MANUAL
DE INTRODUÇÃO AO SOX

SX0588-01.0

MANUAL
DE INTRODUÇÃO AO SOX

1º Edição

setembro /88

SX0588-01.0

Este manual é uma publicação da Divisão de Testes e Integração de Sistemas / Setor de Documentação, aplicável a produtos comercializados pela Cobra

O Fabricante reserva-se o direito de alterar a qualquer tempo, sem notificação prévia, as características e especificações técnicas do produto retratado nesta documentação, bem como promover, sempre que necessário, alterações e correções na forma e conteúdo desta publicação

Av. Comandante Guaranys, 447 - Jacarepaguá

22785 Rio de Janeiro RJ BRASIL

REGISTRO DE ATUALIZAÇÃO

Manual	Emissão
Introdução ao SOX	setembro/88

Esta página é uma orientação ao usuário quanto à identificação e aplicação específica deste manual, sua estrutura e o histórico das atualizações aplicáveis ao mesmo. Mantenha esta folha anexa ao início do manual, para controle de sua atualização.

APLICAÇÃO DO MANUAL:

Equipamento: Linha X

Sistema: SOX

Produto: SOX

CONTROLE DE PÁGINAS DO MANUAL:

Cap.	1 -	10	Cap.	4 -	48	Gloss.	-	4
Cap.	2 -	95	Cap.	5 -	16			
Cap.	3 -	35	Anexo	A -	5			

HISTÓRICO DAS ATUALIZAÇÕES:

1 - Esta documentação encontra-se em sua primeira edição.

SISTEMA DE ATUALIZAÇÃO DE DOCUMENTAÇÃO:

Objetivando manter o usuário sempre a par das constantes evoluções de seus produtos, a COBRA fornece o Sistema de Atualização de Documentação (SAD), que consiste em um conjunto de arquivos reunidos no disco flexível **altera-doc**, fornecido com o sistema operacional

SOX. Os arquivos contêm as informações necessárias para a atualização dos manuais (enquanto ainda não-atualizados ou reeditados) e pertencem ao diretório **/man/altera.** No diretório existe um arquivo disponível para cada manual referente aos produtos X-10/SOX, e outro contendo informações a respeito de todos os manuais (arquivo **indice**).

Os arquivos que descrevem as alterações de cada manual são formados por blocos de informações, cada um dos quais contém as alterações liberadas numa mesma data e é dividido em duas partes: a primeira, com informações gerais e um sumário das alterações aplicáveis ao manual, e a segunda, com as alterações propriamente ditas, precedidas de informações que possibilitam sua localização no manual.

O arquivo **indice** permite obter, a partir do título de cada manual, o nome do arquivo associado (no qual se encontram as atualizações, caso existam), o código do manual (que reflete a edição e versão correntes) e os números e respectivas datas de todas as atualizações efetuadas através do SAD. O campo **ALTERACOES** indica o número e a data de cada bloco de informações liberadas e, se estiver em branco, significa que o manual, até a presente data, não sofreu alteração alguma. Caso haja alteração em um manual, o nome do arquivo contendo as alterações correspondentes ao mesmo pode ser obtido através do campo **ARQUIVO.**

Para se tomar conhecimento dos arquivos acima referenciados, utilizar o comando **am**, descrito no "Manual de Referência - Comandos/SOX".

Este manual destina-se a usuários que não conheçam o sistema operacional SOX e desejam usufruir das facilidades oferecidas por este sistema.

O manual é composto por cinco capítulos, um anexo e um glossário.

No capítulo 1 é ensinado ao usuário como se conectar e desconectar do SOX, o que é um terminal, o que é uma sessão e como criar ou modificar sua senha.

O capítulo 2 instrui o usuário quanto a manipulação de arquivos e diretórios, apresentando a estrutura de arquivos. É mostrada também a utilização dos comandos usados para a criação de diretórios, listagem de arquivos, mudança de diretórios, remoção de arquivos e mudança de permissões.

No capítulo 3 são apresentados os conceitos sobre entrada e saída-padrão, redirecionamento de entrada e saída-padrão, canalizações, execução de comandos em plano secundário e execução de arquivos de comandos.

No capítulo 4 é apresentada uma classificação funcional dos comandos do SOX e uma descrição detalhada de alguns comandos.

O capítulo 5 contém uma descrição da estrutura do SOX.

No Anexo A são apresentadas as mensagens de erro ou advertência que podem ser exibidas durante a etapa de identificação junto ao SOX.

Finalmente, o glossário apresenta a definição de termos pouco usuais utilizados neste manual.

Uma recomendação, que para nós é de grande importância, é que o cartão de cadastro e comentários de usuário, anexo ao final do manual, seja preenchido e remetido à COBRA. Esta realimentação é que nos permite o constante aprimoramento da documentação dos nossos produtos. Além disso, nos permitirá cadastrar nossos usuários, habilitando-os a receber boletins técnicos, informes de novos lançamentos e atualizações deste manual.

CAPÍTULO 1

BEM-VINDO AO SOX

1.1 SEU TERMINAL: INSTRUMENTO DE INTERAÇÃO COM O SOX

Na sua interação com o SOX, você utilizará um terminal. O terminal é formado de duas partes:

- Teclado;
- Monitor de vídeo.

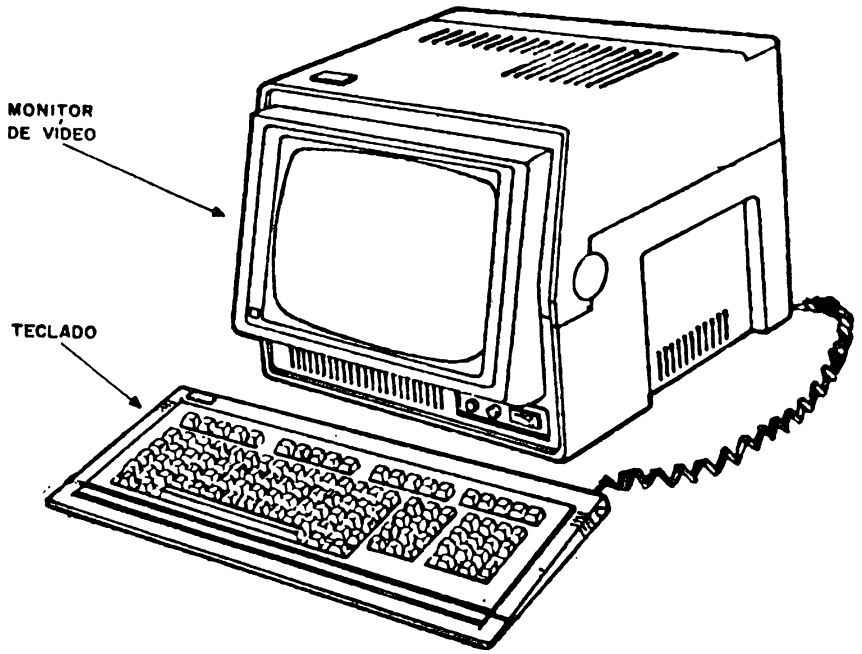


Figura 1-1

Vejamos qual a função de cada uma dessas partes.

. Teclado

É através do teclado que você comunica ao SOX o que deseja que ele faça. Você digita no teclado caracteres que serão recebidos pelo SOX, reproduzidos na tela do monitor de vídeo e agrupados de tal forma que sejam interpretados como pedidos de serviço ao SOX ou como informações que você necessita fornecer para a execução de seus pedidos.

O SOX permite que você interaja com ele através de vários tipos de teclado.

Durante a sua interação com o SOX, você precisará comunicar a ele que terminou de digitar o seu pedido de serviço. Para isto, você deve digitar uma tecla, que depende do tipo de teclado que esteja sendo utilizado. Como convenção adotaremos o símbolo <**CR**> ou **CR** para representar esta tecla.

As teclas que correspondem ao símbolo <**CR**>nos teclados dos terminais Cobra TI 200, TI 300 e TI 300-PC, são, respectivamente, **RET**, ↵ e **Enter**.

Outra função que pode ser obtida através do teclado é a interrupção do fornecimento de dados durante a execução de um serviço. Nos teclados com os quais o SOX pode ser utilizado, esta função é exercida pelas combinações de teclas **ALT D** ou **CRTL D**. Verifique o seu teclado. Para efeito de padronização, as teclas que exercem esta função serão representadas neste manual pelos símbolos <**ALT**><**D**> ou **ALT D**.

Outra convenção utilizada neste manual é que, nos exemplos, todo texto que deve ser digitado aparecerá em negrito. Assim, no exemplo:

```
$ cat   <CR>
EXEMPLO
EXEMPLO
<ALT><D>
```

devemos digitar cat, a tecla correspondente ao símbolo **CR**, EXEMPLO, e as teclas correspondentes a **ALT D**.

. Monitor de vídeo

Exibe os caracteres digitados por você no teclado, além de exibir freqüentemente os resultados dos pedidos de serviço ao SOX.

Um componente fundamental na sua interação com o SOX através do monitor de vídeo é o **cursor**. Ele é um objeto que está quase sempre na tela do monitor de vídeo, e indica a posição da tela onde será exibido o próximo caractere. Ele será representado neste manual pelo caractere "_". Assim, na tela:

```
$ cat      <CR>
EXEMPLO
EXEMPLO
<ALT><D>
$ _
```

o símbolo "_"indica a posição da tela onde será exibido o próximo caractere.

1.2 IDENTIFICANDO-SE AO SOX

Para que você possa se conectar com o SOX e usufruir das suas facilidades, é preciso que ele o reconheça como um usuário válido. Isto significa que ele deve ter alguma informação que o identifique. Esta informação permite que:

- O SOX possa identificar você como seu usuário;
- Usuários que não tenham autorização para utilizar o sistema não possam se conectar a ele.

Para que o SOX possa ter essa informação, é preciso que você a forneça a alguém que esteja autorizado a recebê-la e a cadastrá-lo junto ao SOX como um novo usuário. Se no sistema em que você está trabalhando existir alguém responsável pela administração, gerência ou operação do sistema, então procure esta pessoa para que ela possa cadastrá-lo junto ao SOX. Se não existe alguém com estas atribuições, então consulte o "Manual do Administrador do Sistema SOX", seção 4.1 ("Cadastramento de Novos Usuários").

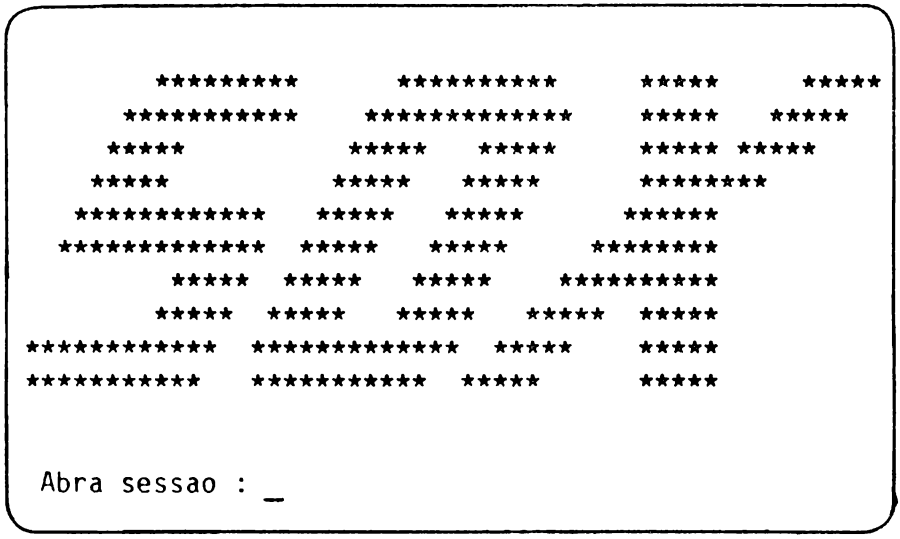
O SOX identifica você por um nome. Este nome é a sua **identificação de usuário**, ou simplesmente **identificação**. Esta é a informação que você deve fornecer para se cadastrar junto ao SOX. É por este nome que você é identificado por ele.

Você poderá também ter associada à sua identificação uma **senha**, que é um nome (de preferência diferente da sua identificação) que você deve manter em absoluto sigilo. Esta senha é uma garantia (nunca total) de que, mesmo que outros usuários conheçam a sua identificação, eles não possam utilizá-la para se conectar ao SOX. Você pode ou não fornecer uma senha ao se cadastrar. Você pode mudar sua senha ou criar uma durante a sua interação com o SOX, como veremos adiante.

1.3 COMO SE CONECTAR AO SOX

Para a discussão que se segue, supomos que o terminal que você está utilizando já está conectado ao SOX. Isto significa que na tela do seu terminal você deve estar vendo um logotipo (se houver algum logotipo) de sua empresa, universidade, ou do SOX, e uma mensagem inicial que você deve responder para poder começar a interagir com o SOX.

Uma tela típica é:



Note que a mensagem inicial vem seguida do cursor.

Para responder à mensagem inicial, você deve digitar no teclado do seu terminal a sua identificação de usuário, seguida de **CR**.

Se você teve algum problema durante a etapa de identificação junto ao SOX, veja o anexo A.

Se você tiver uma senha associada à sua identificação de usuário, o SOX exibirá uma mensagem solicitando que você digite sua senha. Uma mensagem típica é:

```
Senha :
```

Você deve então digitar sua senha seguida de **CR**. Não se preocupe com o fato de que os caracteres que compõem a sua senha não aparecerem na tela do seu terminal. Isto ocorre para que outros usuários não possam tomar conhecimento de sua senha e usá-la indevidamente.

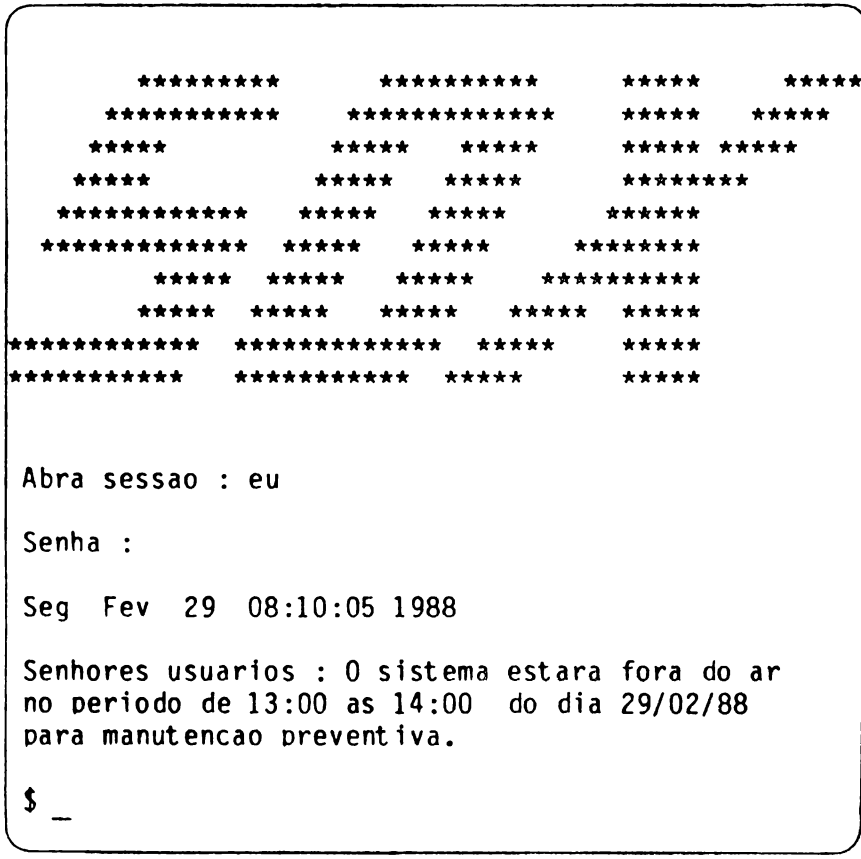
Uma vez terminada com sucesso a etapa de identificação junto ao SOX, são exibidas mensagens (quando houver mensagens) com informações diversas, tais como data e hora corrente, avisos de períodos de indisponibilidade do sistema para manutenção preventiva, etc. Algumas mensagens típicas são:

```
Seg  Fev  29  08:10:05 1988

Senhores usuarios : O sistema estara fora do ar
no periodo de 13:00 as 14:00  do dia 29/02/88
para manutencao preventiva.
```

Terminadas as mensagens, o SOX exibirá um **pronto**, que é um caractere (em geral $) ou conjunto de caracteres, seguido do cursor.

Supondo que um usuário com identificação **eu** tenha aberto sessão, teríamos a tela:



Isto significa que o SOX está pronto para interagir com você, e está esperando suas ordens. O que ocorrerá daqui em diante na sua interação com o SOX, até o momento em que você se desconectar, é chamado de **sessão de trabalho** ou simplesmente **sessão.**

1.4 O QUE OCORRE DURANTE UMA SESSÃO ?

Uma sessão consiste basicamente em um diálogo entre você e o SOX. O SOX comunica a você que está pronto para dialogar exibindo o pronto. Você digita então um pedido de serviço ao SOX. Pedidos de serviço passarão a ser chamados de **comandos**. Se o comando que você digitou for válido, então ele será executado. Caso contrário, será produzida uma mensagem de erro. Em ambos os casos, seja após a execução do comando ou após a produção da mensagem de erro, o SOX exibirá o pronto para reiniciar o diálogo. Este ciclo continua até que você se desconecte do SOX.

1.5 COMO SE DESCONECTAR DO SOX

Existem duas maneiras de você, após o pronto, se desconectar do SOX. A primeira consiste em digitar o comando **exit** seguido de **CR**. A outra é digitando após o pronto a seqüência **ALT D**.

1.6 COMO CRIAR OU MODIFICAR A SUA SENHA

Você pode criar ou modificar a sua senha através do comando passwd.

Para utilizar este comando, digite passwd no seu teclado, seguido de **CR**. Supondo que a sua identificação seja eu, teremos então na tela:

```
$ passwd    <CR>
mudando a passwd de eu
```

Se você está criando uma senha, isto é, se você não tem senha e quer ter uma, aparecerá na tela a mensagem

```
nova_passwd:
```

Você deve então digitar a senha desejada, seguida de **CR.** A senha não aparecerá na tela.

Em seguida, aparecerá a mensagem

```
confirme passwd:
```

Você deve, então, digitar a senha seguida de **CR.**

Se você já tem uma senha e deseja modificá-la, a mensagem que aparecerá na tela será

```
antiga passwd:
```

Você deve então digitar a senha que deseja modificar, seguida de **CR.** A senha não aparecerá na tela. Em seguida, aparecerá na tela a mensagem

```
nova_passwd:
```

Você deve então digitar a nova senha seguida de **CR.** Em seguida, aparecerá na tela a mensagem

```
confime nova_passwd:
```

Você deve então redigitar a nova senha, seguida de **CR.**

CAPÍTULO 2
SISTEMA DE ARQUIVOS

Um dos pontos fortes do SOX é a maneira pela qual o sistema organiza e mantém as informações para os seus usuários, ou seja, é o seu sistema de arquivos.

O objetivo deste capítulo é descrever o sistema de arquivos do SOX. Nele serão apresentadas, entre outras coisas, a estrutura hierárquica do sistema de arquivos do SOX, os tipos de arquivos existentes, as permissões de acesso e associações.

2.1 ARQUIVOS

Intuitivamente falando, um arquivo é um recipiente no qual podem ser armazenados dados.

Cada arquivo tem um nome, pelo qual pode ser referenciado. Através deste nome é possível, desde que se tenha as permissões de acesso apropriadas, manipular todas as informações nele contidas.

Arquivos geralmente são mantidos em dispositivos de memória secundária (discos, disquetes ou fitas). Para a manipulação do conteúdo de um arquivo, é necessário que os dados sejam transferidos da memória secundária para a memória principal, tendo-se, desta forma, a informação disponível em memória principal.

Na manipulação de um arquivo são realizadas determinadas operações, tais como:

- **Criar**

  Significa criar a infra-estrutura necessária para que o arquivo possa ser utilizado. Entre outras coisas, o usuário deve atribuir um nome ao arquivo a ser criado.

- **Carregar** (ou ler)

  Significa trazer o conteúdo (ou parte dele) do arquivo referenciado (através de seu nome) da memória secundária para a memória principal.

- **Editar**

  Significa ampliar, alterar ou remover o conteúdo (ou parte dele) do arquivo referenciado.

- **Gravar** (ou salvar)

  Significa guardar o conteúdo de um arquivo em memória secundária.

2.2 TIPOS DE ARQUIVOS DO SOX

Basicamente, três tipos de arquivos são tratados pelo SOX:

- Arquivos regulares;
- Arquivos diretórios;
- Arquivos especiais.

As características de cada tipo são descritas a seguir.

2.2.1 ARQUIVOS REGULARES

Arquivos regulares são utilizados para armazenar informações.

Um arquivo regular pode conter textos simples (como cartas, dados de funcionários de uma empresa, poesias), programas escritos em linguagem de programação (código-fonte) ou conter código executável referente a algum programa já compilado. Enfim, qualquer tipo de informação que o computador possa processar pode ser armazenada em um arquivo regular.

Em uma sessão no SOX, você pode trabalhar com vários arquivos regulares. Como exemplo, sejam os comandos do SOX. Eles são armazenados em arquivos executáveis, os quais são executados ao serem fornecidos os nomes dos comandos, ou seja, dos arquivos contendo os seus códigos executáveis.

Arquivos regulares podem conter ou não caracteres pertencentes ao conjunto de caracteres ASCII.

Como exemplos de arquivos contendo apenas caracteres ASCII, podem ser citados aqueles contendo o texto de uma carta ou o código-fonte de um programa em linguagem C. O conteúdo destes arquivos pode ser exibido com a utilização de comandos do SOX específicos para realizarem esta tarefa, tal como o comando cat, conforme veremos mais adiante.

Arquivos regulares podem conter também códigos que não fazem parte do conjunto de caracteres ASCII. Estes arquivos não podem ser exibidos diretamente no terminal, pois possuem caracteres não-exibíveis. Entretanto, o conteúdo destes arquivos pode ser verificado com a utilização do comando od, o qual converte os valores do arquivo em caracteres exibíveis.

Arquivos regulares podem possuir as seguintes organizações:

- stream;
- seqüencial;
- seqüencial indexado;
- relativa.

Estas organizações não estão disponíveis a todos os usuários do SOX. Isto porque o SOX é comercializado em duas versões. A primeira, funcionalmente compatível com o UNIX® System V, Release 2.0 da AT&T, dá suporte apenas à organização stream. A segunda oferece um sistema de arquivamento com métodos de acesso mais elaborados, dando suporte às organizações stream, seqüencial, seqüencial indexado e relativa. A organização stream é comum às duas versões. Nesta organização, o arquivo é tratado como uma seqüência de octetos.

Como nem todos os usuários possuem a versão contendo o sistema de arquivamento estendido, trataremos no manual apenas de arquivos regulares com organização stream. Caso você possua a versão contendo o sistema de arquivamento estendido e quiser obter informações a respeito das outras organizações, consulte o "Manual de Referência Sistema de Arquivamento Estendido SOX (SAX)".

2.2.2 ARQUIVOS DIRETORIOS

Um arquivo diretório, ou simplesmente diretório, é um arquivo que contém informações sobre outros arquivos tais como os seus nomes e localizações.

De uma maneira intuitiva, é mais fácil pensar que um diretório é alguma coisa que "contém" arquivos e diretórios.

Os diretórios podem, por sua vez, conter outros arquivos e subdiretórios. Um diretório contido em um outro diretório é chamado de subdiretório ou diretório-filho ou descendente.

Um diretório que contém outro é freqüentemente chamado de diretório-pai ou ascendente.

No momento, apenas estas informações a respeito de diretórios são necessárias. A medida que for sendo apresentada a estrutura de arquivos do SOX, o conceito de diretório se tornará mais claro.

2.2.3 ARQUIVOS ESPECIAIS

Arquivos especiais são utilizados para acessar dispositivos. Isto significa que toda vez que quisermos acessar ou manipular diretamente um dispositivo temos que fazê-lo através do arquivo a ele associado.

Arquivos especiais podem ser do tipo bloco ou caractere.

A utilização de arquivos especiais foge ao escopo deste manual. Maiores informações a respeito destes arquivos podem ser encontradas no "Manual do Administrador SOX".

2.2.4 COMO SABER O TIPO DE UM ARQUIVO ?

Para saber o tipo de um arquivo pode ser utilizado o comando ls, no formato longo (opção -l). O formato longo apresenta várias informações relativas a um arquivo. O primeiro caractere deste formato é o único que nos interessa neste momento. Isto porque é através dele que é informado o tipo do arquivo.

A tabela a seguir ilustra o significado de alguns caracteres exibidos pelo ls, que indicam o tipo do arquivo:

Caractere	Tipo do arquivo e organização
-	Regular (stream)
d	Diretório
b	Especial tipo bloco
c	Especial tipo caractere

Como exemplo, vamos verificar os tipos dos arquivos de nomes arq_reg e arq_diretorio. Os seguintes comandos geram os resultados mostrados:

```
$ ls -l arq_reg        <CR>
-rwxrwx--- 2 silva alfa 32 Mar 14 10:31 arq-reg
$ ls -l arq_diretorio <CR>
drwxrwx--- 2 silva alfa 32 Mar 14 10:31 documentos
$ _
```

Verificando o primeiro caractere exibido por cada um dos comandos ls, podemos concluir que o arquivo arq_reg é do tipo regular (organização stream) e que o arquivo arq_diretorio é do tipo diretório.

2.3 A ESTRUTURA HIERÁRQUICA DE ARQUIVOS

O sistema de arquivos do SOX se organiza na forma de uma estrutura hierárquica. Esta estrutura oferece uma maneira simples e eficiente de organizar grandes volumes de informações, permitindo que informações logicamente relacionadas sejam agrupadas.

Antes de entrarmos em detalhes a respeito da estrutura de arquivos do SOX, é necessária uma rápida explanação do que vem a ser uma estrutura hierárquica.

2.3.1 ESTRUTURA HIERÁRQUICA

Uma estrutura hierárquica freqüentemente tem a forma de uma pirâmide.

Como exemplo, seja a construção da árvore genealógica de uma família, tomando-se inicialmente um determinado membro, tal como, por exemplo, o avô. Este membro avô pode ter um ou mais filhos. Cada filho pode, por sua vez, ter um ou mais filhos, e assim sucessivamente. Esquematizando, esta árvore genealógica pode ter o seguinte formato:

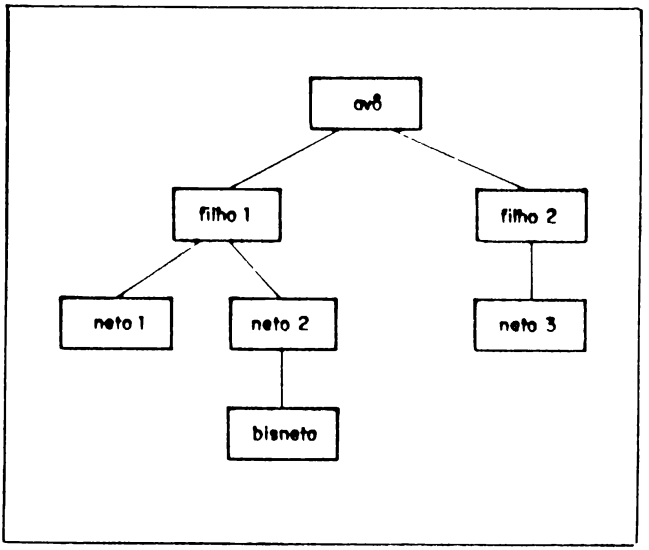


Figura 2-1

A partir do membro avô, ramificam-se os membros filho1 e filho2.

O membro filho1 ramifica-se nos membros neto1 e neto2. Este último ramifica-se no membro bisneto.

O membro filho2, por sua vez, ramifica-se no membro neto3.

O membro avô ocupa o nível mais alto desta estrutura hierárquica e, conforme podemos observar, existe uma hierarquia entre os demais membros desta árvore genealógica.

A árvore genealógica da família tem uma estrutura denominada árvore.

Tal como na árvore genealógica da família, a estrutura de organização de arquivos do SOX é hierárquica, sendo também chamada de árvore.

2.3.2 ESTRUTURA HIERÁRQUICA DO SISTEMA DE ARQUIVOS DO SOX

A estrutura de arquivos do SOX é composta por um conjunto de arquivos conectados entre si, segundo a estrutura de árvore apresentada anteriormente.

Uma das vantagens desta estrutura é permitir uma melhor organização dos arquivos, facilitando a pesquisa de uma informação em particular.

Para um melhor esclarecimento, consideremos como exemplo uma árvore representada pela figura a seguir. Nela é ilustrada a organização de um escritório que se ocupa da correspondência enviada aos clientes, da preparação de documentos referentes ao IAPAS e da confecção/armazenamento dos balanços da empresa.

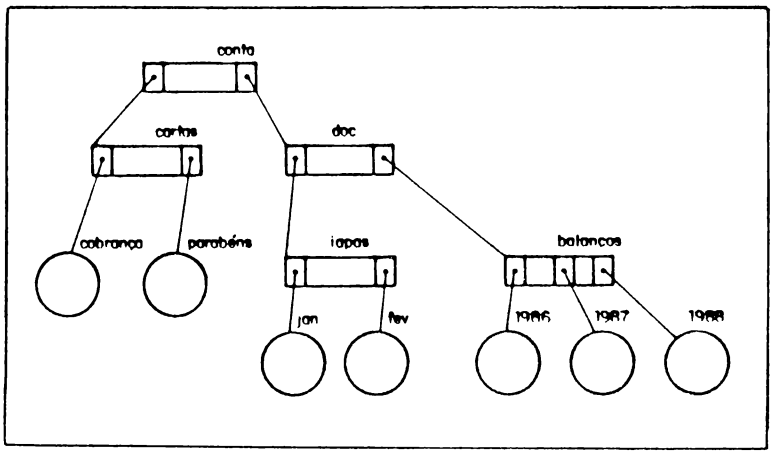


Figura 2-2

A árvore associada à estrutura de arquivos é usualmente representada de cima para baixo, onde a raiz da árvore fica no topo.

Para nos familiarizarmos com a estrutura de árvore, vamos estudar a sua expansão. A árvore se expande para baixo a partir da raiz, com caminhos (ramificações) conectando a raiz e outros arquivos.

Um arquivo regular ou especial é o final de um caminho, no qual não é permitido ter-se mais nenhum caminho a partir dele.

Um diretório, pelo contrário, permite que outros caminhos se ramifiquem a partir dele.

Na Figura 2.2, e nas seguintes, os retângulos simbolizam diretórios e os círculos simbolizam arquivos regulares.

Consideremos um diretório qualquer. A partir deste diretório podem ser criados vários subdiretórios que, por sua vez, podem conter outros subdiretórios, e assim sucessivamente. Desta forma, a estrutura de arquivos pode se expandir de acordo com as suas necessidades. O sistema de arquivos resultante é visto como a coleção dos diversos arquivos (regulares, diretórios e especiais), onde se destaca a interdependência lógica entre os elementos (através de caminhos).

Para tornar clara esta interdependência lógica, vamos considerar o nosso exemplo.

Suponhamos que o nosso escritório necessite armazenar os documentos referentes ao IAPAS dos meses de janeiro e fevereiro e aos balanços dos anos de 1986, 1987 e 1988, e que a correspondência seja preparada a partir de dois modelos, um para cobrança e outro para parabéns. Neste ponto temos um problema: como organizar estas informações utilizando a estrutura hierárquica?

Uma solução possível é utilizarmos um diretório para cada grupo de informações: um para cartas e outro para doc. No diretório **cartas** agrupamos os dois modelos de cartas. No diretório **doc** temos dois outros grupos de informações distintas. Com isto, criamos um diretório para as informações referentes ao pagamento do IAPAS e outro diretório para as informações referentes aos balanços.

No nosso exemplo, o diretório inicial **conta**, que é o de nível mais alto da hierarquia, contém dois outros diretórios: cartas e doc. Assim, conta é o diretório-pai dos diretórios cartas e doc e, por sua vez, os diretórios cartas e doc são diretórios-filhos do diretório conta. O diretório cartas contém apenas dois arquivos regulares: cobrança e parabéns.

O diretório doc desdobra-se em dois outros diretórios: iapas e balanços. Assim, doc é o diretório-pai dos diretórios iapas e balanços e, por sua vez, os diretórios iapas e balanços são diretórios-filhos do diretório doc. O diretório iapas contém os arquivos regulares jan e fev e o diretório balanços contém os arquivos regulares 1986, 1987 e 1988.

Conforme podemos observar no nosso exemplo, a organização dos arquivos pela estrutura hierárquica em árvore nos facilita a visualização do sistema de arquivos. Uma rápida verificação na Figura 2-2 nos 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.

A partir do que vimos, podemos concluir que a estrutura hierárquica permite que informações logicamente relacionadas sejam agrupadas. Com isto, a partir do conhecimento da natureza de uma informação, esta pode ser facilmente recuperada. Cabe observar, entretanto, que esta propriedade se aplica apenas se o usuário souber utilizá-la adequadamente.

Vamos agora aprender a caminhar na árvore.

O termo "subir" refere-se a caminhar na árvore em direção à raiz. O termo "descer" refere-se a caminhar na árvore a partir da raiz ou de um diretório qualquer, afastando-se da raiz.

Observando-se a Figura 2-2 podemos ver que existe um caminho ligando os vários arquivos. Assim, para chegarmos ao arquivo 1986, devemos começar pelo diretório conta e "descer" pela árvore, passando pelos diretórios doc e balanços. Da mesma forma, o arquivo parabéns pode ser encontrado passando-se pelos diretórios conta e cartas. Os nomes utilizados para cada arquivo componente deste caminho seguem determinadas regras e são assunto de um tópico mais adiante.

No SOX, o início do sistema de arquivos fica no diretório-raiz, o qual é representado por /.

Na figura a seguir, apresentamos um sistema de arquivos SOX, onde a árvore apresentada no exemplo anterior é uma subárvore deste sistema, ou seja, é parte componente deste.

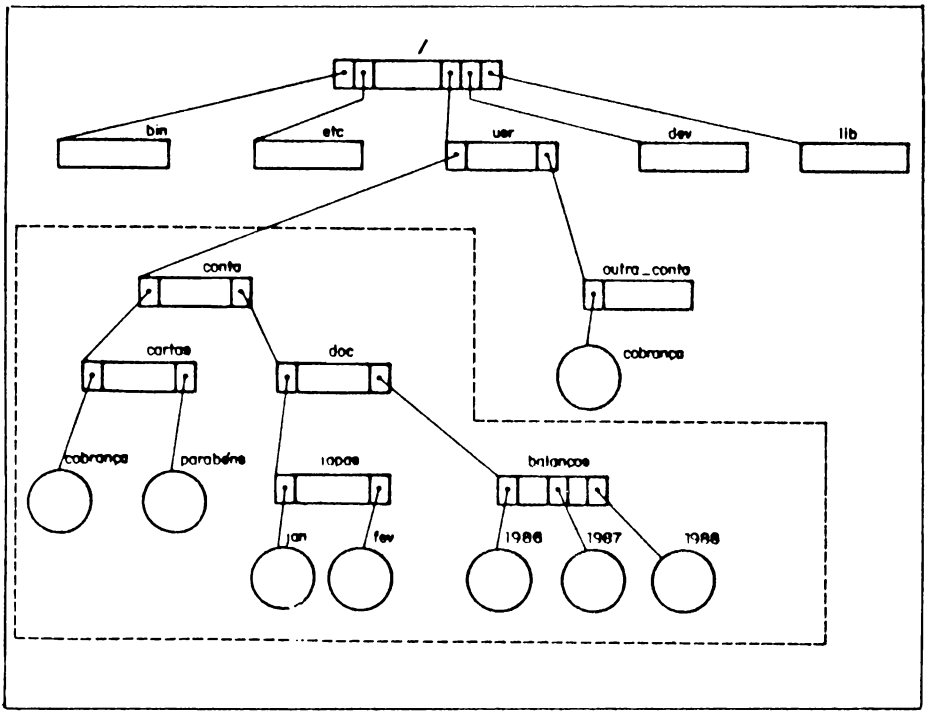


Figura 2-3

Agora, seguem-se alguns comentários sobre esta figura.

Primeiro, devemos notar que, como a figura ilustra o sistema de arquivos numa máquina SOX, lá se alojam os diretórios e arquivos regulares ou especiais dos usuários e do próprio sistema.

Abaixo do diretório-raiz (/) vemos nomes de diretórios, tais como bin, etc, dev, lib e usr. Apenas este último nos interessa aqui. Os demais (podem existir outros no 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. O conteúdo destes diretórios serão vistos mais adiante.

O diretório /usr (este é o nome completo partindo da raiz) é que contém as subárvores, uma para cada sistema de arquivos de cada usuário SOX.

Na subárvore associada ao seu sistema de arquivos é que ficam seus diretórios e outros arquivos. Em geral, você não recebe o seu sistema de arquivos na sua forma definitiva. Você é quem constrói e o modifica, à medida que usa o SOX e realiza suas atividades.

Mas como fazemos para criar diretórios e arquivos regulares? Esta pergunta será respondida em uma próxima seção. Antes, porém, vamos aprender um pouco mais sobre nomes de arquivos.

2.4 EXPRESSÃO DE CAMINHO

Todo arquivo tem uma expressão de caminho a ele associado.

Esta expressão de caminho é construída traçando-se um caminho a partir do diretório raiz (/) ou de outro diretório qualquer, através de todos os subdiretórios até o arquivo propriamente dito. Os nomes dos arquivos utilizados na expressão de caminho são separados entre si pelo caractere "/" (barra).

Para exemplificarmos, consideremos a Figura 2-3. As expressões de caminho

/usr/conta/cartas/parabéns

e

conta/cartas/parabéns

estão associadas ao arquivo parabéns. O primeiro é construído traçando-se um caminho a partir da raiz (/), descendo-se pelos diretórios usr, conta e cartas. O segundo é construído traçando-se um caminho a partir do diretório conta, descendo-se pelo diretório cartas.

Devido à existência da expressão de caminho, o SOX permite a criação de arquivos com o mesmo nome, desde que estejam em diretórios distintos. Por exemplo, observando a Figura 2.3, vemos que o arquivo cobrança existe em dois lugares diferentes. As expressões de caminho desses arquivos, contudo, são diferentes.

Assim, construindo-se estas duas expressões a partir do diretório raiz, temos

/usr/conta/cartas/cobrança

e

/usr/outra_conta/cobrança

Por este motivo, não existe confusão por parte do SOX sobre qual arquivo foi feito referência.

2.4.1 NOMES DE ARQUIVOS COMPONENTES DE UMA EXPRESSÃO DE CAMINHO

Conforme já vimos, todo arquivo tem um nome. Este nome é composto por 1 a 14 caracteres. Porém, convém lembrar que se for usado o comando edx, o nome do arquivo deve ter, no máximo, 10 caracteres, pois os quatro últimos são usados pelo edx para o arquivo .bak. Embora qualquer caractere possa ser utilizado, devem ser evitados nomes de arquivos para o seu uso contendo algum caractere diferente dos descritos na lista a seguir:

- letras maiúsculas [A-Z];
- letras minúsculas [a-z];
- números [0-9];
- sublinha [_];
- ponto [.];
- vírgula [,].

Como exemplos de nomes de arquivos válidos, temos:

1988

contas_do_mes

relatorios

Embora no decorrer do manual não sejam utilizados nomes de arquivos com letras maiúsculas, o uso destas é permitido. Deve-se ter em mente, entretanto, que para o SOX os arquivos de nomes MANUAL, Manual e manual são considerados três arquivos distintos.

O nome escolhido para um arquivo deve ser significativo no sentido de fornecer informações a respeito do conteúdo do arquivo.

Nomes de arquivos podem ter extensões (ou sufixos), a fim de ajudar a descrever o conteúdo do arquivo. O sufixo é a parte do nome do arquivo seguinte ao caractere "." (ponto).

Alguns comandos do SOX, como, por exemplo, o compilador da linguagem C, dependem da existência destes sufixos. Entretanto, na maioria das vezes, como é o caso de arquivos contendo código executável (tais como os comandos SOX), a extensão do nome do arquivo é opcional.

Exemplo:

prog.c - Indica um arquivo contendo um programa em linguagem C.

prog.o - Indica um arquivo contendo um programa em código-objeto.

stdio.h- Indica um arquivo contendo definições declarações a serem incluídas em um programa C.

O uso de extensões é livre dentro do nome do arquivo. Vários caracteres "." (ponto) podem ser utilizados para aumentar a legibilidade. Por exemplo, o nome cartas.1.12.88 é válido.

2.5 NOMES DE ARQUIVOS

O nome de um arquivo no SOX pode ser absoluto ou relativo.

2.5.1 NOMES ABSOLUTOS DE ARQUIVOS

A expressão de caminho construída a partir do diretório raiz (/), através de todos os subdiretórios até o arquivo propriamente dito, é o que chamamos de nome absoluto do arquivo. Isto porque, através deste nome, pode-se localizar um arquivo absolutamente, traçando um caminho a partir do diretório raiz até ele.

Para praticarmos um pouco, vamos construir os nomes absolutos dos arquivos jan e fev. Descendo-se pela árvore a partir do diretório raiz, temos, respectivamente, os seguintes nomes absolutos:

/usr/conta/doc/iapas/jan

e

/usr/conta/doc/iapas/fev

2.5.2 NOMES RELATIVOS DE ARQUIVOS

Para entendermos o que vem a ser um nome relativo, faz-se necessária a discussão de alguns pontos.

2.5.2.1 Diretório de Trabalho Corrente e Diretório Inicial de Trabalho

No seu cadastramento no SOX, é criado pelo administrador do sistema um diretório que lhe é atribuído.

Como vimos, de uma maneira geral, é o diretório /usr que contém as subárvores correspondentes a cada sistema de arquivo de cada usuário SOX. Assim, o diretório criado pelo administrador, quando do seu cadastramento, está contido, geralmente, no diretório /usr.

Supondo-se que o nome fornecido ao administrador para seu cadastramento seja **conta**, é criado por ele o diretório /usr/conta.

Durante uma sessão no SOX, você está sempre associado a um diretório.

O diretório ao qual você está associado (ou trabalhando) é chamado de diretório de trabalho corrente ou, simplesmente, diretório corrente. Algumas vezes esta associação é referenciada num sentido físico mesmo, tal como em "você está no (ou trabalhando no) diretório conta" ou "você está no diretório iapas".

O comando pwd pode ser utilizado para informar o diretório de trabalho corrente. Assim, supondo-se que você esteja no diretório conta, o comando pwd nos fornece o seguinte resultado:

```
$ pwd   <CR>
/usr/conta
$ _
```

Sempre que você abre sessão no SOX, o seu diretório de trabalho corrente é aquele que lhe é associado pelo administrador no seu cadastramento. Este diretório é chamado de diretório inicial de trabalho (ou diretório-casa). No nosso exemplo, o diretório /usr/conta é o diretório inicial de trabalho.

Nem sempre o diretório de trabalho corrente é o diretório inicial de trabalho. Isto porque o diretório de trabalho corrente pode ser alterado para outro diretório qualquer. Para realizarmos esta operação, usamos o comando cd.

Como exemplo, suponhamos que o diretório de trabalho corrente seja /usr/conta e desejamos trocá-lo para /usr/conta/cartas. Para tal, utilizamos o comando cd para trocar o diretório, e o comando pwd para verificar o diretório corrente antes e após a troca.

```
$ pwd                    <CR>
/usr/conta
$ cd /cartas             <CR>
$ pwd                    <CR>
/usr/conta/cartas
$ _
```

A expressão de caminho construída a partir do diretório de trabalho corrente, através de todos os subdiretórios até o arquivo propriamente dito, é o que chamamos de nome relativo de arquivo. Isto porque, o nome do arquivo é relativo ao diretório no qual você se encontra. Um nome relativo não inicia nunca pelo caractere "/" (barra).

2.6 COMANDOS PARA MANIPULAÇÃO DO SISTEMA DE ARQUIVOS

O SOX fornece um conjunto de comandos para manipulação do sistema de arquivos. Nesta seção, dedicaremos uma atenção especial a alguns deles.

Para que possamos exemplificar o uso destes comandos, precisamos que você crie alguns arquivos e os organize em diretórios. Como exercício dos comandos que estudaremos, vamos reproduzir o sistema de arquivos da Figura 2.3, dentro da região tracejada, ou seja, sua subárvore. Primeiro, criaremos todos os diretórios e, finalmente, os arquivos regulares nos diretórios aos quais pertencem.

Neste exercício, vamos supor que você já se cadastrou no SOX com o nome conta e, conseqüentemente, o diretório /usr/conta já foi criado pelo administrador. O diretório criado pelo administrador não está vazio. Esta informação é suficiente no momento.

Antes de começarmos, deve ser feita uma observação sobre a representação de um sistema de arquivos. A Figura 2.3 representa o sistema de arquivos de forma lógica. O SOX não lhe apresenta a estrutura (árvore hierarquizada) do seu sistema de arquivos de forma pictorial (como na Figura 2-3). Em vez disso, você obtém, através do comando ls, por exemplo, uma listagem na tela de seu terminal, do conteúdo do diretório que você especifica ou do diretório de trabalho corrente.

Concluindo, podemos dizer que você poderá formar a estrutura hierarquizada de seu sistema de arquivos na sua mente, a partir das informações que o SOX lhe fornece.

Agora, conecte-se ao SOX, utilizando como identificação o nome conta. Convém lembrar que, feito isto, o seu diretório de trabalho corrente, que é o mesmo que o seu diretório inicial de trabalho, é /usr/conta.

Vamos agora aos comandos!

2.6.1 CRIANDO DIRETÓRIOS

A fim de reproduzirmos o seu sistema de arquivos, vamos inicialmente criar todos os diretórios. O comando SOX que cria um ou mais diretórios é mkdir.

Para criar um diretório, o comando mkdir deve vir acompanhado do nome do diretório a ser criado.

Como exemplo, considerando-se que o diretório de trabalho corrente é /usr/conta, a criação do diretório **cartas** como diretório-filho do diretório corrent é feito como:

```
$ mkdir cartas      <CR>
$ _
```

Deve ser observado que não foi necessário utilizar o nome absoluto do diretório cartas, ou seja, /usr/conta/cartas, uma vez que o diretório de trabalho corrente é /usr/conta. Sendo assim, a utilização do nome relativo do diretório cartas é suficiente para que este seja criado como diretório-filho do diretório de trabalho corrente.

O mesmo procedimento deve ser seguido para criarmos o diretório **doc** como diretório-filho do diretório de trabalho corrente.

```
$ mkdir doc   <CR>
$ _
```

Feito isto, criamos os dois diretórios-filhos do diretório /usr/conta. Os nomes absolutos destes dois diretórios são

/usr/conta/cartas

e

/usr/conta/doc

O comando mkdir pode criar mais de um diretório de uma só vez, informando-se no mesmo comando os vários nomes dos diretórios a serem criados.

Assim, em vez de criarmos os diretórios cartas e doc conforme mostramos, podemos, opcionalmente, fazer:

```
$ mkdir cartas doc    < CR >
$ _
```

Neste ponto, representando-se a subárvore correspondente ao seu sistema de arquivos, temos:

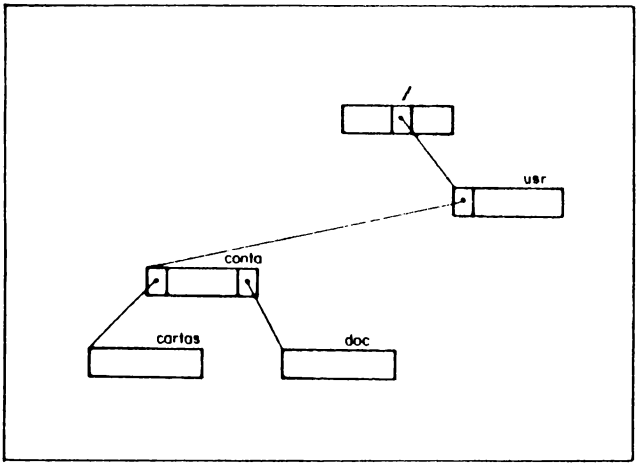


Figura 2-4

Você ainda não tem aqui como certificar-se da criação destes dois diretórios. Não se preocupe! Você poderá fazer isto quando estudarmos o comando ls mais adiante.

Antes de prosseguirmos com o estudo dos comandos, faz-se necessária uma observação sobre o comando mkdir.

2.6.1.1 As Entradas . e ..

O comando mkdir insere, automaticamente, duas entradas em cada diretório criado por você. Por este motivo, como já vimos, um diretório recém-criado não está vazio.

Estas entradas são . (lê-se ponto) e .. (lê-se ponto-ponto), e representam, respectivamente, o próprio diretório e o seu diretório-pai. Isto significa que . (ponto) é sinônimo da expressão de caminho do diretório de trabalho corrente e pode ser usado em seu lugar, e que .. (ponto-ponto) é sinônimo da expressão de caminho do diretório-pai do diretório de trabalho corrente.

O comando ls, o qual é utilizado, entre outras coisas, para listar o conteúdo de um diretório, não exibe, de um modo geral, estas duas entradas; é como se elas não existissem. Para listá-las, deve ser utilizado o comando ls -a, conforme veremos mais adiante.

O tamanho de um diretório é um múltiplo de 16 octetos, com um valor mínimo de 32 octetos. Isto ocorre porque as entradas . e .. sempre fazem parte de um diretório recém-criado, onde cada uma ocupa 16 octetos. Maiores detalhes a este respeito são dispensáveis para o tratamento de seu dia-a-dia com o SOX.

2.6.2 LISTANDO O CONTEÚDO DE DIRETÓRIOS

O comando ls já foi utilizado neste capítulo para fornecer o tipo de um arquivo. Este mesmo comando pode também ser utilizado para listar o conteúdo de um diretório em ordem alfabética.

Era este o comando que precisávamos para verificar o resultado das operações realizadas na seção 2.6.1. Assim, utilizando o comando ls para verificarmos o conteúdo do diretório de trabalho corrente, ou seja, /usr/conta, temos o seguinte diálogo:

```
$ ls     <CR>
cartas      doc
$ _
```

O comando ls considera, uma vez que nenhum diretório foi informado, que se deseja saber o conteúdo do diretório de trabalho corrente.

Caso os nomes exibidos pelo comando ls sejam diferentes de cartas ou doc, significa que você os criou com nomes errados. Neste caso, você poderá eliminá-los com o comando rmdir, a ser visto mais adiante, e criá-los novamente com o comando mkdir, utilizando os nomes corretos.

O comando ls pode fornecer informações mais detalhadas sobre o conteúdo do seu diretório de trabalho corrente ou de um outro diretório qualquer, bastando para isto utilizar a opção -l, formato longo, conforme mostrado a seguir:

```
$ ls -l      <CR>
total    2
drwxrwxrwx 2 conta 133 32 Mar 24 08:52 cartas
drwxrwxrwx 2 conta 133 32 Mar 24 08:52 doc
$ _
```

O formato longo apresenta o número total de arquivos contidos no diretório de trabalho corrente. Em seguida, em ordem alfabética, para cada um deles, é indicado o tipo do arquivo (conforme já vimos), as permissões de acesso, o número de associações, a identificação do dono do arquivo, a identificação do grupo, o tamanho (em octetos), a data e a hora da última alteração realizada no arquivo e o nome do arquivo. No momento, não vamos nos deter explicando o significado destas informações. Ele se tornará mais claro até o final do capítulo.

Na listagem do conteúdo do diretório /usr/conta, que é o diretório de trabalho corrente, pode ser observado que as entradas . e .. não são exibidas. O mesmo acontece com arquivos cujos nomes começam pelo caractere . (ponto). Isto ocorre porque nomes de arquivos começando pelo caractere . (ponto) são conhecidos como arquivos invisíveis e não são exibidos de um modo geral pelo comando ls. Para tanto, deve ser utilizada a opção -a do comando ls, que indica que todos os arquivos, inclusive as entradas . e .. e os arquivos invisíveis, devem ser listados.

```
$ ls -a        <CR>
.          ..          cartas      doc
$ _
```

2.6.3 MUDANDO O DIRETÓRIO DE TRABALHO CORRENTE

O comando cd muda o seu diretório de trabalho corrente para um outro diretório. Assim, o comando:

```
$ cd doc   < CR >
$ _
```

faz com que o diretório de trabalho corrente passe a ser /usr/conta/doc. Devemos observar que aqui foi utilizado o nome relativo do diretório doc, uma vez que o diretório de trabalho corrente é /usr/conta. Caso este diretório não seja o diretório de trabalho corrente, deveremos utilizar o nome absoluto /usr/conta/doc para atingirmos o nosso objetivo, da seguinte maneira:

```
$ cd /usr/conta/doc      < CR >
$ _
```

Utilizando o comando ls neste momento, verificamos que o diretório de trabalho corrente se encontra vazio, ou seja, este não contém arquivo algum.

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

Agora devemos criar os diretórios iapas e balanços como diretórios-filhos de doc, agindo da seguinte forma:

```
$ mkdir iapas balancos   <CR>
$ _
```

Uma nova conferência pode ser feita com o comando ls, a fim de verificarmos a criação destes dois subdiretórios do diretório de trabalho corrente /usr/conta/doc.

```
$ ls     <CR>
balancos  iapas
$ _
```

Neste ponto, podemos ilustrar a subárvore correspondente ao seu sistema de arquivos da seguinte maneira:

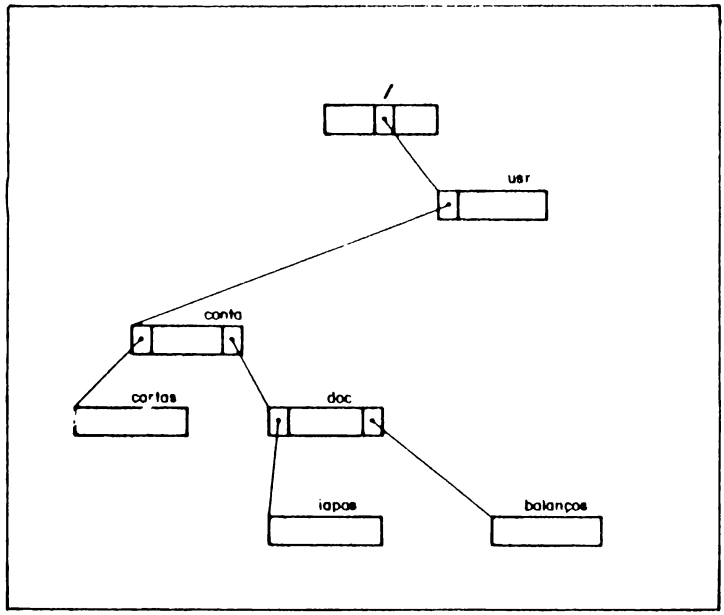


Figura 2-5

A partir dos procedimentos mostrados, podemos concluir que uma maneira de se criar subdiretórios é mover-se (com o comando cd) para o diretório onde ficarão os subdiretórios e lá criar os subdiretórios que desejar (com o comando mkdir).

Para você se deslocar para um dos subdiretórios criados, como por exemplo, iapas, utilize novamente o comando cd.

```
$ cd iapas    < CR >
$ _
```

Feito isto, o seu diretório de trabalho corrente é /usr/conta/doc/iapas.

Para você retornar ao diretório-pai de iapas, ou seja, /usr/conta/doc, pode ser usado o seguinte comando:

```
$ cd ..       < CR >
$ _
```

Devemos lembrar que, conforme já vimos, .. (ponto-ponto) é sinônimo da expressão de caminho do diretório-pai do diretório de trabalho corrente. Assim, o comando cd, utilizado anteriormente, é equivalente a:

```
$ cd /usr/conta/doc   < CR >
$ _
```

A partir de qualquer diretório, você pode sempre retornar ao seu diretório inicial de trabalho com o comando:

```
$ cd        <CR>
$ _
```

Feito isto, o diretório de trabalho corrente é o mesmo que o diretório inicial de trabalho, ou seja, /usr/conta.

2.6.4 EXIBINDO O NOME DO DIRETÓRIO DE TRABALHO CORRENTE

O comando pwd fornece o nome absoluto do seu diretório de trabalho corrente.

```
$ pwd       <CR>
/usr/conta
$ _
```

Como exercício, vamos mudar agora para o diretório doc e utilizar novamente o comando pwd.

```
$ cd doc            <CR>
$ pwd               <CR>
/usr/conta/doc
$ _
```

Ainda como exercício, vamos agora passar para o diretório balanços e utilizar o comando pwd. Em seguida, vamos retornar ao diretório-pai do diretório de trabalho corrente e novamente listar o nome absoluto deste diretório.

```
$ cd balancos          < CR >
$ pwd                  < CR >
/usr/conta/doc/balancos
$ cd ..                < CR >
$ pwd                  < CR >
/usr/contas/doc
$ _
```

Vamos agora retornar finalmente ao diretório inicial de trabalho e verificar este retorno com o comando pwd.

```
$ cd                   < CR >
$ pwd                  < CR >
/usr/conta
$ _
```

Como conclusão, podemos dizer que 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 (comando mkdir), como caminhar por todo o seu sistema de arquivos (comando cd) e como perguntar ao SOX onde você se encontra (comando pwd).

Para chegarmos à estrutura do seu sistema de arquivos, conforme a Figura 2-3, falta-nos criar os arquivos regulares. Para tanto, podemos utilizar um editor de texto, tal como o comando edx.

2.6.5 CRIANDO ARQUIVOS REGULARES

O SOX oferece alguns comandos para a criação de arquivos regulares. Um deles, de nome edx, é um editor de texto.

O comando edx opera em tela plena, sendo, por este motivo, um editor de tela. Com sua utilização vamos criar os arquivos regulares, a fim de completarmos a subárvore correspondente ao seu sistema de arquivos.

O objetivo aqui não é descrever o comando edx, com as facilidades oferecidas. Nós vamos apenas citá-lo. Maiores detalhes a respeito do edx podem ser vistos no "Manual do Usuário Editor EDX/SOX".

Vamos então à criação do arquivo parabéns!

O arquivo parabéns deverá estar contido no diretório cartas. Logo, o primeiro procedimento é mudarmos o diretório de trabalho corrente, que é /usr/conta, para /usr/conta/cartas. Utilizamos o comando pwd para obtermos informações a respeito do diretório de trabalho corrente.

```
$ pwd          <CR>
/usr/conta
$ cd cartas    <CR>
$ pwd          <CR>
/usr/conta/cartas
$ _
```

Feito isto, vamos utilizar o comando edx, fazendo:

```
$ edx          <CR>
```

Logo após, é exibida uma tela pelo edx, pedindo que você informe o nome do arquivo a ser editado. Você deverá digitar então: parabens.

Como este arquivo não existe, a tela anterior é apagada e a mensagem **arquivo novo** é exibida. Neste ponto, você poderá desistir inibindo a criação do arquivo ou iniciá-la. Como o nosso objetivo é criar o arquivo parabéns, vamos atingi-lo.

O edx fornece, através do auxílio de funções específicas, facilidades para:

- Movimentação do cursor;
- Movimentação da tela;
- Tabulação;
- Modificação de texto;
- Manipulação de arquivos.

As funções disponíveis e facilidades oferecidas estão descritas detalhadamente no manual do edx. Consulte-o a fim de editar o arquivo parabéns e obter o seguinte resultado:

Rio de Janeiro, .. de ..... de 19..

Prezado cliente

Vimos atraves desta parabeniza-lo pela data de hoje.

Aproveitamos a oportunidade para nos colocar ao seu dispor.

Esperamos continuar contando com voce.

Cordialmente,

.....
(Gerente geral de vendas)

Após a edição, para gravar o arquivo parabéns, utilize a função **FINALIZAÇÃO DE EDIÇÃO**, descrita no manual do edx.

Neste ponto, representando-se a sua subárvore, temos:

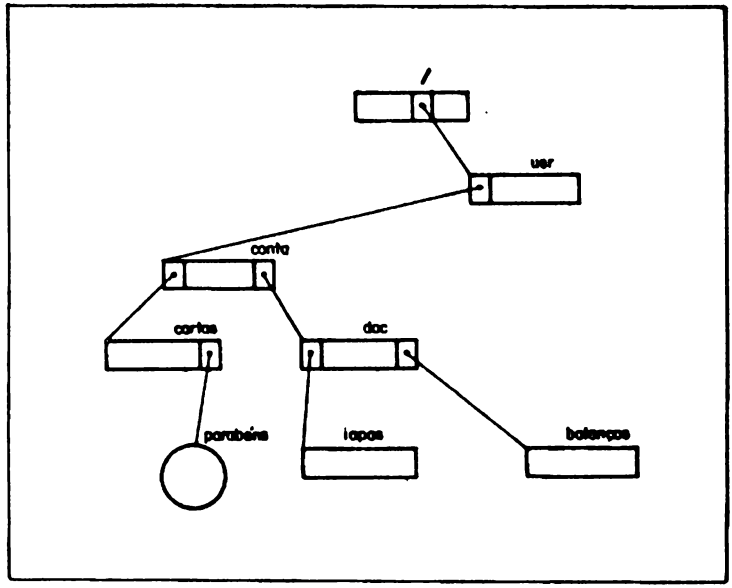


Figura 2-6

Vamos aproveitar que o diretório de trabalho corrente é /usr/conta/cartas e criar o arquivo regular cobrança. A tarefa de criação deste arquivo é deixada como exercício para você. Ao final da edição do arquivo, utilize a função **FINALIZA EDIÇÃO E ABANDONA JANELA**, descrita no manual do edx, a fim de encerrar a atividade de edição e finalizar o edx.

Após a criação destes dois arquivos, temos a seguinte representação da sua subárvore:

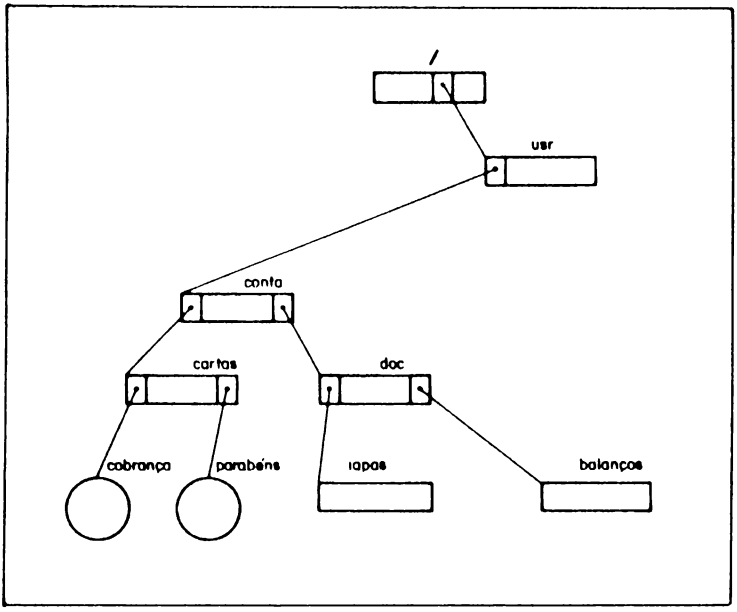


Figura 2-7

Para completarmos a subárvore correspondente ao seu sistema de arquivos, devemos criar os arquivos regulares restantes.

Vamos então nos mover para o diretório /usr/conta/doc/iapas e criar aí os arquivos jan e fev. Como o diretório corrente é /usr/conta/cartas, podemos agir da seguinte maneira:

```
$ pwd                          < CR >
/usr/conta/cartas
$ cd ../doc/iapas              < CR >
$ pwd                          < CR >
/usr/conta/doc/iapas
$ _
```

Com isto, conforme podemos verificar com a utilização do comando pwd, o nome absoluto do diretório de trabalho corrente é /usr/conta/doc/iapas. A criação dos arquivos jan e fev neste diretório, com a utilização de edx, é deixada como exercício para você.

A seguir, devemos criar os arquivos 1986, 1987 e 1988 no diretório balanços. Novamente vamos utilizar o comando cd para nos mover para o diretório no qual estes arquivos deverão ser criados.

```
$ pwd                          < CR >
/usr/conta/doc/iapas
$ cd ../balancos               < CR >
$ pwd                          < CR >
/usr/conta/doc/balancos
$ _
```

A criação dos arquivos 1986, 1987 e 1988 neste diretório é deixada também como exercício para você.

Pronto! Finalmente a subárvore correspondente ao seu sistema de arquivos, representada pela Figura 2-3, está completa.

Resta ainda a apresentação de alguns comandos SOX que são de muita utilidade na manipulação do sistema de arquivos. Vamos estudá-los baseando-nos ainda na subárvore representada pela Figura 2-3.

2.6.6 EXIBINDO O CONTEÚDO DE ARQUIVOS REGULARES

O comando cat permite, entre outras coisas, a exibição na tela do conteúdo de um arquivo regular.

Como exemplo, vamos exibir o conteúdo do arquivo parabéns, criado por nós anteriormente. Inicialmente, vamos mudar o diretório de trabalho corrente para /usr/conta/cartas e, em seguida, listar o arquivo parabéns.

```
$ cd /usr/conta/cartas      < CR >
$ cat parabens              < CR >
Rio de Janeiro, .. de ..... de 19..

              Prezado cliente

Vimos atraves desta  parabeniza-lo pela data
de hoje.

Aproveitamos a oportunidade para nos colocar
ao seu dispor.

Esperamos continuar contando com voce.

                Cordialmente,

             ..............................
                (Gerente geral de vendas)
$ _
```

As linhas do arquivo são exibidas continuamente, até o seu final, havendo rolamento de tela.

O rolamento pode ser interrompido a qualquer instante com a utilização de teclas específicas de controle do terminal. Verifique no manual do seu terminal quais as teclas que realizam esta tarefa.

Pratique o uso do comando cat listando cada um dos arquivos criados por você.

Você pode economizar tempo pedindo para listar vários arquivos, com uma única linha de comando. Neste caso, o comando cat exibe o conteúdo dos arquivos pedidos, um após o outro. Por exemplo, para listarmos o conteúdo dos arquivos 1986 e 1987 vamos nos mover para o diretório /usr/conta/doc/balanços e proceder da seguinte forma:

```
$ pwd                    < CR >
/usr/conta/cartas
$ cd ../doc/balancos     < CR >
$ cat 1986 1987          < CR >
                .
                .
                .
    (Conteúdo do arquivo 1986)
                .
                .
                .

    (Conteúdo do arquivo 1987)
                .
                .
                .
$ _
```

Pratique mais um pouco exibindo o conteúdo dos arquivos jan e fev.

Conforme você pode ver, o comando cat oferece uma alternativa eficiente para examinar o conteúdo de vários arquivos rapidamente.

2.6.7 COPIANDO ARQUIVOS REGULARES

O comando cp copia o conteúdo de um arquivo em outro, tendo o seguinte formato:

```
cp arquivo1 arquivo2
```

O conteúdo do arquivo especificado como arquivo1 é copiado no arquivo especificado como arquivo2, sendo este último criado, caso não exista, ou o seu conteúdo é alterado, caso contrário.

Como exemplo, vamos copiar o arquivo parabéns para o arquivo aniversário no mesmo diretório do arquivo parabéns, ou seja, cartas. Uma vez que o arquivo aniversário não existe no seu sistema de arquivos, ele será criado com a operação de cópia. Vamos conferir a criação deste arquivo utilizando o comando ls.

```
$ cd /usr/conta/cartas     <CR>
$ cp parabens aniversario  <CR>
$ ls                       <CR>
aniversario  cobranca  parabens
$ _
```

Neste ponto, representando-se a subárvore correspondente ao seu sistema de arquivos, temos:

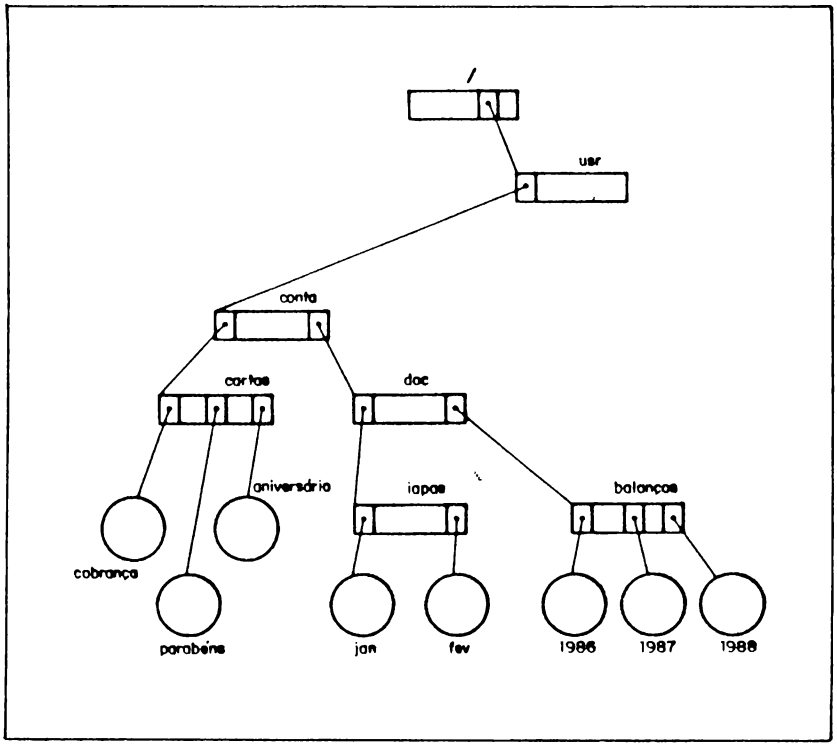


Figura 2-8

Vamos praticar mais um pouco, fazendo uma cópia do arquivo cobrança no diretório cartas para o arquivo cobrança no diretório conta. Considerando-se que o diretório de trabalho corrente é /usr/conta/cartas, a operação de cópia pode ser realizada da seguinte maneira:

```
$ pwd                          < CR >
/usr/conta/cartas
$ cp cobranca ../cobranca      < CR >
$ ls ..                        < CR >
cartas      cobranca    doc
$ _
```

Após a cópia, temos a seguinte representação da subárvore correspondente ao seu sistema de arquivos:

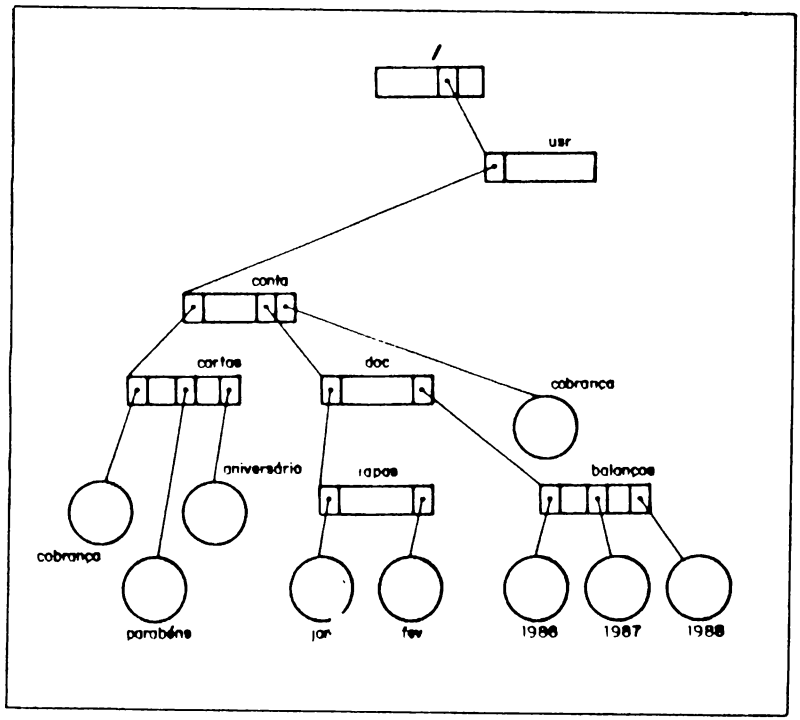


Figura 2-9

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, utilize o comando cp no formato seguinte:

```
cp arquivo1 arquivo2 ... nome-do-diretório-destino
```

Isto fará com que sejam criadas cópias dos arquivos arquivo1, arquivo2, etc, com estes mesmos nomes, no diretório destino especificado.

Para praticar, vamos utilizar esta informação para copiar os arquivos jan e fev do diretório iapas para o diretório inicial de trabalho (/usr/conta), mantendo os mesmos nomes. Procedemos da seguinte maneira:

```
$ pwd                          < CR >
/usr/conta/cartas
$ cd ../doc/iapas              < CR >
$ pwd                          < CR >
/usr/conta/doc/iapas
$ cp jan fev ../..             < CR >
$ cd                           < CR >
$ pwd                          < CR >
/usr/conta
$ ls                           < CR >
cartas   cobranca   doc  fev  jan
$ _
```

Após esta cópia, podemos representar a subárvore correspondente ao seu sistema de arquivos como:

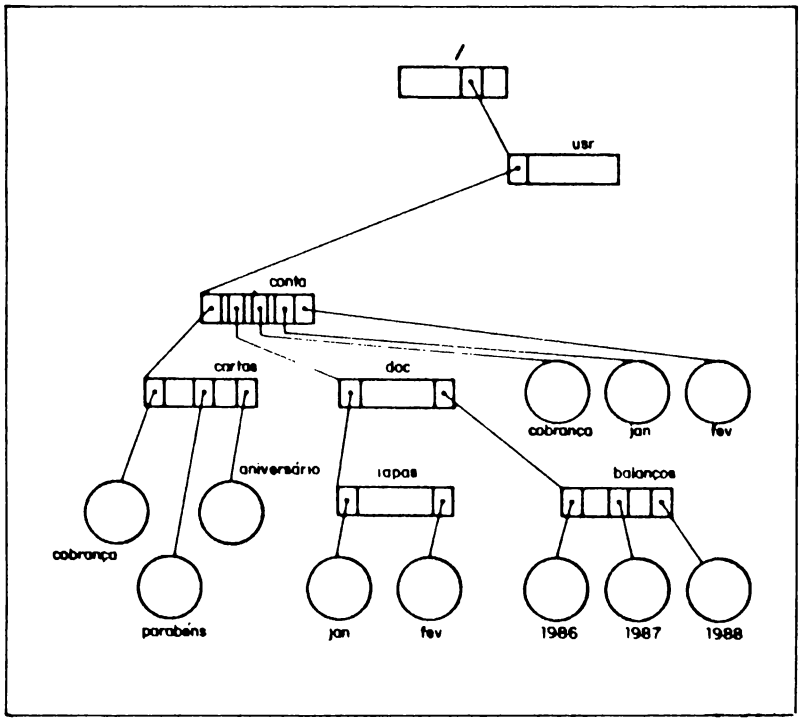


Figura 2-10

2.6.8 REMOVENDO ARQUIVOS REGULARES

Para remover um arquivo de um diretório, deve ser utilizado o comando rm. Para isto, você pode:

a) Mover-se (com o comando cd) para o diretório no qual está contido o arquivo a ser removido e então usar o comando rm;

b) Usar o comando rm a partir do diretório de trabalho corrente, fornecendo a expressão de caminho até o arquivo a ser removido.

Vamos exemplificar primeiro a possibilidade **a**, considerando-se que o arquivo de nome absoluto /usr/conta/cobrança deva ser removido.

```
$ cd              < CR >
$ pwd             < CR >
/usr/conta
$ rm cobranca     < CR >
$ ls              < CR >
cartas  doc  fev  jan
$ _
```

Exemplificando agora a alternativa **b**, vamos considerar que o arquivo de nome absoluto /usr/conta/cartas/aniversário deve ser removido. Assim, tem-se:

```
$ pwd                        < CR >
/usr/conta
$ rm cartas/aniversario      < CR >
$ ls cartas                  < CR >
cobranca   parabens
$ _
```

O comando rm permite que mais de um arquivo seja removido em um único comando. Vamos praticar removendo todos os arquivos contidos nos diretórios cartas e iapas, considerando-se que o diretório de trabalho corrente é /usr/conta.

```
$ cd cartas                    <CR>
$ pwd                          <CR>
/usr/conta/cartas
$ rm cobranca parabens         <CR>
$ ls                           <CR>
$ cd ../doc/iapas              <CR>
$ rm jan fev                   <CR>
$ ls                           <CR>
$ _
```

Após a operação de remoção destes arquivos, os diretórios cartas e iapas se encontram vazios, conforme podemos verificar com a utilização do comando ls.

Ainda para praticarmos, vamos remover os arquivos jan e fev do diretório conta.

```
$ cd          <CR>
$ rm jan fev  <CR>
$ ls          <CR>
cartas    doc
$ _
```

Neste ponto, representando-se a sua subárvore após a remoção dos arquivos regulares, temos:

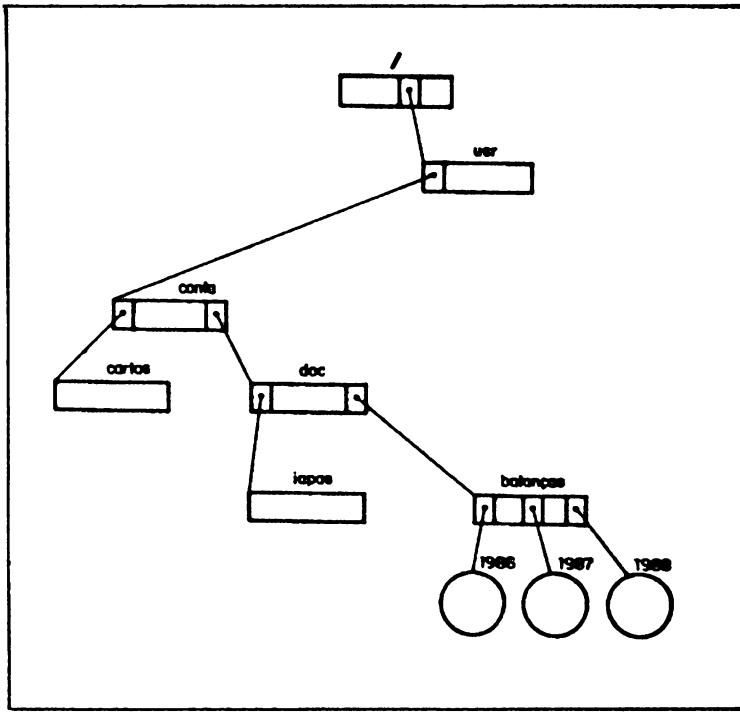


Figura 2-11

No momento, as informações dadas sobre o comando rm são suficientes. Entretanto, a tarefa de remoção de arquivos não é tão simples como apresentada aqui, sendo necessária uma informação adicional, relacionada a associações. Numa próxima seção, abordaremos novamente o comando rm, fornecendo maiores detalhes a respeito de remoção de arquivos.

2.6.9 REMOVENDO DIRETORIOS

Para remover um diretório é necessário que este não contenha nenhuma outra entrada além de . (ponto) e .. (ponto-ponto). Para esta operação, você dispõe do comando rmdir.

Vejamos um exemplo: mude o diretório de trabalho corrente para o seu diretório inicial de trabalho e remova o diretório cartas, verificando o conteúdo do diretório conta logo após.

```
$ cd          <CR>
$ rmdir cartas <CR>
$ ls          <CR>
doc
$ _
```

Conclusão: O subdiretório cartas não faz mais parte do diretório conta.

Você conseguiu remover o diretório cartas porque ele se encontrava vazio.

O comando rmdir não remove um diretório não-vazio. Por exemplo, o diretório doc possui dois subdiretórios: iapas e balanços. Ao tentar removê-lo, considerando que o diretório de trabalho corrente é /usr/conta, teremos o seguinte diálogo:

```
$ rmdir doc    <CR>
rmdir: 'doc' nao se encontra vazio
$ _
```

Utilizando a seguir o comando ls, verificamos que o diretório doc não foi removido.

```
$ ls           <CR>
doc
$ _
```

Para remover o diretório doc você precisa primeiro esvaziá-lo, ou seja, remover os diretórios iapas e balanços. Antes, porém, devemos remover os arquivos regulares contidos no diretório balanços. É deixado como exercício para você a remoção dos arquivos 1986, 1987 e 1988 do diretório balanços. Feito isto, vamos utilizar o comando rmdir com uma facilidade oferecida por ele, que é permitir a remoção de mais de um diretório em um único comando, a fim de removermos os subdiretórios iapas e balanços. Logo após, vamos usar novamente o comando rmdir para removermos, finalmente, o diretório doc. Assim, temos:

```
$ cd /usr/conta/doc      <CR>
$ rmdir iapas balancos   <CR>
$ ls                     <CR>
$ cd ..                  <CR>
$ rmdir doc              <CR>
$ ls                     <CR>
$ _
```

Neste momento, representando-se a subárvore correspondente ao seu sistema de arquivos, temos:

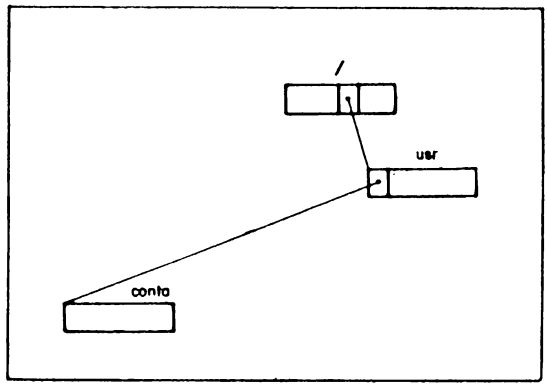


Figura 2-12

2.7 ALGUNS DIRETÓRIOS IMPORTANTES DO SISTEMA DE ARQUIVOS

O conjunto de arquivos que compõem o sistema de arquivos e a sua distribuição varia de uma instalação para outra. Cabe ao administrador, quando da criação do sistema de arquivos, determinar esta distribuição e estabelecer regras para que esta seja mantida.

Alguns diretórios importantes, de uso geral, tais como /bin, /etc, /usr, /dev, /lib, /tmp e /usr/bin, normalmente fazem parte do sistema de arquivos. A figura a seguir ilustra a localização usual destes diretórios.

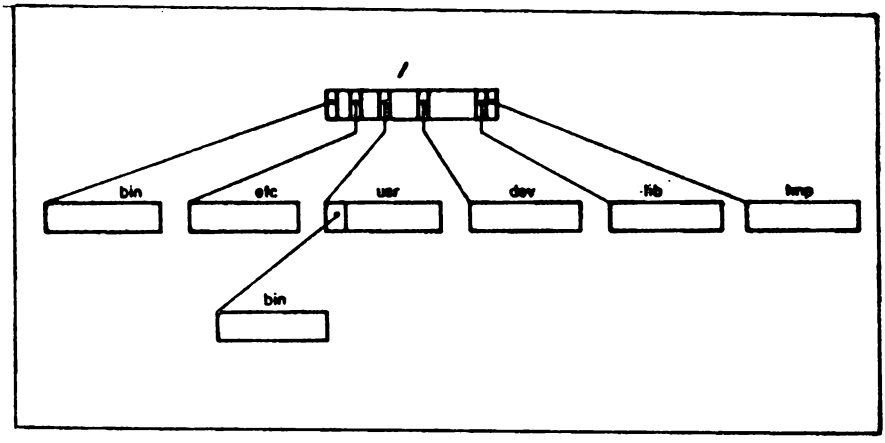


Figura 2-13

O conteúdo destes diretórios é o seguinte:

/ O diretório / (raiz) está presente em todo sistema de arquivos SOX, sendo o de mais alto nível na estrutura hierárquica.

/usr O diretório /usr contém os subdiretórios correspondentes ao diretório inicial de trabalho de cada usuário. Assim, como exemplo, o nome absoluto do diretório inicial de trabalho do usuário conta é /usr/conta.

/bin O diretório /bin contém os comandos SOX usados mais freqüentemente, incluindo todos aqueles necessários à fase de manutenção do sistema.

/usr/bin O diretório /usr/bin contém os comandos SOX usados menos freqüentemente.

/etc O diretório /etc contém programas e arquivos de dados que são essenciais à execução do SOX, os quais geralmente são utilizados pelo administrador do sistema. Como exemplo de um arquivo contido neste diretório, pode ser citado o arquivo passwd, que contém uma lista de todos os usuários que têm permissão para usar o sistema.

/dev O diretório /dev contém os arquivos especiais.

/tmp O diretório /tmp é utilizado para conter arquivos temporários criados por alguns programas.

/lib O diretório /lib contém bibliotecas. Como exemplo, pode ser citada a biblioteca C, cujo nome absoluto é /lib/libc.a.

2.8 PERMISSÕES DE ACESSO

Nas últimas seções você aprendeu como manipular seu sistema de arquivos. No seu dia-a-dia de trabalho, entretanto, você necessita acessar arquivos de outros usuários e vice-versa.

O acesso a arquivos de outros usuários é feito de forma similar à que discutimos. Você só tem (a princípio) que fornecer a expressão de caminho correta. Como exemplo, vamos considerar novamente a Figura 2-3 e supor que o seu diretório inicial de trabalho é /usr/conta. Caso você desejasse a listagem do conteúdo do arquivo regular cobrança contido no diretório outra_conta, bastaria fazer o seguinte:

```
$ cat /usr/outra_conta/cobranca < CR >
```

Nada diferente do que já vimos: bastou você fornecer a expressão de caminho a partir da raiz. Entretanto, não podemos afirmar que o arquivo cobrança seja acessado e seu conteúdo exibido. Continue lendo para descobrir o porquê.

Neste ponto, poderíamos fazer algumas perguntas. O que aconteceria se o outro usuário não quisesse que você examinasse o conteúdo do arquivo? Como ele poderia impedir que alguém, em vez de usar o comando cat, utilizasse o comando rm e, assim, removesse o arquivo?

Estas perguntas são respondidas de forma simples. O SOX permite que você atribua permissões de acesso a seus arquivos, de forma a limitar ou inibir acessos indesejáveis no seu sistema de arquivos.

Existem três tipos ou classes de usuários que podem acessar um arquivo: o dono do arquivo (dono), um membro do grupo ao qual o dono do arquivo pertence (grupo), e qualquer outro usuário (outros), onde:

dono - É geralmente o usuário que criou o arquivo.

grupo - É um conjunto de usuários que recebe uma identificação de grupo em função das afinidades de suas tarefas. Você pode permitir ou negar que membros do grupo ao qual você faz parte acessem seus arquivos.

outros - Todos os demais usuários SOX da sua instalação.

Um usuário pode tentar acessar um arquivo realizando uma das seguintes operações: lendo do arquivo, gravando no arquivo, ou executando o arquivo.

A fim de que o usuário realize qualquer uma destas operações em um arquivo, é necessário que ele tenha a devida permissão de acesso ao arquivo. Assim, no SOX, temos três tipos de permissões, a saber:

leitura - Permite ao usuário ler o conteúdo do arquivo. No caso de um diretório, o usuário pode listar o seu conteúdo, obtendo os nomes dos arquivos nele contidos (por exemplo, com a utilização do comando ls). A permissão de leitura em um diretório não dá, automaticamente, permissão de leitura nos arquivos neles contidos; para tanto, é necessário permissão de leitura nos arquivos individuais.

gravação - Permite ao usuário alterar o conteúdo do arquivo. No caso de um diretório, o usuário pode criar arquivos e remover arquivos do diretório em questão. A alteração de arquivos regulares já existentes no diretório depende das permissões de gravação em cada arquivo regular individual.

execução - Permite ao usuário chamar o nome do arquivo regular como se fosse um comando SOX, iniciando a sua execução. No caso de um diretório, permite que o usuário o transforme no diretório de trabalho corrente (com o comando cd) e que utilize o nome do diretório como componente de uma expressão de caminho. No acesso a um arquivo é verificado se o próximo diretório componente da expressão de caminho tem permissão de execução para o usuário que emitiu o comando. Portanto, é necessário ter-se permissão de execução (pesquisa) em cada um dos diretórios componentes da expressão de caminho correspondente a um arquivo. Assim, por exemplo, quando você utiliza o comando

```
cat /usr/conta/cartas/parabens
```

é necessário que você tenha permissão de execução nos diretórios usr, conta e cartas, para que o arquivo parabéns seja acessado, a fim de que o seu conteúdo seja exibido. Concluindo, podemos verificar que negar a proteção de execução em um diretório é uma proteção contra pessoas não autorizadas utilizarem arquivos naquele diretório.

O SOX combina os três tipos de permissões de acesso (leitura, gravação, execução) para cada tipo de usuário (dono, grupo, outros), obtendo nove modos de acesso a um arquivo, ou seja:

O dono pode ter permissão para:

leitura
gravação
execução

Um membro do grupo pode ter permissão para:

leitura
gravação
execução

Qualquer outro usuário pode ter permissão para:

leitura
gravação
execução

A seguir, vamos aprender como podemos saber as permissões de acesso de um arquivo e como alterá-las.

2.8.1 CONHECENDO AS PERMISSÕES DE ACESSO

O comando ls no formato longo (opção -l) apresenta várias informações relacionadas a um arquivo. Estas informações são dispostas da seguinte maneira:

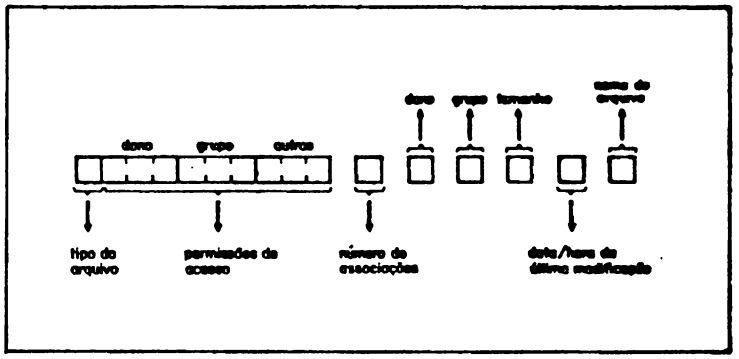


Figura 2-14

Da esquerda para a direita, estas informações significam:

- O tipo do arquivo, que é indicado pelo primeiro caractere, conforme já vimos.

- As permissões de acesso, que são indicadas pelos próximos nove caracteres. Para cada tipo de usuário (dono, grupo, outros) são utilizados três caracteres que indicam as permissões de acesso para aquele usuário. O primeiro caractere destes três pode ser a letra **r**, indicando permissão de leitura, ou o caractere - (traço), indicando que o usuário em questão não tem permissão de leitura; o segundo caractere pode ser a letra **w**, indicando permissão de gravação, ou o caractere - (traço), indicando que o usuário em questão não tem permissão de gravação; o terceiro pode ser a letra **x**, indicando permissão de execução, ou o caractere - (traço), indicando que o usuário em questão não tem permissão de execução.

- O número de associações do arquivo. Este assunto será visto numa próxima seção.

- O nome do dono do arquivo.

- O nome do grupo ao qual o dono do arquivo pertence.

- O tamanho do arquivo. Para arquivos regulares indica o número de caracteres do arquivo. Para diretórios, o tamanho é um múltiplo de 16, com um valor mínimo de 32.

- A data e a hora em que o arquivo foi criado ou modificado por último.

. O nome do arquivo.

Vamos nos deter um pouco mais no campo contendo as informações sobre as permissões de acesso. Utilizando o que foi apresentado, podemos ver que um arquivo que possa ser lido, gravado e executado por qualquer tipo de usuário tem as seguintes permissões:

```
rwxrwxrwx
```

Um arquivo que só pode ser lido, gravado e executado pelo dono tem as permissões de acesso:

```
rwx------
```

Devemos lembrar que o caractere - (traço) indica a ausência da permissão correspondente (no caso do nosso exemplo, para o grupo e para os outros usuários).

Um arquivo que pode ser lido, gravado e executado pelo dono e pelos membros do grupo tem permissões:

```
rwxrwx---
```

Um arquivo que só permita leitura por qualquer tipo de usuário tem permissões:

```
r--r--r--
```

Um arquivo que tenha permissão de leitura e execução para qualquer tipo de usuário tem permissões:

```
r-xr-xr-x
```

Vamos agora praticar um pouco, considerando a Figura 2-12, que representa a situação atual do seu sistema de arquivos após as operações de remoção das seções 2.6.8 e 2.6.9. Deixamos como exercício a criação dos diretórios cartas e doc como subdiretórios do diretório conta (comando mkdir) e dos arquivos regulares cobrança e parabéns contidos no diretório cartas (comando edx). Ao final, podemos representar sua subárvore como:

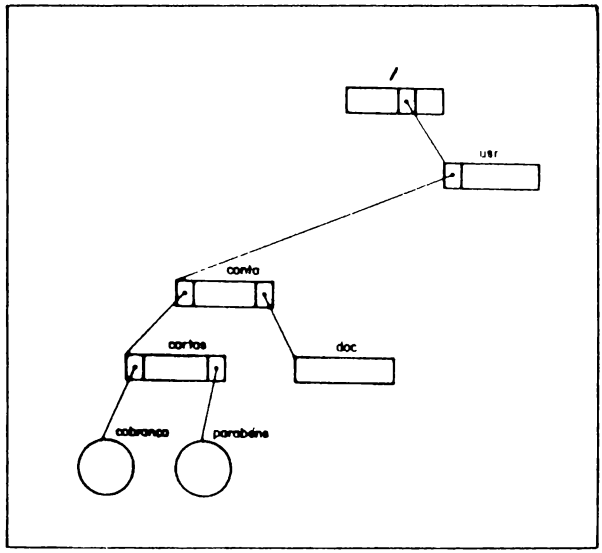


Figura 2-15

Vamos agora utilizar o comando ls para conhecermos as permissões de acesso aos seus arquivos.

Inicialmente, mude-se para o seu diretório inicial de trabalho, caso você já não esteja nele, fazendo:

```
$ cd        < CR >
$ pwd       < CR >
/usr/conta
$ _
```

Utilizando agora o comando ls, formato longo, obtemos o seguinte resultado:

```
$ ls -l     < CR >
total    2
drwxrwxrwx 2 conta 133 96 Mar 24 08:52 cartas
drwxrwxrwx 2 conta 133 32 Mar 24 08:52 doc
$ _
```

Confira as informações obtidas com a Figura 2-14. Podemos observar aqui que qualquer tipo de usuário tem todas as permissões de acesso aos diretórios cartas e doc. Estas permissões são dadas pelo comando mkdir.

Vamos verificar agora as permissões de acesso aos arquivos regulares contidos no diretório cartas, fazendo:

```
$ ls -l cartas      < CR >
total    2
-rw-r--r-- 1 conta 133 17 Mar 08:52 cobranca
-rw-r--r-- 1 conta 133 37 Mar 09:00 parabens
$ _
```

Podemos ver que nos arquivos cobrança e parabéns, o dono tem permissão de leitura e escrita, mas não tem permissão de execução, e que o grupo e outros têm permissão de leitura, mas não de gravação e execução. Estas permissões são dadas pelo comando edx.

Uma observação importante deve ser feita aqui. Antes de utilizar qualquer comando para a criação de um arquivo em um diretório qualquer, verique quais são as suas permissões de acesso naquele diretório. Caso você não proceda desta forma, poderá ter surpresas não muito agradáveis.

O comando ls tem disponível a opção -d, que é utilizada para exibir informações a respeito do próprio diretório. Caso você forneça esta opção sem especificar um diretório em especial, o comando ls exibe informações sobre o diretório de trabalho corrente. Combinando esta opção com a opção -l, obtemos as informações a respeito do diretório no formato descrito na Figura 2-14.

Vamos agora conhecer as permissões de acesso ao diretório /usr/conta, utilizando o comando ls com as opções -l e -d, ou seja, -ld:

```
$ pwd      <CR>
/usr/conta
$ ls -ld   <CR>
drwxrwxrwx 4 conta root 112 Mar 24 08:00 .
$ _
```

Verificamos aqui que todos os tipos de usuários têm todas as permissões de acesso a este diretório.

Como exercício examine as permissões de acesso aos arquivos com comandos SOX contidos no diretório /bin. Isto pode ser feito com a utilização do comando:

```
$ ls -l /bin   <CR>
          .
          .
          .
$ _
```

Como fazer agora para evitar que outros usuários acessem os arquivos contidos nos seus diretórios, tais como cartas e doc, ou para executar um programa criado com o comando edx? Estas perguntas são respondidas na próxima seção.

2.8.2 ALTERANDO AS PERMISSOES DE ACESSO

Após a criação de um arquivo, verifique as permissões que lhe foram atribuídas. Caso você queira mudá-las, o SOX fornece o comando chmod, permitindo que a mudança seja feita.

O comando chmod modifica as permissões de acesso a um arquivo. A modificação pode ser feita pelo dono do arquivo ou por usuários privilegiados, conhecidos como superusuários.

Este comando pode ser usado de duas formas: na primeira, as novas permissões de acesso ao arquivo são especificadas através de uma seqüência de caracteres, que chamamos de palavra de controle; na segunda, as novas permissões são especificadas através de uma seqüência de três dígitos octais. Vamos nos deter aqui apenas na primeira forma. Neste caso, o formato do comando chmod é:

```
chmod palavra-de-controle nome-arquivo
```

A palavra de controle utilizada no comando chmod pode ser dividida em três partes:

```
QUEM OPERADOR PERMISSOES
```

onde:

- A primeira parte (QUEM) indica o tipo do usuário. Os caracteres (pode ser usado mais de um) que podem ser utilizados nesta parte são:

Caractere (QUEM)	Significado
u	Usuário dono
g	Grupo
o	Outros
a	Todos (ugo)

. A segunda parte (OPERADOR) indica a operação a ser realizada. Os caracteres (apenas um de cada vez) que podem ser utilizados nesta parte são:

OPERADOR	Significado
-	Remove permissões
+	Acrescenta permissões
=	Atribui permissões

. A terceira parte (PERMISSÕES) indica as permissões de acesso ao arquivo. Alguns dos caracteres que podem ser utilizados nesta parte são:

Caractere (PERMISSÕES)	Significado
r	Leitura
w	Gravação
x	Execução
u	Permissões do usuário dono
g	Permissões do grupo
o	Permissões de outros

Vamos agora utilizar o comando chmod para alterar as permissões de acesso a alguns arquivos do seu sistema de arquivos.

Conforme vimos com a utilização do comando ls na seção anterior, qualquer tipo de usuário tem todas as permissões de acesso aos diretórios cartas e doc, ou seja,

rwxrwxrwx

Vamos supor que desejássemos proibir o acesso para gravação e execução no diretório cartas ao grupo e a outros usuários. Construindo a palavra de controle, temos:

1) QUEM: go

2) OPERADOR: -

3) PERMISSÕES: wx

Juntando, formamos a seguinte palavra: go-wx. Assim, utilizando o comando chmod, temos:

```
$ chmod go-wx cartas    <CR>
$ _
```

Com isto, removemos do diretório cartas as permissões de gravação e execução para o grupo e outros usuários. Podemos verificar as novas permissões deste diretório com o comando ls:

```
$ ls -l     <CR>
total    2
drwxr--r-- 2 conta 133 96 Mar 24 08:52 cartas
drwxrwxrwx 2 conta 133 32 Mar 24 08:52 doc
$ _
```

Vamos agora remover a permissão de gravação ao diretório doc para todos os tipos de usuários, sabendo que as permissões deste diretório neste momento são:

rwxrwxrwx

Construindo a palavra de controle, temos:

1) QUEM: a

2) OPERADOR: -

3) PERMISSOES: w

Utilizando o comando chmod e verificando em seguida com o comando ls, temos:

```
$ chmod a-w doc                    <CR>
$ ls -l                            <CR>
total    2
drwxr--r-- 2 conta 133 96 Mar 24 08:52 cartas
dr-xr-xr-x 2 conta 133 32 Mar 24 08:52 doc
$ _
```

O diretório doc está, desta forma, protegido contra gravação para todos os tipos de usuários, inclusive você. Assim, nenhum usuário tem permissão de gravar ou remover arquivos neste diretório.

Vamos treinar um pouco mais o comando chmod e mudar as permissões do arquivo cobranças. No momento, verificando com o comando ls, podemos ver que as permissões de acesso são:

```
$ ls -l cartas/cobranca     <CR>
-rw-r--r-- 1 conta 133 17 Mar 08:52 cartas/cobranca
$ _
```

Vamos mudá-las, a fim de que o grupo possa alterar este arquivo, utilizando o comando:

```
$ chmod g+w cartas/cobranca <CR>
$ _
```

Feito isto, acrescentamos no arquivo cobrança a permissão de gravação para o grupo. Podemos verificar utilizando o comando ls:

```
$ ls -l cartas/cobranca   <CR>
-rw-rw-r-- 1 conta 133 17 Mar 08:52 cartas/cobranca
$ _
```

Com a utilização do comando chmod vamos agora atribuir a outros usuários as permissões do dono ao arquivo cobrança, fazendo:

```
$ chmod o=u cartas/cobranca   <CR>
$ _
```

Verificando com ls, temos:

```
$ ls -l cartas/cobranca    <CR>
-rw-rw-rw- 1 conta 133 17 Mar 08:52 cartas/cobranca
$ _
```

2.9 ASSOCIAÇÕES

Na criação de um arquivo qualquer, seja com a utilização dos comandos mkdir, cp, edx ou algum outro, é inserido um ponteiro para o arquivo criado no diretório no qual ele está contido. Este ponteiro é utilizado para associar um nome de arquivo a um lugar no disco. Assim, quando você fornece o nome de um arquivo em um comando, você está fornecendo a localização no disco onde se encontra (ou encontrará) a informação desejada.

Uma associação é um ponteiro em um diretório para um arquivo. Assim, quando você cria um arquivo é estabelecida, automaticamente, no seu diretório-pai, uma associação para ele.

O SOX permite a criação de associações adicionais para um arquivo, além daquela estabelecida automaticamente para ele na sua criação.

A criação de associações adicionais para um arquivo permite que este seja compartilhado por vários usuários. Isto pode ser de grande utilidade quando várias pessoas trabalham em um mesmo projeto e necessitam compartilhar informações. Através da criação de associações, cada usuário participante do projeto pode ter acesso ao arquivo pelo seu próprio diretório.

A utilização de associações pode ser também muito eficiente quando o seu sistema de arquivos é muito grande e você acha conveniente que um mesmo arquivo esteja contido em mais de um diretório. Isto pode ser feito sem a necessidade de uma cópia física do arquivo (comando cp) de um diretório para outro, o que causaria um gasto desnecessário de espaço em disco. Neste caso, basta que você crie uma outra associação (ou seja, um ponteiro) para o arquivo no diretório desejado.

O SOX fornece alguns comandos que permitem a verificação do número de associações para um arquivo, a criação de associações adicionais para um arquivo e a remoção de associações. Vamos, então, estudá-los.

2.9.1 VERIFICANDO O NÚMERO DE ASSOCIAÇÕES PARA UM ARQUIVO

O número de associações para um arquivo pode ser verificado com a utilização do comando ls, formato longo (opção -l), conforme podemos observar na Figura 2-14.

Como exemplo, vamos considerar a Figura 2-15, que representa a sua subárvore no momento. Inicialmente, mova-se para o seu diretório inicial de trabalho (/usr/conta) e utilize o comando ls, fazendo:

```
$ cd          <CR>
$ pwd         <CR>
/usr/conta
$ ls -l       <CR>
total    2
drwxr--r-- 2 conta 133 96 Mar 24 08:52 cartas
dr-xr-xr-x 2 conta 133 32 Mar 24 08:52 doc
$ _
```

O número de associações dos diretórios cartas e doc é igual a 2. Vamos analisar este número.

Anteriormente, vimos que, ao ser criado um diretório (comando mkdir), automaticamente são inseridas neste diretório duas entradas: . (ponto) e .. (ponto-ponto). Estas entradas significam, respectivamente, um ponteiro para o próprio diretório e um ponteiro para o seu diretório-pai. O diretório-pai deste diretório, por sua vez, contém um ponteiro para o diretório em questão. Assim, o número de associações (ponteiros) para um diretório é no mínimo igual a 2, correspondendo à entrada . (ponto) e ao ponteiro do diretório-pai para o diretório-filho. Caso o diretório tenha um diretório-filho, o número de associações é acrescido de um, pois, neste caso, deve ser levada em consideração a entrada .. (ponto-ponto) do diretório-filho, que é um ponteiro para o diretório-pai. Assim, podemos concluir que o número de associações de um diretório é igual a 2 acrescido do número de diretórios-filhos. Por exemplo, um diretório com três diretórios-filhos tem 5 (2+3) associações.

Seguindo esta regra, podemos entender agora o número de associações informado pelo comando ls para os diretórios cartas e doc. Como nenhum dos dois tem diretórios-filhos, o número de associações de ambos é igual a 2.

Como exercício, vamos verificar o número de associações para o diretório conta, fazendo:

```
$ ls -ld        < CR >
total    1
drwxrwxrwx 4 conta root 112 Mar 24 08:00 .
$ _
```

O número de associações para o diretório conta é 4, que corresponde a 2 acrescido do número de diretórios-filhos, ou seja, 2.

Vamos agora verificar o número de associações para o arquivo parabéns, fazendo:

```
$ ls -l cartas/parabens  < CR >
-rw-r--r-- 1 conta 133 37 Mar 09:00 cartas/parabens
$ _
```

O número de associações é igual a 1. Isto ocorre porque o diretório cartas contém um ponteiro para o arquivo parabéns, existindo, assim, apenas uma associação para este arquivo. Nenhum outro diretório contém um ponteiro para o arquivo parabéns. Mas, como fazer para criar uma outra associação para o arquivo parabéns? É o que veremos a seguir.

2.9.2 CRIANDO ASSOCIAÇÕES PARA UM ARQUIVO

A criação de uma associação estabelece um novo caminho para o arquivo, permitindo, assim, que ele seja referenciado por um outro nome. Esta referência é feita através de uma outra expressão de caminho.

O comando ln é utilizado para criar uma associação adicional para um arquivo.

Como exemplo, considere a Figura 2-15. Vamos criar uma associação para o arquivo parabéns do diretório cartas no diretório doc, supondo que o diretório de trabalho corrente seja /usr/conta. Inicialmente, uma vez que trocamos as permissões de acesso ao diretório doc, devemos alterá-las novamente, a fim de que o dono tenha permissão de gravação neste diretório.

```
$ ls -ld doc                          <CR>
dr-xr-xr-x 2 conta 133 32 Mar 24 08:52 doc
$ chmod u+w doc                       <CR>
$ ls -ld doc                          <CR>
drwxr-xr-x 2 conta 133 32 Mar 24 08:52 doc
$ ln cartas/parabens doc/congrat      <CR>
$ _
```

Com isto, foi criado um ponteiro no diretório doc para o arquivo parabéns contido no diretório cartas. Neste ponto, representando a sua subárvore, temos:

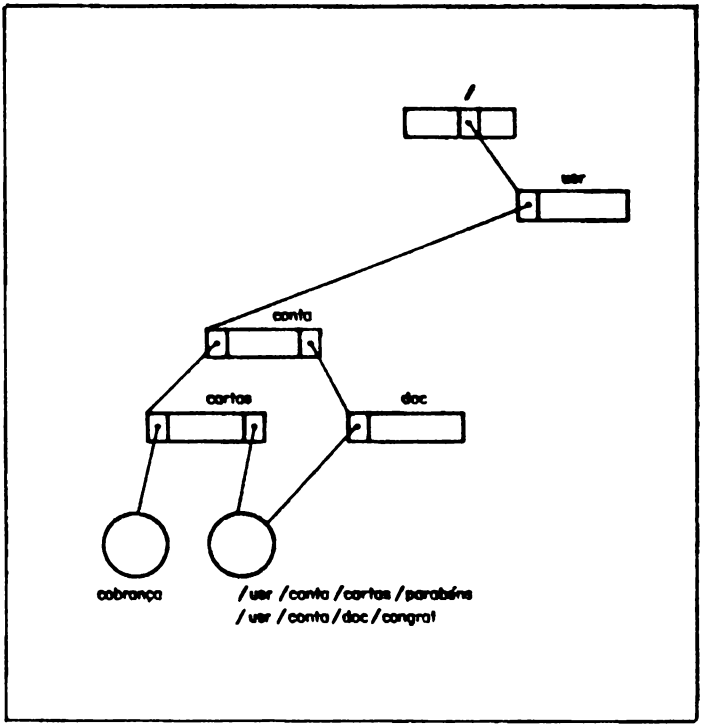


Figura 2-16

Estabeleceu-se, assim, um novo caminho para o arquivo parabéns, tendo-se as expressões de caminho /usr/conta/cartas/parabéns e /usr/conta/doc/congrat.

Verificando com o comando ls o número de associações para o arquivo parabéns, temos:

```
$ ls -l cartas/parabens <CR>
-rw-r--r-- 2 conta 133 37 Mar 09:00 cartas/parabens
$ _
```

A criação de uma associação não faz com que o arquivo seja fisicamente copiado para um outro arquivo, o que difere do comando cp, onde o arquivo é realmente duplicado.

Utilizando o comando ls, podemos verificar que, por se tratar do mesmo arquivo, as informaçoes relacionadas a /usr/conta/cartas/parabens e /usr/conta/doc/congrat, tais como, as permissões de acesso, dono e a data e a hora da última modificação, são as mesmas. Somente os nomes dos arquivos diferem.

```
$ ls -l doc/congrat <CR>
-rw-r--r-- 2 conta 133 37 Mar 09:00 doc/congrat
$ _
```

Vamos agora verificar a diferença entre os comandos cp e ln.

Utilize o comando cat para acrescentar uma linha ao conteúdo do arquivo parabens:

```
$ cat>> cartas/parabens <CR>
Linha a ser digitada <CR>
<ALT><D>
```

A seqüência <**ALT**><**D**> é usada para informar ao comando cat que não há mais linhas a serem digitadas.

A explicação detalhada para o uso do comando cat nesta forma, está descrita no item 3.4.2.2.

Verificando agora o conteúdo do arquivo parabens com cat, temos:

```
$ cat cartas/parabens <CR>
Este e o conteudo do arquivo parabens apos a
sua alteracao.
$ _
```

Exiba o conteúdo do arquivo /usr/conta/doc/congrat, fazendo:

```
$ cat doc/congrat  <CR>
Este e´ o conteudo do arquivo parabens apos a
sua alteracao.
$ _
```

Este era o resultado esperado, uma vez que o arquivo em questão é um só.

Vamos agora experimentar o comando cp e ver o que acontece.

Primeiro, copie o arquivo parabéns para o arquivo felicidade no diretório doc:

```
$ cp cartas/parabens doc/felicidade <CR>
$ _
```

Altere o conteúdo do arquivo felicidade com a utilização do comando edx e obtenha o seguinte resultado:

```
$ cat doc/felicidade  <CR>
Arquivo felicidade foi alterado.
$ _
```

Compare os conteúdos dos dois arquivos. Verificamos que estes são agora diferentes. Isto ocorre porque o comando cp copia o arquivo para um outro lugar. Com a utilização do comando edx, foi alterado apenas o arquivo felicidade. O arquivo parabéns ficou inalterado. Representando a sua subárvore, temos:

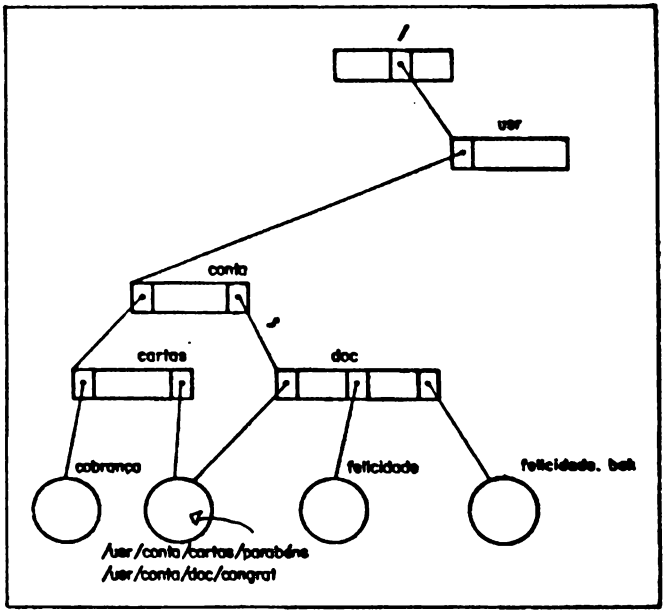


Figura 2-17

Observe que a Figura 2-17 apresenta o arquivo felicidade.bak. Isto se deve ao fato de que ao usar o comando edx para alterar um arquivo, este comando cria automaticamente um arquivo que é a cópia do arquivo original, com o objetivo de, se necessário, você poder recuperar o arquivo original, isto é, antes da última edição.

Maiores detalhes podem ser vistos no "Manual do Usuário Editor EDX/SOX".

Vamos agora aprender como remover associações para um arquivo.

2.9.3 REMOVENDO ASSOCIAÇÕES

Conforme vimos, um arquivo pode ter várias associações para ele. Para o SOX, todas as associações têm o mesmo valor, ou seja, o SOX não faz distinção da ordem em que as associações foram realizadas.

O comando rm remove associações para um arquivo.

Como exemplo, considere a Figura 2-17. Vamos remover a associação /usr/conta/doc/congrat para o arquivo parabéns, supondo que o diretório de trabalho corrente seja /usr/conta:

```
$ rm doc/congrat      <CR>
$ _
```

Com este comando, foi retirada a associação do diretório doc para o arquivo parabéns. Representando a subárvore correspondente ao seu sistema de arquivos, temos:

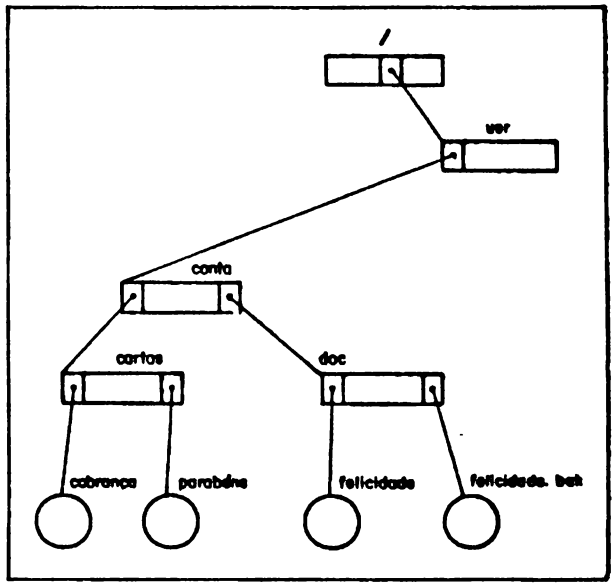


Figura 2-18

Verificando o diretório doc com o comando ls, temos:

```
$ ls doc      <CR>
felicidade  felicidade.bak
$ _
```

Deve ser verificado, entretanto, que o arquivo parabéns não foi removido da sua estrutura de arquivos. Ele ainda pode ser acessado. Experimente utilizar o comando cat para exibi-lo.

Vamos agora utilizar novamente o comando rm neste mesmo arquivo:

```
$ rm cartas/parabens      <CR>
$ _
```

Experimente agora acessá-lo, exibindo o seu conteúdo. Você não vai conseguir, porque este arquivo agora não faz mais parte do seu sistema de arquivos. Verifique utilizando o comando ls no diretório cartas.

```
$ ls cartas      <CR>
cobranca
$ _
```

Assim, representando-se a subárvore correspondente ao seu sistema de arquivos, temos:

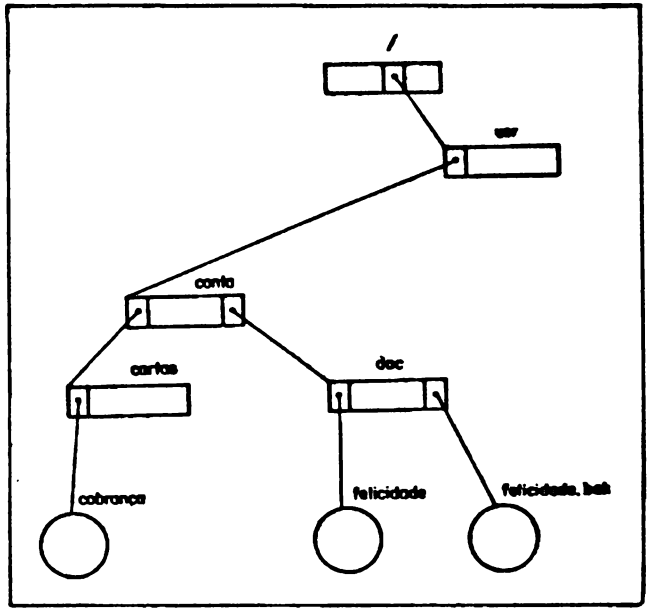


Figura 2-19

O que ocorre aqui é que um arquivo só é removido do seu sistema de arquivos quando não há mais associação alguma para ele, ou seja, quando o número de associações é igual a zero. Assim, como o arquivo parabéns tinha inicialmente duas associações para ele, com a utilização do comando rm pela primeira vez, foi removida uma associação, restando ainda outra. Com a utilização do segundo comando rm, foi removida a outra associação. Com isto, o arquivo não pode ser mais acessado, sendo removido do sistema de arquivos.

2.9.4 RENOMEANDO ARQUIVOS

O comando mv é utilizado para renomear arquivos, permitindo, assim, que o arquivo passe a ser referenciado por outro nome. Funcionalmente, o comando mv "move" um arquivo pelo estabelecimento de uma associação entre o novo nome do arquivo e o arquivo físico e então desfaz a associação entre o nome antigo do arquivo e o arquivo físico.

Como exemplo, considere a Figura 2-19, que representa a sua subárvore no momento. Vamos mudar o diretório de trabalho corrente para /usr/conta/doc e renomear o arquivo felicidade para parabéns, utilizando o comando mv.

```
$ cd doc                          <CR>
$ pwd                             <CR>
/usr/conta/doc
$ mv felicidade parabens          <CR>
$ _
```

O nome absoluto do arquivo, após a renomeação, passa a ser /usr/conta/doc/parabéns. Para confirmar a operação de renomeação do arquivo, utilize o comando ls.

```
$ ls     <CR>
felicidade.bak  parabens
$ _
```

Visto que o arquivo felicidade.bak foi criado anteriormente de forma automática pelo comando edx e o nosso objetivo é ter apenas o arquivo parabens no diretório corrente, vamos removê-lo, usando o comando rm.

```
$ rm felicidade.bak  <CR>
$ _
```

Vamos usar novamente o comando ls para confirmar a operação de remoção.

```
$ ls <CR>
parabens
$ _
```

Representando-se a subárvore correspondente ao seu sistema de arquivos no momento, temos:

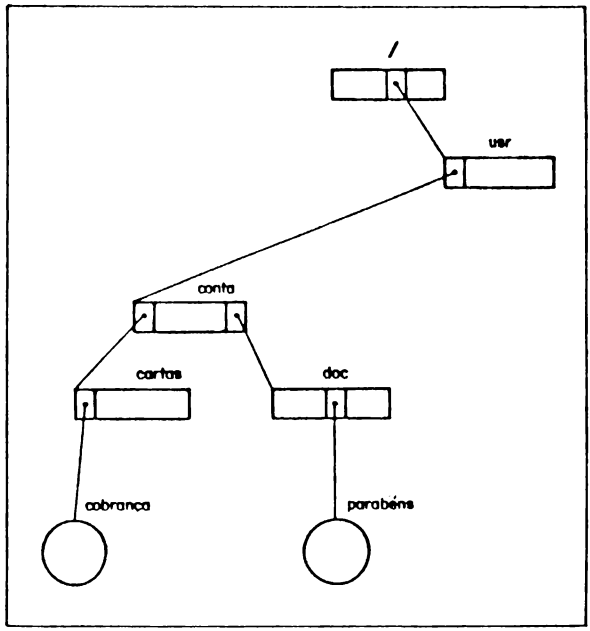


Figura 2-20

Prosseguindo, vamos agora utilizar o comando mv para "mover" o arquivo parabéns, contido no diretório doc, para o diretório cartas.

```
$ pwd                          <CR>
/usr/conta/doc
$ mv parabens ../cartas  <CR>
$ _
```

O nome absoluto do arquivo passa a ser /usr/conta/cartas/parabéns. Podemos observar que foi retirada a associação para o arquivo parabéns do diretório doc e criada uma outra associação para ele no diretório cartas. Verificando os conteúdos dos diretórios cartas e doc, temos:

```
$ ls                  <CR>
$ ls ../cartas     <CR>
cobranca     parabens
$ _
```

A representação da subárvore correspondente ao seu sistema de arquivos pode ser feita como:

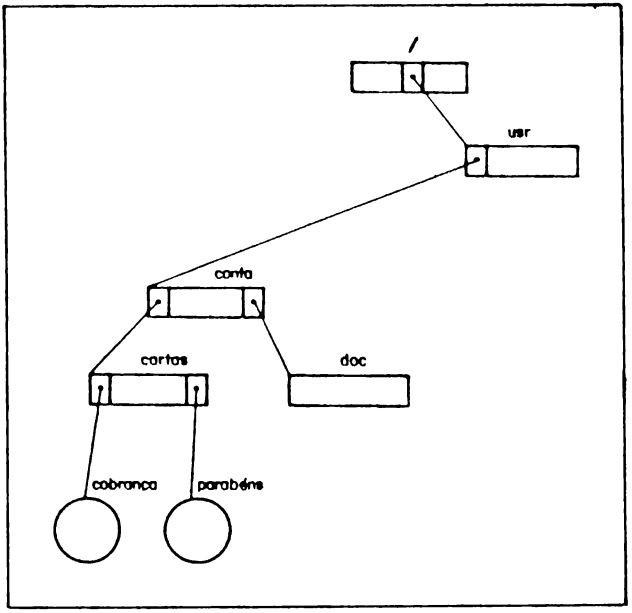


Figura 2-21

Cabe aqui uma observação final sobre o comando mv. O comando mv difere funcionalmente do comando cp. No primeiro, o arquivo é "movido" pelo estabelecimento de uma associação entre o novo nome do arquivo e o arquivo físico, e a associação anterior entre o nome antigo do arquivo e o arquivo físico é desfeita. No segundo comando, por outro lado, é realmente realizada uma cópia do arquivo e a associação já existente para o nome anterior do arquivo permanece inalterada.

CAPÍTULO 3
RECURSOS DO SHELL

Uma das características mais importantes do SOX é a sua capacidade de dialogar, interagir com você. Quando você digita uma linha de comando através do teclado, o SOX a recebe, interpreta, executa e devolve resultados ou mensagens de erro, passando em seguida a aguardar um novo comando.

Este capítulo tem como objetivo apresentar o componente do SOX responsável pela sua capacidade de interagir com você. Este componente é o SHELL. São apresentados o funcionamento básico do SHELL, a estrutura de uma linha de comando e os principais recursos do SHELL. Estes recursos permitem melhorar a sua produtividade e a qualidade de seu trabalho, pois permitem ampliar as ações dos comandos.

Este capítulo não apresenta todos os recursos do SHELL. Uma apresentação completa destes recursos pode ser encontrada no "Manual de Referência Linguagem SHELL/SOX".

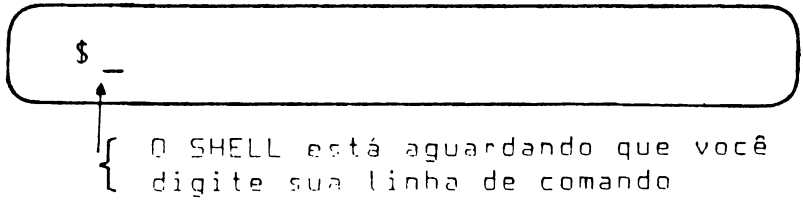
3.1 O QUE É O SHELL ?

O SHELL é um programa que começa a executar a partir do momento em que você abre sua sessão, terminando somente quando você fecha a sua sessão. Ele é responsável por receber linhas de comandos, interpretá-las e ativar os programas que executam os comandos.

3.2 FUNCIONAMENTO BÁSICO DO SHELL

Nesta seção é apresentado o ciclo básico de funcionamento do SHELL. O objetivo é lhe dar fundamentos para uma melhor compreensão dos recursos oferecidos pelo SHELL. Este ciclo consiste nos seguintes passos:

1- Inicialmente, o SHELL emite um pronto no seu terminal (usualmente o caractere "$"), indicando que está aguardando que você digite uma linha de comando. O SHELL passa então a aguardar a sua ação.



2- Você digita a sua linha de comando.

3- Você digita **CR** após a linha de comando.

Esta ação "acorda" o SHELL, avisando-o de que a linha de comando esperada já está disponível. Cabe observar que o SHELL, neste ponto, percebe a linha de comando como um conjunto de caracteres sem nenhuma estrutura.

4- O SHELL passa então a interpretar a linha de comando recebida. Isto consiste em separar os diversos componentes da linha e, a partir deles, tomar as providências necessárias (os componentes de uma linha de comando são separados entre si por um ou mais brancos). Veremos nas seções seguintes quais são os principais componentes de uma linha de comando. Por ora, podemos adiantar que o primeiro componente de uma linha é o nome de um comando. Este comando é um programa que passará a ser executado, como veremos no passo 5.

5- De posse do nome do comando encontrado na linha de comando e dos outros componentes, o SHELL passa o controle para o programa correspondente, aguardando o término do mesmo.

6- O programa correspondente ao comando passa a ser executado, produzindo resultados ou mensagens de erro. Freqüentemente, estes resultados e mensagens de erro são produzidos na tela do seu terminal, como veremos nas seções seguintes.

7- Após o término do programa, o SHELL é "acordado", emitindo em seguida um caractere de pronto como no passo 1, reiniciando o ciclo.

3.3 A LINHA DE COMANDO DO SHELL

A linha de comando do SHELL é o conjunto de caracteres que você digita entre o pronto e **CR**.

Nesta seção é apresentada a estrutura mínima de uma linha de comando, tal como ela é interpretada pelo SHELL. Esta apresentação não abrange todas as possibilidades, mas é básica para a compreensão de outras estruturas mais complexas.

A estrutura mínima de uma linha de comando é:

comando [**arg1**][**arg2**] **...** [**argn**]

Onde:

comando, arg1, arg2, ... , argn são os componentes da linha de comando. Os componentes são separados entre si por um ou mais espaços. Os colchetes indicam componentes opcionais.

comando é o nome do comando. Toda linha tem que conter o nome de um comando. A menor linha reconhecida pelo SHELL contém apenas o nome do comando.

arg1, arg2, ..., argn são argumentos que podem ou não ser opcionais, dependendo do comando.

Os argumentos podem ser:

- Nomes de arquivos;
- Opções;
- Cadeia de caracteres.

Nomes de arquivos

Tomemos como exemplo o comando mkdir. Este comando requer pelo menos um nome de diretório como argumento, podendo aceitar mais de um.

Opções

São argumentos que modificam os efeitos de um comando. A maioria dos comandos permite que você especifique mais de uma opção, modificando assim o comando de várias maneiras.

Por convenção, opções são colocadas imediatamente após o nome do comando. Em geral, os comandos só aceitam opções precedidas de um hífen.

Tomemos como exemplo o comando wc. Este comando exibe o número de linhas, palavras (onde uma palavra é um conjunto de caracteres separados por espaço) e caracteres de um ou mais arquivos especificados como argumentos. Assim, se tivermos os arquivos arq1 e arq2 com os seguintes conteúdos:

arq1	arq2
palavra 1	palavra 3
palavra 2	palavra 4
	palavra 5

teremos o seguinte diálogo:

```
$ wc arq1 arq2     <CR>
     2       4       20  arq1
     3       6       30  arq2
     5      10       50  total
$ _
```

A primeira linha produzida por wc nos indica que arq1 é formado por 2 linhas, 4 palavras e 20 caracteres, a segunda, que arq2 é formado por 3 linhas, 6 palavras e 30 caracteres, enquanto a terceira apresenta os totais.

O comando wc tem as seguintes opções:

Opção	Descrição
-l	conta linhas
-w	conta palavras
-c	conta caracteres

Assim, o comando:

```
wc -l arq1
```

apresentará apenas o número de linhas existentes no arquivo arq1. Ele produz o resultado:

```
2  arq1
```

que nos informa que arq1 tem 2 linhas.

Já o comando:

```
wc -lc arq1
```

nos daria a resposta:

```
2       20  arq1
```

indicando que arq1 é formado por 2 linhas e 20 caracteres.

O comando

```
wc -wl arq1
```

nos daria a resposta:

```
4       2         arq1
```

. Cadeia de caracteres

Tomemos como exemplo o comando grep. Numa de suas formas mais simples, este comando aceita dois argumentos. O primeiro é uma cadeia de caracteres e o segundo é um nome de arquivo. Nesta forma, grep procura a cadeia de caracteres dada pelo primeiro argumento no arquivo cujo nome é dado pelo segundo argumento.

Como exemplo, suponhamos que o conteúdo de um arquivo chamado arq seja:

```
almirante
mirante
antes
```

Se quisermos determinar todas as ocorrências da cadeia de caracteres **mir** no arquivo arq, devemos utilizar o comando:

```
grep mir arq
```

A resposta de grep é:

```
almirante
mirante
```

indicando que a cadeia **mir** se encontra nas palavras **almirante** e **mirante**, existentes no arquivo arq.

3.4 RECURSOS OFERECIDOS PELO SHELL

Nas seções anteriores, apresentamos os conceitos necessários à compreensão dos recursos oferecidos pelo SHELL. Nesta seção, apresentaremos os recursos mais freqüentemente usados e que são responsáveis pela grande flexibilidade e produtividade que podem caracterizar a sua interação com o SOX. Serão vistos os seguintes recursos:

- entrada-padrão, saída-padrão e saída-padrão de erro;
- redirecionamento;
- canalização;
- execução de comandos em plano secundário;
- metacaracteres;
- execução de arquivos de comandos.

3.4.1 ENTRADA-PADRÃO, SAÍDA-PADRÃO E SAÍDA-PADRÃO DE ERRO

Todo comando do SOX está associado a três entidades. Essas entidades são chamadas de entrada-padrão, saída-padrão e saída-padrão de erro. Alguns comandos lêem dados da entrada-padrão, grande parte dos comandos produz resultados na saída-padrão e todos dirigem suas mensagens de erro para a saída-padrão de erro.

Figura 3-1

Cada uma dessas entidades deve estar associada a algum arquivo do SOX para que a operação relativa à entidade (leitura, produção de resultados ou mensagens de erro) possa ser efetivamente realizada.

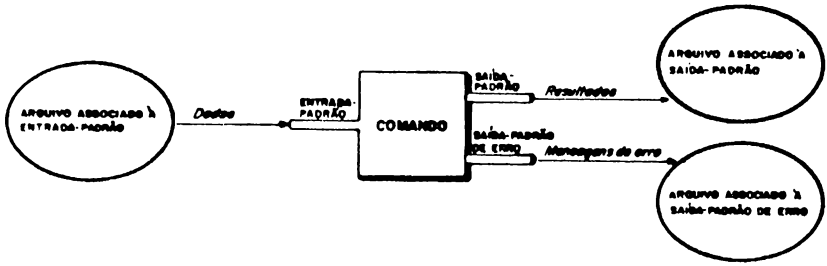


Figura 3-2

Estas associações podem ou não ser especificadas por você na sua linha de comando. Se você especificar alguma associação, o SHELL associará o arquivo especificado à entidade desejada. Nas seções seguintes, veremos como você pode especificar associações de arquivos a estas entidades. As entidades que não tiverem associações especificadas na linha de comando serão associadas ao seu terminal. Chamamos esta não especificação de associação por omissão.

Isto significa que se um comando espera ler dados da sua entrada-padrão e você não especificar qual o arquivo que está a ela associado, então o comando lerá os dados digitados por você no teclado do seu terminal. Por omissão, será assumido que o arquivo associado à entrada-padrão é o terminal. Analogamente, se um comando deve produzir resultados na saída-padrão e você não especificar qual o arquivo a ela associado, então o comando produzirá os resultados na tela do seu terminal. Por omissão, será assumido que o arquivo associado à saída-padrão é o terminal.

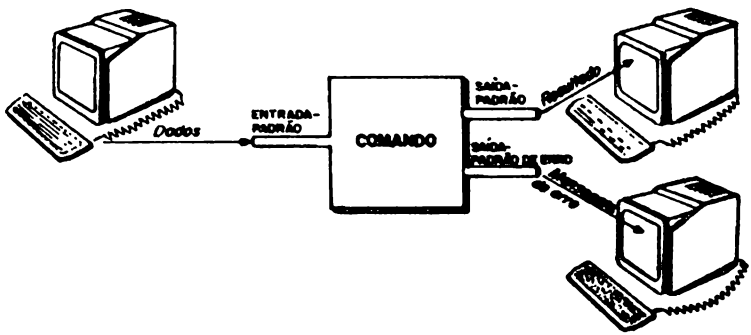
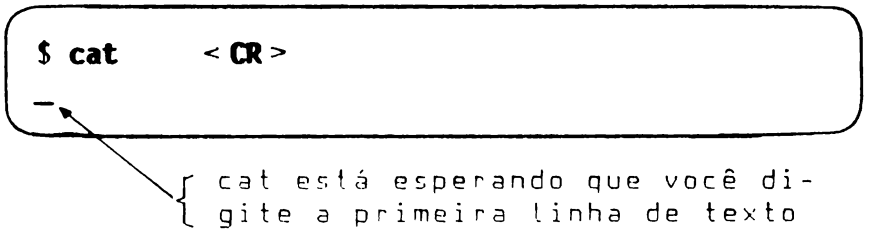


Figura 3-3

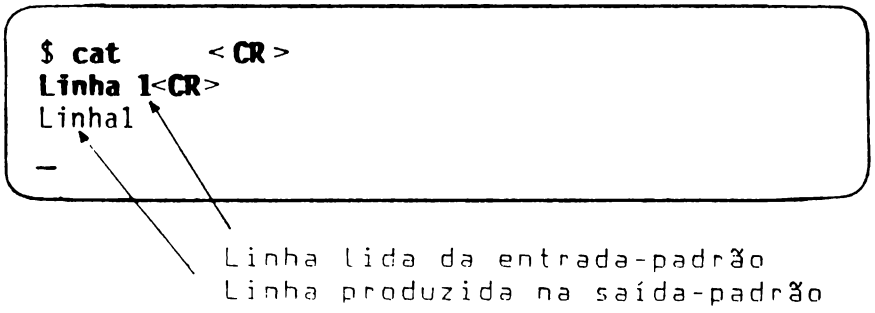
Tomemos como exemplo de associação por omissão de entrada-padrão e saída-padrão, a forma mais simples de utilização do comando cat.

Este comando, quando usado sem argumentos, lê linhas da entrada-padrão e as copia para a saída-padrão.

Se usarmos cat sem especificar os arquivos "padrão", teremos :



Como a entrada-padrão não foi especificada, é assumido que ela será o seu terminal. Você deve então digitar a primeira linha de dados, seguida de **CR**. Teremos, então :



O cursor indica que cat está esperando que você digite a segunda linha seguida de **CR**.

Após a digitação de todas as linhas desejadas, devemos digitar a seqüência **ALT D**, para informar ao comando cat que não há mais linhas a serem digitadas. Assim, teremos :

```
$ cat      <CR>
Linha 1<CR>
Linha 1
Linha 2<CR>
Linha 2
.
.
.
.
.
.
.
Linha final<CR>
Linha final
<ALT> <D>
$ _
```

3.4.2 REDIRECIONAMENTO DE ENTRADA E SAÍDA

Conforme vimos na sessão anterior, as entidades "padrão" de um comando serão associadas ao terminal em caso de omissão. O usuário pode, entretanto, redefinir estas associações, explicitando quais serão os arquivos associados às entidades "padrão". Esta operação de redefinição é chamada de **redireção.**

Nas sessões seguintes veremos como você pode especificar a redireção de cada uma das entidades "padrão".

3.4.2.1 Redirecionamento da Entrada-padrão

Consiste em associar um arquivo à entrada-padrão de um comando. Isto é feito da seguinte forma:

```
comando < arquivo
```

Esta linha indica que a entrada-padrão do comando está associada ao arquivo cujo nome está após o símbolo "<". Isto significa que quando o comando precisar obter dados da entrada-padrão, eles serão obtidos do arquivo cujo nome está após o símbolo "<".

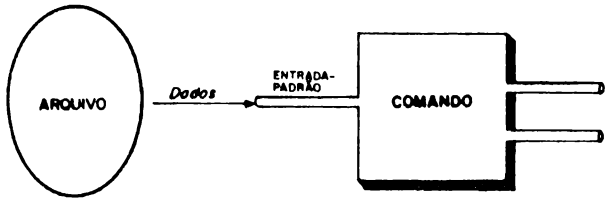


Figura 3-4

Tomemos como exemplo o redirecionamento da entrada-padrão do comando write. Este comando copia mensagens da sua entrada-padrão para o terminal de um usuário cuja identificação é especificada como argumento.

Se usarmos write sem redireção, a entrada-padrão estará associada ao seu terminal. Supondo que queiramos enviar uma mensagem ao usuário cuja identificação é usu2, teremos :

```
$ write usu2      <CR>
_
```

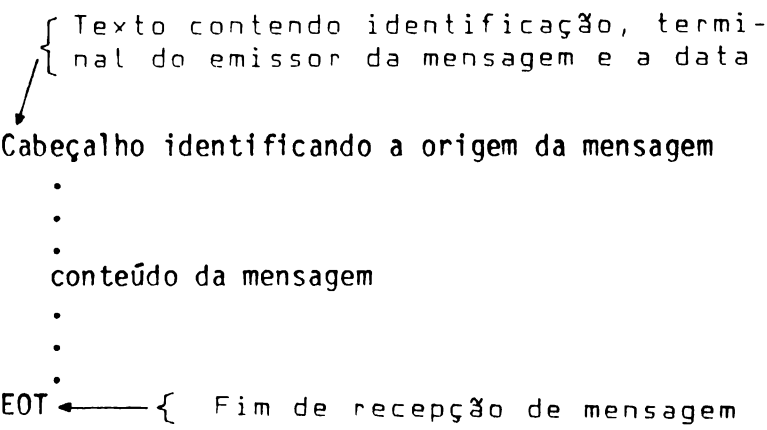
Write está esperando que você digite a primeira linha da mensagem

A mensagem deve ser então digitada, seguida de **ALT D** para indicar fim de mensagem. Assim, teremos :

```
$ write usu2    <CR>
    .
    .
    .
    conteúdo da mensagem
    .
    .
    .
  <ALT><D>
$ _
```

Indica fim de transmissão de mensagem

Na tela do usuário que recebe a mensagem, teremos:



Se a mensagem que desejamos enviar é muito longa ou é enviada freqüentemente, é preferível, ao invés de digitar a mensagem, criar um arquivo que a contenha, e enviá-la, utilizando o redirecionamento da entrada-padrão, associando o arquivo que contém a mensagem à entrada-padrão de write. Supondo que o nome do arquivo que contém a mensagem seja messg, teremos:

```
$ write usu2 < messg      <CR>
$ _
```

Na tela do usuário que recebe a mensagem, teremos:

```
Cabeçalho identificando a origem da mensagem
    .
    .
    .
    conteúdo da mensagem
    .
    .
    .
  EOT
```

Uma observação final sobre o uso do redirecionamento de entrada. A grande maioria dos comandos do SOX não lê dados da entrada-padrão, mas sim de arquivos cujos nomes são argumentos do comando. Por este motivo, não é comum o redirecionamento da entrada-padrão.

3.4.2.2 Redirecionamento da Saída-padrão

Consiste em associar um arquivo à saída-padrão de um comando. Duas situações podem ocorrer :

1. O arquivo associado à saída-padrão é esvaziado antes da execução do comando, perdendo assim, o conteúdo anterior. Você especifica este comportamento da seguinte maneira :

   comando > arquivo

   Esta linha indica que a saída-padrão está associada ao arquivo cujo nome está após o símbolo ">" e que, antes da execução do comando, o arquivo associado à saída-padrão será esvaziado.

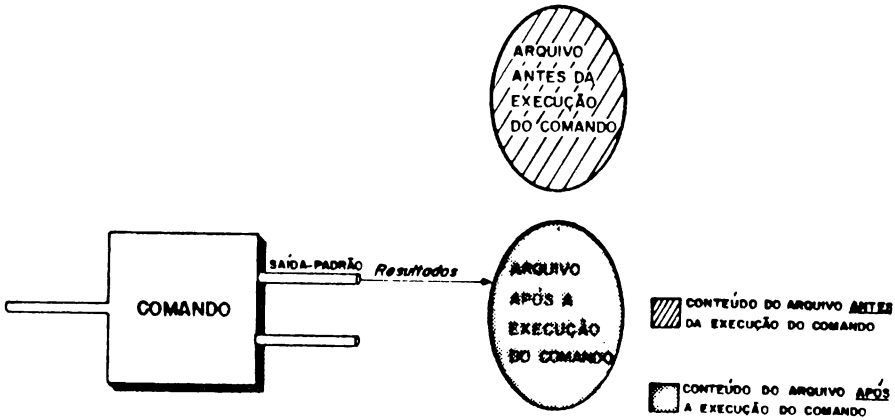


Figura 3-5

2. O conteúdo do arquivo associado à saída-padrão é preservado, e os resultados produzidos pelo comando são gravados após o final do arquivo. Você especifica este comportamento da seguinte maneira:

```
comando >> arquivo
```

Esta linha indica que a saída-padrão do comando está associada ao arquivo cujo nome está após os símbolos ">>" e que os resultados produzidos pelo comando são gravados após o final deste arquivo.

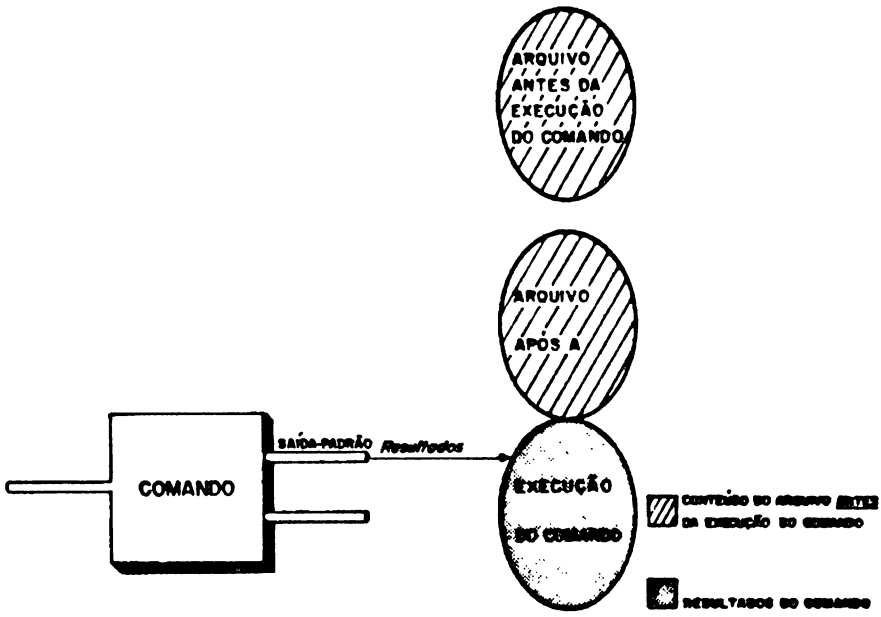


Figura 3-6

Tomemos como exemplo o redirecionamento da saída-padrão do comando cat.

Conforme foi visto, cat sem argumentos copia linhas da entrada-padrão para a saída-padrão.

Para efeito deste exemplo, suponhamos que um arquivo denominado arq contenha as seguintes linhas:

LINHA 1
LINHA 2

Assim, se usarmos cat com redirecionamento da saída-padrão, teremos:

```
$ cat > arq      <CR>
_
```

Em seguida, digitamos as linhas:

LINHA 3<CR>
LINHA 4<CR>
<ALT><D>

Verificando agora o conteúdo do arquivo arq com cat, temos:

```
$ cat arq     <CR>
LINHA 3
LINHA 4
$ _
```

Note que o conteúdo do arquivo arq anterior à execução do primeiro comando cat foi perdido.

Se, ao invés de:

```
$ cat > arq     <CR>
```

tivéssemos:

```
$ cat >> arq     <CR>
```

o conteúdo do arquivo arq após a execução deste comando seria:

```
LINHA 1
LINHA 2
LINHA 3
LINHA 4
```

Note que o conteúdo do arquivo arq anterior à execução de cat foi preservado, sendo os resultados gravados a partir do final do arquivo.

3.4.2.3 Redirecionamento de Saída-padrão de Erro

Consiste em associar um arquivo à saída-padrão de erro de um comando. Isto é feito da seguinte maneira :

```
comando 2 > nome-de-arquivo
```

Esta linha indica que a saída-padrão de erro do comando está associada ao arquivo cujo nome está após os símbolos "2>".

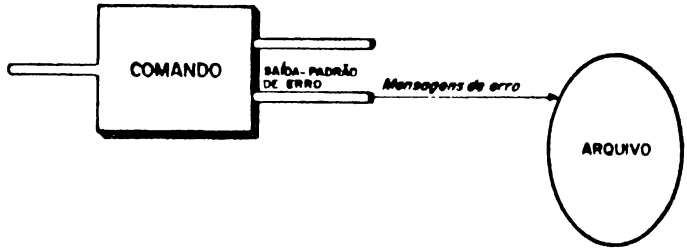


Figura 3-7

Tomemos como exemplo o redirecionamento da saída-padrão de erro do comando cat.

Este comando quando usado com um único argumento, copia as linhas do arquivo cujo nome é dado pelo argumento para a saída-padrão. Se não existir nenhum arquivo com este nome, cat produz uma mensagem correspondente na saída-padrão de erro.

Assim, supondo que não exista nenhum arquivo com o nome fantasma no diretório corrente, teremos o seguinte diálogo:

```
$ cat fantasma      <CR>

cat  : Arquivo ou algum componente no nome
       do caminho não existe
$ _
```

Se quiséssemos que esta mensagem fosse produzida num arquivo denominado erro, teríamos o seguinte diálogo:

```
$ cat fantasma 2> erro      <CR>
$ _
```

Verificando o conteúdo do arquivo erro, temos:

```
$ cat erro      <CR>

cat  : Arquivo ou algum componente no nome
       do caminho não existe
```

3.4.3 CANALIZAÇÃO

Um canal é um mecanismo que faz com que a entrada-padrão de um comando seja associada à saída-padrão de outro comando. Poderíamos representar este mecanismo através do seguinte desenho :

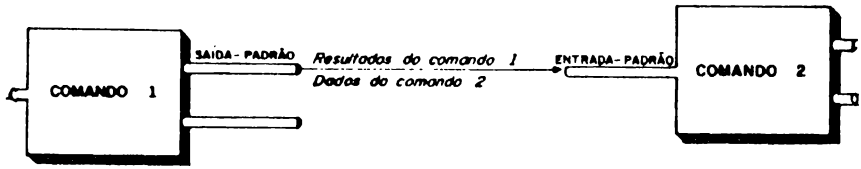


Figura 3-8

Você especifica uma canalização através do símbolo "|". Assim, se desejarmos criar um canal entre dois comandos, teremos :

```
comando1 | comando2
```

Esta linha indica que a saída-padrão do comando1 está associada à entrada-padrão do comando2.

Isto significa que os resultados do comando1 serão os dados de entrada do comando2.

Tomemos como exemplo uma canalização entre os comandos who e wc com opção -l. Esta canalização tem como objetivo determinar quantos usuários estão utilizando o sistema. Vejamos como isto é feito.

O comando who produz na saída-padrão uma listagem dos usuários que estão utilizando o sistema no momento de sua execução. Esta lista é formada de tantas linhas quantos forem os usuários, sendo uma para cada usuário. Cada uma dessas linhas contém o nome do usuário, a identificação do terminal utilizado e a hora em que foi aberta a sessão.

Uma execução típica de who é:

```
$ who
joao          tty0      Mar  22 09:05
carlos        tty1      Mar  22 10:29
lia           tty5      Mar  22 09:47
$ _
```

O comando wc com opção -l produz na saída-padrão o número de linhas do arquivo associado à entrada-padrão.

Consideremos agora a canalização:

```
who | wc -l
```

Esta linha nos indica que a saída-padrão de who está associada à entrada-padrão de wc -l.

Isto significa que wc -l tomará como dados as linhas produzidas por who e irá contá-las. No caso da saída produzida acima por who, teremos:

```
$ who | wc -l
          3
$ _
```

Este resultado nos indica que três usuários estão utilizando o sistema.

Este exemplo pode ser ilustrado pela figura a seguir:

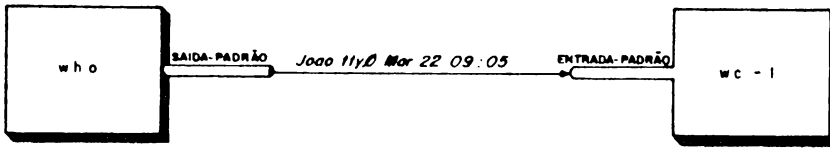


Figura 3-9

Um comando muito útil quando trabalhamos com canais é tee. Ele lê dados do arquivo associado à sua entrada-padrão e produz resultados no arquivo associado à sua saída-padrão. Ele permite que a sua entrada-padrão seja associada à saída-padrão de um comando, e a sua saída-padrão seja associada à entrada-padrão de outro comando, de tal forma que as informações que entram pela sua entrada-padrão sejam copiadas para a sua saída-padrão e para um arquivo cujo nome é dado como argumento de tee. Assim, a linha de comando:

```
comando1 | tee arq | comando2
```

faz com que os resultados do comando1 sejam os dados de entrada do comando2, e que os resultados sejam também copiados no arquivo de nome arq, como ilustra o desenho seguinte.

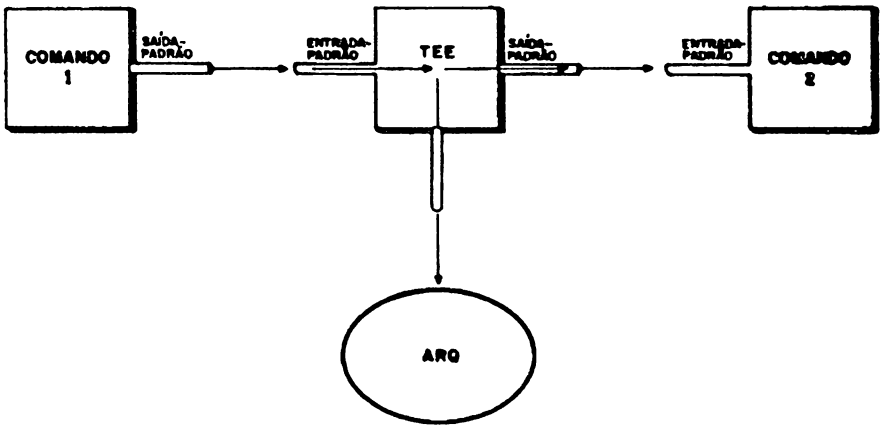


Figura 3-10

Tomemos como exemplo a canalização:

```
who | tee arq | wc -l
```

Ela exibirá na tela do seu terminal o número 3, como no exemplo anterior, e arq terá o seguinte conteúdo:

```
joao        tty0     Mar  22 09:05
carlos      tty1     Mar  22 10:29
lia         tty5     Mar  22 09:47
```

3.4.4 EXECUÇÃO DE COMANDOS EM PLANO SECUNDÁRIO

Um comando cujo tempo de execução é muito longo obriga você a esperar desnecessariamente pelo seu término. O tempo de execução do comando poderia ter sido gasto de forma mais produtiva.

Para evitar este transtorno, o SHELL oferece a possibilidade de executar comandos em plano secundário. Neste tipo de execução de comandos, o SHELL inicia a execução do comando desejado, mas não aguarda o seu término.

Você informa ao SHELL que deseja executar um comando em plano secundário colocando o símbolo "&" após o comando. Assim,

```
comando &
```

solicita ao SHELL que o comando seja ativado, e que após a ativação o SHELL volte a interagir com você, sem que ele espere o término do comando.

A resposta do SHELL a uma solicitação de execução de um comando em plano secundário é um número que identifica a execução do comando.

Este número é usado para que possamos rastrear a execução do comando ou mesmo terminá-lo. O rastreamento pode ser feito através do comando ps, e o término através do comando kill.

Devemos estar atentos ao fato de que qualquer comando executado em plano secundário termina, se for fechada a sessão na qual ele foi iniciado. Para que ele não termine após o fechamento da sessão, é necessário que se use o comando nohup.

Uma aplicação típica de execução de comandos em plano secundário é a ordenação de arquivos extensos.

O comando que ordena arquivos é sort. Na sua versão mais simples, ele produz, na saída-padrão, uma versão ordenada do arquivo cujo nome é seu primeiro argumento. Assim, se tivermos um arquivo extenso chamado, por exemplo, arq_ext, e quisermos ordená-lo, é recomendável que utilizemos o comando:

```
sort arq_ext > arq_sai &
```

Este comando produzirá no arquivo de nome arq_sai uma versão ordenada do arquivo de nome arq_ext. O SHELL iniciará a execução do comando sort, emitirá a identificação da execução do comando e exibirá o caractere de pronto para informar que está esperando a próxima linha de comando. Teríamos então o seguinte diálogo:

```
$sort arq_ext > arq_sai &    <CR>
78
$ _
```

Se quisermos saber em um determinado instante se este comando ainda está sendo executado, devemos utilizar o comando ps. O comando ps exibe na saída-padrão informações sobre todos os programas que estão sendo executados no sistema. Cada linha exibida por ps corresponde à execução de um programa. Cada linha exibe a identificação da execução do programa (coluna PID), a identificação do terminal que iniciou o programa (coluna TTY), o tempo de execução do programa (coluna TIME), e o nome do programa.

No caso do último programa apresentado, basta acharmos uma linha com PID igual a 78. Isto indicará que o comando sort ainda está sendo executado. Assim, podemos ter :

```
$ ps      <CR>
 PID        TTY       TIME     COMMAND
   1         ?        0:01      init
   3        tty0      0:43      -sh
   4        tty1      0:06      -sh
  78        tty2      0:52      sort
$ _
```

A execução deste comando indica que o comando sort ainda está sendo executado.

3.4.5 METACARACTERES

Outra facilidade oferecida pelo SHELL é que você pode usar certos caracteres que tem significado especial em linhas de comandos para referenciar conjuntos de arquivos. Estes caracteres são chamados de metacaracteres. São eles:

```
?     *     [     ]
```

Vejamos a seguir o significado de cada um desses metacaracteres.

. Metacaractere *

Usado para representar zero ou mais ocorrências de qualquer caractere em nomes de arquivos em linhas de comando.

Como exemplo, suponha que no diretório corrente tenhamos os arquivos de nome arq, arq1, arq2, arq3, arq10, arqa e arquivo. O comando:

```
ls arq*
```

nos dará o resultado:

```
arq  arq1  arq10  arq2  arq3  arqa  arquivo
```

Note que o SHELL procurou no diretório corrente, à partir do padrão "arq*", todos os nomes de arquivo que começassem com "arq" e fossem seguidos de zero ou mais caracteres.

. Metacaractere ?

Usado para representar a ocorrência de um caractere qualquer em nomes de arquivos em linhas de comandos.

Como exemplo, suponha que no diretório corrente tenhamos os arquivos de nome arq, arq1, arq2, arq3, arq10, arqa e arquivo. O comando:

```
ls arq?
```

nos dará o resultado:

```
arq1  arq2   arq3   arqa
```

Note que o SHELL procurou no diretório corrente, a partir do padrão "arq?", todos os nomes de arquivo que começassem com "arq" e fossem seguidos de apenas um caractere.

. Metacaracteres []

Usados para representar a ocorrência de um conjunto de caracteres em nomes de arquivos em linhas de comando. Este conjunto de caracteres é especificado entre os metacaracteres "[" e "]", através da menção explícita de cada elemento do conjunto, ou, quando os caracteres forem consecutivos, da menção do menor caractere do conjunto, seguido do caractere "-", seguido do maior caractere do conjunto.

Como exemplo, suponha que no diretório corrente tenhamos os arquivos de nome arq, arq1, arq2, arq3, arq4, arq5, arq10, arq20, arq30, arq40, arq50 e arquivo. O comando:

```
ls arq[123456]
```

nos dará o resultado:

```
arq1  arq2  arq3  arq4  arq5
```

Note que o SHELL procurou, a partir do padrão "arq[123456]", todos os nomes de arquivo que começassem com "arq" e fossem seguidos de um dos caracteres "1", "2", "3", "4", "5" ou "6". Note também que estes caracteres são consecutivos. Por este motivo, podemos utilizar a versão abreviada de [123456], que é [1-6], já que o menor caractere do conjunto desejado é "1", e "6" o maior. Assim, o comando:

```
ls arq[1-6]
```

é equivalente a:

```
ls arq[123456]
```

3.4.6 EXECUÇÃO DE ARQUIVOS DE COMANDOS

Existem situações em que é conveniente utilizarmos um conjunto de comandos para realizar uma certa tarefa. Se necessitarmos repetidamente de um conjunto de comandos, gostaríamos de não ter que digitar os comandos todas as vezes que fôssemos utilizá-los.

Para evitar este transtorno, o SHELL permite que você possa executar comandos agrupados em um arquivo. Arquivos contendo comandos são chamados de arquivo de comandos. Para que você possa executar um arquivo de comandos, você deve criar o arquivo contendo os comandos, dar permissão de execução ao arquivo, e quando quiser executar os comandos nele existentes, basta fornecer o nome do arquivo como sendo um comando SHELL.

Tomemos, como exemplo, um arquivo de comandos que exibe a data atual e o número de usuários que estão utilizando o sistema.

Inicialmente, você deve criar um arquivo, utilizando, por exemplo, o comando edx, com o seguinte conteúdo:

```
echo
echo -n "Data e hora :"
date
echo
echo -n "Número de usuários utilizando o sistema :"
who|wc -l
echo
exit 0
```

Suponha que o nome do arquivo seja informa.

Em seguida, devemos dar permissão de execução a este arquivo. Isto é feito através do comando:

```
$ chmod +x informa     <CR>
$ _
```

O passo seguinte consiste na execução do arquivo de comandos, através do comando:

```
$ informa     <CR>
      .
      .
   resultado do comando
      .
      .
$ _
```

CAPÍTULO 4
COMANDOS SOX

O objetivo deste capítulo é apresentar os comandos disponíveis no SOX em grupos relacionados, de acordo com a sua função. Estes grupos são:

- Controle de acesso;
- Manipulação de terminal;
- Controle de impressora;
- Manipulação de arquivos;
- Manipulação de diretórios e nomes de arquivos;
- Processamento de texto;
- Desenvolvimento de programas;
- Execução de programas;
- Informações de estado;
- Comunicação entre usuários;
- Manipulação de informações;
- Manutenção do sistema;
- Formatação de documentos;
- Controle de atualização da documentação;

- Ambiente da linguagem C;
- Integridade de arquivos com organização seqüencial indexado;
- SCCS (Sistema de Controle de Programas-fonte);
- Outros.

Para cada grupo são apresentados os comandos pertencentes a ele, seguidos de uma breve descrição. A descrição detalhada de cada comando pode ser encontrada no "Manual de Referência Comandos SOX".

Neste capítulo, serão apresentados também, a título de ilustração, os aspectos mais importantes de alguns dos comandos do SOX.

4.1 GRUPOS DE COMANDOS

A seguir são apresentados os grupos de comandos e os seus componentes.

4.1.1 CONTROLE DE ACESSO

newgrp Troca a identificação do grupo de usuário.

passwd Define ou troca a senha associada à identificação do usuário que executa o comando.

su Transforma usuário em usuário privilegiado ou em outro usuário.

4.1.2 MANIPULAÇÃO DE TERMINAL

edcar Editor de caracteres para terminais.

stty Modifica atributos do terminal.

4.1.3 CONTROLE DE IMPRESSORA

lpa Envia mensagens que atuam sobre as saídas em curso e sobre arquivos da fila de spool.

lpd Imprime um arquivo no periférico físico associado à saída lógica.

lpi Associa uma saída lógica a um periférico físico.

lpl Mostra fila de saída do spool.

lpr Inclui um arquivo na fila do spool de saída.

lps Escolhe próximo arquivo para impressão.

4.1.4 MANIPULAÇÃO DE ARQUIVOS

cat Concatena arquivos.

cmp Compara o conteúdo de dois arquivos.

cp Copia arquivos.

csplit Particiona arquivos.

dd Copia arquivos, realizando conversões de códigos e representações de caracteres.

line Lê e imprime linha de arquivo.

pack Compacta arquivos.

pcat Lista arquivos compactados.

split Particiona arquivos.

sum Calcula o checksum de um arquivo e o número de blocos de 1024 octetos ocupados pelo arquivo.

tail Lista trecho de um arquivo.

touch Atualiza datas de acesso e modificação de arquivos.

unlink Desassocia um arquivo de outro.

4.1.5 MANIPULAÇÃO DE DIRETORIOS E NOMES DE ARQUIVOS

basename Remove partes do nome de um caminho.

cd Troca o diretório de trabalho corrente.

chgrp Troca o grupo do arquivo.

chmod Altera as permissões de acesso a arquivos.

chown Troca o dono do arquivo.

dircmp Compara diretórios.

dirname Lista o nome de caminho correspondente ao diretório onde está um arquivo.

find Procura arquivo com características especificadas.

ln	Cria associações para um arquivo.
mkdir	Cria diretórios.
mv	Renomeia arquivos.
rm	Remove arquivos.
rmdir	Remove diretórios vazios.
umask	Estabelece máscara de permissões para criação de arquivos.

4.1.6 PROCESSAMENTO DE TEXTO

edx	Permite a edição de textos.

4.1.7 DESENVOLVIMENTO DE PROGRAMAS

ar	Permite a manutenção de bibliotecas.
as	Montador 68000.
false	Retorna condição diferente de zero.
ld	Ligador.

make	Mantém, atualiza e gera grupos de programas.
nl	Numera as linhas de um arquivo.
nm	Lista informações da tabela de símbolos.
od	Dump em octal, hexadecimal, decimal ou ASCII.
size	Imprime tamanho das sessões de um arquivo objeto.
strip	Remove informações da tabela de símbolos.
time	Informa tempo de execução de processos.
true	Retorna condição zero.

4.1.8 EXECUÇÃO DE PROGRAMAS

chroot	Troca raiz de sistema de arquivos.

echo	Ecoa texto.
env	Define ambiente para execução de programas.
expr	Avalia expressões aritméticas.
kill	Cancela processo.
nice	Muda prioridade de um processo.
nohup	Executa comandos sem interrupção externa.
sh	Interpretador de comandos.
sleep	Suspende a execução de um processo por tempo determinado.
tee	Permite obter resultados intermediários de uma canalização.
test	Avalia expressões.

4.1.9 INFORMAÇÕES DE ESTADO

date Requisita ou insere nova data.

du Estima o espaço ocupado por arquivos.

id Imprime as identificações de usuário e de grupo.

logname Fornece o nome da sessão.

ls Lista conteúdo de diretórios.

ps Informa estados dos processos ativos.

pwd Mostra caminho do diretório de trabalho corrente.

tty Obtém o nome do terminal corrente.

uname Informa o nome do sistema SOX corrente.

who Lista informações sobre usuários ativos no sistema.

4.1.10 COMUNICAÇÃO ENTRE USUÁRIOS

mesg Habilita/desabilita o recebimento de mensagens.

write Envia mensagem a um usuário.

4.1.11 MANIPULAÇÃO DE INFORMAÇÕES

comm Identifica linhas comuns ou linhas diferentes existentes em dois arquivos ordenados.

cut Suprime campos ou colunas de um arquivo.

fgrep Procura por cadeias de caracteres em arquivos.

grep Procura por padrões em arquivos.

join Concatena pares de linhas de dois arquivos que tenham campos idênticos.

paste Intercala linhas de vários arquivos.

sort Ordena e/ou intercala arquivos.

tr Copia um arquivo, substituindo ou retirando caracteres selecionados.

tsort Realiza ordenação topológica.

uniq Informa as linhas repetidas de um arquivo.

wc Conta linhas, palavras e caracteres de arquivos.

4.1.12 MANUTENÇÃO DO SISTEMA

bkpcp Realiza a cópia total de um volume.

clri Transforma conteúdo de BCAs em zeros.

copdos Realiza o transporte de arquivos entre os sistemas MS-DOS/SISNE e SOX.

copsel Realiza cópia seletiva, em disquete, de arquivos em formato padrão COBRA.

cpio Copia arquivos.

devnm Obtém nome do arquivo especial.

df	Informa espaço livre no disco.
formimp	Formata a impressora.
fsck	Realiza a manutenção física de um volume de disco.
grpck	Verifica a consistência das informações contidas no arquivo que descreve os grupos.
killall	Cancela todos os processos do sistema.
link	Cria associações de arquivos.
mkfs	Cria um sistema de arquivos em um volume.
mknod	Cria arquivos especiais.
mount	Monta um sistema de arquivos.

mvdir	Renomeia diretórios.
pwck	Verifica consistência das informações contidas no arquivo que descreve os usuários do sistema.
setmnt	Cria tabela de volumes montados.
sync	Grava buffers do sistema.
tar	Salva e/ou restaura arquivos em fita.
umount	Desmonta um sistema de arquivos.
wall	Envia mensagens a todos os usuários.

4.1.13 CONTROLE DE ATUALIZAÇÃO DA DOCUMENTAÇÃO

am	Permite a atualização de manuais.

4.1.14 AMBIENTE DA LINGUAGEM C

cc Compilador C.

cxref Obtém referências cruzadas.

4.1.15 INTEGRIDADE DE ARQUIVOS COM ORGANIZAÇÃO SEQUENCIAL INDEXADO

manut Verifica/consiste arquivo com organização seqüencial indexado.

recutra Recupera transação.

4.1.16 SCCS (SISTEMA DE CONTROLE DE PROGRAMAS-FONTE)

admin Cria e administra um arquivo SCCS.

val Valida um arquivo SCCS.

what Identifica um arquivo SCCS.

4.1.17 OUTROS

banner Amplia caracteres.

cal Imprime calendário.

calendar Imprime agenda.

4.2 APRESENTAÇÃO DE ALGUNS COMANDOS

A seguir, apresentamos os aspectos mais importantes de alguns dos comandos SOX.

Na sintaxe dos comandos apresentados, utilizaremos os caracteres para indicar que o argumento que se encontra entre eles é opcional.

4.2.1 ORDENANDO ARQUIVOS COM O COMANDO sort

Muitas vezes precisamos que as informações estejam ordenadas. Como exemplo, podemos citar uma lista telefônica. O que seria da operação de encontrar um número de telefone de um determinado assinante se a lista não estivesse apropriadamente ordenada?

Assim, certas informações não fazem sentido se não estiverem ordenadas. Deste modo, no seu dia-a-dia, você necessitará ordenar vários de seus textos. Os exemplos são diversos: uma lista telefônica particular por nomes, uma lista de projetos por orçamento, eventos por data, etc.

Para realizar a operação de ordenar um arquivo, o SOX oferece o comando sort.

Os arquivos podem ser ordenados de duas formas: por ordem alfabética e por ordem numérica. Qualquer que seja a forma 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?

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

Vamos então descrever como especificar seus campos de ordenação e o significado das opções do comando sort. O nosso objetivo aqui não é sermos completo. Por isto, não descreveremos todas as opções do comando, mas somente as mais importantes.

Para efeito dos exemplos seguintes, suponha que você tenha uma pequena lista telefônica armazenada no arquivo de nome lista. Você pode criá-lo, se desejar, utilizando o editor de tela edx. Seu conteúdo é exibido a seguir, através do comando cat.

```
$ cat lista     <CR>
Alzira Oliveira                          - 3311100
Claudia Serenario                        -2211341
Edio Nogueira                            -4010081
Felipe de Lima Saldanha                  -2222014
Henrique Fausto Ferreira                 -3211384
Luiz Arlindo Junqueira                   -1681012
Moema Cortes Senra                       -3221514
Paula Monteiro de Barros                 -4011984
Rogerio Ferreira                         -4011305
Maria Angelica Fausto de Souza           -3221515
$ _
```

Observe que o nome Maria Angelica Fausto de Souza foi colocado fora de seu devido lugar na lista. É claro que você poderia fazer as modificações necessárias através do editor edx, mas não é isto que nos interessa neste momento. O que nos interessa agora são os recursos do SOX para resolver problemas de ordenação de textos. Pensando melhor a este respeito, o comando edx não seria o mais indicado para resolver a maioria desses problemas. Imagine um texto de mil linhas, desordenado (ou até parcialmente ordenado) e o trabalho enorme que seria ordená-lo por meio do edx. Isto acontece simplesmente porque o edx não foi criado para tal tipo de operação.

Para resolver este nosso problema, vamos estudar o comando sort.

4.2.1.1 Ordenação Alfabética

Vamos agora aplicar o comando sort ao arquivo lista, fazendo:

```
$ sort lista     <CR>
Alzira Oliveira                    - 3311100
Claudia Serenario                  -2211341
Edio Nogueira                      -4010081
Felipe de Lima Saldanha            -2222014
Henrique Fausto Ferreira           -3211384
Luiz Arlindo Junqueira             -1681012
Maria Angelica Fausto de Souza     -3221515
Moema Cortes Senra                 -3221514
Paula Monteiro de Barros           -4011984
Rogerio Ferreira                   -4011305
$ _
```

Vamos entender 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 para o comando sort, este entende que todo o texto deve ser ordenado alfabeticamente, linha a linha.

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, por exemplo, aceitamos que a letra **a** é menor que a letra **e**, ou então que a letra **c** é maior que a letra **a**. Mas como comparar uma letra com um número ou com um sinal menos, por exemplo?

Para a ordenação, 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), o . (ponto) e o espaço, temos a seguinte ordem de valores:

espaço + . 0 ... 9 A ... Z - a ... z

Exemplo de ordem alfabética ascendente:

```
 123
+123
-12.3
Maria
maria
```

Voltando ao exemplo dado anteriormente sobre o comando sort, resta ainda uma observação sobre a operação realizada: não foi especificado como o comando sort deveria salvar o texto agora ordenado. Nestes casos, ele entende que você deseja que ele exiba o texto ordenado na saída-padrão, que por omissão é a sua tela. Desta forma, deve estar claro que o arquivo lista não sofreu qualquer alteração. Esta informação pode ser comprovada utilizando o comando cat para exibir o conteúdo do arquivo lista.

```
$ cat lista     <CR>
Alzira Olivcira                    - 3311100
Claudia Serenario                  -2211341
Edio Nogueira                      -4010081
Felipe de Lima Saldanha            -2222014
Henrique Fausto Ferreira           -3211384
Luiz Arlindo Junqueira             -1681012
Moema Cortes Senra                 -3221514
Paula Monteiro de Barros           -4011984
Rogerio Ferreira                   -4011305
Maria Angelica Fausto de Souza     -3221515
$ _
```

O que ocorreu é que o arquivo lista foi a entrada para o sort, e o conteúdo ordenado deste arquivo foi a sua saída.

Caso você deseje, a saída do comando sort pode ser redirecionada para um arquivo qualquer, tal como lista_ordem por exemplo, fazendo:

```
$ sort lista >lista_ordem   <CR>
$ cat lista_ordem           <CR>
Alzira Oliveira                    - 3311100
Claudia Serenario                  -2211341
Edio Nogueira                      -4010081
Felipe de Lima Saldanha            -2222014
Henrique Fausto Ferreira           -3211384
Luiz Arlindo Junqueira             -1681012
Maria Angelica Fausto de Souza     -3221515
Moema Cortes Senra                 -3221514
Paula Monteiro de Barros           -4011984
Rogerio Ferreira                   -4011305
$ _
```

4.2.1.2 Ordenação Numérica

A ordenação numérica seqüencia números (em ordem ascendente ou descendente) pelos 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.

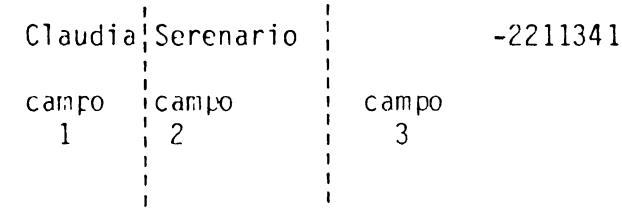
No caso de termos apenas números positivos, não importando se a ordenação é alfabética ou numérica, deve ser levado em consideração que o caractere espaço é menor que o caractere + (sinal mais). Assim, como exemplo de ordenação alfabética ou numérica, temos:

```
 12.3
 123
+123
```

4.2.1.3 Campo de Ordenação

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

Um campo para o comando sort é um conjunto de caracteres consecutivos delimitado por espaço, se nenhum outro for especificado, ou **CR**. Dizemos então que o caractere espaço é, a princípio, um separador de campos, desde que nenhum outro seja especificado. Um campo começa imediatamente após o separador, mesmo que o próximo caractere seja também um espaço. Tomemos, como exemplo, a seguinte linha do arquivo lista:



Você pode optar por ordenar o texto a partir de um determinado campo até o final da linha, ou somente por um campo em particular, ou mesmo a partir de um caractere qualquer de um determinado campo. Você pode, inclusive, combinar estas opções, conforme veremos a seguir.

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

Para especificar ao comando sort que você quer ordenar somente pelo campo 2, você deve utilizar 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 especificar +1.3 como opção, estará dizendo ao comando sort: salte o primeiro campo e salte os três primeiros caracteres do campo 2.

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

Exemplos de opções que podem ser especificadas:

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

4.2.1.4 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.

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 do comando sort serão discutidas ao longo dos vários exemplos que se seguem.

A especificação dos campos para ordenação deve ser feita tal como mostrada na seção anterior.

Conforme podemos verificar no formato do comando sort, podem existir vários arquivos de entrada para o sort. Veremos também 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 produzida no arquivo associado à saída-padrão.

Vamos agora às opções do comando sort.

4.2.1.5 A Opção -t

Tratemos do exemplo de ordenar o arquivo lista por número de telefone. Já sabemos que devemos especificar o campo correspondente aos números de telefone. Como fazê-lo? Pelo que aprendemos e verificando o arquivo lista, na primeira e segunda linhas ele seria o campo 4, na terceira linha ele seria o campo 3, na quarta linha, o campo 5, na quinta linha, o campo 4. 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 - (traço). Sua colocação antes dos números foi, como você vê, intencional. Assim, usaremos a opção -t, seguida de - (traço), que indicará o novo separador de campos - (traço). 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>
Alzira Oliveira                    - 3311100
Luiz Arlindo Junqueira             -1681012
Claudia Serenario                  -2211341
Felipe de Lima Saldanha            -2222014
Henrique Fausto Ferreira           -3211384
Moema Cortes Senra                 -3221514
Maria Angelica Fausto de Souza     -3221515
Edio Nogueira                      -4010081
Rogerio Ferreira                   -4011305
Paula Monteiro de Barros           -4011984
$ _
```

4.2.1.6 A Opção -n

A opção -n 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 um único campo de ordenação, tanto faz especificar -n como global ou local.

```
$ sort -t- +1n lista      <CR>
Luiz Arlindo Junqueira                -1681012
Claudia Serenario                     -2211341
Felipe de Lima Saldanha               -2222014
Henrique Fausto Ferreira              -3211384
Moema Cortes Senra                    -3221514
Maria Angelica Fausto de Souza        -3221515
Alzira Oliveira                       - 3311100
Edio Nogueira                         -4010081
Rogerio Ferreira                      -4011305
Paula Monteiro de Barros              -4011984
$ _
```

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

4.2.1.7 A Opção -b

A opção -b, 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.

```
$ sort -t- +1b lista     < CR >
Luiz Arlindo Junqueira                -1681012
Claudia Serenario                     -2211341
Felipe de Lima Saldanha               -2222014
Henrique Fausto Ferreira              -3211384
Moema Cortes Senra                    -3221514
Maria Angelica Fausto de Souza        -3221515
Alzira Oliveira                       - 3311100
Edio Nogueira                         -4010081
Rogerio Ferreira                      -4011305
Paula Monteiro de Barros              -4011984
$ _
```

Devemos observar aqui que, embora os resultados dos dois últimos exemplos sejam os mesmos, o procedimento do comando sort foi diferente em cada caso.

Vamos agora ordenar por números de telefone, sem considerar os prefixos, ou seja, os três primeiros números.

```
$ sort -t- +1.3b lista      <CR>
Edio Nogueira                          -4010081
Luiz Arlindo Junqueira                 -1681012
Alzira Oliveira                        - 3311100
Rogerio Ferreira                       -4011305
Claudia Serenario                      -2211341
Henrique Fausto Ferreira               -3211384
Moema Cortes Senra                     -3221514
Maria Angelica Fausto de Souza         -3221515
Paula Monteiro de Barros               -4011984
Felipe de Lima Saldanha                -2222014
$ _
```

4.2.1.8 A Opção -r

A opção -r, global ou local, indica para o comando sort que a ordenação é descendente, ou seja, o maior primeiro, o menor por último.

Vamos ordenar, a seguir, o arquivo lista por ordem descendente de números de telefone, fazendo:

```
$ sort -t- +1br lista      <CR>
Paula Monteiro de Barros               -4011984
Rogerio Ferreira                       -4011305
Edio Nogueira                          -4010081
Alzira Oliveira                        - 3311100
Maria Angelica Fausto de Souza         -3221515
Moema Cortes Senra                     -3221514
Henrique Fausto Ferreira               -3211384
Felipe de Lima Saldanha                -2222014
Claudia Serenario                      -2211341
Luiz Arlindo Junqueira                 -1681012
$ _
```

4.2.1.9 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 recebe as linhas de seu texto do arquivo associado à entrada-padrão, que, por omissão, é o terminal, até que você digite **ALT D**, que indicará para ele o fim de seu texto. A partir daí, o comando sort fará a ordenação do texto digitado.

Cada linha do nosso texto constará dos seguintes campos: departamento, nome, divisão, área. O separador é o caractere espaço. 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>
<ALT><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.

4.2.1.10 A Opção -m

Se você tiver vários arquivos de entrada e quiser intercalá-los num único arquivo de saída, poderá usar a opção -m, sempre global. Para que isto funcione corretamente é necessário, no entanto, que os arquivos de entrada estejam previamente ordenados.

Crie três arquivos de números ordenados. Por exemplo, o primeiro arquivo, de nome arq1, com os números 1, 5, 7 e 8; o segundo arquivo, de nome arq2, com os números 3 e 9; o terceiro arquivo, de nome arq3, com os números 2, 4, 6, 7 e 9. Vamos agora intercalá-los com a utilização do comando sort, opção -m.

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

4.2.1.11 A Opção -u

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

Vamos refazer o exemplo apresentado na opção -m, evitando a duplicação de linhas.

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

4.2.2 REUNINDO DADOS ESTATISTICOS DE UM TEXTO COM O COMANDO wc

Em algumas situações pode ser necessário obter vários dados estatísticos de um texto, tais como o número de linhas, o número de palavras e o número de caracteres. Você pode obter todas essas informações utilizando o comando wc.

A forma geral deste comando é:

wc [-lwc] [arquivos de entrada]

A opção -l informa o número de linhas; a opção -w informa o número de palavras; a opção -c informa o número de caracteres. Os valores correspondentes às opções especificadas são informadas na ordem em que elas são indicadas.

Se nenhuma opção é indicada, são informados, nesta ordem, o número de linhas, o número de palavras e o número de caracteres.

Para exemplificarmos o comando wc é necessário que você crie o arquivo de nome arq2 com o seguinte conteúdo:

```
$ cat arq2      <CR>
3
9
$ _
```

Vamos agora utilizar o comando wc para obter informações a respeito do número de linhas, palavras e caracteres deste arquivo.

```
$ wc arq2    <CR>
       2       2       4  arq2
$ _
```

O primeiro valor informado (ou seja, 2) indica que o arquivo contém duas linhas. O segundo valor (ou seja, 2) indica que o arquivo contém duas palavras (a saber: 3 e 9). O terceiro valor (ou seja, 4) indica o número de caracteres do arquivo. Neste ponto, deve ser observado que o caractere de fim de linha (CR) conta como um caractere do arquivo, devendo, portanto, ser contabilizado neste total. Assim, como o arquivo possui duas linhas, temos que levar em consideração os dois caracteres CR que indicam o fim de cada linha. A quarta informação dada pelo comando wc diz respeito ao nome do arquivo em questão.

Como outros exemplos, temos:

```
$ 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
$ _
```

Podem existir vários arquivos de entrada: quando isto acontece, o SOX informa também um total geral, de todos os arquivos de entrada, sob o rótulo "total". Assim, como exemplo, supondo-se que os arquivos arq1, arq2 e arq3 possuem o conteúdo exibido a seguir, temos o seguinte resultado com a utilização do comando wc:

```
$ cat arq1                    < CR >
1
2
3
4
$ cat arq2                    < CR >
3
9
$ cat arq3                    < CR >
1
2
3
4
5
$ wc arq1 arq2 arq3           < CR >
       4       4       8  arq1
       2       2       4  arq2
       5       5      10  arq3
      11      11      22  total
$ _
```

Se nenhum arquivo de entrada é especificado, o comando espera pelo seu texto no arquivo associado à entrada-padrão e informa as estatísticas do texto.

```
$ wc        <CR>
duas palavras<CR>
<ALT><D>
        1       2       14
$ _
```

4.2.3 CALENDÁRIO E AGENDA ELETRÔNICOS COM OS COMANDOS cal E calendar

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 em certas datas.

4.2.3.1 Imprimindo um Calendário com o Comando cal

O comando cal imprime um calendário para todo o ano especificado ou somente para um mes em particular.

O formato geral deste comando é:

```
cal [ -i ] [[ mes[-mes]] ano]
```

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").

No caso de nenhum argumento ser utilizado, o comando cal exibe o calendário para o mês corrente.

A opção -i indica que o calendário será exibido através da impressora.

Como exemplo de utilização do comando cal, vamos exibir o calendário para o mês de maio do ano de 1987.

```
$ 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
```

4.2.3.2 Controlando a Agenda com o Comando calendar

Imaginemos como você cuidaria de se lembrar de seus compromissos ou de suas tarefas. Você anotaria todas as atividades na sua agenda, nas datas certas. 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 o comando calendar funcione, você precisa registrar suas anotações em um arquivo, através do editor edx, por exemplo. Este arquivo é usualmente chamado calendar. É necessário que você anote, para cada atividade, sua data, no formato mês/dia. 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). São aceitos nomes de mês abreviados, desde que não haja ambigüidade. A seguir, exemplos de datas válidas:

```
Abr.29
Abril 29
4/29
```

Como exemplos de datas não-válidas podemos citar "29 abril" e "4-29".

O comando calendar consulta o arquivo de mesmo nome, calendar, no seu diretório de trabalho corrente, e imprime as linhas que contêm as datas de hoje e amanhã.

A forma de proceder do comando calendar é procurar sequencialmente, 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.

Como exemplo, vamos criar o arquivo calendar com o seguinte conteúdo:

```
$ cat calendar      <CR>
4/28 :  9:30    Reuniao com Vendas
4/29 : manha    Telefonar para Faturamento sobre
4/29 :          Fatura no.5622
4/29 : 10:30    Visita ao Depto. de Compras
4/29 : 14:30    Reuniao com o gerente do CPD
4/29 : 17:00    Audiencia com o Diretor-Presidente
4/29 :          para tratar da situacao de Vendas
4/30 : 14:00    Almoco com pessoal de vendas na
4/30 :          Cantina do Manoel
4/30 : 10:00    Viagem a Sao Paulo
4/31 : tarde    Medico
$ _
```

A seguir, utilizando o comando calendar e supondo-se que a data do dia seja 29 de abril, temos:

```
$ calendar      <CR>
4/29 : manha    Telefonar para Faturamento sobre

4/29 :          Fatura no.5622

4/29 : 10:30    Visita ao Depto. de Compras

4/29 : 14:30    Reuniao com o gerente do CPD

4/29 : 17:00    Audiencia com o Diretor-Presidente

4/29 :          para tratar da situacao de Vendas

4/30 : 14:00    Almoco com pessoal de vendas na

4/30 :          Cantina do Manoel

4/30 : 10:00    Viagem a Sao Paulo

$ _
```

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.

4.2.4 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 conectado ao sistema naquele momento.

A forma geral do comando write é a seguinte:

```
write usuário [tty]
```

Onde:

- usuário é a identificação de um usuário do SOX;
- tty é a identificação do terminal utilizado pelo usuário que receberá a mensagem.

O comando write transmite as linhas da mensagem, uma a uma, para a tela do terminal do destinatário.

O comando write obtém a mensagem do arquivo associado à entrada-padrão, que pode ser o seu terminal, por omissão, ou um arquivo qualquer, caso você tenha usado do redirecionamento da entrada-padrão. 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).

Quando a entrada-padrão está associada ao seu terminal, a mensagem é composta de linhas digitadas através do teclado do seu terminal, terminadas com **CR**. 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 **ALT D**. Ao fazê-lo, o pronto do SHELL volta a aparecer.

Se você deseja enviar uma mensagem a um usuário qualquer, você deve verificar se ele está conectado com o SOX.

Suponha, por exemplo, que você queira enviar uma mensagem para um usuário que tenha identificação joão. Inicialmente, você deve verificar se joão está conectado com o SOX. Para isso, você deve utilizar o comando who.

```
$ who <CR>
antonio          ttj03        17:50
andre            ttj02        17:53
joao             ttj00        17:59
$ _
```

Como joão está conectado (veja a terceira linha da resposta de who), podemos enviar-lhe a mensagem desejada.

```
$ write joao <CR>
Gostaria de conversar com voce sobre o livro<CR>
<ALT><D>
$ _
```

Vejamos o que foi enviado para o terminal de joão:

```
Mensagem de eu (tty02) Seg Mai 08:52:26  1988

Gostaria de conversar com voce sobre o livro
EOT
```

A expressão EOT, no terminal de joão, corresponde ao <**ALT**><**D**> que você digitou.

Se você quiser enviar uma mensagem para um usuário que está conectado em vários terminais, você tem duas possibilidades: a primeira identificando o terminal e a segunda não identificando o terminal. Neste último caso, a mensagem irá para o terminal de menor número de identificação.

Suponha, a título de exemplo, que queiramos enviar uma mensagem para o usuário cuja identificação seja pesquisa. Para isso, vejamos se pesquisa está conectado.

```
$ who    <CR>
.
.
.
pesquisa   tty10   09:13
.
.
.
pesquisa   tty09   09:53
.
.
pesquisa   tty05   10:22
$ _
```

Como os usuários que têm identificação pesquisa estão conectados em 3 terminais com identificações tty05, tty09 e tty10, o comando

```
write pesquisa
```

enviará a mensagem para o usuário que está utilizando o terminal de identificação tty05, pois este tem a menor identificação dentre as três possíveis. Se quisermos enviar mensagens aos outros usuários com identificação pesquisa, devemos mencionar no comando write a identificação dos terminais por eles utilizados. Teríamos assim:

```
write pesquisa tty09
```

ou

```
write pesquisa tty10
```

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 o destinatário recebeu sua mensagem, pois não houve um diálogo entre você e ele.

- As linhas da mensagem que você enviou são exibidas no terminal do destinatário à medida que são recebidas, independente do que esteja sendo exibido neste terminal. Isto implica em que na tela do terminal do destinatário, as linhas da mensagem podem estar intercaladas com as linhas do que está sendo exibido no momento da recepção.

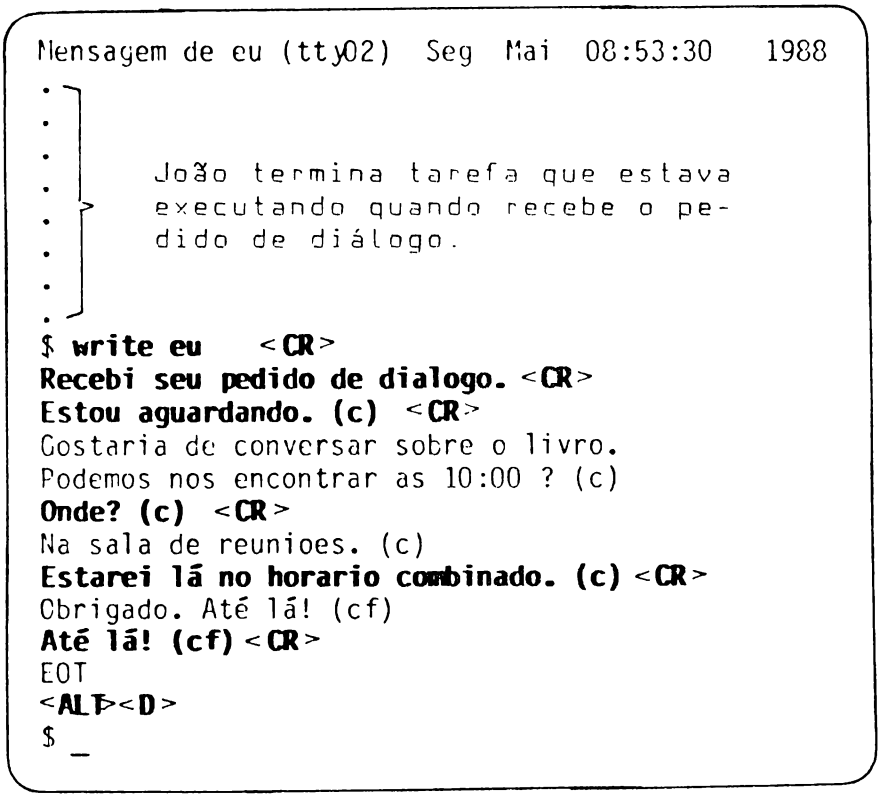
Quanto ao primeiro problema, ele pode ser resolvido pelo estabelecimento de um diálogo entre você e o destinatário. Este diálogo consiste em uma troca de mensagens entre você e seu interlocutor. Como ocorre este diálogo? A simples digitação do comando write faz aparecer o cabeçalho "Mensagem de..." na tela do destinatário. Se este desejar estabelecer o diálogo, ele deve utilizar também o comando write para enviar a sua resposta. Deste modo, para começar a dialogar, você deve esperar até que o destinatário confirme que está pronto para receber sua primeira mensagem.

O comando write transmite linha a linha, isto é, após o **CR** que finaliza cada linha, esta é transmitida. Uma mensagem pode ocupar várias linhas. Como, então, 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 câmbio) para terminar uma mensagem e (cf) (de câmbio final) para encerrar o diálogo.

Vamos então reproduzir um diálogo com o comando write, mostrando primeiro a sua tela e depois a do destinatário, que, vamos supor, se chame joão.

```
$ write joao    <CR>
Mensagem de joao (tty03)  Seg  Mai  08:55:01   1988
Recebi seu pedido de dialogo.
Estou aguardando. (c)
Gostaria de conversar sobre o livro. <CR>
Podemos nos encontrar as 10:00 ? (c)<CR>
Onde? (c)
Na sala de reunioes. (c) <CR>
Estarei lá no horario combinado. (c)
Obrigado. Até lá! (cf) <CR>
Até lá! (cf)
EOT
<ALT><D>
$ _
```

Agora a tela de joão:



4.2.5 FECHANDO-SE PARA MENSAGENS COM O COMANDO mesg

Existem ocasiões em que o destinatário não deseja receber mesagens enviadas através do comando write. O comando mesg tem como objetivo habilitar ou desabilitar a recepção destas mensagens. O formato geral do comando mesg é:

```
mesg [-n] [-y]
```

mesg com argumento -n impede a recepção de mensagens enviadas através do comando write, isto é, torna você inapto para receber estas mensagens.

mesg com argumento -y estabelece que você está aberto (apto) a receber mensagens de outros usuários enviadas através do comando write.

mesg sem argumento mostra seu estado de aptidão corrente.

Assim, podemos ter o seguinte diálogo:

```
$ mesg -n   <CR>
$ mesg      <CR>
inapto
$ mesg -y   <CR>
$ mesg      <CR>
apto
```

Quando você se conecta ao SOX, você está apto a receber mensagens enviadas através do comando write.

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 : Permissão negada para o modo de acesso
        feito no arquivo
```

CAPÍTULO 5
ESTRUTURA DO SOX

Este capítulo apresenta os componentes do SOX. Ele se divide em três sessões: a primeira (Introdução) apresenta os conceitos básicos envolvidos na compreensão da estrutura do SOX; a segunda (Arquitetura do SOX) apresenta a condensação dos conceitos apresentados na seção anterior; a terceira (Exemplo) ilustra, através de um exemplo, o funcionamento da estrutura do SOX.

5.1 INTRODUÇÃO

O SOX é constituído de três componentes distintos:

- Ambiente da Máquina Virtual (AMV);
- Ambiente da Máquina Real (AMR);
- Núcleo.

5.1.1 AMBIENTE DA MÁQUINA VIRTUAL (AMV)

É o conjunto das entidades denominadas Máquinas Virtuais.

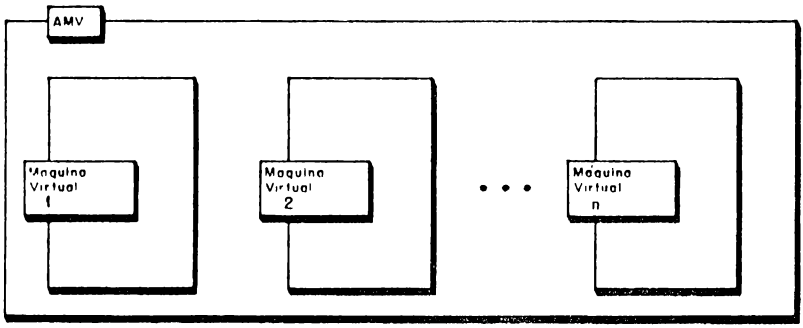


Figura 5-1

Cada Máquina Virtual (MV) é constituída de duas componentes:

. Aplicação;

. Servidor.

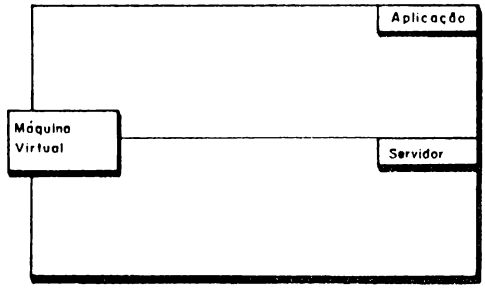


Figura 5-2

5.1.1.1 Aplicação

A Aplicação pode ser um programa escrito pelo usuário, um comando do SOX ou um arquivo de comandos. A Aplicação, durante sua execução, necessita de serviços os quais não é capaz de executar. Exemplos desses serviços são criação, abertura, leitura, gravação de arquivos, criação e destruição de Máquinas Virtuais. Quando a Aplicação necessita da execução de alguns desses serviços, ela emite um pedido de execução de serviço para a outra componente da Máquina Virtual, o Servidor. Esses pedidos são denominados Chamadas ao Servidor (CSERV). Imediatamente após a emissão da CSERV, o Servidor começa a executar e a Aplicação passa a aguardar o término da execução da CSERV.

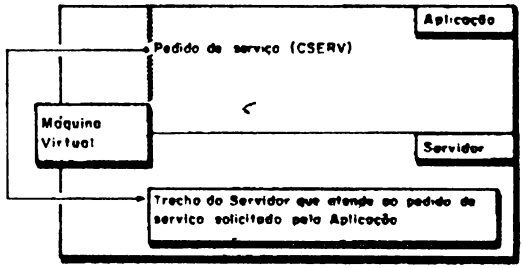


Figura 5-3

5.1.1.2 Servidor SOX

O Servidor do SOX é o componente do SOX que atende, ou seja, serve aos pedidos de serviço das Aplicações (daí o seu nome: SERVIDOR). Ele é igual em todas as Máquinas Virtuais. É também a interface entre a Aplicação e as entidades que controlam o Hardware. Isto significa que, ao atender a uma CSERV, se o Servidor necessitar executar uma ação que dependa da manipulação de algum componente de Hardware, ele emitirá um pedido de serviço à entidade que controla o componente.

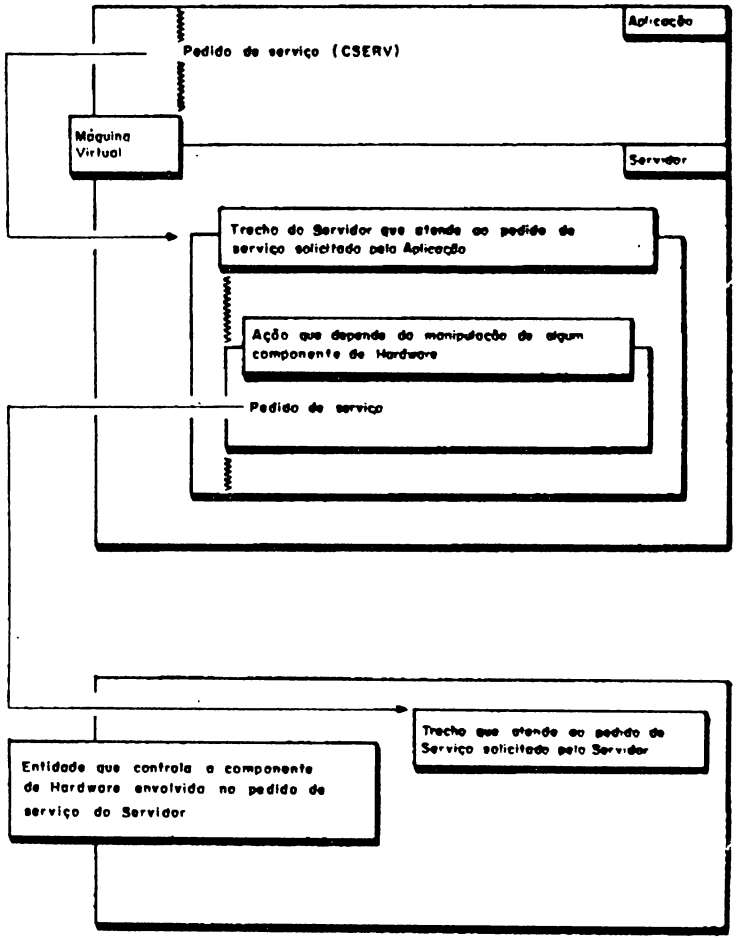

Aplicação
Pedido de serviço (CSERV)
Máquina Virtual
Servidor
Trecho do Servidor que atende ao pedido de serviço solicitado pela Aplicação
Ação que depende da manipulação de algum componente de Hardware
Pedido de serviço
Trecho que atende ao pedido de Serviço solicitado pelo Servidor
Entidade que controla o componente de Hardware envolvido no pedido de serviço do Servidor

Figura 5-4

Os pedidos de serviço solicitados pelo Servidor são feitos de tal forma que não precisem ser alterados nos diversos tipos de equipamento e configurações em que o SOX pode ser executado. Por isso, dizemos que o Servidor não depende do Hardware sobre o qual o SOX esteja executando.

5.1.2 AMBIENTE DA MÁQUINA REAL (AMR)

É o conjunto das entidades denominadas Processos da Máquina Real.

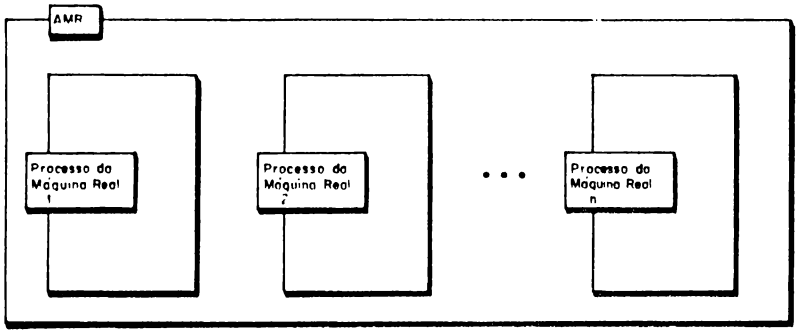


Figura 5-5

Os Processos da Máquina Real (PMR) têm como principal objetivo controlar e manipular os componentes de Hardware sobre o qual o SOX executa. Eles recebem pedidos de serviço do Servidor. Durante o atendimento destes pedidos, eles emitem comandos para os componentes de Hardware que controlam.

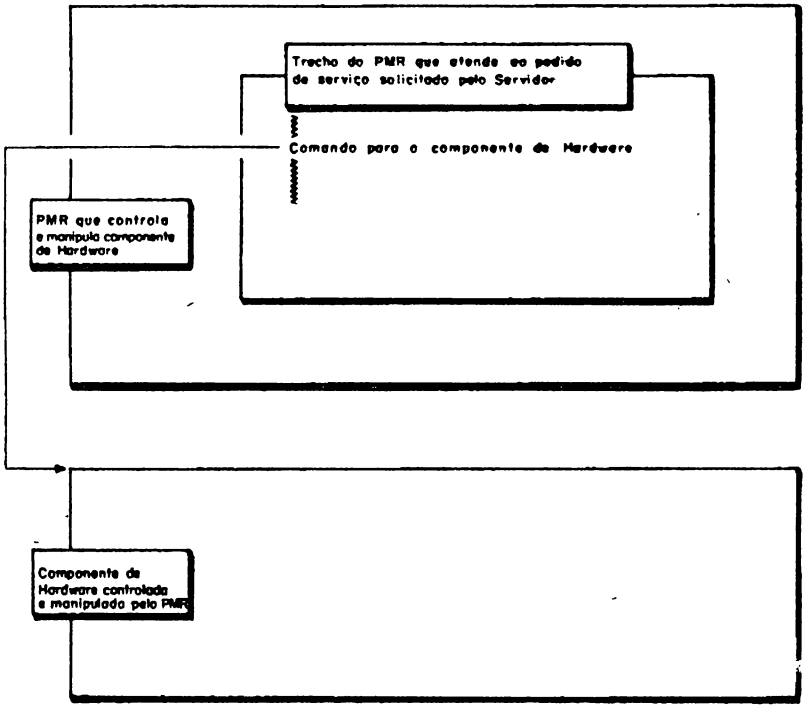


Figura 5-6

Desta maneira, todas as dependências de Hardware ficam isoladas no Ambiente da Máquina Real. Isto implica em que equipamentos diferentes demandam Ambientes da Máquina Real diferentes.

5.1.3 NÚCLEO

É o componente do SOX responsável por criar e controlar as abstrações existentes tanto no Ambiente da Máquina Virtual como no Ambiente da Máquina Real. Isto significa que é o Núcleo que cria e controla as informações necessárias para que possam existir as Máquinas Virtuais e os Processos da Máquina Real. Ele também é responsável pela comunicação e controle de fluxo entre estas entidades, bem como pela gerência de memória.

5.2 ARQUITETURA DO SOX

A arquitetura do SOX pode ser resumida na ilustração seguinte:

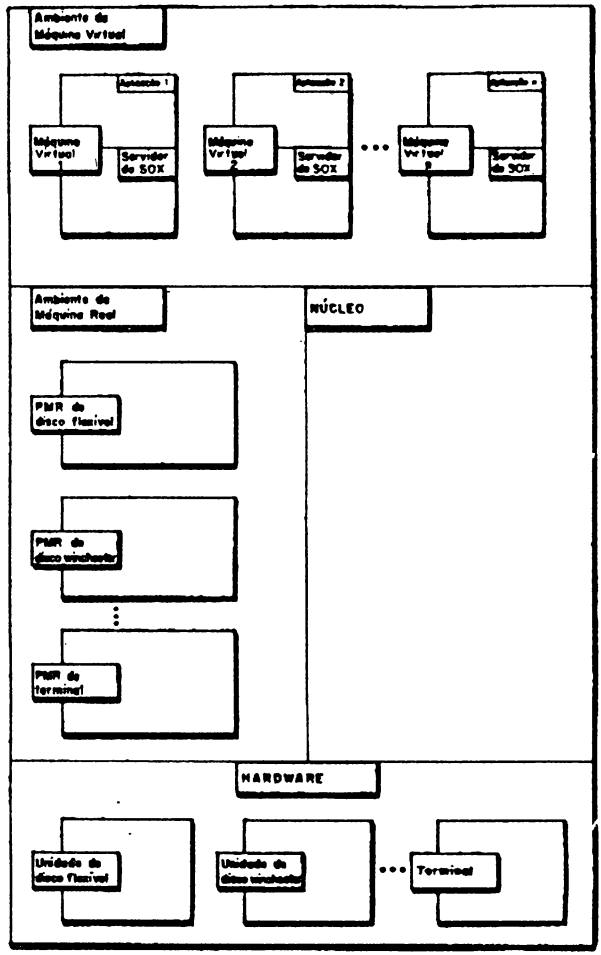


Figura 5-7

No Ambiente da Máquina Virtual executam as Máquinas Virtuais. Cada Máquina Virtual é constituída de Aplicação e Servidor. A Aplicação é um comando do SOX, um programa do usuário ou um arquivo de comandos. O Servidor é a parte do SOX que atende a pedidos de serviço da Aplicação de forma independente do Hardware. O Servidor repassa as ações que dependem do Hardware aos PMRs do Ambiente da Máquina Real, através de pedidos de serviço.

No ambiente da Máquina Real executam os Processos da Máquina Real, responsáveis por atender a pedidos de controle e acesso ao Hardware feitos pelo Servidor ou por outros PMRs. Eles emitem comandos diretamente aos componentes de Hardware que controlam.

5.3 EXEMPLO

Nesta seção apresentaremos um exemplo que ilustra o fluxo de controle entre os diversos componentes do SOX quando você executa um comando. Foram feitas algumas simplificações no que realmente ocorre, com objetivo de melhorar a sua compreensão, evitando detalhes desnecessários. Neste exemplo serão apresentados os diversos passos executados pelo SOX, desde a abertura de sua sessão até a exibição do resultado no seu terminal.

1 - Quando você se conecta com o SOX, a sua sessão corresponde a uma Máquina Virtual, cuja Aplicação é o SHELL.

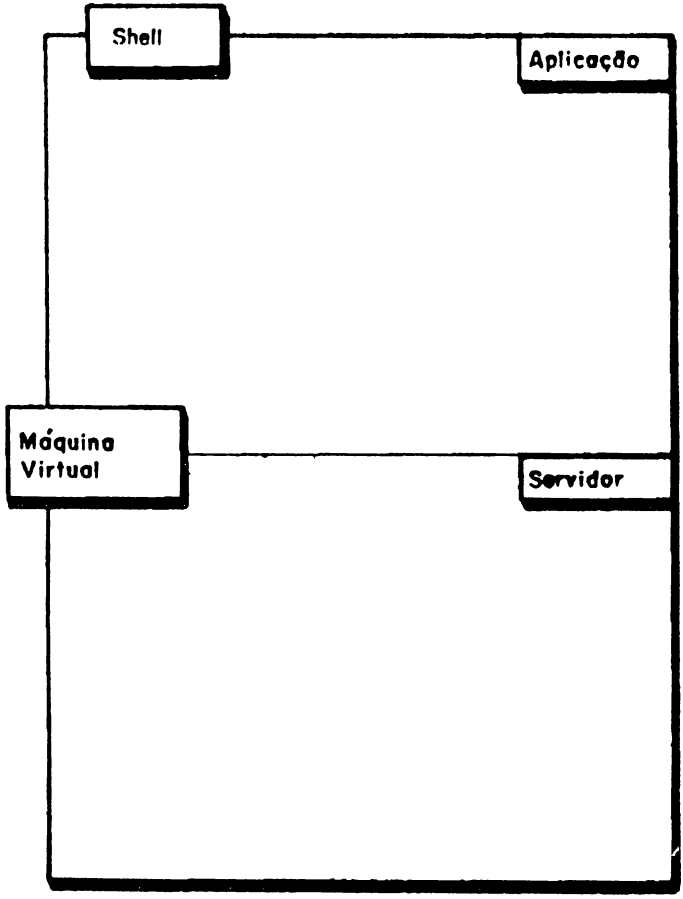


Figura 5-8

2 - Você faz um pedido de execução de comando. Por exemplo:

```
$ ls -l arq     <CR>
```

O SHELL recebe o seu pedido e cria uma Máquina Virtual para executar o comando desejado.

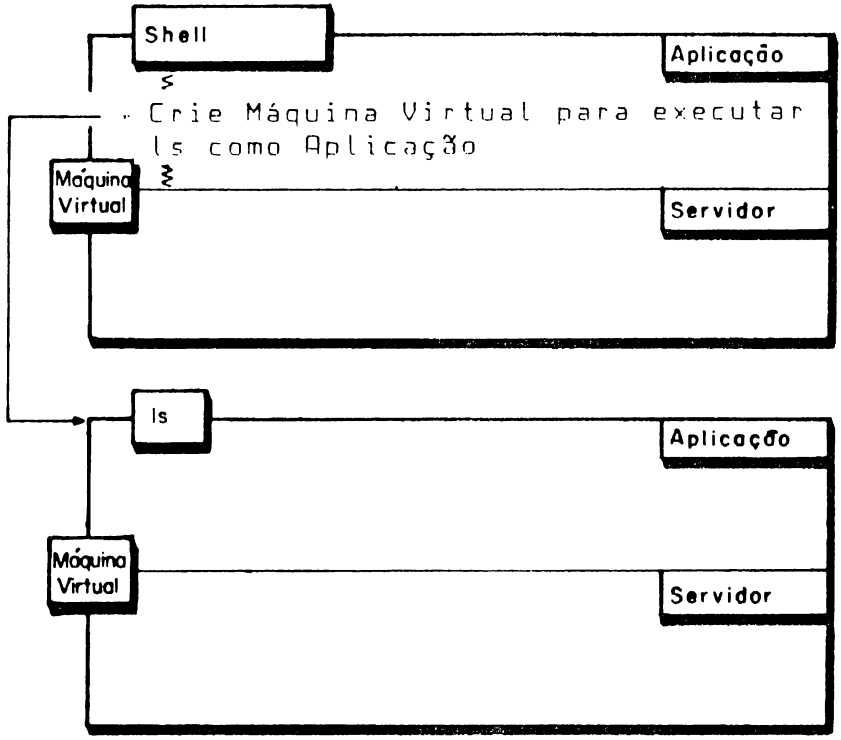


Figura 5-9

3 - O SHELL ativa a execução da Aplicação da nova Máquina Virtual e aguarda o seu término. Isto corresponde ao início da execução do comando desejado. Durante a execução da Aplicação, são feitos pedidos de serviços ao Servidor.

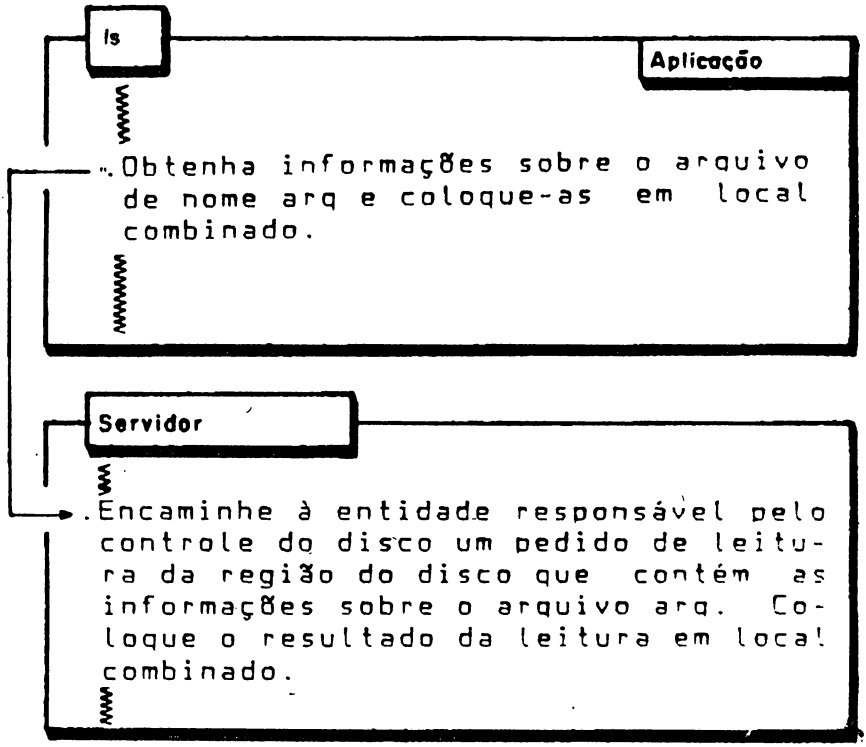


Figura 5-10

4 - Se os pedidos de serviço ao Servidor mencionados no passo 3 envolverem a manipulação de alguns componentes de Hardware, então o Servidor solicita serviço ao PMR responsável pela manipulação do componente.

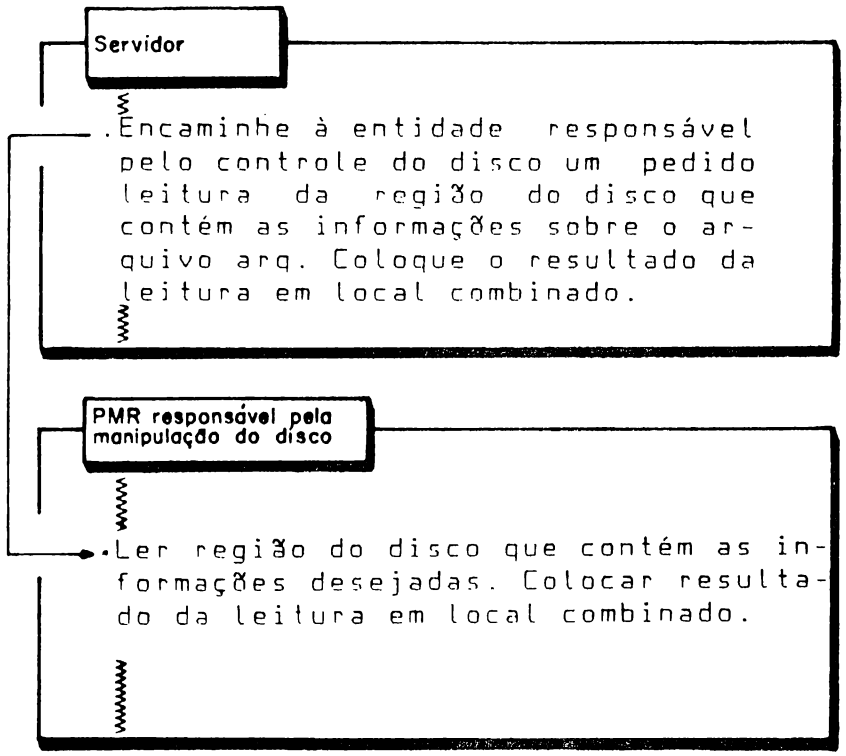


Figura 5-11

5 - O PMR responsável pelo controle do componente de Hardware a ser acessado manipula este componente de modo a realizar o serviço desejado.

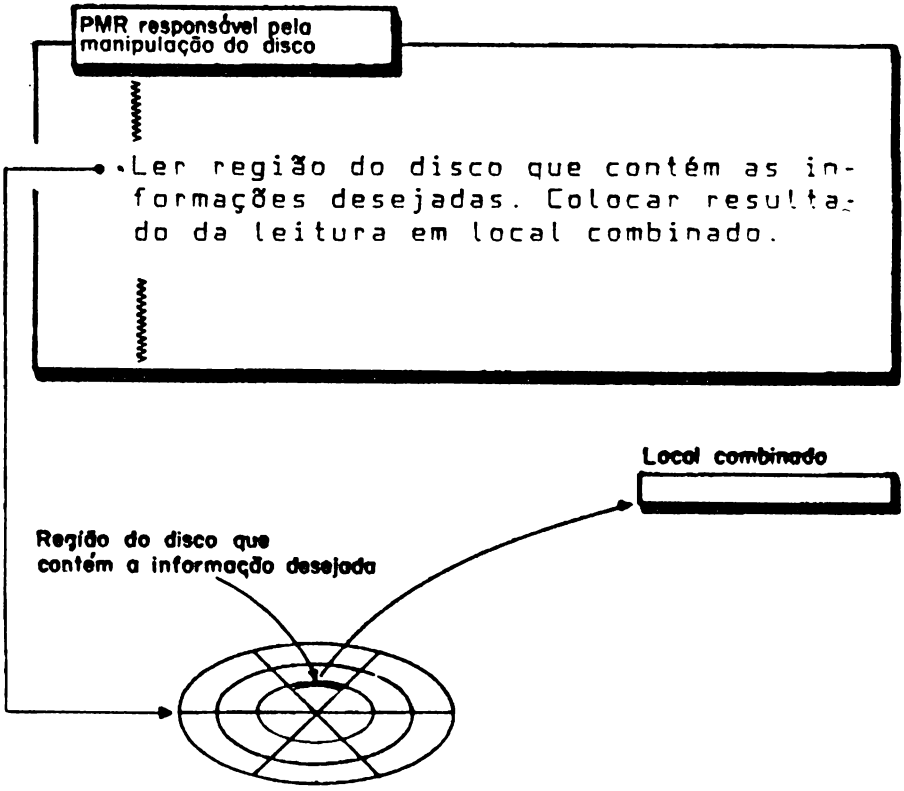


Figura 5-12

6 - Após a ação do Servidor e dos PMRs envolvidos na execução de um serviço, a Aplicação utiliza os resultados da execução do serviço.

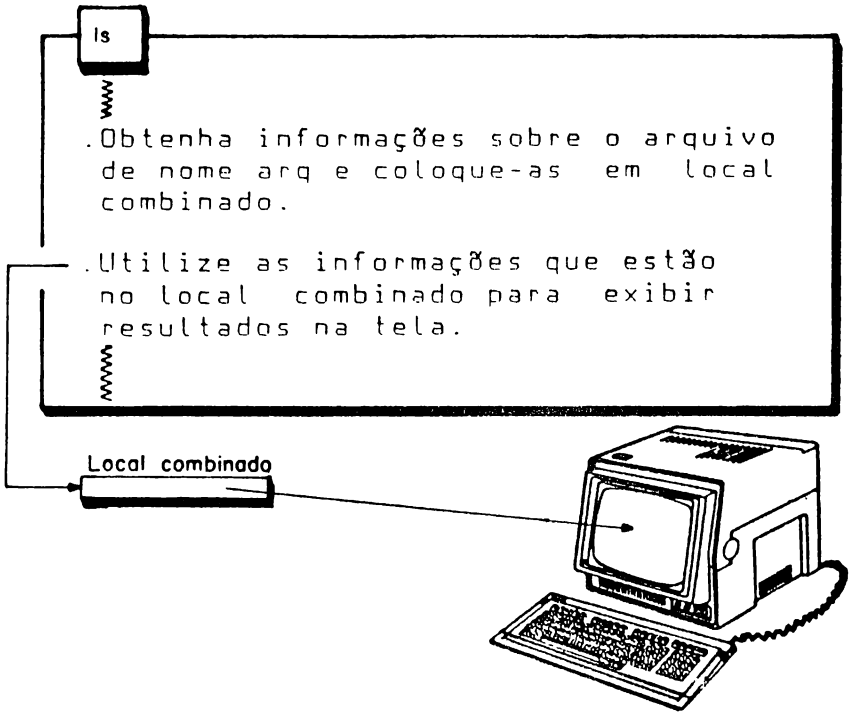


Figura 5-13

ANEXO A

ERROS NA IDENTIFICAÇÃO DO USUÁRIO

A seguir são apresentadas as mensagens de erro ou advertência ocorridas na identificação junto ao SOX.

1.Mensagem:

"Apenas caracteres alfabeticos"

Causa:

Você digitou um caractere não-alfabético na sua identificação.

Ação:

Redigitar a sua identificação.

2.Mensagem:

"Nome ate 8 caracteres."

Causa:

Você digitou sua identificação com mais de 8 caracteres.

Ação:

Redigitar a sua identificação.

3.Mensagem:

"Confirma nome sem minuscula ? (S/n)"

Causa:

Você digitou sua identificação com letras maiúsculas.

Ação:

Se você digitou sua identificação com letras maiúsculas intencionalmente, responda "S". Caso contrário, digite "N". Você deve responder "S" se o seu terminal gerar somente letras maiúsculas.

4.Mensagem:

"Nome obrigatorio"

Causa:

Não foi digitado nenhum caractere da identificação.

Ação:

Digitar sua identificação.

5.Mensagem:

"login: nenhuma entrada em utmp.

recarregue novamente o sistema."

Ação:

Procure o Administrador do Sistema para que ele possa tomar as providências necessárias.

6.Mensagem:

"login : nao foi possivel atualizar /etc/wtmp"

Causa:

Não há mais espaço disponível em disco.

Ação:

Procure o Administrador do Sistema para que ele possa tomar as providências necessárias.

7.Mensagem:

"login : nao foi possivel alterar diretório <nome-do-seu-diretório-casa>"

Causa:

Seu diretório-casa não existe.

Ação:

Procure o Administrador do Sistema para que ele possa tomar as providências necessárias.

8.Mensagem:

"login : nao ha memoria disponivel"

Causa:

Não há memória disponível para novas variáveis do ambiente.

Ação:

Procure o Administrador do Sistema para que ele tome as providências necessárias.

9.Mensagem:

"login : usuario nao valido"

ou

"login : grupo nao valido"

Causa:

Identificações do usuário ou grupo não são válidas, isto é, têm valores não válidos.

Ação:

Procure o Administrador do Sistema para que ele tome as providências necessárias.

10.Mensagem:

"login : nao conseguiu executar shell"

Causa:

Nome de Shell invalido.

Ação:

Procure o Administrador do Sistema para que ele tome as providências necessárias.

11.Mensagem:

"Senha invalida"

Causa:

Você não está cadastrado junto ao SOX como seu usuário, ou você está cadastrado junto ao SOX e a senha fornecida em resposta ao pedido de senha não coincide com a sua senha.

Ação:

Verifique se você está cadastrado, ou forneça a senha correta.

Ambiente de Máquina Real (AMR)
Ambiente criado a partir de um conjunto de recursos oferecidos pelo núcleo do SOX. Este ambiente é dependente do Hardware e tem por finalidade prover ferramentas para construção de módulos prestadores de serviços (de E/S ou outros) e para oferecer suporte aos outros processos que executam no sistema.

Ambiente de Máquina Virtual (AMV)
Ambiente lógico mantido pelo núcleo do SOX, constituído de um espaço de endereçamento lógico e de uma série de mecanismos para atendimento de pedidos de serviços solicitados pelas aplicações que estão sendo processadas. Tem por objetivo tornar transparente para estas as características particulares do ambiente real que as está suportando.

Arquivo
Conjunto de registros inter-relacionados, tratados como um todo.

ASCII
Tipo de código para a representação de caracteres alfanuméricos, que utiliza 8 bits. Sete são usados para o caractere em si, enquanto o oitavo, para paridade.

Associação
Diz-se que um arquivo possui uma associação sempre que houver alguma entrada em algum diretório apontando para ele. Um arquivo pode ter uma ou mais associações.

Canalização
Mecanismo que possibilita que um comando utilize como entrada-padrão os dados da saída-padrão de um outro comando.

Diretório
Área lógica de um meio magnético de armazenamento de dados onde são identificados os arquivos gravados neste meio.

Entrada Ponto (.)
Entrada de diretório que aponta para o próprio diretório, isto é, contém a expressão de caminho do diretório corrente.

Entrada Ponto-ponto (..)
Entrada de diretório que aponta para o diretório-pai do diretório corrente.

EOT
Caractere EOT (End Of Transmission). Representa o fim da recepção de uma mensagem.

Expressão de Caminho
Explicitação dos diretórios e subdiretórios, a partir do diretório raiz (/) ou outro diretório qualquer, que compõe o "caminho" até se atingir o arquivo a ser referenciado. Os componentes da expressão de caminho separam-se entre si pelo caractere "/".

Hardware
Constituição física e permanente de um computador ou equipamento. Circuitos eletrônicos, dispositivos e componentes elétricos e mecânicos empregados em sua construção.

Metacaractere
Caractere que tem significado especial em linhas de comando, e que possibilitam a referência a conjuntos de arquivos.

Núcleo
Componente do SOX responsável pela criação, controle e intercâmbio das informações necessárias para a implementação e interação entre os Ambientes de Máquina Virtual e Máquina Real.

Octeto
Unidade formada por um conjunto de 8 bits que pode assumir valores de 0 a 255 (decimal). Em inglês, "byte".

Organização stream
Tipo de organização de arquivo em que os caracteres são tratados como simples seqüências de octetos, sem nenhuma forma de organização mais elaborada.

Pedido de Serviço (CSERV)
Solicitação emitida por uma aplicação ao servidor, para que determinada ação de mais baixo nível seja executada, como por exemplo, criação, abertura, leitura e gravação de arquivos.

Pronto
É um caractere (em geral "$") que precede o cursor, usado para indicar que o SOX está pronto para interagir com o usuário, aceitando seus comandos.

Redireção
Operação que permite ao usuário redefinir as entidades-padrão (entrada, saída e saída de erro) associadas a um comando.

Servidor SOX
Módulo do SOX que atende os pedidos de serviço das aplicações, constituindo-se, conforme a ação a ser executada, na interface com as entidades que controlam componentes de hardware.

Shell
Programa responsável por receber linhas de comandos, interpretá-las e ativar os programas que executam esses comandos.

SOX
Sistema operacional compatível com o UNIX, desenvolvido pela COBRA para computadores de 16 a 32 bits. É utilizado na linha de computadores X.

Superusuário
Usuário que possui todos os privilégios oferecidos pelo sistema para sua operação.

COMPUTADORES E SISTEMAS BRASILEIROS S.A.

CARTÃO DE CADASTRO E COMENTÁRIOS DO USUÁRIO

MANUAL DE INTRODUÇÃO AO SOX

SX0288-01.0

Prezado leitor:

Suas criticas, comentarios e sugestões são valiosos instrumentos para que a COBRA possa, cada vez mais, aprimorar a documentação de seus produtos. Solicitamos, portanto, sua colaboração em preencher este cartão, colocando-o em qualquer caixa de correio (porte pago). Obrigado.

AVALIAÇÃO DO MANUAL

Nos itens que se seguem, indique o conceito aplicável, comentando se assim o desejar:

CONCEITOS: • Excelente (E) • Bom (B) • Regular (R) • Precisa melhorar (P) • Insuficiente/Inadequado (I)

ITEM	CONCEITO
APRESENTAÇÃO DO MANUAL (Encadernação, acabamento, formato)	
QUALIDADE GRÁFICA (Tipo de impressão, diagramação, papel utilizado)	
CLAREZA DAS INFORMAÇÕES (Linguagem empregada, facilidade de compreensão do texto)	
ABORDAGEM DOS ASSUNTOS (Organização, precisão e objetividade)	

ITEM	CONCEITO
DESENHOS, ILUSTRAÇÕES E EXEMPLOS (Suficiência, adequabilidade e clareza)	
UTILIDADE (Adequação à utilização pretendida, facilidade de consulta)	
EDITORAÇÃO E REVISÃO (Existência de falhas de impressão, paginação, erros ortográficos ou gramaticais)	
QUALIDADE GERAL DO MANUAL (Avaliação global do usuário)	

COMENTÁRIOS E SUGESTÕES

IDENTIFICAÇÃO DO USUÁRIO

Nome:	Profissão:
Empresa:	Cargo:
Endereço p/correspondência:	CEP:
	Tel.:

Av. Comandante Guaranys, 447 - 22785 - Rio de Janeiro - RJ
(021) 342-9393 - Telex (021) 22480

COLE ESTA ABA NO VERSO DO CARTÃO

ISR-52-850/87
UP APT PRES. VARGAS
DR-RJ

CARTA RESPOSTA COMERCIAL

NÃO É NECESSÁRIO SELAR

O SELO SERÁ PAGO POR

COBRA COMPUTADORES E SISTEMAS BRASILEIROS S/A

(DIV. ASSIST. SOFTWARE - DAS)

20299 RIO DE JANEIRO - RJ

CEP

Endereço:

Remetente:

COLE AQUI A ABA DO CARTÃO

IMPRESSO NA GRÁFICA DA COBRA COMPUTADORES

cobra

Computadores e Sistemas Brasileiros S/A