos então +3 para transpor 3 campos. Depois, a ordenação deve ser feita pelo terceiro campo (+2 –3 significa "transpõe" 2 campos e ordena até o terceiro). Continuando, a ordenação deve considerar o primeiro campo (+0 –1) e, finalmente, o segundo campo (+ 1 –2).

A lógica de ordenação quando se tem múltiplos campos de ordenação é a seguinte: os campos posteriores serão comparados apenas se todos os campos anteriores resultarem em igualdade na comparação.

A OPÇÃO m

Se você tiver vários arquivos de entrada e quiser intercalá-los num único arquivo de saída, poderá usar a opção **m** (do inglês **m**erge: intercale), sempre global. Para que isto funcione corretamente é necessário, no entanto, que os arquivos de entrada estejam previamente ordenados. Vejamos por quê.

Crie três arquivos de números ordenados e intercale-os. Por exemplo, o primeiro arquivo **arq1** com os números 1, 5, 7 e 8; o segundo arquivo **arq2** com os números 3 e 9; o terceiro arquivo **arq3** com os números 2, 4, 6, 7 e 9.

```
$ sort -m arq1 arq2 arq3 <CR>
1
2
3
4
5
6
7
7
8
9
9
$
```

Vamos agora desordenar **arq2**, que passará a ser 9 e 3, nesta ordem. O efeito da opção **m** será:

```
$ sort -m arq1 arq2 arq3 <CR>
1
2
4
5
6
7
7
8
9
9
3
$
```

Como agora a primeira linha de **arq2** é o número 9, ele só pode ser intercalado no final, com relação aos demais arquivos, porque o comando **sort** supõe que o arquivo **arq2** está ordenado. Quando a segunda linha (número 3) é lida do arquivo **arq2** já é tarde para intercalar no lugar correto.

A OPÇÃO u

Se quisermos, o comando **sort** pode remover linhas duplicadas da saída, *deixando somente uma delas*. Para isto, a opção é **u**, global.

Refaça o primeiro exemplo com a opção **m**, para evitar linhas duplicadas na saída da intercalação.

```
$ sort -mu arq1 arq2 arq3<CR>
1
2
3
4
5
6
7
8
9
$
```

GRAVANDO O TEXTO ORDENADO

Ordene, definitivamente, seu arquivo **lista**.

```
$ sort -o lista lista<CR>
$
```

Verifique de novo seu arquivo **lista**: agora, Maria Clara Machado deverá estar em seu devido lugar.

SUMÁRIO DAS OPÇÕES APRESENTADAS PARA O COMANDO SORT

Damos a seguir um quadro das opções para o comando **sort** discutidas nesta seção, com seus respectivos significados.

Tabela 9.1 Sumário de Opções para o Comando sort.

Opção	Significado
b	ignora brancos iniciais nas comparações de campos.
m	apenas intercala. Os arquivos de entrada já estão ordenados.
n	uma cadeia numérica inicial (podendo ter brancos não-significativos, sinais + e – e ponto decimal) é ordenada de acordo com seu valor aritmético.
r	reverte o sentido das comparações.
tx	estabelece o caractere x como separador de campos.
u	remove todas as linhas, exceto uma, em cada conjunto de linhas idênticas. Os caracteres fora dos campos de ordenação não estão incluídos nessa comparação.

Opções Globais: todas
Opções Locais: **b, n, r**

9.4 PESQUISANDO UM TEXTO COM O COMANDO grep

Continuemos pensando em sua lista telefônica particular. Uma operação constante é, dado o nome de uma pessoa, encontrar na lista seu telefone. Imagine que sua lista pode ser razoavelmente extensa (você certamente é uma pessoa bem relacionada). Neste caso, quer achar o número o mais rapidamente possível. Você pode, para isso, recorrer tanto ao **cat** quanto ao **ed**: nenhuma dessas soluções o deixará contente.

Chegou a hora de apresentar-lhe o comando **grep** (do inglês **g**lobal **r**egular **e**xpression **p**rinter: globalmente localize expressões regulares e imprima o resultado), ideal quando você quer localizar linhas de seu texto que contenham uma determinada expressão. É claro que o **ed** também tem recursos para isso, mas existe o inconveniente de você ter que entrar no **ed** e depois sair dele, além de outras deficiências que ficarão claras à medida que formos expondo as potencialidades do comando **grep.**

O comando **grèp** é então um comando que pesquisa um ou vários arquivos, procurando linhas que contenham uma determinada expressão. As linhas achadas são exibidas

no seu terminal, devidamente identificadas em caso de a pesquisa ser feita em vários arquivos.

O formato geral deste comando é o seguinte:

grep [opções] expressão [arquivos de entrada]

Aqui, **expressão** é a razão de ser do uso do comando **grep**, logo, ela é obrigatória.

Da mesma forma que para o **sort**, a ausência de arquivos de entrada faz com que o comando **grep** aceite como entrada linhas de texto digitado no terminal, até que você digite <CTRL D>. A pesquisa se fará sobre o texto recém-digitado. Note, entretanto, que isto aparentemente não faz sentido. Veremos, num capítulo posterior, como o **grep** pode ser usado sem nome de arquivo.

A saída, ou seja, as linhas que contêm a expressão pesquisada, é exibida no terminal. Da mesma forma que para o comando **cat**, você pode interromper e depois continuar a exibição das linhas recuperadas, digitando <CTRL S> e <CTRL Q>.

De tudo o que falamos, **expressão** merece uma dedicação especial, tal a flexibilidade que ela lhe oferece. Depois de explorarmos este assunto veremos algumas das opções do comando.

EXPRESSÕES

Podemos considerar dois tipos de expressão:

- cadeia de caracteres;
- expressão regular.

Já estamos familiarizados com o primeiro tipo: é tal e qual vimos com o **ed**.

Vejamos dois exemplos do comando **grep** usando cadeias de caracteres. Ambos tratam de encontrar em **lista** o número do telefone de Maria Creuza.

```
$ grep "Maria Creuza" lista<CR>
Maria Creuza Dantas                    -3221514
$
```

```
$ grep "Maria" lista<CR>
Maria Alzira Dornelles Vargas                    -1681012
Maria Clara Machado                              -3221515
Maria Creuza Dantas                              -3221514
$
```

Vejamos um outro exemplo, agora fazendo a pesquisa em mais de um arquivo.

```
$ grep "9" lista arq2<CR>
lista: Orlando Abid Jateme                       -4011984
arq2: 9
$
```

Imagine agora a seguinte situação: na sua lista você anota os nomes tanto com iniciais maiúsculas como com iniciais minúsculas. Assim, para encontrar as "marias" de sua lista seria preciso uma expressão mais ou menos como "Maria" ou "maria": tal forma, no entanto, não é permitida. Outra situação: e se quiséssemos todos os nomes que começam com a letra m (maiúscula ou minúscula)? Estas e muitas outras situações, importantes para atender às suas necessidades de recuperação de informações, não podem ser atendidas por uma simples cadeia de caracteres. Em vez disso, você pode utilizar *expressões regulares*, poderoso recurso para atender às situações apresentadas, bem como de outras.

EXPRESSÕES REGULARES

Uma *expressão regular* também é uma cadeia de caracteres com um importantíssimo detalhe: certos caracteres são abrangentes e têm significado especial – eles são chamados de *metacaracteres* porque são caracteres que representam *outros* caracteres.

Veremos somente os metacaracteres que julgamos os mais usáveis por um usuário não-programador. Além disso, veremos o conceito de *classe de caracteres* e o conceito de *quantificador* numa expressão.

O METACARACTERE PONTO

O *ponto* (.) é um metacaractere que representa *qualquer* caractere. Assim, por exemplo, ele tanto representa o *M* (maiúsculo) ou o *m* (minúsculo) ou qualquer outro

caractere, letra, número, sinal, parênteses etc. Agora você pode resolver o problema das "marias".

```
$ grep ".aria " lista<CR>
Maria Alzira Domelles Vargas                    –1681012
Maria Clara Machado                             –3221515
Maria Creuza Dantas                             –3221514
maria domitila de castro                        –524107
$
```

Observe que o espaço no final da expressão é importante. Experimente usar a expressão sem tal espaço.

```
$ grep ".aria" lista<CR>
Maria Alzira Domeles Vargas                     –1681012
Maria Clara Machado                             –3221515
Maria Creuza Dantas                             –3221514
Zacarias Rolim de Moura                         –4011305
maria domitila de castro                        –524107
$
```

Agora execute o seguinte exemplo: recupere todos os nomes cujo primeiro nome tenha cinco caracteres e comece por M (maiúsculo), e cujo segundo nome comece por C (maiúsculo).

```
$ grep "M.... C" lista<CR>
Maria Clara Machado                             –3221515
Maria Creuza Dantas                             –3221514
$
```

Você já deve estar se perguntando: "e se eu quiser usar o ponto como um ponto mesmo e não como um metacaractere?" Neste caso, ao ponto você deve preceder o caractere \ (contra-barra ou barra invertida): ele não faz parte da expressão – só indica que o próximo caractere da expressão é mesmo um caractere (inclusive \ ) e não um metacaractere. Exemplos:

"123 \.45"	A expressão é 123.45
"/\\ /\\ "	A expressão é / \ / \

Mas ainda não estamos satisfeitos só com o metacaractere ponto. Voltando às "marias", e se existisse uma pessoa cujo primeiro nome fosse Faria? Essa pessoa também seria recuperada (em choque com sua vontade de só querer as "marias"). Deste modo, há necessidade de definirmos uma classe de caracteres que possamos aceitar.

CLASSE DE CARACTERES E O METACARACTERE –

Uma classe de caracteres é um subcojunto do conjunto de todos os caracteres. Uma classe de caracteres é especificada entre colchetes: [e]. Dentro dos colchetes podemos usar o metacaractere –, que significa todos os caracteres compreendidos entre o caractere antes de – e o caractere depois de –, inclusive.

A classe de caracteres indica que, na posição relativa da expressão, pode aparecer qualquer caractere da classe.

Assim, para nos restringirmos às "marias" você deve fazer:

```
$ grep "[Mm]aria" lista<CR>
Maria Alzira Dornelles Vargas                    –1681012
Maria Clara Machado                              –3221515
Maria Creuza Dantas                              –3221514
maria domitila de castro                         –524107
$
```

Neste exemplo, a classe de caracteres que pode aparecer na primeira posição da expressão é composta de dois caracteres: **M** ou **m**.

Vejamos outros exemplos:

[abcdef]	Qualquer caractere entre **a** e **f**.
[a–f]	Mesmo significado anterior. O metacaractere deve ser usado sempre que possível pois vem a favor de sua comodidade (sem falar que fica mais elegante).
[a–dw–z]	Qualquer caractere entre **a** e **d** ou entre **w** e **z**.
"[Aa] [Bb] [Cc]"	Uma expressão composta de três caracteres em que o

primeiro pode ser **A** ou **a**, o segundo pode ser **B** ou **b** e o terceiro pode ser **C** ou **c**.

[S–b] Qualquer caractere entre **S** e **b**. Lembre-se de que entre estes dois caracteres estão o – (reveja a seção 9.3) e vários outros caracteres que não são letra e que podem não ser o que você deseja. Na verdade, na seção 9.3 apresentamos somente parte dos caracteres com seus valores segundo a codificação **ASCII** (American National Standard Code Information Interchange). Peça ao administrador de sua instalação uma *Tabela ASCII*.

Ainda não estamos satisfeitos. Vamos supor a seguinte situação: "desejamos todos os nomes que comecem por *J* e que tenham um outro nome ou um sobrenome que comece com *N* ou *P* ou *V*" (este exemplo é meramente didático, mas você pode se deparar com uma situação similar). Não dá para usar aqui pontos entre **J** e **N** (ou **P** ou **V**), pois é impossível saber quantos pontos colocar. Há, pois, a necessidade de se poder especificar um número variável de caracteres. Introduziremos então a noção de *quantificador* de caracteres.

QUANTIFICADORES DE CARACTERES

Um número, ou uma faixa de números, entre chaves { e } numa expressão significa o número de vezes que o caractere que vem antes deve ser repetido. Exemplos:

"A[34]{2}" É o mesmo que "A[34] [34]"

"A[34]{1,2}" É o mesmo que "A[34]" ou "A[34] [34]"

{2} e {1,2} são *quantificadores*. A forma geral de um quantificador é

{m,n}

onde **m** é o número mínimo de repetições e **n** é o número máximo de repetições. Algumas variações são permitidas:

{m, } Significa um mínimo de **m** repetições, diferentemente de **m** que significa exatamente **m** repetições

{ ,n} Significa de **uma** a **n** repetições

Alguns metacaracteres representam determinadas faixas de números (as de maior interesse prático, talvez):

* Equivale a {0, }

\+ Equivale a {1, }

? Equivale a {0,1}

Agora você tem condições de resolver o último problema proposto.

```
$ grep "J.*[NPV]" lista<CR>
João José da Silva Nicolletti                -2222014
Jussara Marques Viana                        -3211384
$
```

Entenda os dois exemplos abaixo:

```
$ grep "4011+" lista<CR>
Orlando Abid Jateme                          -4011984
Zacarias Rolim de Moura                      -4011305
$
```

```
$ grep "4011?" lista <CR>
Frederico Sidnei César                       -4010081
Orlando Abid Jateme                          -4011984
Zacarias Rolim de Moura                      -4011305
$
```

E fiquemos por aqui em relação a expressões. Estamos satisfeitos? Muito provavelmente a resposta é não (o ser humano é um eterno insatisfeito, não é mesmo?). Mas de uma coisa podemos estar certos: o que aprendemos sobre expressões regulares permitirá resolver, seguramente, boa parte desse tipo de recuperação de informações.

ALGUMAS OPÇÕES DO COMANDO grep

Veremos apenas duas opções, as quais julgamos de maior interesse. A primeira será a opção **v** (do inglês invert: inverta); a segunda será **n** (do inglês number: numere).

A OPÇÃO v

Se você usar esta opção, estará fazendo uma recuperação *inversa*, ou seja, as linhas exibidas na sua tela serão aquelas que *não contêm* a expressão de recuperação.

Por exemplo, para recuperar de **lista** todas as pessoas que têm números de telefone com prefixo diferente de "401" você poderia, mais comodamente, usar a opção **v**.

```
$ grep -v "-401" lista<CR>
Alceu Aboim                          - 3311000
Djenane Regina Machado               -2211341
João José da Silva Nicolletti        -2222014
Jussara Marques Viana                -3211384
Maria Alzira Dornelles Vargas        -1681012
Maria Clara Machado                  -3221515
Maria Creuza Dantas                  -3221514
$
```

A OPÇÃO n

Com esta opção você está solicitando ao **grep** que numere as linhas recuperadas.

```
$ grep -vn "-401" lista<CR>
1:Alceu Aboim                        - 331100
2:Djenane Regina Machado             -2211341
4:João José da Silva Nicolletti      -2222014
5:Jussara Marques Viana              -3211384
6:Maria Alzira Dornelles Vargas      -1681012
7:Maria Clara Machado                -3221515
8:Maria Creuza Dantas                -3221514
$
```

9.5 REUNINDO DADOS ESTATÍSTICOS DE SEUS TEXTOS

Pode ser desejável obter vários dados estatísticos de seus textos, como o número de linhas, o número de palavras e o número de caracteres (este último também fornecido pelo editor de linhas **ed**). Você pode obter todas essas informações com o comando **wc** (do inglês word counter, que traduziremos por seu real significado que é "conte linhas, palavras e caracteres de textos").

Vejamos a forma geral deste comando:

wc [–lwc] [arquivos de entrada]

A opção **l** (do inglês **l**ine: linha) exibe o número de linhas; a opção **w** (do inglês **w**ord: palavra) exibe o número de palavras; a opção **c** (do inglês **c**haracter: caractere) exibe o número de caracteres. São exibidos os valores correspondentes às opções indicadas, na ordem em que elas são especificadas. Se nenhuma opção é indicada, são exibidos o número de linhas, o número de palavras e o número de caracteres, nesta ordem.

Como está indicado, pode haver vários arquivos de entrada: quando isto acontece, o SOX exibe também um total geral, de todos os arquivos de entrada, sob o rótulo "total". Se nenhum arquivo de entrada é especificado, o comando espera pelo seu texto a ser digitado naquele momento e exibe as estatísticas do texto recém-digitado.

Acompanhe os vários exemplos abaixo.

```
$ cat arq2<CR>
3
9
$ wc arq2<CR>
                2               2               4 arq2
$ wc –l arq2<CR>
               2 arq2
$ wc –w arq2<CR>
               2 arq2
$ wc –c arq2<CR>
               4 arq2
$ wc –cw arq2<CR>
                4              2 arq2

$
```

```
$ wc arq1 arq2 arq3<CR>
                4               4               8 arq1
                2               2               4 arq2
                5               5              10 arq3
               11              11              22 total
$
```

```
$ wc<CR>
duas palavras<CR>
  CTRL-D
                1           2           14
$
```

São estes os comandos do SOX de operações com textos que escolhemos para lhe ajudar nas tarefas de tratamento de suas informações. Operações mais refinadas requerem programas de serviço especializados: é nosso desejo que você os encontre, cada vez com mais facilidade, no mercado de produtos de software para o SOX.

CAPÍTULO **10**

MANIPULANDO ARQUIVOS E DIRETÓRIOS

Um dos pontos fortes do SOX é a maneira pela qual o sistema organiza e mantém as informações para os seus usuários. Dizendo a mesma coisa em linguagem técnica: um dos pontos fortes do SOX é o seu Sistema de Arquivos.

No Capítulo 4, introduzimos os conceitos relativos ao sistema de arquivos do SOX, suas características e as vantagens que traz para a organização de seu trabalho e atividades da sua empresa. Para não repetirmos esta introdução, sugerimos que você releia a seção que trata do assunto (seção 4).

Este capítulo apresenta os comandos do SOX de uso mais freqüente, para você criar, se informar, alterar e restringir acesso ao seu sistema de arquivos. Ao todo, são examinados 10 utilitários ou comandos. Cada comando será examinado com o enfoque limitado à sua utilidade mais usual, do ponto de vista de um usuário *normal* do SOX. Assim, não abordaremos funcionalidades sofisticadas. Estas últimas funcionalidades são conseguidas com ligeiras alterações no formato do comando, através do emprego de outras opções e/ou argumentos – distintos daqueles a serem apresentados.

O estudo da funcionalidade completa ou estendida de cada comando é, sem dúvida, instrutivo e imprescindível para os usuários responsáveis pela operação do SOX e pelo desenvolvimento de software na sua empresa. Nossa experiência tem mostrado, contudo, que o feijão-com-arroz a ser-lhe oferecido atende, e muito bem, ao trato diário com o sistema da maioria dos usuários. Informações mais aprofundadas estão disponíveis nos manuais do SOX.

As explicações dadas devem servir pelo menos para tranqüilizá-lo. Apesar do capítulo abordar 10 comandos do SOX, eles serão vistos e absorvidos muito rapidamente. Em questões de alguns minutos você os estará usando desenvoltamente.

Os comandos serão apresentados a partir da seção 10.2, com uma seção sendo dedicada para cada um deles. A seção 10.1 traz algumas observações sobre o sistema de arquivos associado à sua conta SOX.

10.1 O SEU SISTEMA DE ARQUIVOS

O SOX organiza todos os arquivos e diretórios no sistema como uma árvore invertida, com a *raiz* em cima. Cria-se, desta forma, uma hierarquia de diretórios. Cada diretório na hierarquia pode conter qualquer número de outros diretórios e/ou arquivos. A um diretório contido em um outro chama-se *subdiretório, diretório-filho* ou *descendente*. Um diretório que contém outro é freqüentemente chamado de *diretório-pai* ou *ascendente*. Usaremos estes termos livremente por todo o capítulo.

O início dos sistemas de arquivos SOX fica no diretório-raiz, o qual é representado por /. A Figura 10.1 apresenta um sistema de arquivos SOX onde um retângulo representa um diretório, e um círculo, um arquivo. Observe que o sistema de arquivos da Figura 4.6 é reproduzido na Figura 10.1 como uma subárvore, ou seja, é parte componente do sistema na Figura 10.1.

Agora seguem alguns comentários sobre a Figura 10.1. Primeiro, deve-se notar que a figura ilustra *todo* o sistema de arquivos numa máquina SOX. Lá se alojam todos os diretórios e arquivos dos usuários e do próprio sistema. Abaixo do diretório-raiz, você vê diretórios com nomes **lib, bin, dev, etc** e **usr.** Apenas este último nos interessa aqui: os demais (podem existir outros no seu sistema) contêm informações relativas à operação e uso do sistema SOX. Tais diretórios, na verdade, não estão vazios como ilustrado. Eles são de interesse de programadores e do pessoal responsável pela operação, manutenção e administração do SOX.

No diretório **/usr** (este é o nome *completo* – partindo da raiz) é onde se amarram as subárvores – uma para o sistema de arquivos de cada usuário do SOX. Dependendo da preferência do administrador e do número de usuários SOX, podem existir um ou mais diretórios **/usr**, p. ex., **/usr1, /usr2, ...** e assim por diante. Por simplicidade, mas sem incorrer em particularidades, vamos supor apenas um destes diretórios e tratá-lo pelo nome genérico **/usr.**

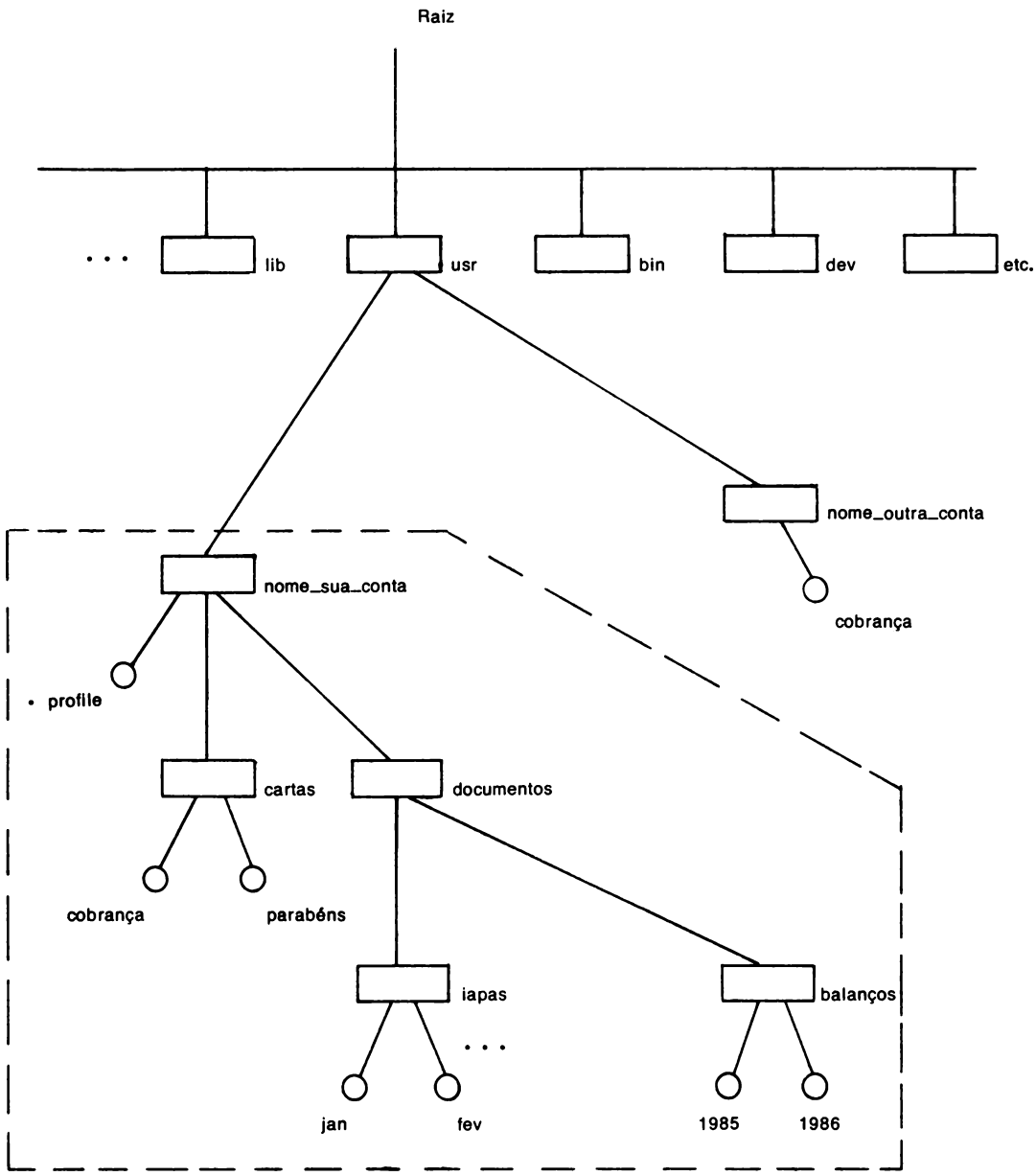


Figura 10.1 Ilustração de um Sistema de Arquivos SOX.

A subárvore associada à sua conta é o *seu* sistema de arquivos. Lá ficam seus diretórios e arquivos. Em geral, você não recebe o seu sistema de arquivos na sua forma definitiva. Você é quem o constrói e o modifica, pedaço por pedaço, à medida que usa o SOX e realiza suas atividades. Mas recebe algo com que começar.

Quando o administrador abriu sua conta no SOX, ele criou um diretório para seu uso, cujo nome é **nome_sua_conta** (ou **silva**, se quisermos ser coerentes com o Capítulo 6). É este diretório que você recebe para começar seu sistema de arquivos. O nome *completo* deste diretório é **/usr/nome_sua_conta** (vide Figura 10.1).

A questão de *nome* de arquivos ou de diretório será melhor discutida adiante. Uma das coisas que o SOX faz para você no procedimento de **login** (conexão) é colocá-lo ao nível do diretório **/usr/nome_sua_conta**. É por isto que este diretório é freqüentemente chamado de **diretório-casa**. Você pode alterar este procedimento, se desejar (fale com o administrador).

Inicialmente, seu **diretório-casa** está vazio, exceto, talvez, pela presença do arquivo **.profile**, cortesia do seu administrador. Este arquivo é utilizado para definir o ambiente de trabalho em sua conta e pode atender às necessidades específicas de suas tarefas. Por exemplo, se você faz muito processamento de textos, o **.profile** pode já colocar o processador à sua disposição. Converse com o administrador sobre o seu **.profile**. Evitamos maiores discussões aqui para não abordarmos conceitos do SOX que fogem ao objeto introdutório deste livro. Voltaremos pois, ao assunto principal: seu **diretório-casa** se encontra vazio inicialmente. À medida que você usar sua conta, irá criando arquivos e diretórios que paulatinamente aumentarão a sua subárvore dentro do sistema de arquivos.

Neste capítulo, você verá como aumentar e reduzir seu sistema de arquivos. Para tanto, você precisa de mais um conceito: *Nome de percurso*.

NOME DE PERCURSO

Observe a Figura 10.1 mais uma vez. Você e outro usuário criam arquivos com o mesmo nome (no caso, os arquivos **cobrança**). Os nomes completos desses arquivos, contudo, são diferentes e por isto não há confusão por parte do SOX ou dos usuários. O nome completo (e exclusivo) de um diretório ou arquivo no SOX é especificado pelo Nome de Percurso, começando na raiz e descendo pela árvore até atingir o diretório ou arquivo. Assim, o Nome de Percurso do seu arquivo **cobrança** é:

/usr/nome_sua_conta/cartas/cobrança

e do outro usuário:

/usr/nome_outra_conta/cobrança

Pratique um pouco, dando o nome de percurso do seu arquivo **1986.** Os nomes de percurso acima são completos, pois eles sempre incluem a raiz. Um nome de percurso completo sempre começa com /.

Um nome de percurso relativo pode ser construído se você inicia num nível mais baixo no sistema de arquivos. Um nome de percurso relativo nunca começa com /. Por exemplo, se você se encontra no seu **diretório-casa**, o nome de percurso para **cobrança** será:

cartas/cobrança

e para arquivos **1986:**

documentos/balanços/1986

Observe que nomes de percurso relativos são tomados *relativamente* ao diretório onde você se encontra.

Daqui em diante, quando mencionarmos o nome de um arquivo ou diretório no SOX você poderá utilizar o nome de percurso completo ou relativo. Desde que o percurso esteja correto, tanto faz para o SOX.

COMANDOS PARA MANIPULAÇÃO DO SISTEMA DE ARQUIVOS

Nas suas atividades com o SOX, você freqüentemente manipulará e terá que se deslocar pelos seus vários arquivos e diretórios, organizando-os segundo suas necessidades. O SOX tem recursos próprios – chamados de utilitários ou comandos – para assisti-lo nestas tarefas. Examinaremos nas próximas seções os seguintes comandos para manipulação do seu sistema de arquivos:

mkdir
ls
cd
pwd
cat
cp
rm
rmdir
mv
chmod

Como já foi dito, dedicaremos uma seção a cada um deles. Não podemos prosseguir sem sua ajuda.

Precisamos que você crie alguns arquivos e os organize em diretórios para que possamos exemplificar o uso dos comandos a serem examinados. Conecte-se ao SOX. Pronto, você está no seu **diretório-casa**. Como exercício dos comandos que estudaremos, vamos reproduzir, em sua conta, o sistema de arquivos mostrado na Figura 10.1 dentro da região, tracejada, ou seja, *sua* subárvore. Primeiro criaremos todos os diretórios e, finalmente, os arquivos nos diretórios aos quais pertencem. Antes, contudo, valem duas observações.

A primeira observação refere-se ao número de caracteres que o SOX permite nos nomes de arquivos e diretórios. O SOX permite até 14 caracteres. Assim, **nome_sua_conta** aparecerá como tal, pois tem exatamente 14 caracteres. Já **nome_outra_conta** aparecerá como **nome_outra_con**, já que, no máximo, 14 caracteres são mostrados. Você pode usar qualquer caractere nos nomes de seus arquivos e diretórios, exceto a /, pois este é o separador de percurso. Evite o uso de caracteres como *, ?, [e] (os metacaracteres), pois eles podem complicar a sua vida, já que têm significado especial para o SOX. Também não é aconselhável o uso de espaços em branco (principalmente no início e no fim de um nome) já que não são visíveis na tela e podem confundi-lo quando você tentar acessar os arquivos ou diretórios correspondentes.

A segunda observação é sobre a representação de um sistema de arquivos. A Figura 10.1 representa o sistema de arquivos (que reproduzimos neste capítulo) de forma lógica. O SOX não lhe apresenta a estrutura (árvore hierarquizada) do seu sistema de arquivos de forma pictorial (como na Figura 10.1). Em vez disto, você obtém na tela de seu terminal uma listagem do conteúdo (arquivos e diretórios descendentes) do diretório que você especifica ou, alternativamente, no qual você se encontra. O diretório no qual você se encontra é chamado de **diretório corrente** ou **diretório de trabalho.** Inicialmente, o seu

diretório de trabalho é o seu **diretório-casa.** O SOX dispõe de comandos para informar o nome e para mudar de diretório de trabalho; os comandos são **pwd** e **cd**, respectivamente. Esses comados serão descritos neste capítulo.

Concluindo: você poderá formar a estrutura hierarquizada do seu sistema de arquivos na sua mente, a partir das informações que o SOX lhe fornece. Via de regra, esta não é a forma de apresentação na tela de seu terminal. Vamos aos comandos.

10.2 mkdir - CRIA UM OU MAIS DIRETÓRIOS

Como dissemos, para reproduzir o seu sistema de arquivos (delimitado pela linha tracejada na Figura 10.1), vamos primeiro criar todos os diretórios. O comando SOX que cria um ou mais diretórios é **mkdir.**

Para criar um diretório, o comando **mkdir** deve ser acompanhado do nome do diretório a ser criado e concluído com um <CR>. O formato é mostrado abaixo:

mkdir nome-diretório<CR>

O comando deve ser fornecido após o sinal de prontidão do Shell. Assim, para criar o diretório **cartas** (vide Figura 10.1), como descendente (ou filho) do seu diretório atual, faça:

```
$ mkdir cartas<CR >
$ –
```

Após a criação, o Shell reapresenta o sinal de prontidão e o cursor (vide acima), à espera do seu próximo comando.

Pratique o uso do comando **mkdir**, criando o diretório **documentos.**

Observe que você terá criado os diretórios **cartas** e **documentos** no seu **diretório-casa**, pois este será o seu diretório de trabalho após concluir o procedimento de conexão (login) ao SOX. Neste ponto, você ainda não tem como certificar-se da criação deste diretório. Não se afobe. Você poderá fazer isto quando estudar o comando **ls** na seção seguinte.

O comando **mkdir** pode ser empregado para criar mais de um diretório de uma só vez. Neste caso, o formato seria (para o caso de três diretórios a serem criados):

mkdir nome-diretório1 nome-diretório2 nome-diretório3<CR>

Experimente criar os diretórios **cartas** e **documentos** de uma vez só. Dê o comando:

```
$ mkdir cartas documentos <CR>
```

Como estes dois diretórios já foram criados, o SOX emitirá uma mensagem informando-lhe que os diretórios já existem e em seguida o Shell apresentará o sinal de prontidão (observe a tela de seu terminal). Você não pode criar algo que já existe. Não se decepcione. Você terá a chance de usar **mkdir** para criar mais de um diretório quando se deslocar para o diretório **documentos** e lá criar os subdiretórios **iapas** e **balanços.** Espere só um pouco: vamos primeiro examinar o comando **ls.**

10.3 ls - LISTA DE CONTEÚDO DE DIRETÓRIO

O comando **ls** lista o conteúdo de um diretório em ordem alfabética.

Era este comando que você precisava para verificar o resultado de suas ações criativas no seu **diretório-casa** na seção anterior. Quando aparecer o sinal de prontidão do Shell($), dê o comando seguido de <CR> e observe o resultado.

```
$ ls<CR>
```

Como você não indicou ao comando **ls** qual era o diretório a listar, ele assumiu que você queria ver o conteúdo do diretório corrente (seu diretório-casa). O SOX apresentará na

tela todos os arquivos e diretórios que você criou no diretório corrente. No presente caso devem constar os diretórios **cartas** e **documentos**. Verifique se estes diretórios foram criados, caso contrário reutilize **mkdir**. Se você criou algum diretório com nome errado ou simplesmente criou um diretório descendente que se tornou indesejável, poderá eliminá-los com o comando **rmdir** a ser visto adiante.

O comando **ls** pode fornecer informações mais detalhadas sobre o conteúdo do seu diretório de trabalho, se você utilizá-lo com a opção **-l**, como mostrado a seguir:

```
$ ls -l<CR>
```

O SOX agora lista o conteúdo do diretório corrente no formato *longo* (opção **-l**). O formato longo apresenta o número total de arquivos/diretórios contidos no diretório corrente. Em seguida, em ordem alfabética, para cada um deles, aparece a indicação se é um diretório ou um arquivo normal (sinal **d** ou – no início da linha), as permissões de leitura, gravação e execução (**rwx**) do dono, grupo e outros usuários, o número de ligações, identificação do dono, identificação do grupo, tamanho (em bytes), data e hora da última alteração e o nome do próprio arquivo ou subdiretório. A Figura 10.2 ilustra as informações de uma linha na listagem de **ls-l.**

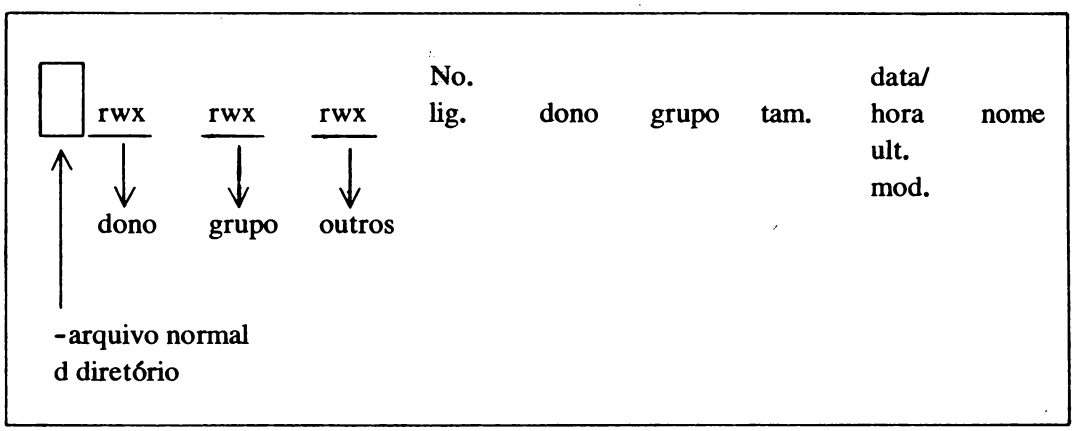


Figura 10.2 Informações numa linha de saída de "ls - l".

Número de ligações informa, no caso de um arquivo, quantos nomes ele tem no sistema de arquivos SOX; no caso de um diretório, é igual a 2 mais o número de diretórios-filhos. Por exemplo, um diretório com 3 descendentes tem *Número de ligações* igual a 5(2 + 3). Para nossos propósitos, bastam essas explicações no que toca a *Número de ligações*; explicações mais pormenorizadas exigem conceitos avançados do SOX.

Tamanho informa o número de caracteres do conteúdo. No caso de um arquivo, este número é simplesmente a soma de todos os caracteres gravados (letras, brancos, sinais de pontuação, acentos, caracteres de controle, retorno do carro etc.). Para um diretório, é um múltiplo de 16 com um valor mínimo de 32. um diretório recém-criado, vazio, tem tamanho 32. Mais uma vez, detalhes adicionais são dispensáveis para o trato cotidiano com o SOX e para não complicar a apresentação, os omitimos aqui.

As demais informações na linha da Figura 10.2 são autoexplicativas. As informações sobre os modos de permissão, no início da linha, serão abordadas com o comando **chmod** na seção 10.11.

Pronto. Você pode examinar na tela de seu terminal o resultado da execução do comando **ls –1**. Supondo que seu nome_de_conta SOX fosse *silva*, que você participasse do grupo envolvido no projeto *alfa* e que houvesse restrições de acesso de leitura, gravação e execução para outros usuários (veremos como fazer isto depois), o resultado seria parecido com:

```
total2
drwxrwx --- 2 silva alfa 32 Mai 14 10:30 cartas
drwxrwx --- 2 silva alfa 32 Mai 14 10:30 documentos
$ _
```

Perceba os seguintes pontos. *Um*: o arquivo **.profile** não aparece na listagem. Nenhuma entrada com nome iniciado com ponto (.) é listada com a opção **–1** (há outra opção para isto). *Dois*: o resultado que você vê na tela pode diferir do que mostramos (você pode ter outros arquivos no seu **diretório-casa**, por exemplo). *Três*: o número de ligações igual a 2 para ambos os diretórios **cartas** e **documentos** implica que eles não têm subdiretórios como filhos. Esta informação deve mudar, no caso de **documentos**, quando criarmos seus filhos (**iapas** e **balanços** – vide Figura 10.1). *Quarto* e último ponto: o tamanho igual a 32 bytes indica um diretório recém-criado e vazio. Verificaremos isto dentro em breve.

10.4 cd - MUDA DIRETÓRIO CORRENTE

O comando **cd nome-diretório** muda o seu diretório de trabalho (ou diretório corrente) para o diretório identificado em **nome-diretório**. Por exemplo (vide Figura 10.1)

```
$ cd /usr/nome_sua_conta/documentos<CR>
```

fará com que você se *desloque* no seu sistema de arquivos para o diretório *documentos*.

O uso do comando **ls –1** agora mostrará um diretório vazio. Experimente. Crie, neste ponto, os diretórios **iapas** e **balanços**, usando, mais uma vez, o comando **mkdir** seguindo as instruções fornecidas anteriormente. Estes diretórios são, na verdade, subdiretórios do seu diretório atual de trabalho, ou seja, o diretório **documentos.**

Uma maneira, portanto, de você criar subdiretórios é mover-se para o diretório onde ficarão os subdiretórios (com o comando **cd**), e lá criar os subdiretórios que desejar (com o comando **mkdir**).

Verifique se os subdiretórios foram realmente criados com o comando **ls –1.**

Para você se deslocar para um dos subdiretórios criados, por exemplo., iapas, digite simplesmente:

```
$ cd iapas<CR>
$
```

Nada diferente do que você já aprendeu.

Para você retornar ao diretório **pai** (ou seja, o diretório localizado um nível acima na hierarquia do sistema de arquivos), use o comando

```
$ cd ..<CR>
$ –
```

O diretório **pai** é representado por .. (fala-se "ponto-ponto"). Dê o comando **ls –1** após o comando acima para se certificar de que você retornou para o diretório **documentos** (o qual é o pai do diretório **iapas** e **balanços**; vide Figura 10.1).

A partir de qualquer diretório, você pode sempre retornar para o seu **diretório-casa** com o comando:

```
$ cd<CR>
$_
```

Observação: Lembre-se de que o comando **ls –1** serve para lhe informar do conteúdo do diretório corrente. O comando adequado para informar em qual diretório você se encontra chama-se **pwd.**

10.5 pwd - IMPRIME NOME DE PERCURSO DO DIRETÓRIO DE TRABALHO

O comando **pwd** fornece o nome-de-percurso completo do seu diretório de trabalho (ou diretório corrente), começando da raiz do sistema de arquivos e continuando por todos os diretórios intermediários, até atingir o diretório corrente.

Como exercício vá para seu diretório-casa (**nome_sua_conta**), dando o comando **cd**, e então digite **pwd:**

```
$ cd<CR>
$ pwd<CR>
```

Após o último comando, o SOX responderá:

```
/usr/nome_sua_conta
$ _
```

Mude agora para o diretório **documentos** (com o comando **cd documentos**). Repita o comando **pwd**, ao qual deve aparecer na tela:

```
$ cd documentos<CR>
$ pwd<CR>
/usr/nome_sua_conta/documentos
$ _
```

Desça mais um nível (vide Figura 10.1) no seu sistema de arquivos, para o diretório **balanços** (com o comando **cd balanços**), e digite, então, **pwd**

```
$ cd balanços<CR>
$pwd<CR>
```

Deve surgir na tela, a informação:

```
/usr/nome_sua_conta/documentos/balanços
$ _
```

Suba um nível, para o diretório-pai de **balanços** com o comando:

```
$ cd .. <CR>
$ –
```

Mais uma vez, digite **pwd**. Agora aparecerá

```
$ pwd<CR>
/usr/nome_sua_conta/documentos
$ –
```

Retorne, finalmente, para seu **diretório-casa** e dê **pwd** para assegurar-se do retorno. Como vê, o comando **pwd** é muito útil para lembrar-lhe onde você se encontra no seu sistema de arquivos.

Excelente! Você já sabe como criar diretórios (**mkdir**) na sua conta, como se deslocar com desenvoltura por todo seu sistema de arquivos (**cd**) e como perguntar ao SOX onde você se encontra (**pwd**).

Os diretórios recém-criados encontram-se inicialmente vazios. Para chegarmos à estrutura do seu sistema de arquivos conforme a Figura 10.1, só nos falta criar os arquivos **cobrança, parabéns, 1985, 1986, 1987, jan** e **fev**, alocando-os aos seus respectivos diretórios.

Conclua a tarefa de reprodução da Figura 10.1, deslocando-se para cada um dos diretórios **cartas, iapas** e **balanços** e lá criando os arquivos (cada um com conteúdo a seu critério). Use o editor de linhas **ed** do SOX conforme aprendeu nas lições do livro ou então um processador de textos. Está pronta a Figura 10.1.

Restam ainda alguns comandos do SOX que são de muita utilidade. Passemos a eles.

10.6 cat - CONCATENA E LISTA ARQUIVOS

O comando **cat** do SOX serve para concatenar (juntar) arquivos e listar o conteúdo na tela de seu terminal (ou na impressora) ou para outro arquivo especificado.

Vejamos a funcionalidade de **cat** para listar o conteúdo de um arquivo na tela de terminal. Para tanto, dê o comando:

```
$ cat nome-arquivo<CR >
```

onde o argumento **nome-arquivo** deve ser substituído pelo nome de um dos arquivos que você criou (p. ex., **parabéns** ou **cobrança**). Pratique o uso de **cat**, listando cada um dos arquivos criados.

Se o Shell lhe responder com uma mensagem do tipo:

cat: não pode abrir nome_arquivo

significa que o arquivo a ser listado inexiste (talvez você tenha cometido um erro de digitação) ou, se existe, você não tem permissão de leitura. O assunto de permissões é tratado mais adiante.

Se, ao listar um arquivo, aparecer algo ilegível na tela, é porque o arquivo contém caracteres de controle (*não-imprimíveis*).

Você pode economizar tempo, pedindo para listar vários arquivos, com uma única linha de comando. Por exemplo, para dois arquivos:

```
$ cat nome-arquivo1 nome-arquivo2<CR>
```

Como você pode ver, o comando **cat** oferece uma alternativa eficiente para examinar vários arquivos rapidamente.

Com **cat** você pode também concatenar os conteúdos de dois ou mais arquivos e depositar o resultado em um outro. O formato do comando **cat** para concatenar três arquivos (arquivo1, arquivo2 e arquivo3) e depositar o resultado em um quarto (arquivo4) é

```
$ cat arquivo1 arquivo2 arquivo3 > arquivo4<CR>
```

Perceba o uso do > no formato acima (detalhes sobre isto no próximo capítulo). Ele serve para avisar ao Shell que a saída de **cat** deve ser redirecionada para **arquivo4** e não ser apresentada na tela de seu terminal (saída-padrão do SOX). Caso você deseje observar o resultado da concatenação (conteúdo de arquivo4) na tela do terminal, simplesmente use **cat** mais uma vez, no formato:

```
$ cat arquivo4<CR>
```

A fim de praticar esta última facilidade do comando **cat**, crie dois arquivos **poesia-1** e **poesia-2** – contendo cada um deles uma estrofe da poesia *O Escravo* (Ferreira Gullar, 1955), como mostra a Figura 10.3. Coloque estes arquivos no diretório que desejar – crie um para prática dos comandos SOX. Muito bem. Criados os arquivos, faça:

```
$ cat poesia_1 poesia_2 > poesia <CR>
$_
```

O SOX criará automaticamente o arquivo poesia. Certifique-se disto com o comando **ls –1.** Agora peça para ver o conteúdo do arquivo **poesia** na tela de seu terminal com o comando:

```
$ cat poesia<CR>
```

Devem aparecer as duas estrofes da poesia na Figura 10.3.

Detrás da flor me subjugam,
atam-me os pés e as mãos.
E um pássaro vem cantar
para que eu me negue.

(a) Crie o arquivo poesia_1 com a estrofe acima

Mas eu sei que a única haste do tempo
É o sulco do riso na terra
– a boca espedaçada que continua falando.

(b) Crie o arquivo poesia_2 com esta última estrofe

Figura 10.3 Estrofes para prática de cat.

10.7 cp - COPIA ARQUIVOS

O comando:

```
$ cp arquivo1 arquivo2 <CR>
$ _
```

cria o **arquivo2** e nele copia o conteúdo do **arquivo1**. Se algum dos arquivos (ou os dois) estiver num diretório distinto do diretório corrente, você deve fornecer o nome de percurso necessário para atingir o nível hierárquico desejado. Pratique o uso de **cp** fazendo uma cópia do arquivo **cobrança**, no seu diretório-casa:

```
$ cp cobrança ../cobrança<CR>
$ _
```

Verifique com os comandos **ls –1** e **cat** se a cópia realmente foi feita. Observe que assumimos que você se encontra no diretório **cartas**. Repita o exercício para cada um dos arquivos restantes (vide Figura 10.1).

O comando **cp** é muito útil para fazer cópias de reserva (**backup**) de arquivos importantes. Talvez você já tenha tomado estas precauções com as facilidades do seu próprio processador de textos. Às vezes, porém, é mais rápido usar **cp**. Há uma dica no uso do comando **cp** que aumentará ainda mais sua produtividade. Quando você tiver que copiar vários arquivos para um mesmo diretório e desejar manter os arquivos a serem copiados com o mesmo nome no diretório de destino, use o comando **cp** no formato abaixo:

```
$ cp arquivo1 arquivo2 ... nome_do_diretório_destino<CR>
$ _
```

Isto fará com que sejam criadas cópias dos arquivos **arquivo1, arquivo2** etc. (com estes mesmos nomes) no diretório de destino especificado.

Para praticar, use a dica acima para copiar os arquivos **jan** e **fev** do diretório **documentos/iapas** no seu **diretório-casa**. Uma coisa você deve ter notado: todos os arquivos copiados permanecem com cópia no seu diretório de origem. Vejamos como eliminá-los. Depois examinaremos o comando para *mover* arquivos.

10.8 rm - REMOVE ARQUIVOS

Para remover um arquivo, simplesmente dê o comando:

```
$ rm nome_arquivo <CR>
```

Você pode remover vários arquivos de um diretório com o comando:

```
$ rm arquivo1 arquivo2 ...<CR>
```

Pratique o uso do comando **rm** removendo todos os arquivos que você criou para exercício (mas somente estes!) dos seus diretórios de origem. Deixe apenas as cópias no seu diretório-casa.

Para remover um arquivo de um diretório você pode:

a) Posicionar-se (com o comando **cd**) no diretório onde reside o arquivo a ser removido e então usar o comando **rm**.

b) Usar o comando **rm** a partir do diretório corrente, fornecendo o nome de percurso até o arquivo a ser removido.

Exemplificamos a possibilidade a), para o caso dos arquivos no diretório **cartas:**

```
$ cd cartas<CR>
$ pwd<CR>
/usr/nome_sua_conta/cartas
$ rm cobrança parabéns<CR>
$ _
```

Se você agora der o comando **ls –1**, o sistema deve indicar um diretório vazio. Retorne ao seu **diretório-casa** (com o comando **cd**) para ilustrar a alternativa *b*).

Vamos remover os arquivos **jan** e **fev** do diretório **documentos/iapas**, segundo a alternativa b) delineada acima. É só você fornecer o comando abaixo:

```
$ rm documentos/iapas/jan documentos/iapas/fev<CR>
$_
```

Usando a alternativa a), remova o arquivo **1985** do diretório **documentos/balanços**, e a alternativa b) para remover os arquivos **1986** e **1987** do mesmo diretório.

Todos os diretórios do seu **diretório-casa** devem estar vazios mais uma vez (Exceto **documentos,** que tem como conteúdo dois subdiretórios: **iapas** e **balanços).**

Para coroar nosso trabalho destrutivo do sistema de arquivos na Figura 10.1, só nos falta remover os diretórios criados. É o que faremos na próxima seção.

10.9 rmdir - REMOVE DIRETÓRIOS

Para remover um diretório vazio do seu sistema de arquivos, você dispõe do comando **rmdir.** Vejamos um exemplo: vá para seu diretório-casa, fazendo:

```
$ cd<CR>
$ _
```

Para remover o subdiretório **cartas**, faça:

```
$ rmdir cartas<CR>
$ _
```

Use o comando **ls –1** para ver o resultado. O subdiretório **cartas** não deve fazer mais parte do seu diretório-casa.

Você conseguiu remover o diretório **cartas** porque ele se encontrava vazio. O comando **rmdir** não remove um diretório não-vazio. Por exemplo: o diretório **documentos**, apesar de não conter mais nenhum arquivo ainda contém os subdiretórios **iapas** e **balanços**. Tente remover o diretório **documentos**, com o comando:

```
$ rmdir documentos<CR>
```

O SOX vai lhe responder:

```
rmdir: diretório não está vazio
```

e não removerá o diretório (verifique com **ls –1**).

Para remover o diretório **documentos**, você precisa primeiro esvaziá-lo, ou seja, remover os diretórios **iapas** e **balanços**. Eliminaremos primeiro **iapas**. Qualquer uma das duas maneiras abaixo fará o serviço:

```
$ rmdir documentos/iapas<CR>
$ _
```

ou

```
$ cd documentos<CR>
$ rmdir iapas<CR>
```

Se você escolheu a última maneira, dê o comando **cd** para retornar ao seu diretório-casa. Use, então, uma das duas maneiras anteriores para remover o diretório **balanços**. Finalmente, remova **documentos.**

Você poderia ter economizado tempo, removendo os dois diretórios **iapas** e **balanços** de uma só vez. Para remover mais de um diretório (digamos três) de uma só vez, use o formato abaixo:

rmdir diretório1 diretório2 diretório3 <CR>

Assim, você poderia ter removido **iapas** e **balanços**, com o comando

```
$ cd documentos<CR>
$ rmdir iapas balanços<CR>
$ _
```

Retorne ao seu diretório-casa e dê o comando **ls –1**. Lá devem restar apenas arquivos (pelo menos os diretórios da subárvore na Figura 10.1 devem ter desaparecido). Tenha um pouco mais de paciência e crie mais uma vez os diretórios **cartas** e **documentos** (com seus dois subdiretórios **iapas** e **balanços**). Vamos precisar deles para exemplificar os efeitos do comando **mv.**

10.10 mv - RENOMEIA ARQUIVOS E DIRETÓRIOS

Tomamos o cuidado, no final da seção 10.7, de pedir-lhe para copiar cada um dos arquivos na Figura 10.1, no seu diretório-casa, antes de removê-los todos quando da seção 10.8. Vamos precisar das cópias agora.

O comando:

mv arquivo1 arquivo2<CR>

transfere o conteúdo do arquivo1 para o arquivo2, removendo arquivo1 após a transferência. Note que o comando acima é equivalente a:

```
$ cp arquivo1 arquivo2<CR>
$ rm arquivo1<CR>
$ _
```

Conseqüentemente, o uso do comando **mv** (**move**) é mais eficiente quando se deseja eliminar o arquivo original. **mv** efetivamente troca o nome do arquivo original (*arquivo1*) para o nome do arquivo-destino (*arquivo2*); ou seja **mv** renomeia arquivos.

Com **mv**, terminaremos (mais uma vez) a tarefa de reproduzir a Figura 10.1 no seu sistema de arquivos. Primeiro, contudo, desloque-se para o seu diretório-casa (**cd**) e veja como se apresenta o seu conteúdo (use o comando **ls –1**). Em seguida, vamos transferir (mover) os arquivos **cobrança** e **parabéns** para o diretório **cartas**. Fazendo isso com o comando:

```
$ mv cobrança cartas/cobrança<CR>
$ _
```

Examine agora o conteúdo do seu **diretório-casa**. O arquivo **cobrança** deve ter desaparecido. Ele agora se encontra no diretório **cartas** e tem o nome de percurso (completo) dado por **/usr/nome_sua_conta/cartas/cobrança**. Verifique.

Pratique o uso de **mv**, transferindo o segundo arquivo (**parabéns**) que deve pertencer ao diretório **cartas** (vide Figura 10.1).

Quando você tem vários arquivos para transferir para um mesmo diretório-destino, onde terão os mesmos nomes, é mais rápido você usar o comando **mv**, como mostrado abaixo:

```
$ mv arquivo1 arquivo2... nome_diretório_destino<CR>
$ _
```

Utilizando esta última dica, você pode transferir os arquivos **jan** e **fev** para o diretório documentos/iapas do seu **diretório-casa** assim:

```
$ mv jan fv documentos/iapas<CR>
$ _
```

Certifique-se do resultado. Conclua a Figura 10.1, transferindo os três arquivos restantes, **1985, 1986** e **1987.**

Só mais um detalhe: o comando **mv** pode também ser usado para trocar o nome de diretório. Por exemplo, se você deseja chamar de **nome_novo_dir** um diretório que antes se chamava **nome_velho_dir,** basta fazer:

```
$ mv nome_velho_dir nome_novo_dir<CR>
$ _
```

Como prática, troque o nome do diretório **cartas** na Figura 10.1 para **correio,** usando **mv**. Verifique se o SOX atendeu ao seu comando corretamente (use o comando **ls –1**).

10.11 chmod - ALTERA PERMISSÕES DE ARQUIVOS E DIRETÓRIOS

Nas últimas seções, você aprendeu como mexer com seu sistema de arquivos. Na vida diária, você poderia acessar, inclusive, arquivos e diretórios de outros usuários (e vice-versa). O acesso aos arquivos de outros usuários é feito de forma similar à que discutimos. Você só tem que fornecer o nome de percurso correto. Por exemplo (vide Figura 10.1), se você quisesse listar o conteúdo do arquivo **cobrança** em **nome_outra_conta**, bastaria fazer:

```
$ cat /usr/nome_outra_conta/cobrança<CR>
```

Nada diferente, bastou você fornecer o nome de percurso a partir da raiz. E se o outro usuário não quisesse que você examinasse o conteúdo do arquivo? E como é que ele impediria alguém de, em vez do inofensivo **cat**, emitir um desaforado **rm**?

O SOX permite-lhe atribuir permissões de acesso a seus arquivos e diretórios, de forma a limitar ou inibir intrusos indesejáveis no seu sistema de arquivos.

Como já visto com o comando **ls –1,** o SOX define três classes de usuário que podem ter acesso a cada arquivo ou diretório seu (vide Figura 10.2):

dono – é geralmente o usuário que criou o arquivo ou diretório. O superusuário pode alterar o dono com o comando **chown**. Como este livro não é dirigido ao superusuário, não estudaremos este último comando aqui.

grupo – conjunto de usuários que recebe uma identificação de grupo (por ex., *alfa*) em função das afinidades de suas tarefas (vide Capítulo 4). O grupo não precisa ter uma conta explícita no SOX. Há, todavia, a possibilidade de você dar acesso a seu arquivo ou diretório ou grupo ao qual você faz parte.

outros – todos os demais usuários SOX da sua instalação, identificados por uma conta.

Conforme já discutimos no Capítulo 4, cada arquivo ou diretório tem 3 tipos de permissões:

leitura (r) – permite ao usuário ler o conteúdo do arquivo; no caso de um diretório, o usuário pode descobrir o que o diretório contém (p. ex., com o comando **ls**). A permissão de leitura de um diretório não dá, automaticamente, permissão de leitura dos arquivos nele contidos; para tanto, é necessária permissão de leitura nos arquivos individuais.

gravação (w) – permite ao usuário alterar o conteúdo do arquivo; no caso de um diretório, o usuário pode criar novos arquivos e remover arquivos (ou diretórios) no diretório em questão. A alteração de arquivos já existentes no diretório depende das permissões de gravação em cada arquivo individual.

execução (x) – permite ao usuário chamar o nome do arquivo como se fosse um comando SOX; no caso de um diretório, permite que o usuário o transforme no diretório de trabalho (com o comando **cd**) e que acesse arquivos residentes no diretório (desde que o usuário também tenha permissão de leitura no diretório).

O SOX combina os três tipos de permissões acima para cada classe de usuários e denomina a combinação resultante de **modo** do arquivo ou diretório. O modo é obtido juntando-se as permissões **rwx** para o dono, grupo e outros, da esquerda para a direita. O comando **ls –l** apresenta o modo de cada arquivo ou diretório imediatamente após a indicação – ou **d** (vide Figura 10.2).

Um arquivo que possa ser lido, gravado e executado por qualquer usuário tem modo:

rwxrwxrwx

Um arquivo que só pode ser lido, gravado e executado pelo dono tem modo:

rwx------

O traço – indica, pois, a ausência da permissão correspondente (no caso, para o grupo e para os outros usuários).

Um arquivo que pode ser lido, gravado e executado pelo dono e pelos usuários do grupo tem modo:

rwxrwx---

Um arquivo que só permita leitura por qualquer usuário tem modo:

```
r--r--r--
```

Como prática, examine os modos de permissão dos arquivos com comandos SOX. A maioria deles encontra-se no diretório **/bin** (vide Figura 10.1). Para saber os modos de permissão nestes arquivos, faça:

```
$ ls -l /bin<CR>
```

Observe a listagem na tela de seu terminal. O primeiro traço em cada linha indica um arquivo, em seguida você vê o modo.

Você deve observar o seguinte:

```
-r-xr-xr-x
```

ou seja, os arquivos podem ser lidos (ou copiados) e executados.

Após criar um arquivo ou diretório em sua conta, verifique o modo que lhe foi atribuído. Se não estiver contente com o modo recebido, você pode alterá-lo com o comando **chmod.**

O comando **chmod** modifica o modo de um arquivo ou diretório. A modificação só pode ser feita pelo dono ou pelo superusuário. A alteração é feita com o comando no formato:

chmod numero nome-arquivo<CR>

Neste caso, o modo é mudado segundo o **número** fornecido.

Suponha que você criou um arquivo de nome **despesas** e pretende que só você e seu grupo possam lê-lo, gravá-lo e executá-lo. Suponha também que este não foi o modo fornecido pelo SOX na sua criação. Você pode alterar o modo para o desejado, fazendo:

```
$ chmod 770 despesas<CR>
$_
```

Se você agora fizer **ls –1** no diretório que contém **despesas**, verá que o modo deste arquivo agora é

rwxrwx---

o que é exatamente o que você desejava.

A Tabela 10.1 fornece o número correspondente para as combinações mais comuns do modo.

Tabela 10.1 Correspondência entre Números e Modos de Permissão.

número	modo
0	---
1	--x
2	-w-
3	-wx
4	r--
5	r-x
6	rw-
7	rwx

Para exemplificar, vamos recordar o último exemplo (**chmod 770**): entrada na tabela acima, para os números 7 (duas vezes) e 0 (zero), temos a reprodução de **rwxrwx---**, como queremos.

Caso você deseje um modo e o número correspondente não se encontre na Tabela 10.1, consulte os manuais do SOX ou então o seu amigo administrador.

Missão cumprida. Era tudo o que queríamos falar sobre os comandos para manipulação do seu sistema de arquivos no SOX. Agora, mãos à obra.

CAPÍTULO **11**

FERRAMENTAS DE USO GERAL DO SOX

Este capítulo trata de uma série de facilidades do SOX, visando melhorar a produtividade e a qualidade de seu trabalho. Elas são *ferramentas de uso geral*, no sentido de que atuam sobre outros comandos (quaisquer comandos), para propiciar um melhor uso deles.

As ferramentas que discutiremos são as seguintes (na ordem em que estão apresentadas):

- Entrada/Saída Padrão;
- Redirecionamento de Entrada/Saída;
- Dutos;
- Comandos na Retaguarda;
- Arquivo **.profile.**

11.1 ENTRADA/SAÍDA PADRÃO

Um comando do SOX, como já vimos, é um *programa* (conjunto de instruções) que geralmente lê dados de um ou vários arquivos em algum meio de armazenamento de dados (a *entrada*), pode fazer algum tipo de processamento com os dados de entrada (o *processamento*) e, finalmente, produz os resultados num arquivo em algum meio de armazenamento (a *saída*). O meio pode ser um terminal, um disco flexível, um disco rígido,

uma fita magnética, uma impressora etc. Uma impressora, se usada por um comando, é sempre um lugar de saída, diferentemente dos demais lugares, que podem ser tanto meios de entrada como de saída. Uma impressora tem tantas particularidades que mereceu um capítulo só para ela (Capítulo 13).

Para ilustrar o que acabamos de dizer a respeito do funcionamento (grosso modo) de um comando do SOX, revisemos o comando **sort**, visto no Capítulo 9, mais especificamente os seguintes exemplos:

```
sort –o lista lista
sort lista
sort
```

No primeiro caso, tanto a entrada (o arquivo **lista**) como a saída (também o arquivo **lista**) foram especificadas; o processamento consistiu no ordenamento alfabético dos dados de entrada – o resultado foi que o arquivo **lista** está agora ordenado.

No segundo caso, somente a entrada (o arquivo **lista**) foi especificada: sendo assim, o **sort** produziu seus resultados numa *saída-padrão* (seu terminal).

No terceiro caso, nem a entrada, nem a saída foram especificadas: deste modo, o **sort** usa uma *entrada-padrão* (seu terminal) e produz seus resultados da mesma forma que no caso anterior (saída-padrão).

Agora recordemos o comando **cat** (visto nos Capítulos 9 e 10). Vejamos o exemplo

```
cat gatos
```

do Capítulo 9. Neste caso, a entrada é **gatos**, enquanto o conteúdo deste arquivo é exibido no terminal (a saída-padrão); **cat** não faz qualquer processamento: apenas transfere dados de uma entrada para uma saída.

Vamos concluir: em geral, se um comando espera uma entrada e nenhum arquivo de entrada é especificado, então esse comando vai buscar seus dados de uma entrada-padrão, que é o terminal do usuário; se um comando produz uma saída e nenhum arquivo de saída é especificado, então esse comando vai jogar a saída em uma saída-padrão, que também é o terminal do usuário.

O que são então *Entrada-Padrão* e *Saída-Padrão*?

– *Entrada-Padrão* Um lugar (o terminal, por omissão) de onde um comando espera ler seus dados, se um *outro lugar* não tiver sido especificado.

– *Saída-Padrão* Um lugar (o terminal, por omissão) onde um comando produz sua saída, se um *outro lugar* não tiver sido especificado.

Quase todos os comandos do SOX seguem a filosofia de trabalho que acabamos de ver. Justifiquemos o "quase": há algumas exceções, bem poucas. Vejamos duas: o comando **ls**, que vimos no Capítulo 10, tem como entrada seu diretório de trabalho, no caso em que você não especifique um nome de diretório, ou seja, ele nunca usará o terminal como entrada (no que ele, o comando **ls**, tem total razão, não é verdade?); uma outra exceção é o comando **lpr** (que será visto no Capítulo 13), cuja saída é *sempre* uma impressora.

11.2 REDIRECIONAMENTO DE ENTRADA/SAÍDA

Agora vamos ver uma outra facilidade do SOX, que consiste no seguinte: se um comando vai usar uma entrada-padrão, que tal dizer a ele que essa entrada-padrão não será o terminal mas um outro arquivo, em disco, por exemplo? (Em outras palavras, que tal *redirecionar a entrada-padrão*?) Por outro lado, se um comando vai usar uma saída-padrão, que tal dizer a ele que essa saída-padrão não vai ser o terminal mas um outro arquivo, em disco, por exemplo? (Em outras palavras, que tal *redirecionar a saída-padrão*?)

À primeira vista, podem lhe parecer estranhas e pouco claras as vantagens que adviriam de um redirecionamento: para lhe acalmar, enquanto não nos aprofundamos na questão, pense nas inconveniências de um terminal como um meio de armazenamento de dados – por exemplo, um relatório produzido num terminal, a menos que ele seja pequeno e fácil de memorizar, é um relatório perdido.

Por motivos meramente didáticos, veremos primeiramente o redirecionamento de saída e, só depois, o redirecionamento de entrada.

REDIRECIONAMENTO DE SAÍDA

Comecemos com um exemplo: o comando **cat** normalmente exibe o(s) conteúdo(s) do(s) arquivo(s) lido(s) no terminal (a saída-padrão); se quisermos mudar esta saída para um arquivo permanente, o que temos a fazer é redirecioná-la para o arquivo pretendido, usando a notação

> nome_do_arquivo

ou seja, o símbolo > é um aviso ao Shell de que a saída-padrão é **nome_de_arquivo** e não o terminal. Em tempo: pode haver espaços, ou não, entre > e o nome do arquivo.

```
$ cat > gatos<CR>
gato angorá<CR>
gato siamês<CR>
gato viralata<CR>
<CTRL D>
$cat gatos<CR>
gato angorá
gato siamês
gato vira-lata
$
```

Analisemos bem o que acabamos de fazer. Em primeiro lugar, estamos diante de uma nova maneira de criar um arquivo (**gatos**, no caso) – a outra maneira, mais "normal", seria criar o arquivo com o editor de linhas **ed**, conforme fizemos no Capítulo 9. No primeiro **cat**, fizemos o seguinte: a entrada foi a padrão (o terminal), onde os dados para o arquivo foram digitados; a saída foi a padrão, redirecionada para o arquivo **gatos** – com isto, todos os dados sobre gatos digitados no terminal ficaram registrados no arquivo **gatos**. Pense agora como seria se você tivesse usado simplesmente **cat<CR>** . Experimente.

Comparando esta nova maneira de se criar um arquivo com a outra, através do editor de linhas **ed**, podemos perceber que a primeira forma é mais rápida e direta; no entanto, usando o editor de linhas, você pode corrigir, no ato, erros de digitação porventura cometidos.

Uma outra observação que queremos fazer é sobre o conceito geral por trás do uso das teclas<CTRL D>: elas indicam o fim dos dados de entrada quando a entrada-padrão é o terminal – isto não impede, contudo, que outras formas de indicação de fim dos dados possam ser utilizadas, como, por exemplo, o . no comando **mail** (veja o próximo capítulo).

Para encerrar nossos comentários sobre o último exemplo, o segundo **cat** simplesmente exibiu o conteúdo do arquivo **gatos**, recém-criado: a entrada foi o arquivo citado, enquanto a saída foi o terminal (padrão).

Vejamos este outro exemplo:

```
$ sort –o lista1<CR>
Alceu Aboim<CR>
Frederico Sidnei César<CR>
Djenane Regina Machado<CR>
<CTRL D>
$ sort > lista2<CR>
Alceu Aboim<CR>
Frederico Sidnei César<CR>
Djenane Regina Machado<CR>
<CTRL D>
$ cat lista1<CR>
Alceu Aboim
Djenane Regina Machado
Frederico Sidnei César
$ cat lista2<CR>
Alceu Aboim
Djenane Regina Machado
Frederico Sidnei César
$
```

Como podemos ver, temos duas estratégias diferentes de obter o mesmo resultado, qual seja o de salvar a ordenação de nomes digitados no terminal. No primeiro **sort**, o arquivo de saída **lista1** foi especificado. No segundo **sort**, a saída-padrão foi redirecionada para o arquivo **lista2**. Você pode usar qualquer uma dessas estratégias, a seu gosto.

Um lembrete importante: se o arquivo para onde o redirecionamento é feito já existir, todo o seu conteúdo anterior será destruído – isto porque, a primeira coisa que o Shell faz, quando encontra o caractere >, é criar um arquivo vazio com o nome especificado. Assim, tenha muito cuidado ao escolher um nome para a saída-padrão, para que não haja o perigo de você perder informações importantes.

```
$ date > agora<CR>
$ cat agora<CR>
Qua Jun 17 18:15:13 1987
$ date > agora<CR>
$ cat agora<CR>
Qua Jun 17 18:16:04 1987
$
```

Existe uma possibilidade de você redirecionar a saída-padrão para um arquivo já existente, sem destruir seu conteúdo anterior: para isto, você deve usar

>> nome_do_arquivo

Neste caso, a saída é colocada no final do arquivo (funcionando da mesma forma que a saída do **cat**).

```
$ ls /usr/voce/dir1 > meus_arquivos<CR>
$ cat meus_arquivos<CR>
arq1
arq2
arq3
arq4
$ ls /usr/voce/dir2 >> meus_arquivos<CR>
$ cat meus_arquivos<CR>
arq1
arq 2
arq3
arq4
arquivo 1
arquivo2
arquivo3
$
```

O que aconteceria se especificássemos, por exemplo

>> saída

e o arquivo **saída** não existisse antes? A resposta é que **saída** seria criado (como se tivéssemos digitado **> saída**). Este fato nos oferece uma grande possibilidade: a de usarmos sempre >> em vez de >, quando quisermos redirecionar a saída-padrão, prevenindo-nos contra qualquer possibilidade de perda do conteúdo de um arquivo. Entendeu por quê?

Vejamos um exemplo ilustrativo:

```
$ sort lista > lista<CR>
$ cat lista <CR>
$
```

Observe que **lista** está vazio! Relembrando: a primeira coisa que o Shell fez (primeiro comando) foi **zerar** o arquivo **lista**, já existente. Cuidado!

REDIRECIONAMENTO DE ENTRADA

O redirecionamento de entrada pode ser feito em todos os comandos que lêem seus dados de uma entrada-padrão. Um exemplo de tal comando é **write**, que copia mensagens de seu terminal no terminal de um outro usuário (você verá este comando com detalhes no próximo capítulo). Se a sua mensagem for muito longa, é preferível criá-la com o editor de linhas **ed** (permitindo-lhe corrigir prováveis erros de digitação) e usar o comando **write** redirecionando a sua entrada para o arquivo assim criado.

Para redirecionar a entrada, você deve usar a seguinte notação:

< nome_do_arquivo

onde o símbolo < indica para o Shell o redirecionamento da entrada-padrão do terminal para o arquivo especificado.

Vejamos duas formas de usar o comando **write**: a primeira, digitando a mensagem de seu terminal; a segunda, redirecionando a entrada para o arquivo **memorando**, previamente criado.

```
$ write andre<CR>
  .
  .
  .
  conteúdo do memorando
  .
  .
  .
<CTRL D>
$
```

```
$ write andre <memorando<CR>
$
```

Diferentemente de redirecionamento de saída, quando muitos comandos do SOX têm saída-padrão (podendo, portanto, ser redirecionada), a grande maioria dos comandos pede um arquivo como argumento de entrada: sendo assim, não é comum o redirecionamento de entrada, ou, muitas vezes, seu efeito é irrelevante, como no exemplo a seguir:

```
$ cat < gatos<CR>
```

que significa a mesma coisa que:

```
$ cat gatos <CR>
```

11.3 DUTOS

A palavra "duto" nos dá a idéia de um canal, por onde flui alguma coisa. No contexto do SOX, um duto é um canal conectando dois comandos (ou programas, ou *processos*) por onde escoa um *fluxo de dados* operado pelos dois comandos. O fluxo é constituído da seguinte forma: a saída-padrão de um comando é usada como a entrada-padrão de um próximo comando, como sugere a Figura 11.1.

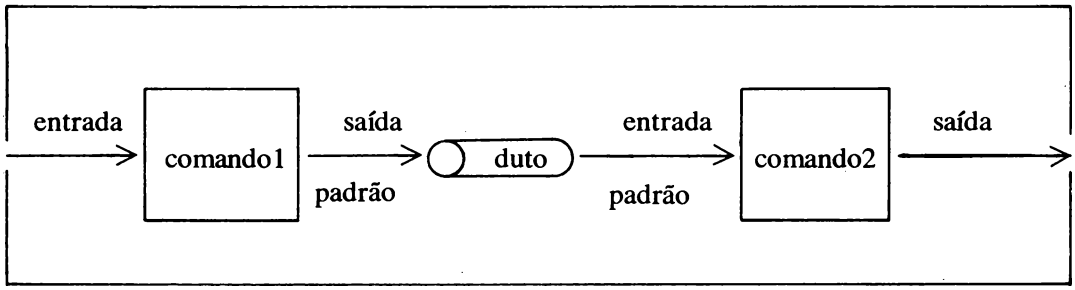


Figura 11.1 Comandos Conectados por um Duto.

Detenhamo-nos um pouco nesta figura. Observe como a seqüência é *fechada*, ou seja, você só tem acesso à entrada *antes* do comando1 (essa entrada, em verdade, é opcional, pois o comando 1 poderia ser **who, ls,** ou um outro que pode não ter uma entrada sua) e à saída *depois* do comando2. Dizendo com outras palavras, do seu ponto de observação, você só tem uma idéia macroscópica do duto: é como se comando1 e comando2 se fundissem num terceiro "comando" e, deste modo, você estaria fornecendo a entrada e obtendo a saída desse "comando". Mas, como sempre, o SOX é flexível: há um jeito de você obter dados intraprocessos de um duto – é com o comando **tee**, que será visto adiante.

Um duto é um poderoso e flexível mecanismo para facilitar e tornar mais eficiente o seu trabalho. Vamos já convencê-lo disto.

Suponhamos que você queira saber quais são as pessoas, trabalhando em um determinado projeto, que estão no ar, num dado momento. Admitamos que todo usuário desse projeto é catalogado como **proj3xxx,** onde **proj3** poderia significar "projeto 3" e **xxx** seriam as iniciais de um usuário. Apresentemos, primeiro, a maneira ineficiente de se obter tal informação.

```
$ who<CR>
andre       tty05      Jun 18      10:35
marcilio    tty01      Jun 18      10:36
joao        tty04      Jun 18      10:40
proj3mcs    tty02      Jun 18      10:42
maruzia     tty10      Jun 18      10:43
maira       tty09      Jun 18      10:50
antonio     tty11      Jun 18      10:51
proj3ddb    tty03      Jun 18      10:51
nivaldo     tty06      Jun 18      11:00
lolo        tty07      Jun 18      11:01
projeto     tty12      Jun 18      11:03
cacilda     tty14      Jun 18      11:04
projeto     tty18      Jun 18      11:05
proj3ec1    tty17      Jun 18      11:10
voce        tty08      Jun 18      11:15
$
```

Você agora terá de percorrer com os olhos a lista de usuários (que pode ocupar mais de uma tela), procurando por aqueles cujas iniciais sejam "proj3" – convenhamos que tal modo de proceder é por demais primitivo, além do que você poderia facilmente se enganar. Vejamos uma maneira mais rápida e segura:

```
$ who | grep "proj3"<CR>
proj3mcs      tty02      Jun 18      10:42
proj3ddb      tty03      Jun 18      10:51
proj3ec1      tty17      Jun 18      11:10
$
```

Para indicar ao SOX que você deseja montar um duto entre dois comandos, você usa o caractere especial | (barra vertical) separando os seus diversos comandos.

Recapitulemos o que acabamos de fazer: nosso duto conectou os comandos **who** e **grep**, já conhecidos; a saída-padrão de **who** é a entrada para o comando seguinte, **grep** – este último exibe as linhas procuradas no terminal (saída-padrão).

Quer melhorar o resultado? Ordene, então, os usuários.

```
$ who | grep "proj3" | sort<CR>
proj3ddb      tty03      Jun 18      10:51
proj3ec1      tty17      Jun 18      11:10
proj3mcs      tty02      Jun 18      10:42
$
```

Perceba que, nestes dois últimos exemplos, usamos o comando **grep** sem um nome de arquivo de entrada. Isto atende à explicação que ficamos lhe devendo desde o Capítulo 9, qual seja, mostrar um exemplo da utilidade do comando **grep** sem argumento de entrada.

Perceba também que poderíamos mudar a ordem dos comandos **grep** e **sort** no duto, ou seja, se digitássemos

```
$ who | sort | grep "proj3"<CR>
```

obteríamos o mesmo resultado. Confira e entenda o porquê.

Se você não tivesse usado os dois dutos, o processo de obtenção da resposta à sua pergunta seria bem mais lento, como o mostrado a seguir:

```
$ who > usuarios_proj3<CR>
$ grep "proj3" usuarios_proj3 > usuarios_p3<CR>
$ sort usuarios_p3<CR>.
proj3ddb        tty03          Jun 18          10:57
proj3ec1        tty17          Jun 18          11:10
proj3mcs        tty02          Jun 18          10:42
$ rm usuarios_proj3 usuarios_p3<CR>
$
```

Na terminologia do SOX, todo comando conectado a um duto (a menos, possivelmente, do primeiro e do último) é um *filtro*. Um filtro, de um modo geral, é qualquer programa que lê seus dados de uma entrada-padrão e grava seus resultados em uma saída-padrão. Desta forma, **grep** e **sort**, nos exemplos de dutos, funcionaram como filtros, diferentemente de **who**, que não usa sua entrada-padrão.

OBTENDO RESULTADOS INTERMEDIÁRIOS DE UM DUTO

Como já afirmamos, é possível que você tenha acesso a uma saída de um comando conectado a um duto. Para isto, você deve usar o comando **tee**, cujo formato é:

tee nome_do_arquivo

Este comando simplesmente copia uma saída de um dos comandos do duto no arquivo **nome_do_arquivo**, permitindo que você possa ter acesso ao seu conteúdo (com **cat**, por exemplo). Usando este comando, devemos fazer uma modificação na Figura 11.1 – a Figura 11.2, abaixo, mostra um duto com o comando **tee**.

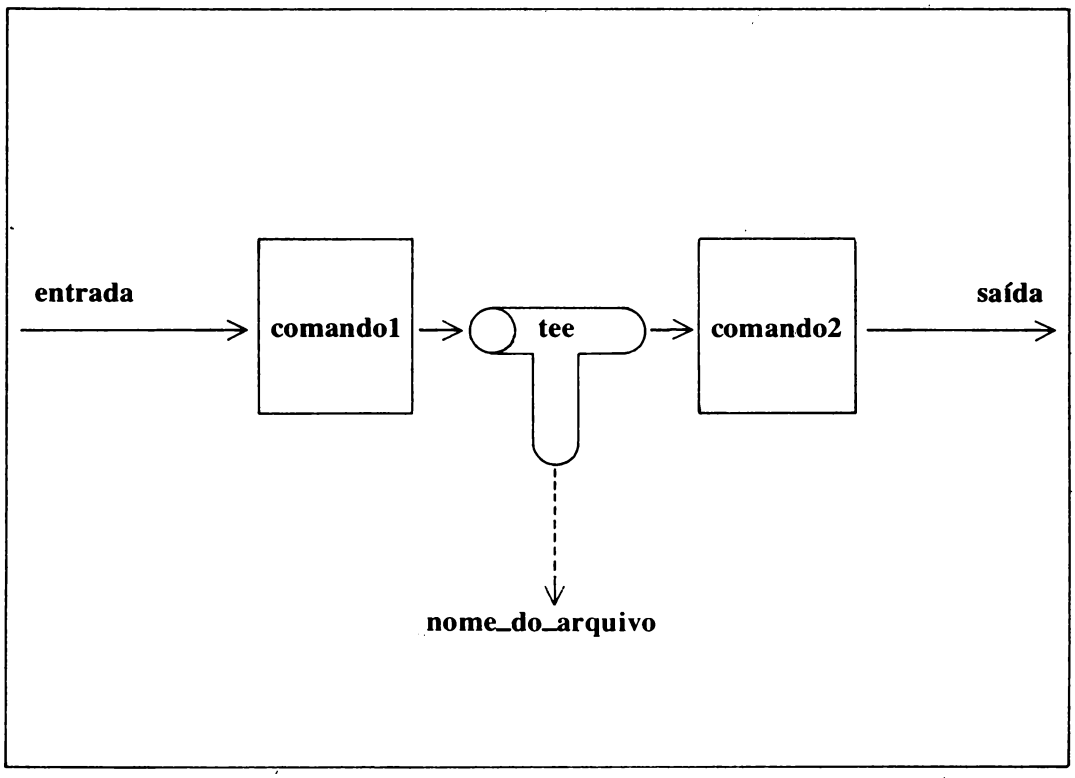


Figura 11.2 Um Duto com o Comando tee.

Para ilustrar o uso do comando **tee**, vamos registrar todos os usuários que entraram no ar após as 11 horas. Somente os usuários do "projeto 3" devem aparecer no terminal.

```
$ who | sort | grep "11:" | tee apos11 | grep "proj3"<CR>
proj3ec1        tty17           Jun 18          11:10
$ cat apos11<CR>
cacilda         tty14           Jun 18          11:04
lolo            tty07           Jun 18          11:01
nivaldo         tty06           Jun 18          11:00
proje3ec1       tty17           Jun 18          11:10
projeto         tty12           Jun 18          11:03
projeto         tty18           Jun 18          11:05
voce            tty08           Jun 18          11:15
$
```

Suponhamos agora que queiramos também registrar, tanto no terminal como em um arquivo permanente, os usuários do "projeto 3" que estão no ar em um determinado momento (por exemplo, 11:15):

```
$who ı grep "proj3" ı sort ı tee usrproj3_1115<CR>
proj3ddb        tty03           Jun 18          10:57
proj3ec1        tty17           Jun 18          11:10
proj3mcs        tty04           Jun 18          10:40
$ cat usrproj3_1115<CR>
proj3ddb        tty03           Jun 18          10:57
proj3ec1        tty17           Jun 18          11:10
proj3mcs        tty04           Jun 18          10:40
$
```

Como vemos, o comando **tee** pode ser o último de uma seqüência de dutos.

11.4 PROCESSOS NA RETAGUARDA

Vamos recapitular um pouco o que de comum acontece em uma sessão de trabalho no SOX: quando você entra no ar, um processo é acionado, o **sh** (do inglês **sh**ell: que, como você já sabe, é um processo – ou programa, ou comando – que faz a comunicação entre você e o SOX); o **sh** emite, então, o caractere de prontidão **$**, que é o sinal de que ele está esperando um comando seu; o processo **sh** dura, em verdade, todo o tempo da sessão, ou comunicando-se ou esperando que um comando acabe, quando então ele volta a se comunicar, para conhecer sua próxima ação e, assim, sucessivamente; quando você manda executar um comando, dois processos seus, no mínimo, existem simultaneamente – o **sh** e o processo correspondente ao comando ora sendo executado; deste ponto de vista, uma sessão tem sido, até aqui, uma sucessão de processos, um após o outro (com a sua coordenação a cabo de um terceiro processo, o **sh**) traduzindo, assim, um método de tabalho que consiste em acionar um comando, esperar que ele termine, acionar um próximo comando e, assim, sucessivamente.

Você pode confirmar o que afirmamos sobre a existência de mais de um processo simultaneamente, com o comando **ps** (do inglês **p**rocess **s**tatus: imprima os processos em execução, em tradução livre):

```
$ ps<CR>
   PID      TTY      TIME     CMD
   50       01       0:08     sh
   71       01       0:01     ps
$
```

Temos, no exemplo, dois processos, o próprio **ps** e o **sh**, este último, sempre. Vejamos as outras informações referentes a processos, fornecidas pelo comando **ps: PID** (do inglês **P**rocess **ID**entification número interno associado ao processo) – cada processo seu e de cada um outro usuário recebe do SOX um número de identificação (dá para você perceber que, a cada momento, muitos processos novos são criados e isto se reflete nos números criados); **TTY** indica o terminal onde o processo foi iniciado; **TIME** indica o tempo (min:seg) que o processo está em execução – no exemplo, o processo **sh** gastou, até

agora, oito segundos de tempo de CPU, enquanto o processo **ps** levou um segundo para ser executado; **CMD** (do inglês **COMMAND**: comando) imprime a linha de comando usado para disparar o processo.

Há várias opções para o comando **ps** que não vamos apresentar, por achar que elas não são importantes para os propósitos deste livro.

É importante saber que um processo (processo *pai*) pode, por sua vez, desencadear outros processos (processos *filhos*), estes últimos, por sua vez, também podem desencadear outros processos, e assim por diante. Todo processo em atividade, qualquer que seja a sua origem, aparece na lista fornecida pelo comando **ps**. Portanto, não se surpreenda se aparecerem mais processos do que você estaria imaginando.

Um desdobramento natural da filosofia de trabalho do SOX, que é aquela de um sistema multitarefa, ou seja, que pode executar várias tarefas (processos) simultaneamente, seria permitir que um mesmo usuário, num mesmo terminal, pudesse mandar executar mais de um processo ao mesmo tempo (não estamos falando, é óbvio, do processo **sh**, nem também de processos-filhos) – do ponto de vista do SOX, tratar-se-ia apenas de processos a mais. E do seu ponto de vista? Poderia ser uma alternativa interessante. Muitas vezes, um processo demorado e não-interativo far-lhe-ia ficar esperando um bom tempo até que ele terminasse e você pudesse realizar sua próxima ação. Para evitar esse incômodo, você poderia quebrar o ritmo seqüencial de suas ações dizendo ao shell "execute este processo, que é demorado e que age sozinho, *na retaguarda*, e continue se comunicando comigo porque, enquanto ele não termina, tenho outras coisas a fazer."

Para ordenar ao shell que execute um processo na retaguarda (ou seja, sem inibir a comunicação usuário-shell), basta colocar o caratere **&** no fim de um comando (você pode, ou não, colocar espaços entre o fim do comando e o **&**) – fazendo isto, imediatamente após digitar o comando aparece novamente o caractere de prontidão do shell.

Quer um exemplo de um comando que pode ser demorado? O **sort**, por exemplo. Imagine ordenar 1.000 nomes, 2.000 etc. Está aí um bom pretexto para fazer processamento na retaguarda.

```
$ sort lista >>lista1 &<CR>
128
$ ps<CR>
   PID     TTY     TIME    CMD
    90      01      0:08     sh
   128      01      0:01    sort lista
   129      01      0:01    ps
$
```

Observe que o número do processo na retaguarda é-lhe informado, antes do próximo caractere de prontidão.

É preciso, entretanto, cuidado e critério ao decidir colocar processos na retaguarda. Quanto a ser criterioso, que fique bem claro que a existência de vários processos na retaguarda pode sobrecarregar demasiadamente o sistema, podendo não compensar, em termos do tempo total de uma sessão de trabalho, que poderia até ser menor com comandos em seqüência. Quanto aos cuidados, nunca esqueça estas três máximas:

- Enquanto um processo estiver na retaguarda, não inicie um outro comando que tente modificar um arquivo que seja também entrada para aquele processo na retaguarda.

- Nunca deixe que a saída de um processo na retaguarda seja o terminal.

- Um processo na retaguarda não deve ter entrada do terminal.

Assim, você não deveria fazer jamais como indicado abaixo:

```
$ sort lista >> lista1 &<CR>
$ ed lista<CR>              Um exemplo do que não
    .                           deveria ser feito.
    .
    .
```

Veja em que situação difícil você colocaria o SOX: o arquivo **lista** em que o **sort** está "pensando" já está obsoleto com as modificações que você agora está introduzindo nele – o resultado dessa bagunça pode ser imprevisível!

Violando a segunda máxima, a tela de seu terminal poderia virar uma salada nada apreciável, misturando as saídas de vários comandos. Este pequeno exemplo já dá uma amostra do problema:

```
$ sort -n numeros&<CR>
93
$ 1
2
3
4
5
```

O exemplo tratou da ordenação de números contidos no arquivo **números.** Observe como está difícil concluir que o SOX está esperando seu próximo comando, uma vez que o caractere de prontidão está misturado com a saída do **sort.**

A maneira correta de proceder seria:

```
$ sort -n -o num_ord numeros &<CR>
93
$
```

ou então:

```
$ sort -n numeros >> num_ord &<CR>
```

Já a maneira seguinte resultará em nada:

```
$ sort -n >> num_ord &<CR>
93
$ ps
   PID      TTY      TIME     CMD
    80       01      0:08      sh
    88       01      0:01      ps
$ cat num_ord<CR>
$
```

Isto porque, o Shell, ao ver que você colocou um comando na retaguarda sem redirecionar a entrada, fez com que a entrada-padrão do comando sort fosse nula.

INTERROMPENDO UM PROCESSO NA RETAGUARDA

A questão que se coloca agora é: como controlar um processo que está na retaguarda, uma vez que ele não é mais diretamente visível ao usuário?

O comando **ps**, já visto, é parte da resposta: através dele poderíamos saber se um processo ainda estaria em execução (quando tal processo apareceria na lista de processos fornecida pelo comando) ou se já teria sido encerrado (quando a lista dos processos não mais acusaria o processo em foco).

Se você, por alguma razão, desejasse interromper um processo em execução na retaguarda (até porque você poderia desconfiar que estaria havendo algum problema com ele), usaria o comando **kill** (do inglês **kill**: cancele, interrompa, para ficar numa tradução mais suave). Para isso, você teria que especificar o número de identificação do processo (obtido de **ps**).

```
$ kill 38<CR>
$
```

Com este exemplo, o processo 38, que estava na retaguarda, foi cancelado.

Agora, uma pausa para meditação. O processo **sh** (o shell) está quase sempre esperando; mais precisamente, ele fica *dormindo*, esperando que um comando que não esteja na retaguarda acabe, quando ele volta a *acordar* (não nos alonguemos nesta trilha: basta saber que um processo tem vários *estados*, como os que apontamos, e que, em qualquer desses estados, o processo continua *ativo* e, como tal, aparece na lista dos processos do comando **ps**). Moral da história: o processo **sh**, sendo um processo sempre ativo (embora *dormindo* a maior parte do tempo!) pode ser cancelado com **kill**, o que é o mesmo que dizer que você pode praticar seu próprio "suicídio"!

Para evitar coisas deste tipo, e outras igualmente indesejáveis que não vale a pena citarmos aqui, é que alguns níveis de segurança poderão ser estabelecidos pelo SOX, para que ele possa tomar efetivamente a decisão de cancelar um processo. Trocando em miúdos: é possível que você, ao mandar cancelar um processo, não seja atendido (o processo **sh** é um deles). Neste caso, recomendamos procurar o administrador de sua instalação – deixemos com ele a ação de "obrigar" o SOX a cancelar seu processo, se este for o caso.

11.5 O ARQUIVO .PROFILE

Quando você entra no ar, imediatamente o Shell se prepara para receber e interpretar seus comandos, como já sabemos. Na verdade, antes de exibir o caractere de prontidão **$**, indicando que o sistema está pronto para receber um seu comando, o Shell pesquisa seu diretório-casa para ver se lá existe um arquivo de nome **.profile**: em caso afirmativo, *todos* os comandos que tiverem sido colocados neste arquivo são executados nesse momento, um após outro, na ordem em que foram colocados; somente após a execução desses comandos é que você poderá introduzir seus novos comandos.

Por exemplo, você pode desejar que, cada vez que você entre no sistema, apareçam a data e a hora da sessão e quem está usando o sistema nesse momento. Vejamos o conteúdo de **.profile** para este caso:

```
$ cat .profile<CR>
date
who
$
```

Observe que você não deve colocar o sinal de prontidão **$** no arquivo **.profile.**

Em resumo: **.profile** é um arquivo especial de seu diretório-casa, cujas linhas são comandos (sem o caractere de prontidão **$**) que serão executados, automaticamente, no início de sua sessão (antes, portanto, que o primeiro caractere de prontidão seja exibido); além desta característica toda particular do modo como ele é usado, ele é um arquivo criado e editado como qualquer outro.

CAPÍTULO **12**

O SOX NO ESCRITÓRIO: COMUNICAÇÃO ENTRE USUÁRIOS E OUTROS RECURSOS

Comunicação entre Usuários é outro tema de fundamental importância no complexo de recursos computacionais necessários ao processo de automação de escritórios. Observe que, em verdade, desde o Capítulo 7 temos discutido conceitos e ferramentas naturais do SOX que podem ser utilizadas no processo de automação de seu escritório.

Pensemos um pouco neste assunto. Antes de mais nada, uniformizemos nossos pensamentos: o que tal conceito – *automação de escritórios* – nos sugere? Sem procurarmos uma definição abstrata, pensemos no seguinte grau de automação de um escritório:

– Preparação de documentos feita com o auxílio do computador.

– Guarda de documentos pelo computador.

– Recuperação de documentos pelo computador.

– Correio (envio e recepção de documentos) por computador.

– Controle de agenda por computador.

Os Capítulos 7 e 8 dedicaram-se à preparação e guarda de documentos através do editor de linhas **ed**, enquanto ordenação de textos (considerada aqui como um aspecto da preparação de um texto) foi vista no Capítulo 9, através do programa **sort**. O Capítulo 9 também discutiu a recuperação de documentos por meio dos programas de serviço **grep** e **cat**.

Na seqüência, dedicar-nos-emos neste capítulo às facilidades naturais do SOX para correio e agenda eletrônicos.

Nesta altura é conveniente também discutirmos o cenário em que poderia se dar a comunicação entre usuários. Geralmente, somos levados a pensar em um vasta estrutura de apoio consubstanciada numa *Rede de Comunicação de Dados*, cobrindo grandes distâncias e integrando os mais diversos tipos de equipamentos – típica de grandes empresas com atuação em boa parte do território nacional. No entanto, uma média empresa (e mesmo uma pequena empresa) necessita também de serviços de comunicação de dados, com a vantagem de poder usar uma estrutura de comunicação de dados muitíssimo mais simples.

Para nossos propósitos de mostrar os recursos naturais de comunicação de dados do SOX, imaginemos tão-somente um sem-número de terminais e micros (doravante, simplesmente terminais) conectados a uma máquina SOX: os terminais estariam espalhados por diversas salas, que poderiam ocupar andares e prédios distintos – este é o nosso cenário.

Permita-nos ainda uma pequena digressão sobre as vantagens da automação das atividades de um escritório. Quanto mais automatizado um escritório, mais sobrará tempo para as atividades que realmente interessam – planejamento, pesquisa de novas frentes de mercado, desenvolvimento – até hoje uma fração insignificante das horas de funcionamento de um escritório: 13%, segundo as estatísticas.

Voltando ao nosso assunto específico, estudaremos agora vários comandos do SOX que tratam de problemas de comunicação de dados. Comecemos pelo comando **write.**

12.1 ENVIANDO MENSAGENS COM O COMANDO WRITE

O comando **write** permite que você prepare uma mensagem e a envie a um outro usuário que também esteja ligado ao sistema naquele momento (uma outra maneira de dizer isto é: "... que também esteja no ar").

A forma geral do comando **write** é a seguinte:

write usuário [tty]

usuário aqui é o nome de uma conta do SOX, que normalmente está associado ao nome de seu titular, identificando assim o usuário propriamente dito. Opcionalmente, você pode indicar também a identificação do terminal em que o usuário se encontra (**tty**).

A entrada-padrão para a sua mensagem é o terminal: nada impede, entretanto, que você a redirecione (em caso de dúvida, reveja o Capítulo 11) para um arquivo permanente – isto é recomendado, por exemplo, se você quiser enviar a mesma mensagem para vários usuários digitando sua mensagem somente uma vez (*observe*: num dado instante você só pode enviar a mensagem para um único usuário, através do comando **write**).

Uma mensagem pela entrada-padrão é composta de linhas (terminadas com <CR>, tal como para o **ed**). A transmissão é feita linha-a-linha e você não recebe nenhuma confirmação do usuário receptor. Para indicar o encerramento de sua mensagem, você deve digitar as teclas <CTRL-D> – ao fazê-lo, o caractere do prontidão do **Shell** volta a aparecer.

Para saber quem está no ar no momento, você pode utilizar o comando **who**.

```
$ who<CR>
andre          tty03          14:05
maruzia        tty00          14:05
voce           tty02          14:15
joao           tty08          14:16
estevao        tty07          14:18
$write andre<CR>
Preciso falar com você às 17:00 na minha sala<CR>
<CTRL-D>
$
```

Vamos dar uma olhadela no terminal de André:

```
Mensagem de voce   tty02   [quinta 14/05 14:19:03]
Preciso falar com você às 17:00 na minha sala
EOF
```

A expressão **EOF** (do inglês **E**nd-**O**f-**F**ile: fim da transmissão, em tradução livre), no terminal de André, corresponde ao <CTRL-D> que você digitou.

Se você quiser mandar uma mensagem para um usuário que "está" em vários terminais (pense em várias situações em que isto pode ocorrer, menos que o usuário tenha o dom da ubiqüidade!), você tem duas possibilidades: a primeira identificando o terminal; a segunda, não identificando o terminal – neste caso, a mensagem irá para o terminal de menor número de identificação.

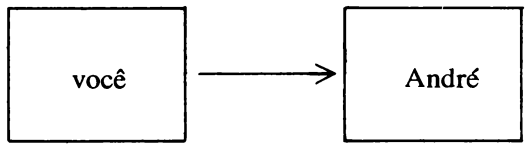
```
$ who<CR>
.
.
.
projeto1        tty13        15:05
.
.
.
projeto1        tty10        15:25
projeto1        tty04        15:25
.
.
.
$ write projeto1 tty10<CR>
```

Ou então:

```
$ write projeto1<CR>
A mensagem será transmitida ao terminal tty04
```

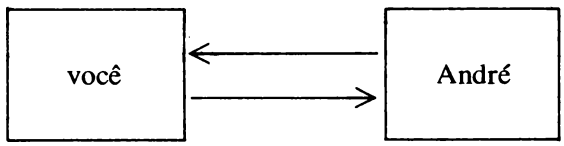
Agora vamos ver alguns inconvenientes do modo como você usou o comando **write**, bem como inconvenientes do próprio comando em si.

- Você não pode ter certeza de que André recebeu seu aviso, pois André não lhe deu sinal de vida, ou seja, não houve um diálogo entre você e André, como mostra o diagrama abaixo.



- A mensagem que você enviou foi "jogada" abruptamente no terminal de André (foi como invadir uma sala, que estava de porta fechada, sem pedir licença!). Este é um problema do próprio comando **write**, e não há como resolvê-lo no âmbito do mesmo.

Quanto ao primeiro problema, ele pode ser resolvido estabelecendo-se um diálogo (ainda assim, você teria que contar com a boa vontade do interlocutor em responder às suas "perguntas"), conforme o diagrama abaixo:



Em primeiro lugar, a simples digitação do comando **write** já faz aparecer um aviso, "Mensagem de...", na tela do receptor. Deste modo, você pode esperar até que André confirme que está pronto para receber sua mensagem. Suponha que cada mensagem ocupe só uma linha, como o comando **write** transmite **linha-a-linha**, então você digita a primeira mensagem, espera a resposta (também em uma linha), digita a segunda mensagem, espera ..., e assim por diante.

Vamos então reproduzir um diálogo com o comando **write**, mostrando primeiro a tela de **voce** e depois a de **andre.**

```
$ write andre<CR>
Mensagem de andre      tty03      [Quinta 14/05 14:19:02]
Oi voce, tudo bem? Estou aguardando.
Preciso falar com você às 17:00, na minha sala. <CR>
Às 17:00 não posso. Pode ser às 17:30?
Certo, 17:30 então. Até logo.<CR>
Até logo.
EOF
<CTRL-D>
$
```

Agora a tela de André:

```
Mensagem de voce    tty02    [Quinta 14/05 14:19:03]
.
.
.
$ write voce<CR>
Oi, voce, tudo bem? Estou aguardando.<CR>
Preciso falar com você às 17:00 na minha sala.
Às 17:00 não posso. Pode ser às 17:30?<CR>
Certo, 17:30 então. Até logo.
Até logo.<CR>
<CTRL-D>
EOF
$
```

Podemos perceber que levou algum tempo (indicado por "...") para que André começasse a responder: provavelmente ele preferiu primeiro terminar alguma tarefa que

estava realizando naquele momento. A propósito, se ele estivesse usando o editor de textos **ed**, não teria que sair dele para digitar o comando **write**: ele poderia usar **!write...**, de dentro do **ed** (lembra-se? se não, reveja o Capítulo 8).

Este último exemplo já melhorou muito em relação ao primeiro. Mas, ainda assim, algumas coisas importantes ainda não estão resolvidas. Por exemplo, a suposição de que cada mensagem caberia dentro de uma linha, é uma suposição pobre: uma mensagem pode ocupar várias linhas de texto. Daí surge a pergunta: como quem está recebendo uma mensagem sabe que ela terminou, para que possa responder? A solução é convencionar um código indicador do fim de uma mensagem. Assim, num diálogo, uma mensagem só seria respondida quando aparecesse o código de fim da mesma. Uma convenção bastante adotada é colocar (**c**) (de **c**âmbio) para terminar uma mensagem e (**cf**) (de **c**âmbio **f**inal) para encerrar o diálogo.

Vejamos então outra forma de André responder:

```
$ write voce<CR>
Oi, voce, tudo bem? Estou aguardando.(c)<CR>
Preciso falar com você às 17:00, na minha sala. (c)
Às 17:00 não posso, a não ser que você<CR>
me diga que o assunto é por demais<CR>
urgente. Do contrário, poderia ser às 17:30?(c)<CR>
Certo, 17:30 então. Até logo.(cf)
Até logo.<CR>
<CTRL-D>
EOF
$
```

Tal como no editor de linhas **ed**, você pode usar o caractere ! para interromper uma mensagem e usar qualquer comando do SOX: ao executar o comando, o SOX lhe imprime um segundo !, indicando que retornou ao comando **write**, permitindo assim que você continue preparando suas mensagens.

```
$ write andre<CR>
        mensagens
!who<CR>
.
.
.
andre       tty03       14:05
.
.
.
!
Você está no terminal desde as 14:05.<CR>
Ainda falta muita coisa?(c)<CR>
        mensagens
<CTRL-D>
$
```

Antes de encerrarmos a discussão sobre o comando **write**, voltemos à questão da "rudeza" do mesmo. Esta sua característica de entrar sem aviso em um terminal pode ser extremamente incômoda para quem está recebendo a mensagem. E se esta for muito longa (daquelas que você gravou em um arquivo, portanto sem dar chance a um diálogo)? Chocante mesmo! Mesmo um diálogo não suavizaria tal situação – pode ser um usuário que desejaria mais que nunca estar "sozinho" naquele momento!

Por tais considerações, o uso do comando **write** deve ser parcimonioso ao extremo. Muito mais apropriado deverá ser o uso do comando **mail** que será visto logo mais. Antes dele, porém, vamos dar uma olhada no comando **mesg**.

12.2 FECHANDO-SE PARA MENSAGENS

Há um antídoto para o comando **write**, que é o comando **mesg** (do inglês **message**: mensagem): com ele você se previne das "rudes" interrupções causadas por outros usuários que enviam mensagens para você através do comando **write**. O formato geral do comando **mesg** é:

mesg [n] [y]

mesg com argumento **n** (do inglês **no**: não) impede a recepção de mensagens enviadas com **write.mesg** com argumento **y** (do inglês **yes**: sim) estabelece que você está aberto à comunicação de outros usuários com **write. mesg** sem argumento mostra seu estado de aptidão corrente.

```
$ mesg<CR>
é s
$ mesg n<CR>
$ mesg<CR>
é n
$
```

Como você pode perceber, normalmente você está aberto a mensagens vindas de outros terminais (tal é o estado de seu terminal quando você entra no sistema).

Se você estiver fechado para mensagens e alguém tentar mandar uma mensagem para você através de **write** ele receberá a seguinte mensagem no terminal dele:

```
$ write voce<CR>
Permissão negada
$
```

12.3 CORREIO ELETRÔNICO

Um sistema de correio eletrônico pode ser estabelecido através do comando **mail** (do inglês **mail**: envie correspondência a usuários ou leia correspondência, traduzindo pelo seu significado). Quais os requisitos mínimos para um serviço desse tipo?

- O primeiro deles, e talvez o mais importante, é a existência de uma **caixa postal** para cada usuário: com isto, o usuário está desobrigado de ter de estar ligado no sistema para receber uma mensagem – em vez disso a mensagem cai diretamente na caixa postal do destinatário. Assim, o usuário pode, quando quiser, consultar sua caixa postal.
- Complementar ao primeiro requisito, deve existir um dispositivo de aviso ao usuário de que tem correspondência nova na sua caixa postal.
- O sistema deve ser capaz de veicular e de tratar eficazmente quaisquer documentos, mesmo os muito longos.
- Uma mesma correspondência pode ser enviada simultaneamente a vários destinatários.

O **mail** apresenta tais requisitos mínimos. Parte de seu esquema de funcionamento é mostrado na Figura 12.1.

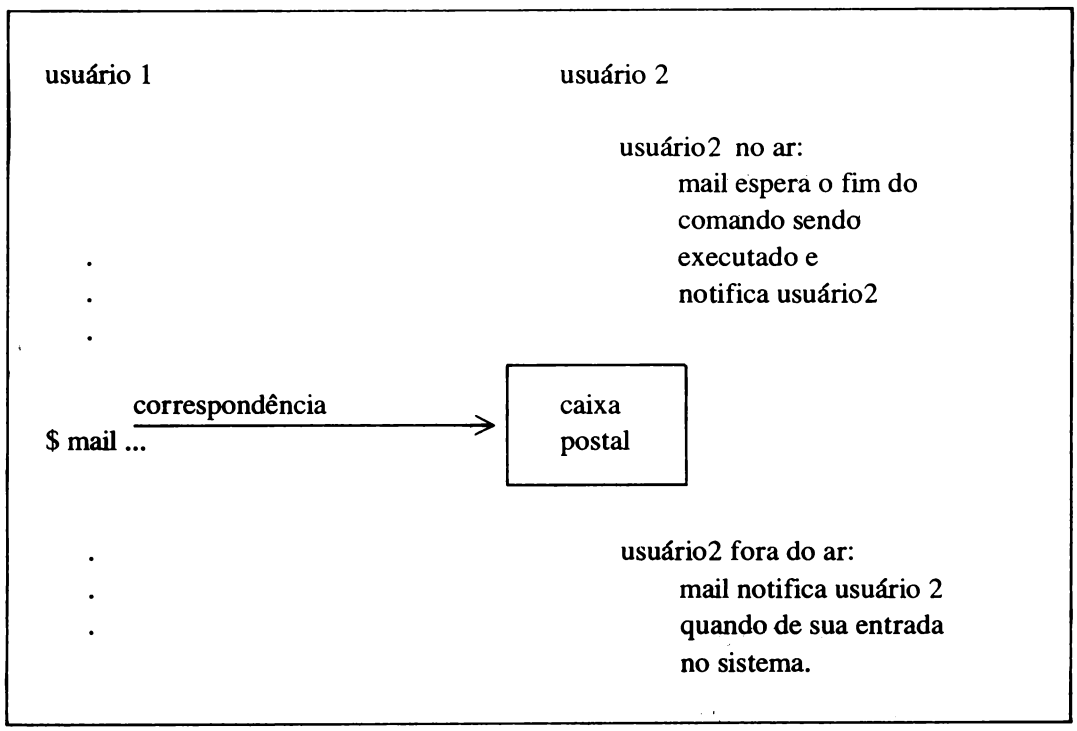


Figura 12.1 Parte do Esquema de Funcionamento do Comando **mail**.

Vamos primeiramente ver como um usuário envia correspondência e, depois, como um usuário lê correspondências que lhe são enviadas.

ENVIANDO CORRESPONDÊNCIAS

Para enviar uma correspondência você deve digitar:

mail usuários<CR>

ou

mail -a<CR>

No primeiro caso, se você quiser enviar a mesma correspondência a vários usuários, você deve digitar a lista de todos eles (nomes separados por espaço); caso contrário, ou seja, se existir um só destinatário, você deve digitar unicamente o nome dele. É importante salientar que, tal como para o comando **write**, o nome de um usuário para o **mail** é o nome de uma conta no SOX (geralmente associado ao nome de seu titular).

No segundo caso, a correspondência é enviada a todos os usuários do sistema: este é o significado da opção **a** (do inglês **all**: todos).

Como dá para perceber, **mail** recebe a correspondência a ser enviada através da entrada-padrão, ou seja, seu terminal. No entanto, você pode redirecionar essa entrada para um arquivo qualquer (desejável para documentos longos ou mesmo curtos, desde que pouco modificáveis e constantemente reenviados). Se a entrada é o terminal, o fim da correspondência é indicado por <CTRL-D> ou por uma linha consistindo em um único ponto na sua primeira posição.

```
$ ed -p -<CR>
-a<CR>
                        Memorando No. 121/87<CR>
<CR>
Fonte: Setor de Documentação<CR>
Destino: Todos os Usuários da Biblioteca<CR>
<CR>
Srs. Usuários:<CR>
<CR>
    A partir de hoje, e por tempo indeterminado, fica<CR>
expressamente proibida a retirada de livros ou manuais da<CR>
Biblioteca. A medida é antipática mas se impõe, até o <CR>
pleno esclarecimento dos casos de livros e manuais <CR>
aparentemente desaparecidos do acervo. <CR>
<CR>
    Somos gratos pelo cumprimento desta diretriz. <CR>
<CR>
                    A Gerência do Setor de Documentação<CR>
<CR>
.<CR>
-w memo121<CR>
<número>
-q<CR>
$
```

```
$ mail -a < memo 121<CR>
$
```

Desta forma, o texto **memo121** é acrescido à caixa postal de cada usuário (à sua, inclusive). Todas as caixas postais ficam localizadas no diretório **/usr/mail** e recebem, cada uma, o nome do usuário.

```
$ mail andre maruzia estevao<CR>
Chopinho na próxima sexta-feira, <CR>
a partir das 18:00, no Chopp do Alemão. <CR>
.<CR>
$
```

Deste modo, o aviso será colocado nas caixas postais:

/usr/mail/andre
/usr/mail/maruzia
/usr/mail/estevao

Sua caixa postal é, portanto, **/usr/mail/voce**: para este arquivo continua valendo toda a política de permissão de acesso a arquivos do SOX. Você pode usar **ls –l** para verificar suas permissões e **chmod** para mudá-las, se achar necessário. Se houver algum problema consulte o administrador de sua instalação. Em todos os exemplos, estamos supondo as permissões normais a um arquivo deste tipo: sua caixa postal pode receber correspondências e somente você pode ler o seu conteúdo.

Em caso de engano quanto ao nome de um usuário, **mail** salva sua correspondência num arquivo especial chamado **dead.letter**, do seu diretório corrente. Assim, ao repetir o comando **mail**, basta redirecionar sua entrada para este arquivo.

```
$ mail estivao<CR>
        .
        .
        .
    mensagem
        .
        .
        .
<CTRL-D>
mail: usuário estivao não existe
Correspondência salva em dead.letter
$ mail estevao < dead.letter<CR>
$ rm dead.letter<CR>
$
```

RECEBENDO CORRESPONDÊNCIAS

Retorne à Figura 12.1 e recapitule parte do esquema de funcionamento do comando **mail**, agora do ponto de vista de **usuário2,** ou seja, como **usuário2** deve ser avisado pelo sistema da entrada de novas correspondências em sua caixa postal. Consideremos que **usuário2** é você. Se você estiver em um terminal, portanto, ligado ao sistema, sempre que chegar uma correspondência (a qual é depositada na sua caixa postal), você será avisado do fato ao término de algum comando do SOX que estiver sendo executado naquele momento. O aviso que surge na tela de seu terminal é:

Você Tem Correspondência

Caso você esteja fora do ar, no momento de entrar no ar receberá o mesmo aviso desde que haja alguma correspondência em sua caixa postal.

Vejamos, então, a estrutura de uma caixa postal: ela é uma **pilha** de correspondências (mensagens, digamos), como ilustra a Figura 12.2, ou seja, a caixa postal tem uma "abertura" superior por onde entram as mensagens, as quais vão sendo empilhadas – a conclusão é que a **última** mensagem recebida é a **primeira** a ser acessada (os americanos sintetizam bem esta característica de uma estrutura de informações tipo pilha dizendo que ela é **LIFO** (**L**ast **I**n, **F**irst **O**ut: O último que entra é o primeiro que sai)).

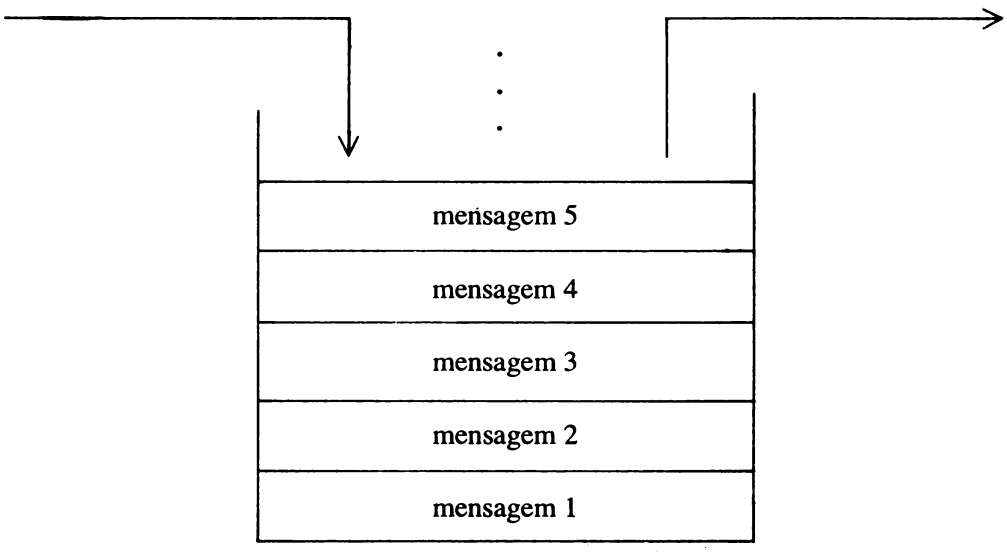


Figura 12.2 Uma Estrutura de Informações Tipo Pilha.

Tratemos da maneira como uma caixa postal é normalmente manuseada – é bom não esquecer que, "vista de fora", ela é um arquivo como qualquer outro, ligado ao diretório **/usr/mail** e que recebe o mesmo nome da conta do usuário (recordando, sua caixa postal é **/usr/mail/voce**). Uma mensagem é guardada em uma caixa postal precedida de um *cabeçalho* (que não faz parte da mensagem). O cabeçalho contém as indicações de quem enviou a mensagem, a data e a hora do envio.

Vamos então aprender a ler as mensagens existentes em sua caixa postal. A forma geral do **mail** para a operação de manuseio de uma caixa postal é:

mail [-rpqf] [arquivo]

Inicialmente, usemos **mail**, simplesmente. Depois, veremos os significados de cada uma das opções permitidas, com exemplos.

mail, sem argumentos, imprime a última mensagem que entrou na caixa postal (conforme o esquema da Figura 12.2, portanto) e, a seguir, exibe o caractere ? que indica que ele está à espera de alguma ação sua de como ele deve proceder dali por diante (por exemplo, você tem de decidir se a mensagem já lida deverá ou não permanecer na caixa postal).

Suponha que você recebeu três mensagens "mensagem1", "mensagem2" e "mensagem3", nesta ordem.

```
Você Tem Correspondência
$ mail<CR>
De Maruzia Seg Maio 25 1987 15:25:03

mensagem3
?
```

Se você responder ao ? com outro ?, **mail** lhe mostrará todas as escolhas possíveis para você indicar como ele deverá prosseguir o manuseio de sua caixa postal – portanto, a escolha de ? é um pedido de ajuda ao **mail.**

```
? ?<CR>
q                encerra
x                encerra sem alterar o conteúdo da caixa postal
p                imprime novamente a mensagem
s[arquivo]
w [arquivo]      salva a mensagem em arquivo ou mbox
_                mesmo anterior, sem cabeçalho
d                imprime a mensagem anterior
+                elimina (deleta) a mensagem
                 próxima mensagem (também <CR>)
m [usuários]
!                envia a mensagem para usuários
?                executa qualquer comando do SOX
?                sumário das escolhas (também*)
```

Vamos dar mais detalhes de suas escolhas possíveis. Atentemos para a diferença entre **q** e **x**: a última escolha encerra o comando **mail** mas, antes disso, cuida para que o conteúdo de sua caixa postal fique inalterado – se, por exemplo, durante o manuseio você chegou a eliminar as mensagens já lidas e se agora achar conveniente mantê-las na caixa postal, é só digitar **x**, que traz de volta à caixa postal aquelas mensagens retiradas durante a execução de **mail** e somente depois é que encerra a sua execução; já com a escolha **q** você simplesmente manda encerrar o **mail**, mantendo as providências que você tomou durante sua execução.

```
? d<CR>
De andre Seg Maio 25 1987 15:10:04

mensagem2
? d<CR>
De estevao Seg Maio 25 1987 14:45:19

mensagem1
? x<CR>
Você Tem Correspondência
$
```

```
$ mail<CR>
De maruzia Seg Maio 25 1987 15:25:03

mensagem3
? d<CR>
De andre Seg Maio 25 1987 15:10:04

mensagem2
? d<CR>
De estevao Seg Maio 25 1987 14:45:19

mensagem1
? q<CR>
Você Tem Correspondência
$
```

```
$ mail<CR>
De estevao Seg Maio 25 1987 14:45:19

mensagem1
? d<CR>
$
```

Observe o funcionamento da escolha **d**: ao fazê-lo, **mail** elimina da caixa postal a mensagem recém-exibida no terminal e apresenta a próxima mensagem da caixa postal – se não existir mais,. encerra. Suponha agora as entradas de "mensagem4", "mensagem5" e "mensagem6", nesta ordem. Vejamos o funcionamento da escolha – que, como vimos, imprime uma mensagem anterior.

```
Você Tem Correspondência
$ mail<CR>
De fulano ...
mensagem6
? <CR>
De sicrano ...
mensagem5
? +<CR>
De beltrano ...
mensagem4
? -<CR>
De sicrano ...
mensagem5
? -<CR>
De fulano ...
mensagem
? - <CR>
De fulano ...
mensagem6
? d<CR>
De sicrano ...
mensagem5
? p<CR>
De sicrano ...
mensagem5
? d<CR>
De beltrano ...
mensagem4
? d<CR>
$
```

A escolha –, como você pode perceber, faz com que o **mail** volte na pilha, uma mensagem de cada vez. Esta é a primeira "violação" da forma normal de atuar do **mail** e existe em seu benefício: é muito comum você esquecer de uma ação que deveria ter tomado sobre uma determinada mensagem "passada" e, assim, pode ser mais eficiente usar a escolha – do que sair do **mail** e depois voltar a ele, para que a mensagem volte a ser lida.

Você pode ordenar ao **mail** que vá para a próxima mensagem da pilha com +, ou simplesmente digitando a tecla <CR>. Esta também é uma ação colateral da escolha **d**, como vimos, e também das escolhas **s**, **w** e **m**, que serão vistas adiante.

A mesma mensagem pode ser reimpressa repetidas vezes com **p**. Agora vejamos as escolhas **s** e **w**.

A escolha **s** faz **mail** guardar a mensagem no arquivo especificado (caso você não diga o nome do arquivo, **mail** guarda a mensagem em um arquivo chamado **mbox**, do seu diretório-casa). **w** é idêntico a **s**, com a diferença de que os cabeçalhos (que, em verdade, não fazem parte das mensagens) não são gravados no arquivo. É importante saber que as mensagens são gravadas sempre no final do arquivo, ou seja, **mail** não destrói nada que porventura já exista no arquivo especificado.

Perceba a utilidade destes dois últimos comandos: uma caixa postal é um meio *temporário* de armazenamento de mensagens, as quais devem ficar lá somente até que você a "abra". Ao consultar sua caixa postal, todas as mensagens nela existentes devem ser ou jogadas fora (que é o que você faz com a escolha **d**), ou então guardadas em um outro lugar (que é o que você faz quando escolhe **s** ou **w** ou **m**. Esta última escolha será vista logo mais).

Voltemos ao exemplo do envio do memorando circular da seção anterior. Como foi usada a opção **-a, memo121** ficou duplicado na caixa postal do emissor, devendo ser retirado de lá. No entanto, pode ser interessante existir um arquivo de nome **circulares.mem**, por exemplo, para controle de todos os memorandos circulares expedidos em um certo período. A operacionalização disso poderia ser:

```
$ mail<CR>
     última mensagem
? d<CR>
     .
     .
     .
     memo121
?s circulares.mem<CR>
     .
     .
     .
$ rm memo121<CR>
$
```

Um lembrete importante: é possível que um operação de salvamento de uma mensagem com **s** ou **w** ou **m** falhe por você não ter permissão de gravação no diretório do arquivo ou no próprio arquivo. Também haverá problemas para a utilização do arquivo **mbox** se você não estiver em seu diretório-casa. Em vista disso, é sempre recomendável que você esteja em seu diretório-casa quando estiver lendo sua caixa postal.

A opção **m** permite que você encaminhe para outros usuários uma mensagem lida de sua caixa postal.

```
$ mail<CR>
      .
      .
      .
Por favor, após a leitura, dê conhecimento desta diretriz a seus subordinados.
? m marina alberto<CR>
      .
      .
      .
      próxima mensagem
      .
      .
      .
```

Finalmente, quando você estiver lendo sua caixa postal, poderá também executar qualquer outro comando do SOX, sem ter que sair do **mail**: basta responder ao ? com !, seguido do comando do SOX.

```
?!pwd<CR>
/usr/voce
!
?
```

Vamos parar um pouco para meditação. O que acabamos de ver foi um conjunto de escolhas que você pode fazer para orientar o **mail** sobre como ler sua caixa postal. O importante é não perder de vista que a escolha, para ser correta, pressupõe o entendimento da forma básica de proceder do **mail**, que é tal como apresentada na Figura 12.2, ou seja, as mensagens são normalmente lidas, uma a uma, da última que entrou na caixa postal até a primeira. Há, no entanto, uma opção do comando **mail** que reverte a maneira de proceder mostrada na Figura 12.2. Esta opção e mais algumas outras serão vistas na seção a seguir. Frisamos que todas elas se aplicam ao modo leitura do comando **mail.**

OPÇÕES PARA LEITURA DE MENSAGENS DO COMANDO MAIL

Relembrando a forma geral do comando **mail** para leitura de mensagens, existem as seguintes opções:

r
q
p
f arquivo

Observe que o uso da opção **f** implica que, junto com ela, você deve especificar o nome de um arquivo.

Vejamos, com algum detalhe, cada uma destas opções.

A opção **r** (do inglês **r**everse: reverta, inverta) simplesmente inverte o modo de leitura normal do **mail**: com esta opção, o **mail** retira as mensagens "pelos fundos" de sua caixa postal, de tal forma que a primeira mensagem que entrou é também a primeira a ser lida (os americanos chamam esse modo de leitura de **FIFO** (**F**irst **I**n **F**irst **O**ut: o primeiro que entra é o primeiro que sai)). Para esta forma de leitura valem também as escolhas vistas para o procedimento normal, com os mesmos significados (pense no funcionamento de **– exibe a mensagem anterior** e de **+ exibe a próxima mensagem** no contexto de **mail –r**).

A opção **q** (do inglês **q**uit: abandone as ações realizadas, traduzindo pelo seu significado) permite manter intacto o conteúdo de sua caixa postal em caso de interrupção na leitura das mensagens. Assim, no meio da execução do comando **mail**, você pode precisar interromper e poderá ser melhor voltar ao estado inicial de sua caixa postal (caso tenha havido remoção de mensagens dela) para que, posteriormente, você possa voltar a lê-la com mais vagar. Observe como a ação da opção **q** é idêntica à da escolha **x**.

A opção **p** (do inglês **p**rint: imprima) faz com que o **mail** leia mensagem a mensagem, sem interrupção (ou seja, com a opção **p**, o **mail** não exibe o ? após cada mensagem lida, esperando por uma escolha sua: em vez disso, ele as vai imprimindo, continuamente).

A opção **-f arquivo** (do inglês **f**ile: arquivo) permite que você leia, com **mail**, um arquivo criado a partir das escolhas **s** ou **w** (por exemplo, **mbox, circulares.mem** etc.).

```
$ mail - f circulares.mem<CR>
```

Se o arquivo tiver sido criado com **s** (portanto, incluindo os cabeçalhos das mensagens), o funcionamento do **mail** é tal e qual para sua caixa postal normal (**/usr/mail/voce**). Já se o arquivo for criado a partir de sua escolha **w** (portanto, sem incluir os cabeçalhos das mensagens), as mensagens são exibidas uma após outra, sem pausa. Você deve agora estar se perguntando: e se meu arquivo for sendo acrescentado, ora com **s**, ora com **w**? Neste caso, o funcionamento do **mail** poderá ser muito confuso!

Para encerrar esta discussão sobre o comando **mail**, o que dizer dele a respeito do terceiro pré-requisito necessário a um correio eletrônico, qual seja, a capacidade de transmitir mesmo documentos longos? A resposta a esta questão é que **mail** faz tal serviço, normalmente – a sugestão é a de transferir rapidamente para outros arquivos seus as mensagens longas de sua caixa postal e, a partir daí, consultá-los através dos comandos do SOX para manipulação de arquivos.

12.4 CALENDÁRIO E AGENDA ELETRÔNICOS

Apresentaremos agora dois comandos do SOX voltados para o controle de suas atividades no dia a dia. O primeiro deles imprime um calendário no seu terminal enquanto o outro lhe faz recordar compromissos assumidos ou coisas que você tem que fazer em certas datas.

IMPRIMINDO UM CALENDÁRIO COM O COMANDO CAL

O comando cal imprime um calendário. O seu formato geral é:

cal [–i] [mês[/mês]] [ano]

Um calendário é impresso para todo o ano especificado. Se um mês também é especificado, um calendário só para esse mês é impresso. O ano é um número entre 1 e 9999. O mês é um número entre 1 e 12. Também é permitido especificar uma faixa de meses (por exemplo "3/6"). Se nenhum argumento é usado, é exibido o calendário para o mês corrente. A opção **–i** indica que o calendário será exibido através da impressora.

```
$ cal 5 1987<CR>
      Maio 1987
 D  S  T  Q  Q  S  S
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
 $
```

Cuidado com o número do ano: você deve incluir todos os dígitos. Por exemplo, se você colocar somente "87", isto será interpretado como "o ano 87 da era cristã" e não "ano 1987" como você estaria pensando!

CONTROLE ELETRÔNICO DE SUA AGENDA

Imaginemos como você, organizado que é, cuidaria de se lembrar de seus compromissos ou de suas tarefas. Você anotaria todas as atividades na sua agenda manual, nas datas acertadas. No começo de cada dia, ou quando fosse necessário, você consultaria sua

agenda, numa determinada data, para ver que atividades estariam reservadas nessa data. No final do expediente, você também consultaria a agenda para ver o que não foi realizado (quando alguma providência deverá ser tomada) e para verificar as atividades do dia seguinte.

Pois bem, o SOX lhe torna disponível uma *agenda automática,* que funciona de maneira similar ao que idealizamos. Isto é possível com o comando **calendar**. Para que este funcione, você precisa registrar seus apontamentos em um arquivo de mesmo nome, **calendar**, através do editor de linhas **ed**, por exemplo. É necessário que você anote, para cada atividade, sua data, no formato mês/dia.

O formato geral do comando **calendar** é:

calendar [–]

Vejamos primeiro **calendar** sem a opção –. Neste caso, ele consulta o arquivo de mesmo nome, **calendar**, no seu diretório corrente, e imprime as linhas que contêm as datas de hoje e amanhã (as datas podem aparecer em qualquer ponto de uma linha). Se você usar somente números, as datas devem vir com o separador "/"; caso use nomes de mês, o separador é o espaço (você pode também dar vários espaços). O SOX aceita nomes de mês abreviados, desde que não haja ambigüidade. Exemplos de datas válidas:

Mai. 29
maio 29
5/29

Já "29 maio", "5-29", não são aceitas.

calendar não leva em conta sábados e domingos. Assim, se o dia de hoje é uma sexta-feira, a data de amanhã será a próxima segunda-feira (o próximo dia útil).

```
$ date<CR>
Sex Maio 29 1987   19:03:14
$ cat calendar<CR>
  5/28:     9:30      Reunião com Vendas
  5/29:     manhã     Telefonar para Faturamento sobre Fatura
  5/29:               No. 5622
  5/29:     10:30     Visita ao Depto. de Compras
  5/29:     14:30     Reunião com o Gerente do CPD
  5/29:     17:00     Audiência com o Diretor-Presidente para
  5/29:               tratar da situação econômica do país
  6/1:      14:00     Almoço com Pessoal de Vendas na
  6/1:                Cantina do Manoel
   6/2      10:00     Viagem a São Paulo
  $
```

```
$ calendar<CR>
  5/29:     manhã     Telefonar para Faturamento sobre Fatura
  5/29:               No. 5622
  5/29:     10:30     Visita ao Depto. de Compras
  5/29:     14:30     Reunião com o Gerente do CPD
  5/29:     17:00     Audiência com o Diretor-Presidente para
  5/29:               tratar da situação econômica do país
  6/1:      14:00     Almoço com Pessoal de Vendas na
  6/1:                Cantina do Manoel
$
```

A forma de proceder do comando **calendar** é procurar seqüencialmente, através do arquivo **calendar**, aquelas linhas que contêm ou a data de hoje ou a data de amanhã. Tais linhas são exibidas, sem qualquer alteração, no terminal. Assim, se a descrição de um evento necessitar de mais de uma linha, é necessário repetir a data em cada uma delas, pois, caso contrário, elas não serão exibidas. Verifique o conteúdo do arquivo **calendar**: foi levada em conta tal observação.

Não é demais repetir que o arquivo **calendar** deve ser inteiramente mantido por você. Assim, muito provavelmente, você terá que atualizá-lo diariamente, acrescentando novos eventos, retirando antigos eventos ou modificando datas de eventos já constantes do arquivo. O editor de linhas **ed**, ou um outro processador de textos, estão aí para isto mesmo.

É possível automatizar a execução do comando **calendar**, ou seja, fazer o SOX disparar este comando num momento previamente especificado. Por exemplo, um tipo de política poderia ser: cada vez que você entrasse no ar você seria lembrado de suas atividades previstas para hoje e amanhã – para isto, o comando **calendar** deveria ser colocado num arquivo especial chamado de **.profile** (maiores detalhes sobre este arquivo no capítulo anterior), com seu arquivo **calendar** colocado no seu diretório-casa.

Agora vejamos como funcionaria o comando **calendar** em presença do argumento–. Quando isto acontecesse, numa hora combinada (geralmente ao alvorecer do dia), o SOX automaticamente executaria o comando **calendar** e todas as atividades dentro das datas pesquisadas (hoje e amanhã) lhe seriam enviadas, via comando **mail**, para sua caixa postal. O acionamento de **calendar** num tempo qualquer especificado é feito com o auxílio de um comando do SOX chamado de **cron** (do inglês **cronology**: tempo, cronologia). Consulte o administrador de sua instalação para a execução desse serviço.

Apresentamos as *Soluções-SOX* para a automação de seu escritório. Esperamos que este capítulo possa ter tido o mérito de motivá-lo a pensar (se ainda não o fez) neste assunto tão palpitante e ainda tão polêmico.

CAPÍTULO **13**

TRABALHANDO COM A IMPRESSORA

Um sistema SOX sempre possui uma ou mais impressoras para a impressão de arquivos e relatórios. Este capítulo discute as várias formas que você, usuário, tem de conversar com o SOX sobre o assunto "impressão".

Vamos primeiro cuidar de alguns conceitos iniciais. O SOX é um sistema multiusuário e, como tal, permite que os diferentes usuários que estão "no ar" compartilhem dos recursos da máquina (UCP, memória, disco etc.). Alguns recursos de máquina, porém, devem ser compartilhados com muito cuidado, devido à sua natureza. Tal é o caso de uma impressora. Se cada usuário pudesse acessar a impressora *diretamente*, teríamos um problema, não acha? Poderíamos mandar imprimir um relatório na impressora enquanto ela já estivesse imprimindo *seu* relatório. O resultado só se tornaria legível após um trabalhão de algumas horas com tesoura e cola! Resultado: num sistema multiusuário, precisamos ter acesso às impressoras de forma controlada, e esse controle é feito através do conceito de *spool de impressão*. O conceito de spool permite que *pedidos de impressão* sejam feitos. Esses pedidos são enfileirados por um programa especial e impressos, um por um, numa impressora. O presente capítulo, portanto, descreve o pacote de spool do SOX, do ponto de vista do usuário.

13.1 IMPRESSÃO DE ARQUIVOS - O COMANDO LPR

O comando do SOX que enfileira pedidos de impressão se chama **lpr** (do inglês **l**ine **p**rinter: impressora de linha). Uma vez enfileirados, os pedidos são impressos por um *servidor de impressão* (chamado de **lpd**, do inglês **l**ine **p**rinter **d**aemon). Neste capítulo, o servidor de impressão não será objeto de enfoque: basta se lembrar de que ele existe para imprimir arquivos; o comando **lpr** apenas enfileira pedidos para o servidor.

Voltando ao comando de impressão **lpr**, a forma mais simples de imprimir um arquivo chamado, digamos, **contas** seria a seguinte:

```
$ lpr contas<CR>
$
```

O comando **lpr** pode ainda ser chamado para imprimir mais de um arquivo de cada vez; isto é feito da seguinte forma:

```
$ lpr fatura1 fatura2 fatura3<CR>
$
```

Os três arquivos serão impressos, um por vez. Imagine agora a seguinte situação. Você vai rodar a folha de pagamento com seu programa **folha** e gostaria que o resultado saísse na impressora. Supondo ainda que o programa **folha** imprima os resultados na sua saída-padrão, você poderia agir da seguinte maneira:

```
$ folha >resultado<CR>
$ lpr resultado<CR>
$ rm resultado<CR>
$
```

Observe o que fizemos. O programa de folha de pagamento **folha** foi executado e, usando os recursos de redirecionamento do Shell, jogou os resultados num arquivo chamado **resultado**; este arquivo foi enfileirado para impressão e removido. Podemos, contudo, lançar mão de um outro recurso do Shell – um duto – para completar a operação de forma mais direta e transparente. Observe:

```
$ folha | lpr<CR>
$
```

A folha de pagamento foi executada e seu resultado foi enviado diretamente para o programa enfileirador **lpr** através de um duto. Muito mais simples, não é?

13.2 AS OPÇÕES DO COMANDO LPR

Vamos agora discutir as várias formas possíveis de pedir a impressão de um arquivo. Por exemplo, sua instalação pode, talvez, possuir mais de uma impressora. Como informar em que impressora deverá sair a impressão? Para este fim, o SOX define o conceito de uma *impressora lógica*. Uma impressora lógica possui um nome, por exemplo, il000. O nome de uma impressora lógica sempre contém as letras il, seguidas de três números. Quem sabe a que impressora física corresponde cada impressora lógica é o administrador do sistema. Fale com ele para descobrir. Também, não se assuste com a palavra *lógica*. Uma impressora lógica nada mais é que um nome dado a uma impressora para você não ter que aprender os nomes que o SOX dá às impressoras físicas. Muito bem. Com o comando **lpr**, você pode pedir que sua impressão saia na impressora que quiser. Por exemplo, se você sabe que a impressora lógica **il000** corresponde à impressora de linha e que aquela chamada **il001** é uma impressora de qualidade de carta, você poderia pedir uma impressão de boa qualidade com o seguinte comando:

```
$ lpr -i il001 arquivo <CR>
$
```

Se você, como nos casos anteriores, não especificar a impressora na qual a listagem deve sair, uma impressora particular será automaticamente escolhida pelo comando **lpr**: essa é a impressora **default** (por omissão), por exemplo, **il000**. Para descobrir qual é a impressora default, execute o comando **lpl –d** – falaremos sobre este comando adiante.

Introduzamos agora o conceito de *formulário*. Cada impressora do sistema está, num determinado momento, processando um tipo de formulário. Exemplos de formulários seriam o formulário-padrão branco, o formulário menos largo para cartas e correspondências, o formulário timbrado da empresa, o formulário de etiquetas, o formulário de contracheques etc. O administrador de seu sistema atribui um número de três dígitos a cada formulário da empresa. Fale com ele para descobrir o que significam. Da mesma forma que você pode escolher uma impressora, pode especificar o formulário no qual quer imprimir seus arquivos. Por exemplo, o seguinte pedido de impressão deve sair no formulário 003:

```
$ lpr –f 003 arquivo<CR>
$
```

Tudo bem, mas a impressora pode não estar processando o formulário 003. E aí? O que ocorre é simples: seu pedido deverá esperar até que o administrador mude o formulário da impressora para aquele chamado 003. Assim que isso for feito, seu pedido finalmente será impresso. Se você não especificar o formulário desejado, o comando **lpr** assumirá que você quer o formulário que está sendo processado atualmente pela impressora.

Para pedir a impressão de mais de uma cópia de algum arquivo, use a opção **–c** do comando **lpr:**

```
$ lpr –c 3 balancete<CR>
$
```

No exemplo anterior, três cópias do arquivo **balancete** serão impressas.

Finalmente, a impressão de um pedido pode incluir ou não uma *folha de rosto*. Normalmente, essa folha não é impressa, para poupar papel. Para pedir a impressão de uma folha de rosto, use a opção **–h** (do inglês **h**eader: cabeçalho):

```
$ lpr –h arquivo<CR>
$
```

A folha de rosto fornece a data, o total de linhas impressas e outras informações úteis.

Como última observação, lembre-se de que o comando **lpr** enfileira uma *cópia* dos seus arquivos a serem impressos. Você pode removê-los após o comando **lpr**, mesmo que a impressão ainda não tenha sido efetuada.

13.3 OBTENDO INFORMAÇÕES SOBRE O SPOOL – O COMANDO lpl

O comando **lpl** permite que você obtenha informação sobre as impressoras do seu sistema e sobre os pedidos de impressão enfileirados. Mais especificamente, você pode obter informação sobre a impressão que está sendo feita correntemente (opção **–a**), sobre as características das impressoras (opção **–p**), sobre os pedidos enfileirados (opção **–o**), sobre a impressora default (opção **–d**) e sobre tudo ao mesmo tempo (opção **–t**). Vamos examinar as opções uma a uma:

Opção **–a**: Se você chamar o comando **lpl** com esta opção, as seguintes informações serão fornecidas:

```
$ lpl –a <CR>
impressora      temporário      arquivo     dono     grupo
i1000           lpr001207       contas      37       2
i1001           lpr001242       faturas      5       1
```

Como você pode observar, algumas informações sobre os pedidos correntemente sendo impressos são fornecidos, uma impressora a cada linha. Abaixo do título *impressora* vem o nome da impressora lógica. Sob o título *temporário* vem uma *identificação de pedido* que pode ser usada para controlar o pedido, conforme veremos adiante. O pedido que está sendo atualmente impresso na impressora **i1000** é aquele chamado **lpr001207.** O arquivo correspondente a este pedido chama-se *contas*. Este é o arquivo que foi submetido para impressão por um usuário. A identificação deste usuário é 37 e ele pertence ao grupo número 2. Uma informação semelhante foi fornecida para a impressora lógica **i1001.** Observe também que se você quiser informação apenas sobre uma única impressora, poderá fornecer seu nome:

lpl –a i1001 <CR>

Opção –d: Esta opção do comando **lpl** pode ser usada para descobrir qual é a impressora default, e quais são suas características. Ao chamar o comando, você recebe as seguintes informações:

```
$ lpl –d <CR>
impressora  dispositivo  formulário  estado  hora      data
i1000       /dev/lp      000         i       10:33.09  12/06/87
```

Estas informações dizem que a impressora default é **i1000**, está processando o formulário **000** e está correntemente imprimindo (estado **i**). O estado também pode indicar **e**, quando a impressora está esperando trabalho para fazer. A informação sob *dispositivo* indica a impressora física e interessa apenas ao administrador do sistema. Lembre-se de que se você quiser imprimir na impressora default (normalmente a impressora principal do seu sistema), você não terá que usar a opção **–i** do comando **lpr.**

Opção **–p**: Esta opção fornece informação idêntica à mostrada para a opção **–d**, mas para as impressoras que você escolher. Por exemplo:

```
$ lpl -p il000 il001 <CR>
impressora  dispositivo  formulário  estado  hora      data
i1000       /dev/lp      000          i       08:01.35  30/08/87
i1001       /dev/tty1    000          e       08:01:38  30/08/87
```

Opção **-o**: Esta opção é usada para examinar tudo o que está enfileirado (esperando impressão) para uma ou mais impressoras. Vamos examinar a saída do comando **lpl**:

```
$ lpl -o il000
impressora: i1000

pr  arquivo  dono  temporário  f    cop  lin    es  hora      data
50  contas   37    lpr001247   000  002  00245  c   12:12:12  12/08/87
51  fat       2    lpr001250   001  001  01089  .   12:12:39  12/08/87
52  fat2      2    lpr001252   001  001  01074  .   12:12:53  12/08/87
```

Cada linha da saída descreve um pedido enfileirado na impressora **il000.** Os títulos *arquivo, dono, temporário* e *f* (formulário) já foram mencionados anteriormente. *hora* e *data* especificam quando o pedido foi submetido. *cop* diz quantas cópias serão impressas. Cada cópia tem um tamanho (em linhas) dado sob o título **lin. Pr** especifica a prioridade dos pedidos. O de número menor será impresso primeiro. Finalmente, o título *es* (estado) pode ser *c,* se o pedido estiver sendo correntemente impresso, ou ".", se o pedido estiver aguardando impressão. Observe que se você imprimir dois ou mais arquivos usando um único comando **lpl**, cada arquivo será considerado como um pedido individual. Isto é, para cada arquivo aparecerá uma linha na saída do comando **lpl -o.**

Opção **-t**: Esta opção imprime tudo, isto é, toda a informação dada pelas opções **-p** e **-o.**

13.4 CONTROLANDO OS PEDIDOS - O COMANDO LPA

Uma vez submetido à impressão, um pedido não está fora do seu alcance. Várias *ações* ainda podem ser aplicadas aos pedidos enfileirados. Essas ações poderiam ser, por exemplo: remover um pedido enfileirado, quando você não quiser mais a impressão, e outras coisas do gênero, que discutiremos oportunamente.

Para aplicar uma ação a um pedido, precisamos primeiro saber como *identificar* um pedido. Já abordamos este assunto brevemente numa seção anterior. Para ver quais são os pedidos enfileirados na área de spool, use o comando **lpl –o**. Na informação fornecida por este comando, você pode observar uma coluna chamada *temporário*, com itens do tipo **lpr001242.** Esta informação identifica cada pedido e é através dela que você pode controlar os seus pedidos. Salientamos que você pode controlar apenas seus pedidos. Não vale remover os pedidos dos colegas para que os seus passem na frente.

A primeira ação, e talvez a mais importante, que você pode aplicar a um pedido seu é removê-lo. Isto é feito da seguinte forma. Primeiro, execute o comando **lpl –o** para descobrir a identificação do pedido que você quer remover: digamos que ele seja **lpr001242** (lembre-se: as demais informações *arquivo, dono, data, hora* etc., ajudam-no a identificar seu pedido). A remoção é feita assim:

```
$ lpa –ci lpr001242 <CR>
$
```

A opção **–ci** significa "**c**ancele **i**mpressão". Observe também que este pedido pode estar sendo impresso atualmente ou pode estar esperando em fila: tanto faz, ele será removido.

Uma segunda situação seria aquela em que você não quer remover um pedido já feito mas quer *prendê-lo* temporariamente para inibir sua impressão. A impressão de um relatório muito grande poderia ser presa até desafogar a máquina, por exemplo. O pedido seria então liberado para impressão, talvez à noite. A prisão de um pedido é feita com a opção **–pi** ("**p**renda **i**mpressão") do comando **lpa**:

```
$ lpa -pi lpr001242 <CR>
$
```

A liberação do pedido usa a opão **-si** ("solte impressão"):

```
$ lpa -si lpr001242 <CR>
$
```

A partir deste comando, o pedido **lpr001242** volta a ser considerado para impressão.

Para mudar o número de cópias associadas a um pedido já feito, o comando **lpa** dispõe da opção **-co** ("cópias"). Por exemplo, se você tivesse pedido duas cópias de um relatório e decidisse que realmente queria três, você usaria:

```
$ lpa -co 3 lpr001242 <CR>
$
```

Da mesma forma, o formulário no qual um pedido deve ser impresso pode ser mudado com a opção **-mf** ("**m**ude formulário"):

```
$ lpa -mf 002 lpr001242 <CR>
$
```

Vamos passar agora ao controle de paginação possível através do spool. Os comandos que veremos a seguir permitem que você controle exatamente o que vai ser impresso para seu pedido. Você pode, por exemplo, pedir a impressão a partir da quarta página de um arquivo; pode pedir para a impressão retroceder algumas páginas, caso o papel da impressora tenha rasgado, e assim por diante. Antes de passarmos a este assunto, contudo, precisamos falar um pouco sobre o *tipo* de um arquivo enfileirado para impressão. Os arquivos que você imprime podem ser orientados por linha ou por página. Um arquivo orientado por página contém caracteres de alimentação de formulário (em inglês – "formfeed") para separar páginas, enquanto arquivos orientados por linha não contêm tais caracteres. Quando você chama o comando **lpr** sem especificar o tipo do arquivo, este supõe que o arquivo tem orientação por linha. Para dizer que o arquivo tem orientação por página, chame o comando **lpr** como segue:

```
$ lpr -t 1 arquivo <CR>
$
```

A opção **-t 1** significa "tipo 1" ou "tipo página".

Para um arquivo orientado por linha, você pode controlar a linha exata onde a impressão deve começar (ou recomeçar, se o pedido estiver sendo impresso). Por exemplo:

```
$ lpa -al  400   lpr001242<CR>
$ lpa -al  -20   lpr001242<CR>
$ lpa -al  +100  lpr001242<CR>
```

A opção **-al** significa "avance (ou posicione) linha". No primeiro exemplo dado, a impressão do pedido **lpr001242** começará (ou recomeçará) na linha 400 do arquivo. No segundo exemplo, a impressão voltará 20 linhas para trás. Finalmente, no terceiro exemplo, a impressão pulará 100 linhas para a frente.

No caso de arquivos orientados por página, basta substituir a opção **–al** por **–ap**. O número se refere então a páginas.

A última opção do comando **lpa** permite reiniciar a impressão na página inicial. Veja:

```
$ lpa –ri lpr001242<CR>
$
```

Observe que **–ri** significa "**r**einicie **i**mpressão".

APÊNDICE A

ÍNDICE ALFABÉTICO DOS COMANDOS DO SOX

APÊNDICE B

ÍNDICE ANALÍTICO

Composição e Arte-Final:
JAG Composição Editorial e Artes Gráficas Ltda.
Praça F. Roosevelt, 208 - 8º andar
Tel. (011) 255-5694 - São Paulo

Impressão e Acabamento
GRÁFICA E EDITORA FCA
AV. HUMBERTO DE ALENCAR CASTELO BRANCO, 3972 - TEL.: 419-0200
SÃO BERNARDO DO CAMPO - CEP 09700 - SP

0-07-450422-3

SAMPAIO/SAUVÉ/MOURA

SOX Conceitos Básicos