

1

Sobre a Instalação

Introdução

Neste capítulo, abordaremos os tópicos referente à instalação do sistema, tais como o particionamento do disco rígido, a utilização dos sistemas Windows e **Linux** juntos, de pacotes, etc.

Suponhamos que o sistema será instalado em um disco rígido onde já existe outro sistema (no caso, o Windows) e não há espaço livre no HD. Sendo assim, é necessário que se faça um reparticionamento do mesmo, sem a perda de dados existentes.

Falaremos, também, um pouco sobre o que é o **Linux**, de onde veio e quais as vantagens em utilizar este sistema que ainda é pouco desconhecido por muita gente.

Linux: Será que é Bom?

Esta pergunta é muito feita por iniciantes ou pessoas que querem fazer uso desse sistema operacional. Comentarei, então, algumas características próprias do **Linux** e suas vantagens.

O **Linux** surgiu da união de milhares de programadores e usuários do mundo inteiro. Tudo começou com um jovem estudante finlandês chamado Linus Torvalds. Em 5 de Outubro de 1991, a seguinte mensagem circulou na Usenet:

“... Como eu mencionei há um mês atrás, estou trabalhando em uma versão ‘free’ de um sistema semelhante ao Minix para computadores AT-386. Ele já alcançou o estágio de ser usável (embora possa não ser, dependendo do que você quer fazer), e pretendo distribuir o código-fonte. É apenas a versão 0.02... mas já consegui rodar o bash, gcc, gnu-make, gnu-sed, compress, etc. nele”.







Esta mensagem foi enviada por Linus, que, na ocasião, não imaginava que mais de 12 milhões de cópias estariam espalhadas pelo mundo.

Falemos, então, um pouco sobre as suas vantagens. O **Linux** oferece uma série de alternativas com muita eficiência e efetividade a baixo custo. Nele, encontramos alguns dos servidores mais usados no mundo, como é o caso do servidor de páginas Apache. Mas não para por aí! Temos os servidores cache, como o squid, servidores de mail (sendmail, qmail, pop3, imap, etc.), NFS e NIS. Há soluções de conectividade com outras plataformas, como Novell, SNA. Ainda existe a possibilidade de usá-lo como servidor de arquivos e impressoras em uma rede Windows .

Enfim, realmente é um sistema muito robusto, com uma infinidade de utilizações (desde rede, até usuários desktop) e inúmeros softwares para aplicações diversas, tanto no modo texto como no modo gráfico. Há, além disso, fator muito forte nesse Sistema Operacional: seu CUSTO. Ele é FREE, ou seja, não há a necessidade de pagar por uma licença. O **Linux** é liberado através da licença GPL, que nada mais é do que um regulamento que especifica o uso do software. Isso significa que você pode ter acesso ao código-fonte do sistema, alterá-lo e distribuí-lo livremente, instalar em quantas máquinas você quiser sem custo adicional.

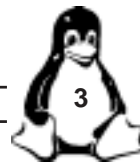
Pré-requisitos

Para instalar uma versão da conectiva **Linux**, por exemplo , é necessário o seguinte :

-  800 Mb de espaço livre no disco para partição raiz e 1 Mb para a partição de troca.
-  16 Mb de Ram (recomendável 32 Mb para utilização de ambientes gráficos).
-  Processador Intel 386 ou superior.
-  CD-ROM e unidade de disquete 3 ½ .

Neste caso, iremos utilizar uma instalação via disquete e CD-ROM, mas existem outras formas de instalação, como via rede, http, ftp, etc.

É sempre bom darmos uma olhada no tipo de hardware que temos e verificar sua compatibilidade com o sistema. Em <http://>



www.conectiva.com.br/suporte/hardware, há uma lista de hardware compatíveis e não compatíveis com o sistema, mas devo lembrá-lo que em alguns casos ainda há salvação, como no caso do hardware onboard; dos winmodems Lucent, Pctel, Motorola; da placa de rede Davicom; da placa de vídeo Sis; etc. Sobre eles falaremos em capítulos posteriores.

Particionamento

Partimos do princípio de que você possui um micro com as características citadas anteriormente, com o Windows da Microsoft instalado em uma partição que ocupa o HD totalmente. E que você tem espaço livre nesta partição onde o Windows está instalado.

Primeiramente, precisamos utilizar o defrag do Windows para unir os arquivos “espalhados” em todo HD. Normalmente, o programa fica no menu *Iniciar/ Programas/Acessórios/Ferramentas de Sistema/ Desfragmentador de Disco*. Isto demora um pouco, por isso recomendo um bom café até que tudo fique pronto.

Finalizando o processo de fragmentação, dê uma olhada na quantidade de HD disponível para que na hora que formos reparticionar o disco não haja dúvidas.

Temos algumas ferramentas “free” e comerciais para reparticionar winchester. Como comercial, posso citar o Particion Magic, uma ferramenta muito boa, totalmente visual. E como “free”, posso citar o fips, que, aliás, vem no CD de instalação do **Linux**. Não é uma ferramenta visual, ela é utilizada no prompt de comando, mas para o que precisamos, ela serve muito bem.

1. Coloque o CD de instalação do **Linux** no drive e abra um prompt de comando (DOS);
2. Em seguida, entre no diretório *dosutils* do CD, digitando *D: enter e cd dosutils enter*;
3. Execute o arquivo com o comando *fips.exe*.

A partir de agora, temos que tomar muito cuidado, pois qualquer erro pode ser fatal para o seu Windows.

Logo após a execução do comando, ele fará uma leitura das partições gravadas na MBR (Master Boot Record), que na realidade é a primeira trilha do seu HD, onde ficam gravadas todas as características do seu HD, tamanho, partições, tamanho da partição, etc.



Feito isso, ele perguntará se você quer gravar um disco de backup, que é o ideal.

Após isso, gravando ou não o disquete de backup, ele mostrará uma linha com a seguinte descrição:

```
old : 3214          new : 2548
```

Onde *old* é o tamanho original da sua partição e *new* será o novo tamanho se você tiver disponível. Por exemplo, se seu HD tem 6 GB e você quer deixar 2 GB livres em *new* será necessário informar o valor de 4 GB. Sugiro algo em torno de 3 GB, pois provavelmente você futuramente gostará de ampliá-lo.

Quando o valor da nova partição de um *enter* é colocado, o programa perguntará se você realmente quer gravar os dados, responda *y*, de *yes*, e pronto. Você terá sua partição diminuída, o restante do HD aparecerá como livre e não será necessário, por enquanto, criar nenhuma partição. Caso ainda não apareça como livre quando você executar o *fdisk*, dê um boot na máquina para completar todo o processo.

Criando Disco de Boot

Se sua bios não suporta boot por CD, será obrigatória a criação de um disco de boot. Caso contrário, pule este tópico e vá direto para instalação.

Mas para você que precisa criar um disco de boot, tenha calma. O processo é rápido e logo estaremos instalando o tão desejado sistema.

No Windows, novamente no prompt de comando, vá ao diretório *dosutils* do CD.

```
C:\> d:  
D:\> cd\dosutils
```

Execute o programa chamado *rawrite*:

```
D:\dosutils\> rawrite
```

O primeiro parâmetro a ser passado será o caminho de onde está a imagem a ser gravada no disco. Digite:

```
..\images\boot.img
```

No caso onde se encontra o arquivo *boot.img*, por exemplo se você fez download, e gravou-o no C:\, deverá informar C:\boot.img)



O segundo parâmetro a ser passado será o caminho da unidade de disco. Digite:

a:

Prontinho! Agora vamos a fase tão esperada.

A Instalação

Abordarei toda a instalação, no entanto, configurações diferentes serão vistas em outros capítulos. Por exemplo, algumas placas de vídeo, a princípio, não são reconhecidas e não falaremos de sua configuração neste momento. E, por esse motivo, resolvi passar para vocês a instalação em modo texto, pois além de ser mais rápida, pode ser mais flexível. Mas, se você estiver instalando no modo gráfico, não terá problema algum em acompanhar a instalação, pois na realidade são poucas as diferenças entre o modo gráfico do modo texto.

Para iniciarmos o processo de instalação, coloque o CD e/ou disquete nos seus respectivos drivers. Em seguida, dê um boot na máquina. Peço aos senhores que não esqueçam de alterar na BIOS o boot para CD-ROM ou floppy.

Logo que iniciado o processo de boot, surgirá a seguinte mensagem:

```
Ben-vindo ao Conectiva Linux 5.1!
o Para instalar o Conectiva Linux 5.1 Beta, pressione a tecla <ENTER>.

Welcome to Conectiva Linux 5.1!
o To install Conectiva Linux 5.1 Beta, press <ENTER>.

¡Bienvenido al Conectiva Linux 5.1!
o Para instalar el Conectiva Linux 5.1 Beta, presione la tecla <ENTER>.

[F1-Principal] [F2-Geral] [F3-Buscue] [F4-Expert] [F5-Kernel]
boot: _
```

É só apertar a tecla enter que o processo de instalação terá início.



A primeira pergunta que o programa de instalação do **Linux** faz é referente ao local onde contém os dados da instalação. Neste caso, com certeza estamos utilizando o CD-ROM.



Você pode mover-se apertando sempre a tecla *Tab*, selecionado o CD-ROM. Vá até a tecla *OK* através do *Tab*. Em seguida, digite *Enter*.

O próximo item é referente a língua. No caso, selecione *Português* e dê um *OK*.

A seguir, vemos a tela de configuração do mouse.



Neste caso, você selecionará seu mouse. Se não estiver na lista, selecione o mouse genérico, *ps2* ou *serial*. Logo após a descrição do mouse, vemos que há um item desmarcado, que é emulação de três botões, ou seja, se você não possuir um mouse de três botões, com essa



opção marcada, ele emulara esse terceiro botão quando você apertar os dois botões do mouse de uma só vez. Para marcá-lo, vá até a seleção e aperte e tecla espaço. Este botão é muito útil, pois quando você seleciona um texto no **Linux**, ele copia. Basta apertar o terceiro botão para colar ou os dois juntos para emular o terceiro. Há muitas outras utilidades que não serão abordadas no momento. Em algumas distribuições, como a Conectiva 6.0, logo após todas essas opções mencionadas, há uma outra opção a ser escolhida, que é a porta COM. Funciona da seguinte forma: COM1 é o ttyS0, COM2 o ttyS1, e assim por diante.

Em seguida, virá a configuração do teclado. Como o anterior, selecione o seu teclado, normalmente o ABNT-2.

A próxima tela é referente aos componentes que serão instalados. Neste caso, estamos utilizando uma versão do **Linux 5.1 Servidor**, que pode variar um pouco conforme a versão. Recomendo novamente não instalar nenhum tipo de servidor, como roteador, banco de dados, servidor FTP, Internet, etc. A configuração desktop é mais do que suficiente. Em capítulos seguintes configuraremos alguns servidores manualmente para um melhor entendimento do usuário.

Selecionado os componentes, prossiga pela instalação, apertando *Enter* sobre o *OK*.

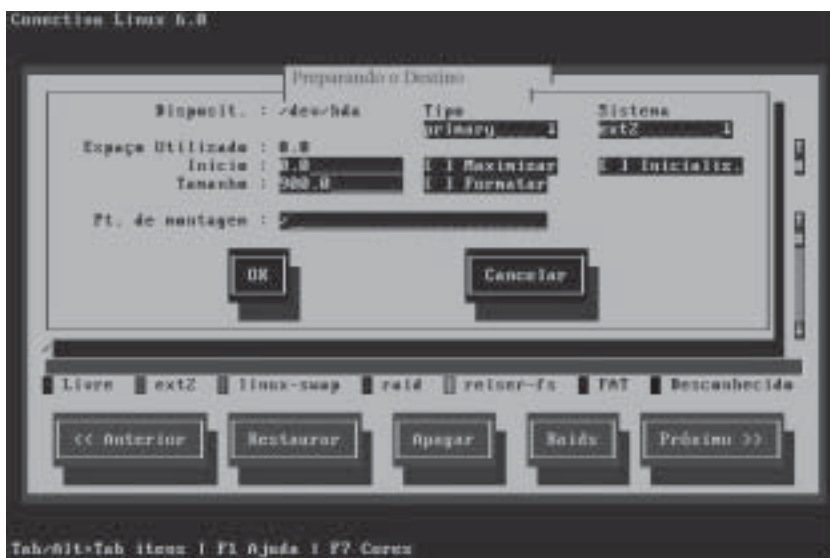


Esta parte é bastante importante, será onde informaremos ao **Linux** quanto de HD estará disponível para ele. Normalmente, se você fizer uma instalação em modo gráfico, este processo será feito automaticamente pelo **Linux**, que ocupará todo espaço livre no HD.







Em *Sumário dos Dispositivos*, será informado quantos HDs e partições de discos possuem a máquina. Neste caso, só um HD, o **hda**. Se houvesse outros, seriam **hdb**, **hdc**, e assim por diante. Provavelmente, você terá uma partição (**hda1**) tipo **Fat** e outra como **Free**.

Utilizado a tecla *Tab*, você colocará o cursor sobre a partição **free** e apertará a tecla *Enter*. Uma nova tela de edição parecida com a seguinte será aberta:






Primeiramente, os dados que devem ser inseridos são os seguintes:



-  **Tamanho:** 3000 (3 GB, conforme disponibilidade).
-  **Type:** *Primary* (para selecionar tais opções, navegue com a tecla *Tab*, aperte o *Enter* no campo editável. No caso acima, você apertaria o *Enter* em cima de *primary* e navegaria com a seta através das opções. Quando escolhida, aperte a tecla de espaço).
-  **Sistema:** ext2.
-  **Pt de montagem:** / (Esse será nossa raiz do sistema).

Feito isso, dê *Enter* sobre o *OK*, volte novamente ao **Free** que restou e refaça todo procedimento, mas com os seguintes dados:

-  **Type:** *primary*.
-  **Sistema:** troca ou swap.
-  **Tamanho:** 100.

Marque a opção *Maximizar*.

Acabou o momento crucial, a partir de agora tudo será mais fácil.

Neste momento, iniciará a instalação do sistema. Sugiro novamente que você prepare um novo café, pois este processo é bem demorado, dependendo do que foi selecionado para instalar, da formatação, etc.

Após a Instalação dos Pacotes

Acabada a instalação dos pacotes, é necessário mais alguns ajustes para a finalização da instalação. O primeiro deles é a configuração de rede, que, por default ou padrão, vêm marcada com a opção sem interface de rede. E é assim que deverá ficar, pois a placa de rede será configurada manualmente para utilização em uma rede interna ou com o ADSL.



Configurando seu Linux

Introdução

Antes de iniciarmos este capítulo, vou comentar um pouco sobre as distribuições.

Em geral, as distribuições detectam automaticamente a maioria dos hardwares, inclusive winmodems. Logicamente, estamos falando destas últimas distribuições, tais como Conectiva 7.0, Red Hat 7, Mandrake 8, Tech Linux 2.0, entre tantas outras. Em particular, adaptei-me melhor a Conectiva. Mas não há a melhor distribuição, e sim a que você se adapta melhor.

Sempre é bom verificar a compatibilidade do seu hardware com a distribuição escolhida.

Há uma distribuição chamada WinLinux, mas não é “free”, custa em torno de US\$ 30. Existe, também, uma versão mais simples, a qual você pode fazer um download na página www.download.com. Essa versão é muito boa para iniciantes, mas sua instalação e detecção de hardware diferem totalmente dos padrões **Linux**. Ela é instalada na mesma partição que a do Windows, sendo obrigatório que a partição seja do tipo **Fat**, ou seja, só é possível instalar em um Windows 95, 98 ou ME. O modo de instalação é muito similar ao Windows da Microsoft.

Não custa dar uma olhada.

Voltemos, então, às distribuições **Linux**.

Calma, meus amigos, sei que estão aguardando ansiosamente para poder usar o sistema, mas primeiro precisamos verificar se está tudo OK.

Proavelmente, se você instalou uma distribuição das mais recentes, acredito que já esteja praticamente tudo configurado.

Ao reiniciarmos a máquina, a primeira coisa que você notará é que agora temos um gerenciador de boot, pode ser o Lilo ou Grub. O que vem a ser isso?

Gerenciador de boot, como o nome o diz, gerencia os vários sistemas operacionais instalados na máquina. Dessa forma, você continuará tendo acesso aos outros sistemas instalados no micro.



Como disse no início do primeiro capítulo, parti do princípio que a sua máquina tem somente o Windows instalado. Esse gerenciador de boot é um software que é gravado na primeira trilha do HD, ou seja, na MBR (Master Boot Record). Para retirar totalmente uma instalação **Linux**, é necessário, além da remoção da partição onde o mesmo se encontra, a remoção dos dados gravados na MBR com o comando *fdisk/mbr*. Mas acredito que depois de vocês terem instalado o **Linux** e aprendido a utilizá-lo, nunca irão querer tirá-lo.

Então, depois dessa breve introdução, vamos botar a mão na massa finalmente.

Neste momento, podem ocorrer duas situações: a placa de vídeo e monitor serem configurados na instalação ou não.

Com as últimas versões das distribuições mais conhecidas, como a Conectiva, Red Hat e outras, as configurações são praticamente automáticas. Neste primeiro caso, uma interface gráfica estará solicitando um nome de usuário e senha, com esta figura abaixo:



Nesse caso, é só fornecer o nome de usuário (o root, por exemplo, que nada mais do que um superusuário com todos poderes para administrar o sistema). Com esse usuário, é possível criar outros, ver logs, iniciar e parar serviços, etc. E, como vamos configurar algumas coisinhas, será necessário usarmos este usuário.

Então, escolha o root e coloque a mesma senha que foi dada na instalação. O tipo de sessão nada mais é do que a interface gráfica que estará sendo usada. Temos o KDE, Gnome, Window Maker, Blanes,



entre outras tantas que falaremos ainda neste livro. Feito isso, escolha um tipo de sessão. Recomendo o KDE ou o Gnome, que são mais amigáveis. Clique em *Ir* para enfim entrar no modo gráfico.

Temos o segundo caso, que não conseguiu configurar a placa de vídeo e o monitor logo na instalação. Lembro novamente que sempre é bom dar uma olhada na compatibilidade de hardware de sua máquina e no sistema a ser instalado.

Neste segundo caso, será apresentada uma tela preta como aquela antiga conhecida de todos, o prompt do DOS. Esta tela chamamos de *console*. Ele terá alguma informação sobre o sistema, tipo Distribuição, Kernel*, e o que não pode faltar, uma linha de login. Nela, você digitará *root*, como descrito anteriormente; teclará *Enter*; digitará a senha. O **Linux** possui case sensitivo, ou seja, maiúsculas são diferenciadas de minúsculas, por isso preste atenção quando digitar a senha.

Feito isso, o console será liberado. E nós iremos dar início a essa empolgante jornada: o conhecimento.

Primeiramente, digite : *pwd* e tecle *Enter*.

A partir de agora, para que fique mais nítido para o leitor, quando for necessário teclar *Enter*, será informado da seguinte forma: *<enter>*.

Rescapitulando, digite *pwd* *<enter>*.

Você verá algo como : */root*.

O que esse comando faz, simplesmente, é mostrar em que diretório você está.

Digite, agora, *ls*.

Este comando mostrará os arquivos e subdiretórios da pasta *root*. Aliás, a pasta *root* é a do diretório de propriedade do superusuário.

* Kernel é o núcleo do **Linux**.

Antes de continuarmos, ou melhor, de começarmos a configuração, algumas coisas têm que ficar claras para que possamos entender bem o que estamos fazendo. Diante disso, iniciarei um subcapítulo para termos noções de como o sistema funciona, dos seus comandos e então configurarmos realmente. Afinal, todo **Linux** user que realmente gosta e usa bem o sistema, com certeza não precisaria de uma interface gráfica (que é muito mais fácil).



Instalei o Linux, e Agora?

Para começarmos a entender o sistema realmente, precisamos saber como ele funciona, como acessá-lo, etc.

Como foi visto anteriormente, o comando *pwd* mostra o diretório que nos encontramos. Mas como pode ser */root*? Onde foi parar o *C:*? Algumas mudanças interessantes. No Windows, usuários estão familiarizados com *C:*, que é a raiz do sistema. Nos sistemas do tipo Unix, isso é semelhante ao *“/”*.

Então, partindo de que *“/”* é a raiz do sistema, */root* é uma sub-pasta da raiz.

Digite *cd ..* <enter> (Entre o *“cd”* e o *..* há um espaço).

Depois *pwd*. Você verá somente *“/”*. Estamos na raiz. Agora digite *ls*.

Você verá algo como :




```
bin      boot      core      dev      etc      home      lib
lost+found mnt
opt      proc      root      sbin     tmp      usr       var
```

O *ls*, como dito anteriormente, é um comando que lista os arquivos e diretórios de uma determinada pasta, semelhante ao *dir* do Windows. Digite *ls-l* <enter>. Você verá, agora, detalhes dos arquivos e diretórios, como no exemplo abaixo:

```
drwxr-xr-x  2 root      root          4096 Mar 22 21:34
bin
```

Não se apague a este detalhe neste momento, veremos o que significa cada item acima descrito. Mas, adianto que os arquivos ou pastas que contêm a letra *d* no início são diretórios e os que nada contêm são outros tipos de arquivos. À frente, os *wxr* estão relacionados com as permissões deste arquivo.

Esses são diretórios que são instalados no sistema. Abaixo, segue a descrição dos mais importantes:

-  **bin:** Armazena executável (binários) dos comandos do próprio sistema. Por exemplo, o próprio comando *ls*, o *pwd*, etc.
-  **boot:** Arquivos utilizados para boot de inicialização.
-  **dev:** Arquivos de dispositivos. Vamos a um exemplo. Você está lembrado de quando configurou o mouse








serial ou ps2, de dois ou três botões e, enfim, informou ao **Linux** em que porta COM estava (COM1, COM2 - que no nosso caso equivalem a ttyS0, ttyS1, ttyS2)?

Veja a tabela abaixo:

ttyS0 = COM1 (ttyS1 é equivalente a COM1)
ttyS1 = COM2
ttyS2 = COM3
ttyS3 = COM4

Bom, matamos duas dúvidas de uma só vez: Primeiro o que era essa tal de ttyS0 que você usou para configurar o mouse, que nada mais é do que um equivalente no **Linux** para COM e, agora, o que vem a ser um arquivo de dispositivo. É justamente o ttyS0, 1, 2... Eles são arquivos de dispositivos, também comumente chamados de “arquivos caracteres”, e ficam neste diretório, o /dev. Mas, compreenda bem: ele é um tipo de arquivo e não um arquivo que pode ser editado e manuseado. Ele é responsável pelo contato dos softwares com o Kernel e, enfim, com o próprio dispositivo ou hardware. Quando listado com o comando `ls -l`, ele apresenta a letra “c” no início da descrição, como no caso abaixo:

```
crw-rw—  1 root      uucp          4,  64 Mar 23 02:40
ttyS0
```

-  **etc:** Onde ficam os arquivos de configuração e administração do sistema. Utilizaremos muito em nossos estudos e, posteriormente, o leitor utilizará também na vida cotidiana.
-  **home:** Diretório local de usuários.
-  **mnt:** Ponto de montagem para vários hardwares, como CD-ROM, floppy, etc.
-  **usr:** Armazena a maioria dos softwares.
-  **var:** Armazena informações variáveis, logs, etc.

Com essas informações, já temos uma idéia do sistema. Mas será que já estamos aptos a mexer entre os subdiretórios, apagar e criar arquivos? Acho que ainda não chegou a hora. Peço um pouco de calma aos leitores, sei que estão ansiosos a utilizar o sistema, mas acredito que será mais proveitoso com conhecimentos prévios.



Agora que já sabemos o que são os diretórios, nada mais justo do que utilizar os de comando no próprio console, aliás, é possível fazer quase tudo no console.

Neste momento, farei apenas uma comparação entre comandos do Windows/DOS e **Linux**. Posteriormente, quando estivermos utilizando mais o **Linux**, serão apresentados novos comandos:

Windows/DOS	Linux
cd diretório	cd diretório
cd..	cd ..
copy	cp
del	rm
deltree dirname	rm -R dirname
edit	vi (muito mais poderoso)
format	fdformat
help comando	man comando
md	mkdir
rd	rmdir
move	mv
dir	ls
type	cat
print	lpr
Windows/Dos	Linux
ren	mv
cls	clear

Um fato que com certeza vai ser estranho para quem nunca usou o **Linux** anteriormente é que no **Linux** é normal não usarmos extensões de arquivos. Não que isso seja impossível, mas como disse é normal esse fato no **Linux**. Com isso, não temos executáveis com extensão **.exe**. Verificaremos isso mais à frente também.

Agora, veremos as permissões de arquivos.

Lembra-se do `ls -l`? Então, o faça novamente.

Teremos uma linha parecida com a seguinte:

```
drwxr-xr-x    2 root    root          4096 Mar 22 21:34
bin
```

A primeira letra sempre representa o tipo de arquivo. Temos os seguintes tipos:



- representa um arquivo sem especificação;
- l representa um link simbólico;
- c representa um arquivo caracter;
- d representa um diretório;
- b representa um arquivo de bloco.

O link simbólico nada mais é do que um arquivo apontando para outro. Vamos supor que exista um log de uma aplicação qualquer em */var/log/logqualquer*. Sempre que você precisar usá-lo, terá que sair do diretório atual e ir para o diretório onde está o log. Podemos fazer o seguinte: como normalmente estamos em */root*, criamos um arquivo de link simbólico da seguinte forma:

```
touch /var/log/logqualquer
ln -s /var/log/logqualquer /root/logqueusaremos
```

Depois disso, *ls -l*.

```
lrwxrwxrwx 1 root root 15 Mar 22:21:34 logqueusaremos
> /var/log/logqualquer
```

Dessa maneira, você pode editá-lo localmente através do *vi* : *vi logqualquer* (supondo que o diretório atual seja o *root*).

Existem formas melhores para fazer isso, como através de Shell scripts prontos. Mas esse foi só para ilustração.



Os caracteres restantes (nove), são divididos em três grupos com três caracteres.

d	rwX	rwX	rwX
	1	2	3

Onde os três primeiros (“1”) são referentes às permissões do dono ou proprietário do arquivo.

Os três seguintes (“2”) são referentes às permissões de grupo no arquivo e os três últimos (“3”) são referentes às permissões globais, ou seja, de todos.

Essas três letras têm uma representação básica e um valor determinado:

-  **r (read):** Dá acesso de leitura. O valor é igual a 4.
-  **w (write):** Dá acesso de gravação. O usuário ou grupo poderá editar. Seu valor é igual a 2.



x (execute): Dá acesso de execução ao usuário ou grupo. Valor igual 1.

Quer dizer que posso dar permissão apenas de leitura, edição e execução juntas ou individuais desde mim, que sou dono, até grupos e usuários distintos?

Sim. Com certeza. Mas tome cuidado, pois dando permissão em um arquivo, não quer dizer que a pessoa vai conseguir acessá-lo. Primeiramente, ele deve ter acesso ao diretório onde se encontra o arquivo.

Vamos ver alguns exemplos:



d rwx r-x r-x: Neste caso, o arquivo é um diretório que o dono tem permissão de ler, gravar e executar, o grupo de ler e executar, mas não de gravar. E, por último, outros usuários que não fazem parte do grupo também têm o direito de ler e executar.



- rwx—x—x: Aqui, o dono tem acesso total, mas o grupo e outros usuários somente execução do arquivo.

Quanto aos números que eles valem, 4, 2 e 1, são de muita utilidade para que possamos alterar as permissões do arquivo. O comando utilizado para a alteração das permissões é o *chmod*, que trabalha de duas formas. Uma diretamente, dando permissões ao dono - *u*, ao grupo - *g*, aos outros usuários - *o* ou, então, a todos os - *a*, com suas respectivas permissões em letras *rwx* ou com os números. Vou mostrar para ficar mais claro.

Temos um arquivo onde o dono pode ler e executar, o grupo e os demais só podem executar.

```
d r-x-x-x
```

Queremos alterar a permissão de leitura para que o dono possa ler, escrever e executar, enquanto o grupo e os demais possam ler e executar.

Da primeira forma, o comando será este:

```
chmod u+w, g+r, o+r arquivo <enter>
```

Onde “u” é o dono, foi acrescido direito de escrita; “g” é grupo, que agora tem permissão de leitura, juntamente com os demais usuários da permissão mundial.



Pode-se usar sinal + para acrescentar permissões, - para retirar e = para deixar igual.

```
ls <enter>
d rwxr-xr-x
```

Esta primeira forma não muito usual, pois você terá que digitar muito mais coisas do que na segunda. Teremos que fazer apenas uma continha de cabeça para realizarmos.

Abaixo, temos valores que são atribuídos às permissões:

- 0 = nenhuma permissão;
- 1 = Execução;
- 2 = Gravação;
- 3 = Execução e Gravação;
- 4 = Leitura;
- 5 = Leitura e Execução;
- 6 = Gravação e Leitura;
- 7 = Execução, Gravação e Leitura.

Vamos fazer na prática para melhor entendimento.

Se você já está no modo console é só seguir. E se está no modo gráfico, execute algum terminal, que provavelmente está no menu K do KDE, o menu Gnome. É possível, no KDE, digitar apenas *Alt + F2*. Será aberta uma janela para escrever, então, digite *xterm*. Um aplicativo que emula ou simula o console será aberto. Você também pode procurar diretamente no menu.

Feito isso, digite:

```
touch arquivo <enter>
```

Esse comando cria um arquivo no diretório atual.

```
ls -l <enter>
```

Agora, veja as permissões do arquivo.

Bom, o que queremos? Vamos dar permissão total para o dono, leitura e escrita, tanto para o grupo como para os demais.

```
-rw-r-r-
```

```
chmod 766 arquivo
```



Onde $7 = 4$ (read) + 2 (write) + 1 (execute) - referente às permissões do dono, aquele primeiro grupo de três caracteres; $6 = 4$ (read) + 2 (write) referente ao segundo grupo com três caracteres referentes às permissões do grupo; e, enfim, novamente 6 , referente ao terceiro grupo de caracteres, aos dos outros usuários que não fazem parte do grupo que o dono faz parte.

```
ls -l <enter>
```

E o arquivo estará da seguinte forma:

```
-rwx-rw-rw
```

Vamos pegar esse mesmo arquivo, dar permissão apenas de leitura e execução para o grupo e deixar o grupo de usuários globais com permissão apenas de execução.

Na primeira forma:

```
chmod g-w+x, o-w-r+x arquivo <enter>
ls -l <enter>
```

Onde “g”, de grupo, tira direito de escrever, “w”, e dá permissão de execução. Dos demais, “o”, está sendo retirada a permissão de editar e ler, permitindo execução do arquivo.

Da segunda forma:

```
chmod 755 arquivo <enter>
ls -l <enter>
```

Onde:

$7 = 4$ (read) + 2 (write) + 1 (execute)

$5 = 4$ (read) + 1 (execute)

$5 = 4$ (read) + 1 (execute)

Mais informações sobre este comando use:

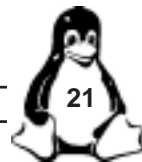
```
chmod --help
```

Ou:

```
man chmod
```

Agora que sabemos como trabalhar com arquivos, devemos aprender como criar os próprios usuários, grupos, etc.

Temos duas formas, uma em modo texto a outra graficamente. Mas, para falarmos do modo gráfico, o mesmo tem que estar instalado e é agora que faremos isso.



Enfim... o Modo Gráfico

São necessários alguns dados antes de iniciarmos a configuração: o modelo e a quantidade de memória da placa de vídeo, o modelo e a frequência vertical e horizontal do monitor (normalmente está gravada em uma chapa de metal atrás do monitor). Normalmente, o próprio **Linux** detecta, mas por vias das dúvidas, execute este comando no Shell (Console): *SuperProbe*. Este comando verifica qual o modelo da sua placa e quantidade de memória que a mesma possui.

Configurando Vídeo

Toda configuração do vídeo fica guardada em um arquivo localizado em */etc/* ou */etc/X11*, chamado *XF86Config* ou *XF86Config-4*, dependendo da versão do Xfree que você estiver usando. O Xfree está em sua quarta versão atualmente, por isso o 4 no arquivo *XF86Config-4*. O Xfree é responsável pelos módulos (drivers, no Windows) que o **Linux** utiliza quando está em modo gráfico. Normalmente, o *XFree-SVGA* contém a maioria dos módulos necessários para podermos configurar o modo gráfico. Mas, não se preocupe com isso neste momento, pois iremos configurar o vídeo através de um programa que gera o arquivo *XF86Config* automaticamente.

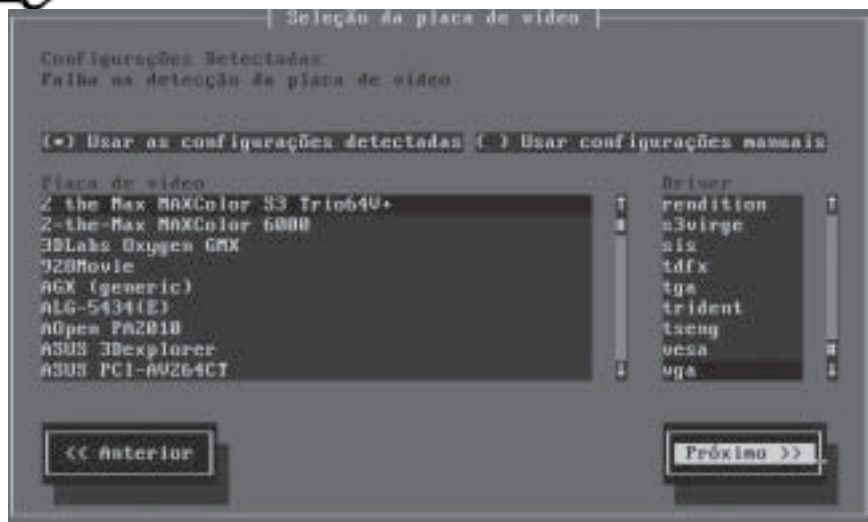
Vamos começar?

Se você já está no modo gráfico, execute um terminal.

No Terminal, digite :

```
Xconfigurator <enter>
```

A primeira tela apresenta as boas vindas e algumas informações. Clique em *OK* e o programa tentará detectar sua placa de vídeo. Caso ele não detecte, você terá que selecioná-la manualmente como na figura seguinte:



Escolha sua placa de vídeo. Caso seja uma placa genérica, selecione-a pelo seu chipset da placa.

Após ter selecionado sua placa de vídeo, tecla *Tab* até estar selecionado o botão *Próxima* e pressione *Enter*. Agora, o *Xconfigurator* tentará detectar seu monitor. Caso também não consiga detectá-lo, verifique qual o modelo e selecione na lista. Uma configuração que costuma ser usada com frequência é a escolha do monitor genérico, onde já é específica o máximo da resolução e a frequência horizontal. Por exemplo, o tipo Monitor Genérico pode fazer 1200x1024 @ 60 Hz. Esta configuração costuma funcionar com a maioria dos monitores de 14 e 15 polegadas sem novas tecnologias do tipo LCD. Feito isso, é necessário informar a quantidade de memória que a placa possui. Caso seja do tipo onboard, que compartilha da memória ram, também deve ser especificada de forma correta, ou seja, deve ser informada a quantidade de memória compartilhada com a memória RAM. A próxima tela de configuração diz respeito à chip set. Se você não conhece muito bem sua placa e todo o dispositivo, deixe como está, sem configuração. A configuração já está quase concluída, pois a próxima tela irá detectar a resolução que sua placa e monitor conseguem suportar. Por isso, você pode deixar que seja detectado automaticamente, pressionando *Enter* sobre *Detectar*.

Caso ocorra algum problema, tente configurar manualmente a resolução selecionando o botão *Ignorar*. É interessante que você comece configurando com resoluções mais básicas, como a de 480x640 16bits ou 8 bits. Depois disso, comece a testar configurações mais pesadas.



Após este processo, ele testará sua placa e monitor. Deverá aparecer uma mensagem perguntando se você consegue lê-la. Diga que sim. Se não aparecer e a tela ficar preta, significa que alguma informação está incorreta, então, aperte as teclas *Ctrl + Alt + Back Space* de uma única vez para encerrar o modo gráfico e voltar ao console. Caso ainda não tenha sucesso, tecele *Ctrl + Alt + F1* que ele irá voltar para o console - mesmo rodando o X (modo gráfico). Tecle *Alt + F2* (ou *F3* ou *F4*) para poder passear entre os vários consoles.

Ainda não falamos disso. O **Linux** possui mais esta qualidade: pode habilitar vários consoles (não emulações), onde estão totalmente independentes, com seus próprios processos. Isto significa que, se um terminal travar por alguma tarefa pesada, o outro não estará com problemas.

Retornando ao assunto, volte ao console original *Alt + F1* e tecele *Ctrl + C* para finalizar o processo do X.

Retome a configuração desde o começo e tenha certeza que as informações estão corretas.

Dessa forma, é feita a configuração. A última mensagem é referente ao modo de inicialização do **Linux** (se você deseja que ele inicie em modo texto ou gráfico). Fica à livre escolha.

O que faz ele iniciar em modo texto ou gráfico é, simplesmente, uma linha do arquivo.

```
/etc/inittab
```

Neste arquivo, existe uma linha deste tipo:

```
id:5:initdefault:
```

O número acima (5) informa ao **Linux** quando inicia o sistema que é para iniciar em modo gráfico. Se colocarmos o 3, ele iniciará em modo texto.

Em http://linux.matrix.com.br/xf86_repos.htm ou em <http://www.dominiolinux.com.br>, existem várias dicas de como configurar placas de vídeo do tipo Diamond, Trident, SIS, S3, etc. Com certeza, é bom dar um pulinho lá para tirarmos algumas dúvidas e, principalmente, verificar se nesse processo de configuração ocorreu algum problema.

XF86Config



Falaremos, agora, um pouco sobre o arquivo *XF86Config*, que, como disse anteriormente, é nele que são guardados muitos configurações, tais como: vídeo, monitor, fontes, teclado, mouse, etc.

Edite o arquivo *XF86Config* com o VI por exemplo:

```
vi /etc/X11/XF86Config
```

Adiante, estarei falando do VI, um ótimo editor de texto com muitos recursos. Mas, nesse caso em particular, queremos somente visualizar o conteúdo desse arquivo. Será suficiente, então, utilizarmos as setas para poder navegar no documento.

A primeira sessão deste arquivo é a *Section Files*, que terá algo parecido com isso:

```
Section "Files"
# The location of the RGB database.  Note, this is the
name of the
# file minus the extension (like ".txt" or ".db").
There is normally
# no need to change the default.
    RgbPath    "/usr/X11R6/lib/X11/rgb"

# Multiple FontPath entries are allowed (they are
concatenated together)
    FontPath   "/usr/X11R6/lib/X11/fonts/misc"
    FontPath   "/usr/X11R6/lib/X11/fonts/misc:unscaled"
    FontPath   "/usr/X11R6/lib/X11/fonts/75dpi:unscaled"
    FontPath   "/usr/X11R6/lib/X11/fonts/
100dpi:unscaled"
    FontPath   "/usr/share/fonts/default/TrueType"
    FontPath   "/usr/share/fonts/default/URW"

EndSection
```

Esta sessão é utilizada para informar o caminho (path) das nossas fontes para o *XFree* e, então, ser usada nos aplicativos gráficos, como Netscape, Star Office, enfim, todos.

A próxima sessão importante é o *Session Keyboard*, que, como o nome diz, é responsável pelo teclado. Veja a linha *XkbModel*, pois ela informa qual o tipo de teclado. No meu caso, por exemplo, uso um us-acentos, que consta com PC101. Se você utiliza um teclado brasileiro, o ABNT, nesta linha estará constando *XkbModel "abnt2"*.

A *Section Monitor* é uma das mais complexas. Para se alterar algo nesta seção, temos que conhecer e saber bem o que iremos fazer, pois



qualquer alteração não-válida irá, provavelmente, travar o X (interfaces gráficas). Para facilitar a configuração do vídeo, sempre é bom termos em mãos as informações sobre nosso monitor, principalmente os dados de refresh vertical e horizontal. É recomendado deixar o próprio *Xconfigurator* achar essas frequências.

A *Section Device* se refere à placa de vídeo. Diz respeito a uma máquina com uma configuração genérica para uma placa SIS 620. Ficará mais claro na próxima sessão, onde é configurada a resolução.

```
Section "Device"
    Identifier      "Generic VGA"
    VendorName      "unknow"
    BoardName       "unknow"
    # Videoram      256
EndSection
```

Enfim, a *Screen Section* define a resolução, profundidade. É onde é conciliado a placa de vídeo e o monitor. Como foi visto na sessão anterior, o driver (módulo) usado é um genérico SVGA, mas a placa de vídeo é a SIS 620. Faz menção ao tipo de monitor e, na Subsection *Dysplay*, ficam as configurações da resolução. Neste caso, *Depth 16*, significa 16 bit e *Modes 800X600* foi a configuração que escolhi. Nada impede de ter várias resoluções, que podem ser alteradas apertando a tecla *Alt + Ctrl + "+"* ou *"-"*. Se uma resolução que foi configurada por você, mas não é suportada pelo monitor e vídeo, provavelmente o X irá congelar. Pressionando *Alt + Ctrl + Back Space*, o X é finalizado.

```
Driver      "svga"
    # Use Device "Generic VGA" for Standard VGA
320x200x256
    #Device    "Generic VGA"
    Device     "SiS 620"
    Monitor    "Generic Monitor that can do 1280x1024 @
60 Hz"
    DefaultColorDepth 16
    Subsection "Display"
        Depth    16
        Modes     "800x600"
        ViewPort  0 0
    EndSubsection
EndSection
```

Terminada a configuração do vídeo e monitor, caso ocorra algum problemas, o ideal é ter os dados do fabricantes, principalmente



do monitor, que pode ser consultado através do manual ou diretamente com o fabricante.

Mouseconfig - Configurando o Mouse

O *mouseconfig*, como o nome diz, é configurador do mouse. Digite *mouseconfig* em um terminal, escolha o tipo do seu mouse e note que há um item para emular três botões. Este item é usado quando o mouse é de apenas dois botões. Quando marcada essa opção, ele emulará o terceiro. Mas qual a utilidade do terceiro botão?

No **Linux**, ele é muito útil, por exemplo, para copiar e colar um texto:

No Windows, você seleciona o texto, digita *Ctrl + C* para copiar e, depois, *Ctrl + V* para colar.

No **Linux**, você seleciona o texto e ele copia automaticamente, bastando pressionar o terceiro o botão do mouse (localizado entre o direito e esquerdo) sobre o local que deseja colar. Caso não tenha o terceiro botão, você poderá apertar o dois botões do mouse ao mesmo tempo (Emular Botão), passando para o **Linux** a informação de que o terceiro botão foi apertado.

Também é muito utilizado na interface gráfica Windows Maker, que falaremos no próximo capítulo, referente a interfaces gráficos.

SndConfig - Configurando o Som

Este aplicativo faz uma autodetecção da sua placa de som. A sintaxe a ser digitada no terminal é *sndconfig*. Ele detectará sua placa logo após ele perguntar se a mesma foi detectada corretamente. Caso isso tenha ocorrido, o aplicativo fará um teste com algumas palavras de Linus, o criador do **Linux**.

Sempre é bom verificarmos a compatibilidade novamente no sites dos fornecedores de **Linux**. No caso da conectiva, www.conectiva.com.br/hardware. Algumas placas não são plug-and-play, ou são ISA's, devem ser verificados os jumpers para não travarem seus endereços de IRQ e DMA.



Outro grande problema são as placas onboard, os fabricantes não liberam drivers (módulos) para **Linux**, e o que se tem no mercado foi produzido por esforços de indivíduos da comunidade **Linux**.

Temos alguns sites muito bons, dentre eles: <http://www.onbord.hpg.ig.com.br/index.htm> , onde existem módulos e How-to de algumas placas de som, CMI8330, CMI8338, CMI8738. Temos também www.dominiolinux.com.br . Aqui, daremos um exemplo de como configurar a placa CMI8730 Onboard.

Primeiramente, você deve baixar o arquivo *cmpci-xx.tar.gz* de qualquer um dos sites acima citados.

O que iremos fazer é utilizar um módulo, como os drivers para Windows. Mas, no nosso caso, teremos que recopilar o Kernel e recopilar o Kernel? O que é isso?

Compilar e fazer instruções em linguagem que o homem entende virarem instruções que os computadores entendam, os famosos binários. Quando instalamos o **Linux** pela primeira vez, é copiado o Kernel com as opções mais usuais. Talvez os novos Kernels, a partir da versão 2.4.x, não necessitem, mas os antigos, como o 2.2.x necessitam deste procedimento. Maiores instruções em www.kernel.org.

Então, vamos colocar a “mão na massa”! Vá para o terminal e acesse o diretório */usr/src/linux/drivers/sounds*, utilizando o comando *cd /usr/src/linux/drivers/sounds* .




Vamos fazer um backup de dois arquivos:

```
cp Makefile Makefile.bak <enter>
cp Config.in Config.in.bak <enter>
```

Agora, vamos descompactar o arquivo *cmpci*:

```
tar -xvzf cmpci-xx.tar.gz
```

O *tar* é usado para compactar e descompactar arquivos. Ele trabalha em conjunto com o *gunzip*. Na realidade, o *tar* guarda as informações dos diretórios originais enquanto o *gunzip* realmente compacta o arquivo. Dessa forma, temos :

-  **-x:** Usado para extrair informações do arquivo .tar;
-  **-v:** O modo *verbose* mostra na tela a lista de arquivos e diretórios extraídos do *tar*;
-  **-z:** Informa que o arquivo foi compactado pelo *gunzip* com extensão .gz;



-f: Informa para o comando *tar* que se trata de um arquivo a ser extraído, no nosso caso, o *cmpci*.

Caso o diretório citado não exista, é necessário instalar o código fonte do Kernel. Abaixo, segue uma tabela com todos os pacotes que devem ser instalados. Caso não saiba instalar novos pacotes, dê uma verificada no capítulo 4, que ensina a instalar pacotes tipo *rpm* e *tar*, montar dispositivos do tipo CD-ROM e floppy.

Construir TABELA

Pacote	Colocação na Instalação	Diretório no CD
glibc-devel	DESENVOLVIMENTO/ BIBLIOTECAS	conectiva/RPMS/glibc-devel (...)
Pacote	Colocação na Instalação	Diretório no CD
ncurses-devel	DESENVOLVIMENTO/ BIBLIOTECAS	conectiva/RPMS/ncurses-devel (...)
Pacote	Colocação na Instalação	Diretório no CD
automake	DESENVOLVIMENTO/ FERRAMENTAS	conectiva/RPMS/automake (...)
Pacote	Colocação na Instalação	Diretório no CD
make	DESENVOLVIMENTO/ FERRAMENTAS	conectiva/RPMS/make (...)
Pacote	Colocação na Instalação	Diretório no CD
egcs	DESENVOLVIMENTO/ LINGUAGENS	conectiva/RPMS/egcs-1.1 (...)
Pacote	Colocação na Instalação	Diretório no CD
Kernel-headers	DESENVOLVIMENTO/ SISTEMAS	conectiva/RPMS/kernel-heade (...)
Pacote	Colocação na Instalação	Diretório no CD
Kernel-source	DESENVOLVIMENTO/ SISTEMAS	conectiva/RPMS/kernel-source (...)

Depois de descompactado o arquivo e instalado os pacotes, deve-se acessar o diretório:

```
cd /usr/src/linux <enter>
```

Se você já estiver utilizando o modo gráfico, pode digitar :

```
make xconfig <enter>
```

No modo texto, digite:



```
make menuconfig <enter>
```

De qualquer forma, é aconselhável utilizar o modo texto.

Procure a opção *sound*, sete o item *Sound Card* para <M>, que significa módulo, e em *CMPCI driver CONFIG_SOUNDSPCI*, também setado para <M>. Dependendo da versão do seu Kernel, pode não haver estas linhas, mas somente a *Support for CMEDIA Sound Chips* também seta para <M> sem problemas.

Digite o comando abaixo e irá recompilar os módulos. Vá tomar um cafezinho, pois pode demorar até uns 15 minutos:

```
make modules <enter>
```

O código abaixo instala os módulos :

```
make modules_install <enter>
```

O seguinte cria a relação de dependência entre outros módulos:

```
depmod -a <enter>
```

Agora, vamos carregar o módulo:

```
modprobe cmpci
```

Ou:

```
insmod cmpci <enter>
```

O comando abaixo mostra todos os módulos carregados *e cmpci* que devem constar nesta lista.

```
lsmod <enter>
```

Você pode testar colocando um CD de música para tocar, ou melhor ainda, digitando o comando:

```
cat arquivo.wav > /dev/audio
```

Mas, verifique se o arquivo áudio existe no diretório */dev*. Para isso, utilize os comando *cd* e *ls*, que foram vistos anteriormente. Caso não exista, entre no diretório *dev* e digite:

```
./MAKEDEV audio
```

O processo está quase concluído, mas precisamos fazer com que o módulo seja carregado automaticamente todas as vezes que o **Linux** for reiniciado.

Vá em */etc* e edite o arquivo *conf.modules*:

```
vi /etc/conf.modules ou modules.conf
```



Aperte a tecla *i* para entrar em modo de edição.

Digite uma linha com o seguinte dados:

```
alias sound cmpci
```

Feito isso, tecele *Esc* para sair do modo de edição.

Pressione “:” (dois pontos) para podermos entrar em modo de linha de comando.

Digite *wq* <enter>, onde *w* grava as alterações e *q* sai do editor *vi*, como na figura abaixo:

```
alias parport_lowlevel parport_pc
alias block-major-58 lvm-mod
alias block-major-109 lvm-mod
alias loop0 loop
if `kernelversion` >= 2.4
alias char-major-108 ppp_generic
alias /dev/ppp ppp_generic
alias tty-ldisc-3 ppp_async
alias tty-ldisc-14 ppp_synctty
alias ppp-compress-21 bsd_comp
alias ppp-compress-24 ppp_deflate
alias ppp-compress-26 ppp_deflate
alias ppp ppp_async
alias rt18139 8139too
alias old_tulip tulip
endif
alias usb usb-ohci
alias eth0 dfe
alias sound-slot-0 es1371
post-install sound-slot-0 /bin/auxix-minimal -f /etc/auxixrc -L
pre-remove sound-slot-0 /bin/auxix-minimal -f /etc/auxixrc -S
"
```

Colocando essa linha sempre que reiniciar o **Linux**, ele carregará o módulo automaticamente, evitando que tenhamos de digitar sempre *lsmod cmpci*.

Outras configurações podem ser achadas nos sites abaixo:

<http://www.onbord.hpg.ig.com.br/index.htm>.

<http://www.dominiolinux.com.br>.



Modems

Vamos configurar o modem com interface gráfica *kppp*, do KDE, que na realidade chama o programa *pppd*, que é o responsável para realizar a conexão.

Os modems devem estar corretamente jumpeados para porta COM correta.

Mas antes de configurarmos uma conta, primeiramente configuraremos o hardware. As portas COM, no **Linux**, são chamadas de *ttyS*. Então:

Para:	Temos:
COM1	ttyS0
COM2	ttyS1
COM3	ttyS2
COM4	ttyS3

Vamos criar um link simbólico, apontando o arquivo */dev/modem* para uma das portas *ttySx*.

Com isso, faremos com que o arquivo responsável pelo modem aponte para a porta COM correta. Na realidade, utilizamos arquivos *devices*.

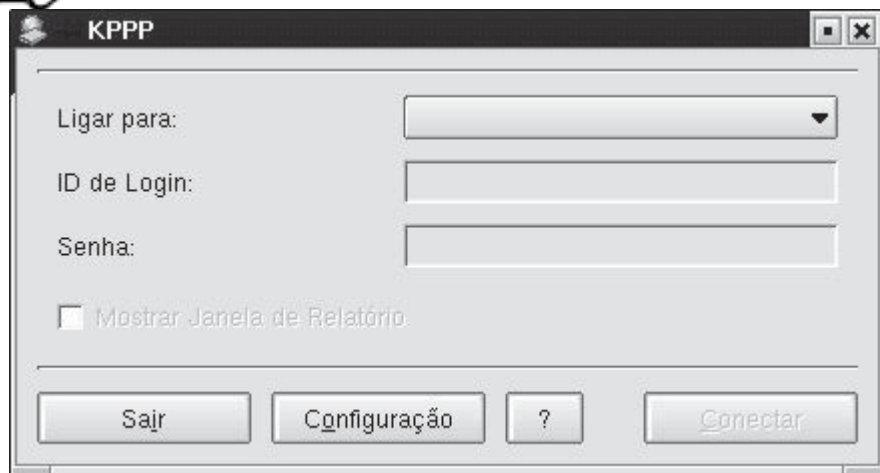
Lembra como criamos arquivos simbólicos? Isso! Utilizando o comando *ln*.

Então, digite:

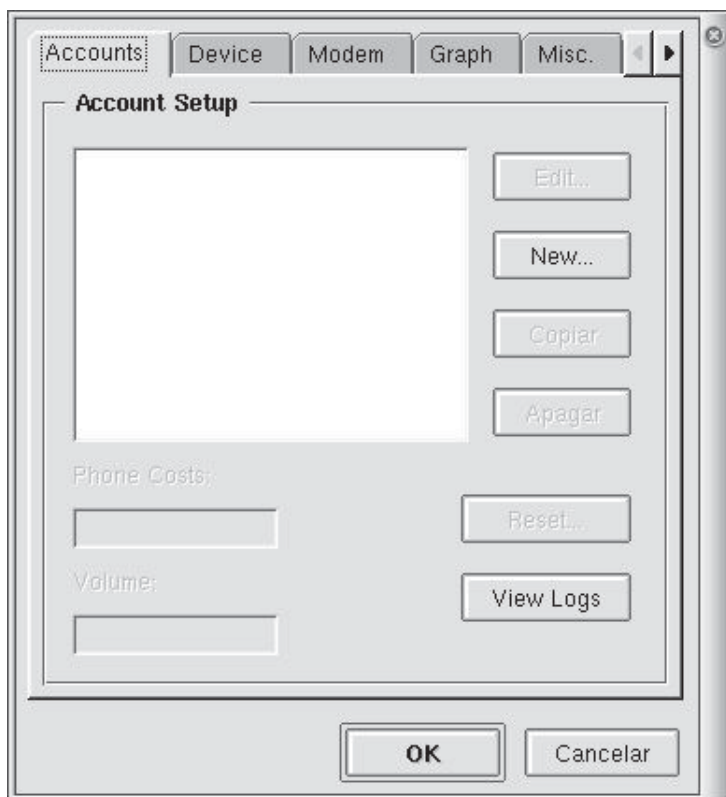
```
ln -sf /dev/ttyS1 /dev/modem <enter>
```

Altera *ttyS1* pela sua respectiva porta COM onde está o modem.

A interface gráfica do *kppp* é a seguinte logo:









A tela de configuração é aberta quando o botão Configuração é selecionado, como na figura abaixo:



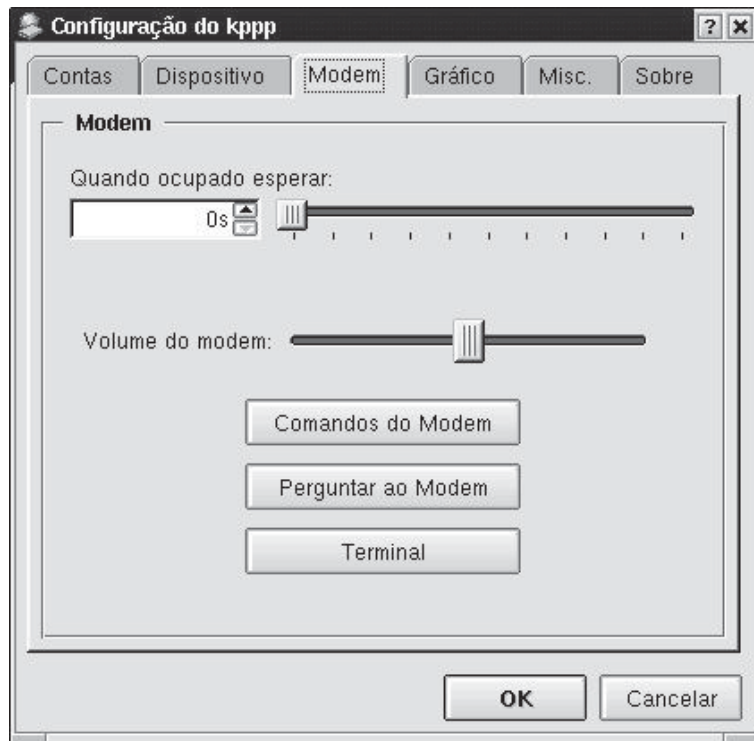
Na guia *account*, clicando em *New*, será solicitado os dados para conexão como:



-  Número do telefone de acesso;
-  Tipo de autenticação;
-  Nome do domínio do provedor (provedor.com[.br]);
-  Número IP do(s) servidor(es) de DNS;
-  Nome de usuário no provedor;
-  Senha do usuário no provedor.

Para verificar se o modem está OK, clique na guia *Modem* e em seguida no botão *Perguntar ao Modem*. Ele deverá responder em que porta se encontra ttyS0, 1 ,2 etc.

Se ele não responder , é porque o modem ainda não foi detectado.



Vamos usar o comando *setserial* para verificarmos o que ocorre na porta específica do modem:

```
setserial /dev/ttyS1
```

(Caso o modem esteja na COM 2, se for COM 3 ttyS2).

Ele deve mostrar algo do tipo:



```
[root@localhost]# /dev/ttyS1, UART: none , Port 0x02f8m
IRQ: 3
```

Vamos pedir para o *setserial* auto configurar o modem da seguinte forma :

```
setserial /dev/modem autoconfig <enter>
setserial /dev/modem
```

Se algum dado estiver incorreto, pode-se alterar como no exemplo abaixo, que muda a porta IRQ e mais alguns itens:

```
setserial /dev/ttyS1 irq 5 uart 16550a port 0x03e8
```

A configuração que fizemos foi alterar a IRQ como 5, a uart do modem e a porta de entrada e saída.

Para tornar isso automático, adicione esta linha de alteração no arquivo */etc/rc.d/rc.local* para que ele altere essas configurações automaticamente.

Altere as permissões do *kppp* e *pppd* da seguinte forma:

```
chmod a+s /usr/sbin/pppd
chmod a+s /usr/bin/kppp
```

E retire a linha onde contém a palavra *lock* do arquivo */etc/ppp/options*. Esse *lock* faz com que quando estivermos conectados não consigamos abrir nenhum outro programa. Não é muito útil.

Ainda temos outra opção, que é o *pnpdump* e *isapnp*, que são aplicativos para configurar modems plug-an-play ou outros dispositivos quando os mesmos apresentam problemas de conflito nas portas IRQ ou de entrada e saída.

Digite :

```
pnpdump -c > /etc/isapnp.conf <enter>
```

O procedimento acima cria o arquivo de configuração *isapnp.conf*, que contém as configurações de dispositivos de hardware. Para testar, digite:

```
isapnp /etc/isapnp.conf
```

Se ocorrerem erros, edite o arquivo, comente o bloco que não está comentado e retire o comentário do bloco desejado.

Vamos supor que o arquivo abaixo, *isapnp.conf*, esteja editado:

```
# Device supports vendor reserved register @ 0x39
Device supports vendor reserved register @ 0x3a
```



A primeira linha está comentada, ou seja, tem o sinal de sustenido (#) na frente. Este tipo de argumento faz com que o **Linux** desconsidere essa linha. No nosso caso é muito útil, pois comentando e retirando os comentários das linhas que já estão no arquivo, você terá a possibilidade de testar várias configurações até acertar a sua.

Existe um software para configurar automaticamente hardwares plug-and-play, chama-se *isapnpcfg*. Pode ser encontrado em sites de procura **Linux**, como:

<http://rpmfind.net/> e <http://freshmeat.net/>.

Quanto aos winmodems do tipo Pctel (HSP), Lucent, Motorola, Cirrus, entre outros, há uma configuração própria para cada. Atualmente, o usual é a instalação de um pacote rpm (veja no capítulo 4), pois ele faz tudo de forma automática.

Segue abaixo alguns sites com módulos de vários tipos de winmodems:

<http://www.jcmp3.cjb.net/>.

<http://www.dominiolinux.com.br>.

<http://www.onbord.hpg.ig.com.br/index.htm>.

Dentro da própria Conectiva, fica disponibilizado para consulta e download todos os livros que acompanham a distribuição.

Pode-se, e é até recomendável, procurar mais informações em listas de discussões. A <http://linux-br.conectiva.com.br/> é uma ótima lista de discussão. Foi por lá que aprendi muitas coisas interessantes do **Linux** que hoje passo para vocês.

Para finalizar todos esses procedimentos, *Xconfigurator*, *sncconfig*, *mouseconfig* e *kbdconfig*, entre outros, podem ser configurados através do setup na Conectiva, ou seja, ele agrupa todas essas configurações facilitando o uso.

Faltou apenas configurar a placa de rede, mas esta eu deixarei para um outro capítulo, onde estaremos configurando não só a placa de rede, mas também uma rede interna com acesso a Speedy e compartilhamento de conexão.

Tenho certeza que será muito interessante o próximo capítulo, pois nele falaremos de interfaces gráficas, desde KDE, Ximiam Gnome, até aquelas que podem rodar com 16 Mb de RAM, ICE WM, Black Box, etc.



Introdução

No **Linux**, a interface gráfica pode ser mudada como uma roupa que você troca. Para cada interface, ainda temos vários temas que podem mudar a sua cara.

Mas antes de falarmos propriamente das interfaces gráficas, temos que falar sobre o seu gerenciador.

Quem é esse “cara”?

Ele é o primeiro que aparece quando o **Linux** é iniciado em modo gráfico. Ele é responsável por gerenciar os vários ambientes Desktop.

Seria esse abaixo:



KDM: Gerenciador de interface do KDE

A figura acima mostra o KDM, que é o gerenciador padrão do KDE, instalado por default em numa instalação Conectiva.



É a forma de fazer logon no modo gráfico. Em *Tipo de sessão*, você irá informar qual o ambiente que deseja iniciar. No caso, temos o KDE, mas poderia, sem nenhum problema, iniciar o Gnome, Window Maker, etc.

Foi dito no capítulo anterior o que diferencia o **Linux** quando inicia em modo gráfico e console. Vamos refrescar a memória?

Temos um arquivo em */etc* chamado *inittab*. Dentro desse arquivo, a seguinte linha:

```
id:5:initdefault:
```

Essa linha informa para o **Linux** que iniciará no modo gráfico, que por sua vez iniciará o gerenciador de interface e logon, o KDM.

Se trocarmos 0 número cinco por três, ele iniciará em modo console.

Em modo console, para acessarmos um ambiente, simplesmente chamamos por comando, por exemplo:

```
kde <enter>, gnome <enter> , wmaker <enter>, etc...
```

Então você tem duas opções: iniciar em modo texto e, através de comandos, iniciar o ambiente gráfico, ou ir diretamente para o ambiente gráfico.

Quando vamos diretamente para o ambiente gráfico, também podemos escolher qual o gerenciador de interface que queremos utilizar.

Podemos optar pelo *xdm*, o primeiro que foi lançado, um pouco mais simples que os outros.

Em seguida, temos o *kdm*, do ambiente KDE, e *gdm*, do ambiente Gnome.

Para alteramos o gerenciador de interface, temos também que editar o arquivo *inittab* dentro do diretório *etc*.

Existe uma linha onde constam tais configurações:

```
#Run xdm in runlevel 5
x:5:once:/usr/X11R6/bin/xdm -nodeamon
```

Para se alterar para o *kdm*, modifique as últimas linhas:

```
x:5:once:/usr/bin/ kdm - nodeamon
```

No caso do *gdm*:



```
x:5:once:/usr/bin/ gdm - nodeamon
```

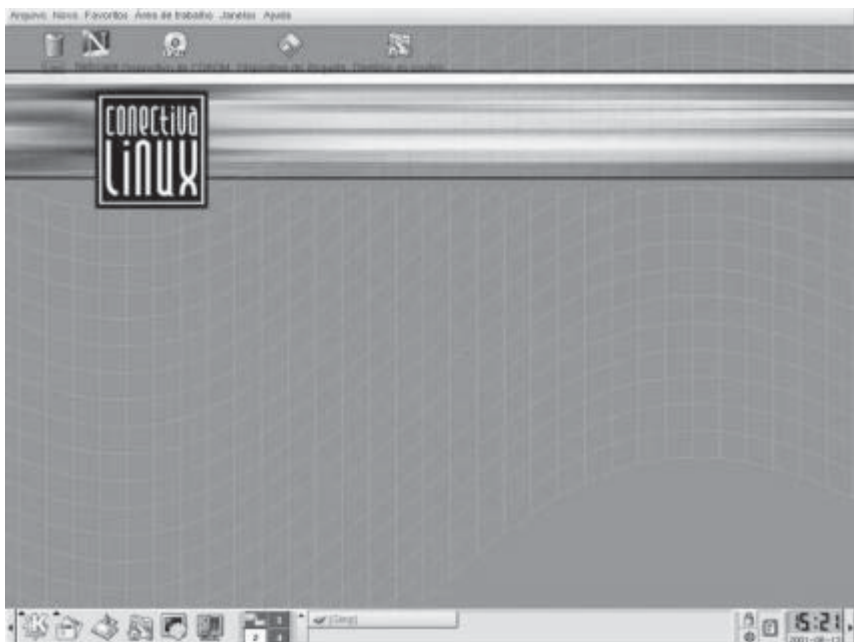
Pode-se alterar o tamanho, a mensagem, a cor , etc. Para mais detalhes, acesse o site:

<http://www.dominiolinux.com.br>

Agora, mostraremos algumas interfaces e suas configurações. Um dos grandes mitos do **Linux** é que ele não tem um ambiente gráfico disponível para usuários que estão migrando. De certa forma, é verdade, pois não existe um ambiente, mas vários. Uns consomem mais recursos, outros feitos para rodar em máquina com poucos recursos de memória RAM e processador. Falaremos, também, de programas para escritório, imagem, som, etc.

KDE - K Desktop Environmet

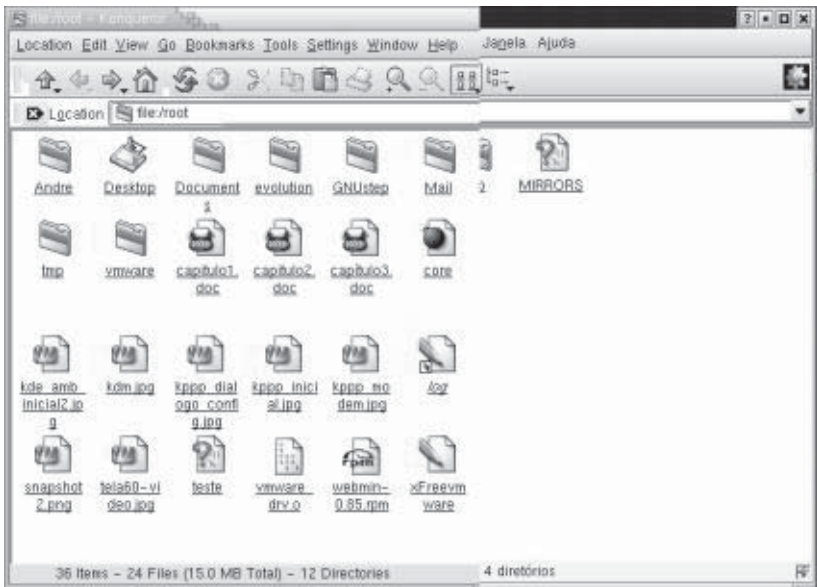
Uma das interfaces mais famosas é o KDE. A distribuição da Conectiva traz a versão KDE 2.2 com muitas inovações. Fácil de usar e disponível em vários idiomas.



Ambiente KDE2

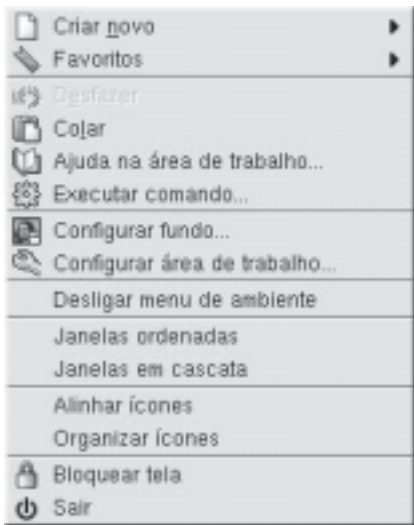


Junto a essa interface, temos muitos aplicativos, como o *konroquer*, que além de ser um gerenciador de arquivos, também é um navegador de web. Veja a figura abaixo:



Konroquer

Clicando com o botão direito na Área de Trabalho, é aberto um menu Pop up, onde se pode executar aplicativos, criar novos ícones, uma gama de utilidades.













Menu do botão direito














Na parte inferior do ambiente, temos a Barra de Ferramentas do KDE, que, como vocês podem, é muito intuitiva. No item K, chamado de menu K, temos várias opções para se abrir programas. O conteúdo você verifica a seguir:

P.S.: Este conteúdo do menu K foi retirado do site da Conectiva, <http://www.conectiva.com.br>.

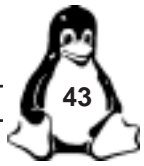


-  **Editores** - Temos, aqui, os ícones referentes a editores de texto.
-  **Gráficos** - Aplicativos gráficos para **Linux**, incluindo capturadores de tela, visualizadores de fax, gerador de fractais, visualizadores de imagens, entre outros.
-  **Internet** - Aplicativos utilizados para se conectar à Internet, navegar (browser), leitura de e-mails e bate-papo.
-  **Multimídia** - Programas utilizados para tocar CD, arquivos midi, além de um mixer.
-  **Escritório** - Neste menu, você encontra os aplicativos do KOffice, incluindo um editor de textos, planilha eletrônica e gerador de apresentações, além de programas para desenho.
-  **Preferências** - Configurações personalizadas do KDE.
-  **Sistema** - Integração do KDE com o sistema, aplicativos que fornecem dados sobre o sistema do usuário, tanto da máquina como sobre o KDE, permitindo efetuar configurações nesses elementos.
-  **Brinquedos** - Aqui, você pode encontrar as dicas do Kandalf (Kandalf é um mago que dá dicas de utilização do KDE).
-  **Utilitários** - Programas simples mas que ajudam bastante no dia-a-dia, como, por exemplo, um armazenador da área de transferência e um livro de endereços.
-  **Ajuda** - Ajuda do KDE em formato HTML. É aberta pelo Konqueror, que é um web browser do KDE.



-  **Centro de Controle** - Aplicativo de configuração do KDE. Agrupa informações dos menus *Preferências* e *Sistema*, permitindo um melhor controle sobre o Gerenciador de Janelas.
-  **Diretório do Usuário** - Abre o Gerenciador de Arquivos (Konqueror) no diretório home do usuário.
-  **Procurar Arquivos** - Aplicativo que procura por arquivos no seu disco rígido.
-  **Favoritos** - Um menu para acesso rápido aos seus favoritos.
-  **Documentos Recentes** - Exibe uma lista com os últimos documentos abertos por aplicativos do KDE, como, por exemplo, o editor avançado.
-  **Navegador Instantâneo** - Abre uma estrutura de menus com os diretórios de seu sistema de arquivos e fornece a opção de abrir um terminal em um diretório.
-  **Executar** - Executa um comando digitado pelo usuário. Guarda um histórico dos comandos digitados anteriormente.
-  **Menu do Painel** - Permite que você configure a Barra de Ferramentas, suas propriedades e seus aplicativos, além do tamanho dos ícones.
-  **Sobre o KDE** - Exibe informações sobre o KDE.
-  **Bloquear a Tela** - Trava a janela do KDE com a proteção de tela atual, pedindo a senha do usuário para liberar a janela.
-  **Sair** - Sai do KDE, perguntando se deseja salvar as configurações feitas.

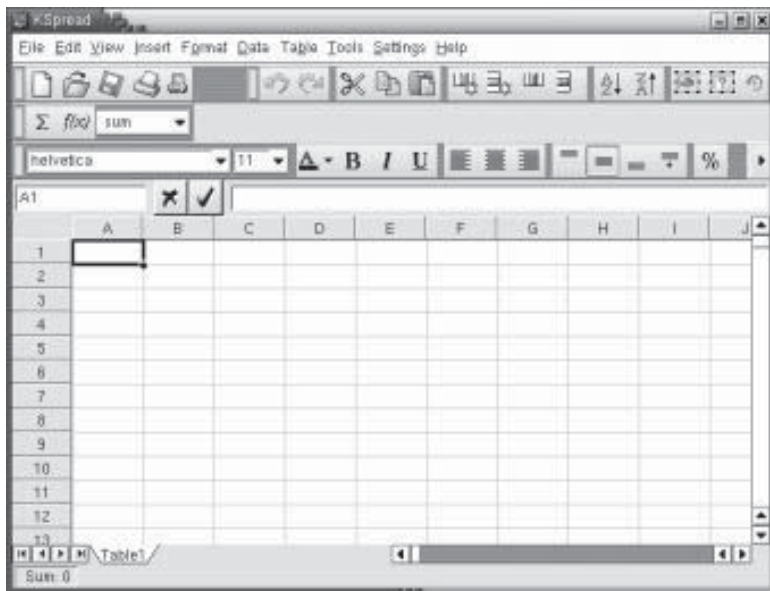
As configurações vistas acima podem ser feitas através do *Centro de Controle*, ou *KDE Control Center*, onde é possível configurar o ambiente de trabalho, tal como a Barra de Tarefas, ícones, cores, fontes, tamanho das fontes, tamanho dos ícones, fundo dos vários ambientes (a princípio 4), protetores de tela, som, temas, etc. Então, quando quiser configurar algo no KDE vá ao *Control Center*.



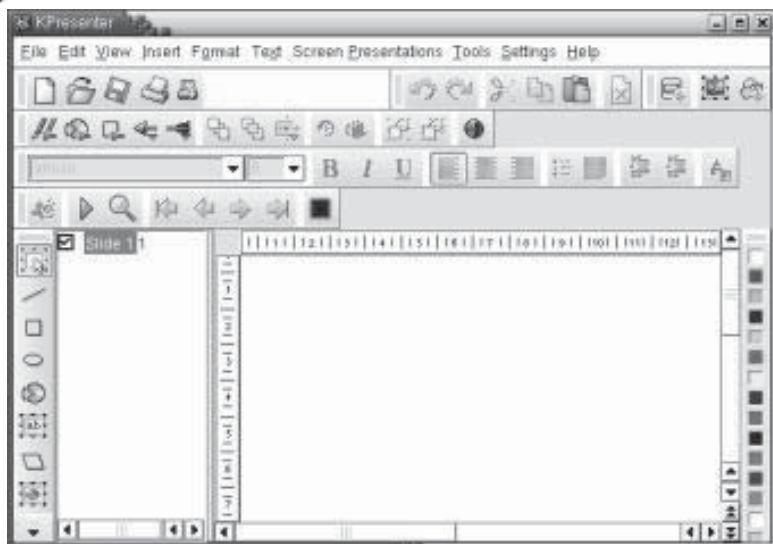
KOffice - Office do KDE

Ainda não acabou. Junto ao KDE, é instalado o KOffice, que são aplicativos de escritório similares ao Office, da Microsoft. Em seguida, mostrarei alguns dos aplicativos.

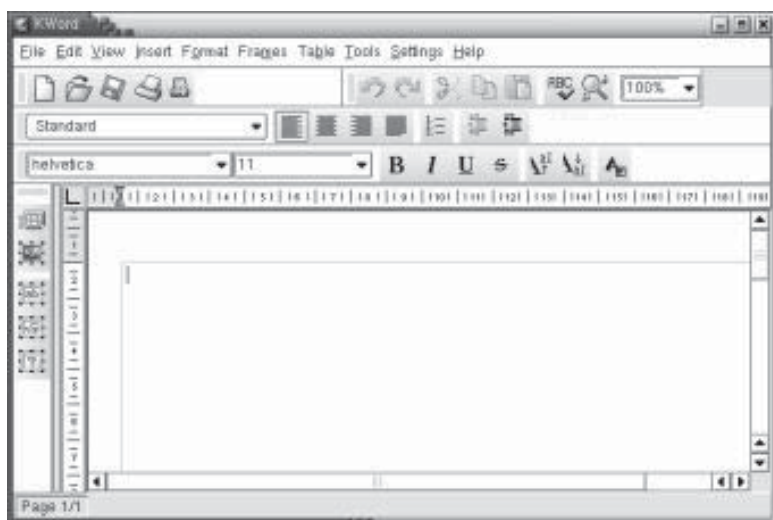
O primeiro é o KSpread, similar ao Excel, da Microsoft. Depois, temos o KWord, não precisamos nem falar, a imagem fala por si. Por último, mostro a figura do KPresenter, similar ao Power Point. Além destes, temos outros, como KMail, KWin (Gerenciador de Janelas), KPersonalizer (ajuda o usuário a configurar o KDE, ícones, imagens, sons, etc.), KIllustrator, KChart, KFormula, etc.



KSpread: Pacote KOffice



KWord



KPresenter

Há muitos outros programas e configurações, mas não serão abordados aqui, pois o ideal deste livro é dar um conhecimento em todo o **Linux**, e não só nas partes gráficas. Um outro livro poderia ser criado para se aprofundar este tema.

Gnome - O GNU Network Object Modeling Environment

O Gnome é outra interface gráfica muito parecida com o KDE, mas tem sua própria personalidade. Atualmente, o projeto está no Gnome 1.4, que acompanha o Conectiva, mas deve ser selecionado na instalação, pois não é padrão. O Gnome está sendo apoiado pela Distribuição Linux Red Hat, com isso, vem crescendo muito neste últimos anos. Existe um projeto que eu, particularmente, chamo de Evolução do Gnome. Ele é chamado de *Ximian*.



Ximian

Na figura acima, vemos a interface do Ximian. Como ele não vem por padrão no Conectiva, é necessário instalá-lo.

Existem duas formas de ser instalado: baixando da Internet os binários ou instalando direto da Internet. Vou fazer uma demonstração do segundo método, tendo em vista que teremos um capítulo só de instalação de pacotes.

Primeiramente, como ele não foi feito exclusivamente para a Conectiva, temos que “enganar” o instalador.



Acesse o diretório */etc* e renomeie o arquivo *versão-conectiva* para *versão-conectiva.old*.

```
cd /etc <enter>
mv versão-conectiva versão-conectiva.old
```

Crie um arquivo chamado *redhat-realease*:

```
vi redhat-realease
```

No capítulo cinco, falaremos mais do editor VI.

Adicione na primeira linha do arquivo: *7.0*;

No VI, digite *i* para entrar em modo de edição;

Escreva : *7.0*.

Aperte a tecla *Esc* e em seguida digite *wq*.

Feito isso, você terá fechado e salvo o arquivo.

Como essa instalação será feita pela Internet e é um pouco grande, pode se tornar demorada. Por isso, é interessante que o usuário tenha acesso a Internet via banda larga, com ADSL, cable modem, ISDN, etc.

P.S.: Teremos um capítulo para configurar o acesso à Internet via banda larga.

Entre em um terminal, é importante que você esteja logado na máquina como root. Caso não esteja, digite:

```
su root<enter>
```

Será solicitado a senha do root. Digite a senha para podermos dar início à instalação do Ximian.

Já logado como root, digite:

```
linx -source http://go-gnome.com / | sh
```

O comando *linx*, na realidade, é um browser no modo texto. O que está sendo solicitado é simplesmente que o site do Gnome VI texto seja acessado e um script seja iniciado. Ao terminar o download deste primeiro item, a instalação continuará em modo gráfico.

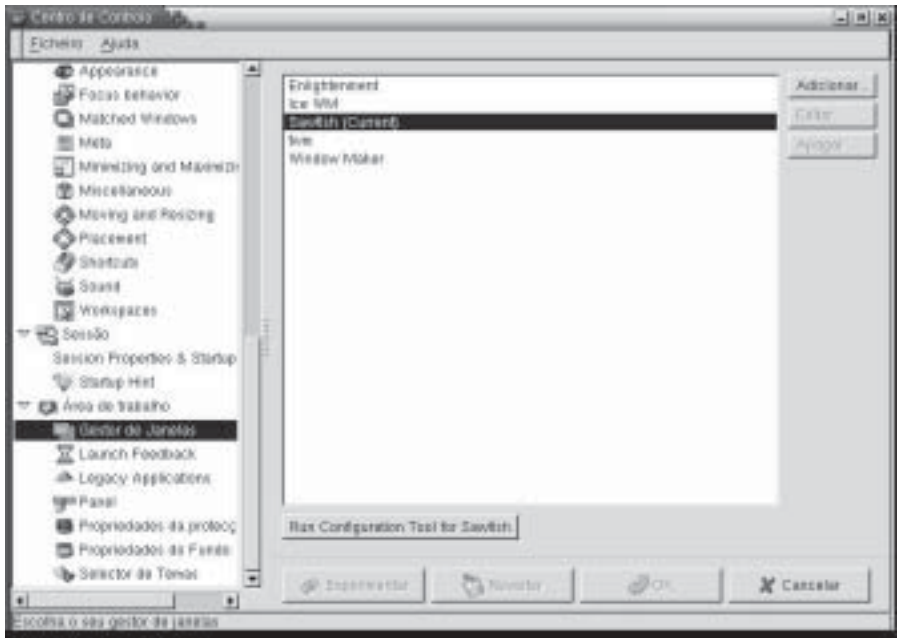
O Ximian tem vários aplicativos que acompanham a interface. Dentre eles, temos AbiWord (processador de texto), Gnumeric, Gnucash (muito familiar ao Money, da Microsoft), Evolution (um ótimo gerenciador de e-mail), Red Carpet (utilizado para instalação e update



de software via Internet - veremos mais a respeito no capítulo sobre instalação), etc.

O Ximian possui seus próprios softwares de editoração, editor de texto, planilhas (Gnumeric), instalador automático de software (Red Carpet), etc.

Suas configurações, seja de teclado, aparência, ícones, atalhos, etc., são efetuadas no centro de controle, que se encontra no menu Gnome\Painel\Global Preferences. A seguir, uma figura mostrando o centro de controle:



Window Maker

O Window Maker também é um “Windows Manager”. É um pouco diferente dos demais, pois trata-se de um ambiente leve e versátil, o que faz dele um ótimo ambiente para máquinas com poucos recursos. Inicialmente, ele foi criado por um brasileiro, e depois, adotado em todo o mundo por vários usuários e programadores. Pode trabalhar juntamente com o KDE ou o Gnome. Seus ícones são criados da forma “clicar e arrastar”, chamado de *Doc app*.



Temos um ícone chamado *Doc* que é onde se deve adicionar mais aplicações. Inicialmente, quando utilizamos pela primeira vez o Window Maker, ele está só. Para adicionar um novo ícone, devemos iniciar a aplicação. Junto com aplicação é criada uma mini-janela. Deve-se clicar sobre ela e arrastá-la até que fique ao lado do Doc.



Ícone Doc

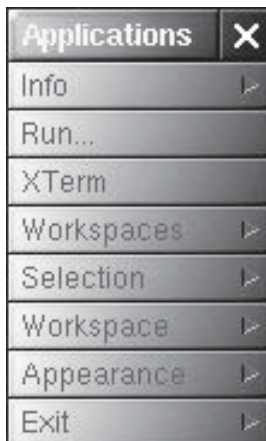
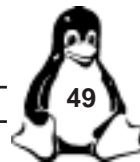
Ele possui muitos temas feitos pelos próprios usuários. Pode-se fazer um download dos mesmos no site :

<http://classic.themes.org/>



Interface gráfica do Window Maker

Clicando com o botão direito do mouse sobre a área de trabalho, surge o menu do Window Maker. Onde podemos acessar vários programas.



Menu Wmaker

Existe um outro ícone chamado *Clip*, utilizado para alternar entre as diversas áreas de trabalho virtuais. Por default, só é criada uma área de trabalho, mas, para que sejam criadas outras, clique com o botão direito no ícone Clip e acesse o menu onde existe a possibilidade de criarmos mais áreas de trabalho virtuais.

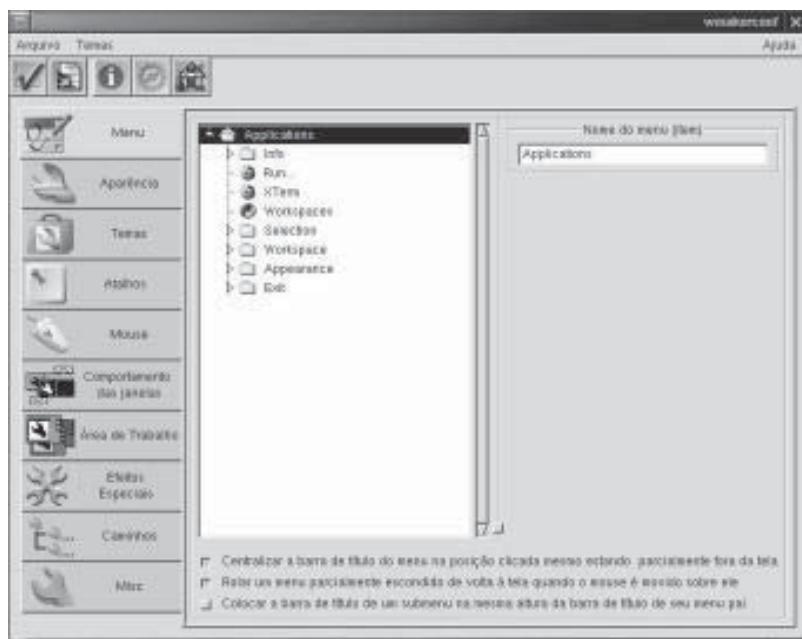


Ícone Clip



Menu do ícone Clip

As configurações do Window Maker são todas feitas através de um programa, O WmakerConf. Através dele, configuramos menus, submenus, aparência, temas, atalhos, mouse, comportamento das janelas, área de trabalho, efeitos, etc. Creio não haver muitos problemas com o configurador, pois trabalha de forma bastante clara. A seguir, uma figura do mesmo:





Este ambiente não deixa a desejar em beleza, porém não necessita de tanto hardware para ser utilizado. Como dito anteriormente, muito útil em equipamento com poucos recursos de hardware.

Gerenciadores Bem Leves

Neste item, falaremos de interfaces leves, que podem rodar até mesmo em um 386 tendo um bom desempenho e flexibilidade. Em especial, falarei do IceWM e o BlackBox. Ambientes como o KDE e o Gnome usam muito mais memória por terem a necessidade de rodar muitos serviços para que seu visual seja elegante e fácil, o que faz perder em performance. Para quem usa microcomputadores que disponham de boa quantidade de memória RAM e processador com alto desempenho, não há problema quanto ao uso de “Windows Managers”, como o KDE e Gnome. Mas, para quem não dispõe, ou até mesmo para quem não se importa em ter uma interface com tantos recursos para ganhar em performance, temos esse outro lado da moeda: os “Window Managers”, como o IceWM e o BlackBox.

IceWM

O IceWM foi construído tendo em vista o bom desempenho e flexibilidade de configuração, utilizando poucos recursos de hardware.

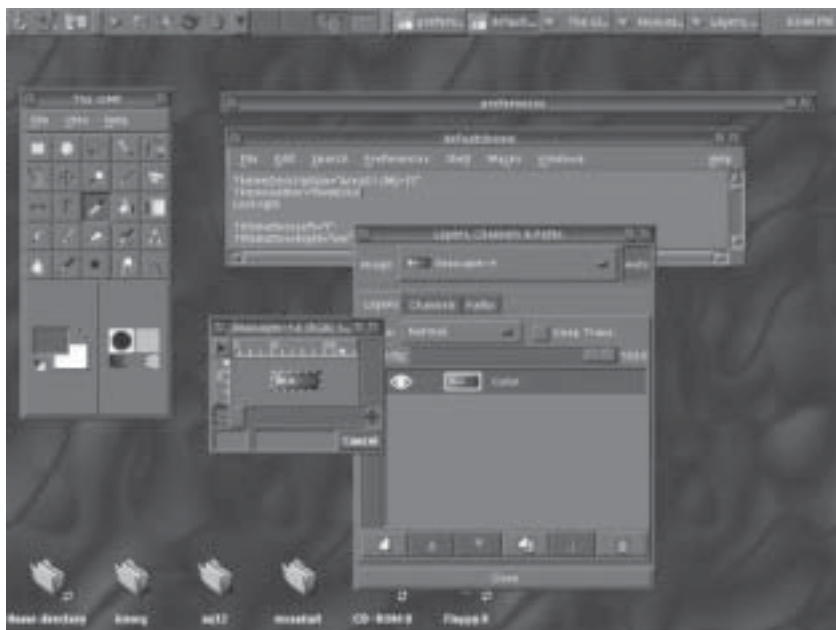
Ele é totalmente configurável. Como o Window Maker, possui vários ambientes e possibilita, por exemplo, que você, simultaneamente, desenvolva algo numa determinada área, como um software com o programa Kylix (Delphi para **Linux**), e em outro ambiente, esteja recebendo e-mail. Torna-se bem mais prático visualmente ter dois ambientes ao invés de um.

Possui suporte a temas, ou seja, é totalmente configurável de acordo com o gosto do usuário. Tem applets, como relógio, monitores que gerenciam várias operações como memória, e-mail, bateria e podem ser adicionados na Barra de Tarefas.

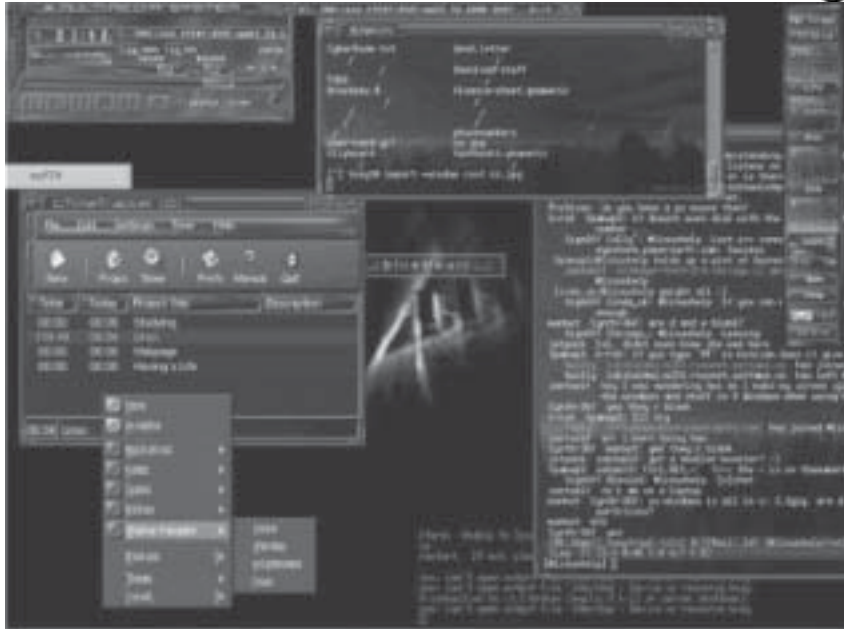
Sua configuração pode ser feita através do IcePref, feito em Python (uma linguagem de programação), ou do IceWMConf, feito em TCL/TK.



O IceWM é compatível com o Gnome. Dessa forma, é possível usá-lo como Gerenciador de Janelas IceWM junto ao Gnome. Logicamente, neste caso, o ganho em performance é insignificante. Mesmo com configurações mais ousadas, ele tem performance melhor que a do KDE e do Gnome.



Interface do IceWM



Uma interface com algumas configurações mais ousadas

A instalação dele é bem simples, consiste em pegar os pacotes *rpm* e descompactá-los. Abordaremos mais sobre instalação no capítulo seguinte.

A maior vantagem dessa interface é o seu uso, que é extremamente simples, e a sua necessidade de menos recursos em hardwares, sendo possível rodar em máquinas mais antigas, como 386 e 486.

BlackBox

O BlackBox será o último gerenciador que estaremos vendo neste livro. Ele é muito parecido com o IceWM no que diz respeito à utilização de memória. Sua aparência não é tão bonita como as outras, pois é um tanto simplista. Porém, possui todos os recursos necessários para utilização com uma estação, sendo possível fazer todas tarefas que são feitas em outros ambientes, como verificação de e-mail, menus configuráveis, Barra de Tarefas simplificada, navegação na Internet, múltiplos ambientes, etc.

Sua economia maior em relação aos outros ambientes está na sua aparência. Ao contrário dos outros ambientes que usam belos pixmap



em todas as suas configurações, o BlackBox usa apenas gradientes, o que deixa a aparência pobre em relação aos demais.

A configuração do BlackBox, diferente dos outros ambientes que têm configuradores, é feita manualmente através de arquivos. Deve-se editar os arquivos e fazer as alterações necessárias.

A configuração é feita através do arquivo *.blackboxrc*. O ponto na frente do arquivo significa que ele é oculto. Para acessá-lo, deve-se colocar o ponto da mesma forma que está. Pode ser visualizado com a opção *-a* do comando:

```
ls:    ls -la <enter>
```

Este arquivo, *./blackboxrc*, é criado automaticamente na primeira vez que saímos do ambiente, sua linguagem não é complicada, sendo quase auto-explicativa. Possui configurações sobre o comportamento de janelas, Barra de Ferramentas, indicação para o arquivo de menus, etc.

Nesse exemplo, iremos configurar nosso próprio menu.

Primeiramente, copiamos o arquivo original para outro lugar, como o diretório do root.

```
cp /usr/local/share/BlackBox/menu /root
```

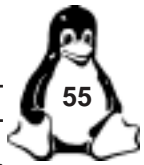
Agora, podemos editá-lo usando o VI ou qualquer outro programa como o Gedit, Kedit, etc.

Abaixo, temos uma pequena parte da configuração de uma menu:

```
[begin] (Blackbox 0.61.x)
[exec] (xterm)      {xterm -ls}
[exec] (rxvt)       {rxvt}

[exec] (StarOffice) {soffice}
[exec] (XEmacs)     {xemacs}
[exec] (Acroread)   {acroread}

[submenu] (Graphics)
  [exec] (XV)          {xv}
  [exec] (The GIMP)    {gimp}
  [exec] (Image Magick) {display}
[end] ...
...
...
...
[end]
```



Parece bem simples. Quando queremos um submenu, o indicamos com o item [submenu], em seguida o seu nome, por exemplo [submenu] *Programas*. Para adicionar um programa de forma similar, mandamos ele executar o binário do programa, como o caso do StarOffice:

```
[exec] (StarOffice) {soffice}
```

Onde *exec* indica que é um programa entre colchetes [], *StarOffice* indica ser um nome para visualizar e, enfim, a chamada ao programa com o *soffice*.

Devo lembrar que só irá funcionar se o programa estiver no path do **Linux**, ou seja, quando iniciarmos o **Linux**. Passado um valor para essa variável path, que mostra os caminhos de onde se encontram os programas, normalmente */usr/bin* (nada impede que você edite o path em */etc/profile*), constará uma linha com o valor do path semelhante a seguinte:

```
PATH="$PATH:/usr/bin:/usr/local/bin"
```

Caso queiramos adicionar, por exemplo, um programa que está no */root*, temos que adicionar no final da linha o */root*:

```
PATH="$PATH:/usr/bin:/usr/local/bin:/root"
```

Depois disso, é necessário dar logoff e logar novamente.

Acabado de editar o arquivo de menu, devemos inserir no seu caminho o arquivo:

```
/.blackboxrc
```

Na linha *session.menufile*, colocamos o novo caminho */root*.

Quando o instalei, os pacotes vieram da forma de binário, ou seja, pacotes do tipo *tar.gz*. Foi necessário descompactá-lo e instalá-lo com comandos predefinidos, que veremos adiante. Ele verificou a falta do X. Na realidade, o que faltava era o pacote de desenvolvimento do X, o *Xfree86-devel-4**. Instalei, mas tive problemas para iniciá-lo através da linha de comando, pois o mesmo reclamava que não conseguia abrir o display, que é uma variável usada para abrir a parte gráfica. Dessa forma, fiz um pequeno arquivo para poder abrir o ambiente. Siga a receita abaixo que tudo dará certo. Falaremos mais adiante sobre este tipo de arquivo, comumente chamado de *Script* ou *Shell Script*.

Crie um arquivo chamado *blackopen*, por exemplo. Pode ser criado através do VI:

```
vi /root/blackopen
```



Ou através de interfaces gráficas como o *gedit*.

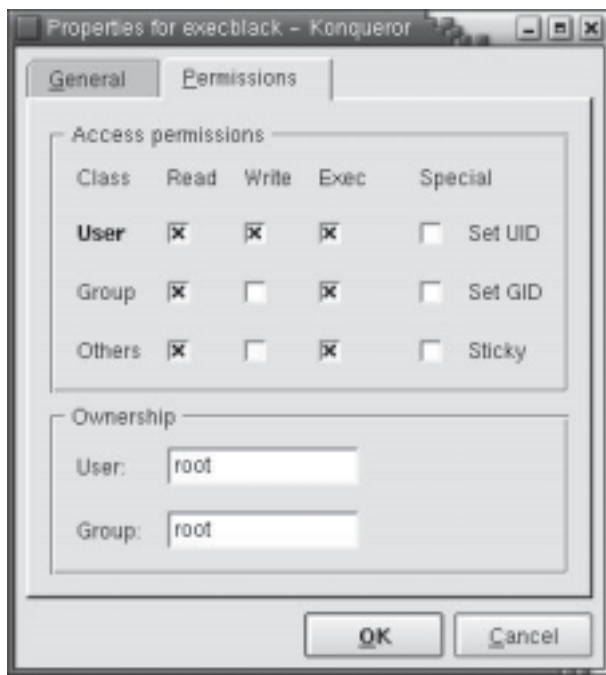
Dentro do arquivo, coloque o seguinte:

```
#!/bin/bash
echo "exec blackbox" >/$HOME/xinitrc
startx
```

Salve e feche o arquivo.

Agora, vamos mudar as permissões dele e transformá-lo em um arquivo executável.

Pode ser feito de duas maneiras. Uma delas é através de qualquer Gerenciador de Arquivos, como o Konqueror, Nautilus e outros. Clique com o botão direito sobre *Propriedades* e, posteriormente, *Permissões*. Marque a opção *Exec* para *User*, *Group* e *Others*, como na figura:



A outra forma é manualmente no terminal gráfico:

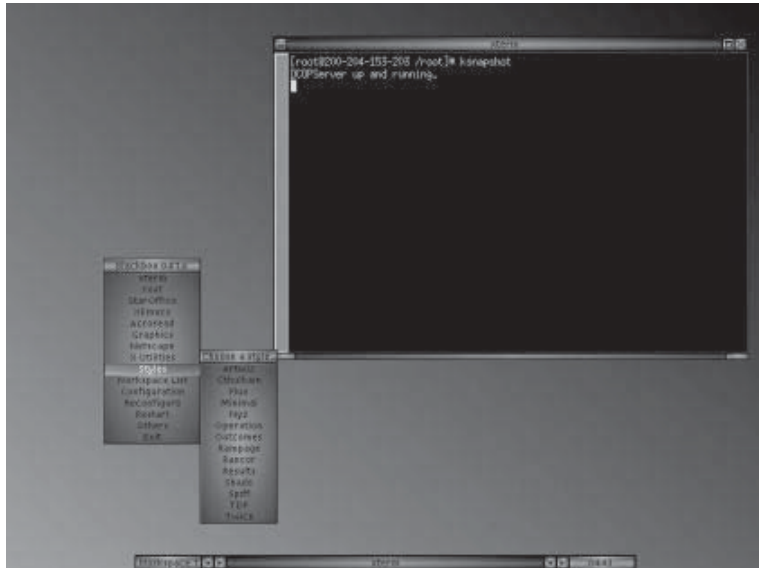
```
chmod 755 /root/blackopen
```

Agora, precisamos colocá-lo em um diretório que esteja no path do **Linux**. Normalmente, coloco em */usr/bin*. Também pode ser feito


```
mv /root/blackopen /usr/bin
```

```
blackopen <enter>
```

Em seguida, a interface se abrirá.



Como vemos acima, a interface é bem simples, mas com certeza vai rodar com uma performance melhor do que a do KDE e Gnome.

Foi feita pela revista do **Linux** (www.revistadolinux.com.br) uma comparação entre as várias interfaces gráficas. Os testes foram realizados com um Pentium com 64 Mb, instalado um Suse Linux 6.2, com as configurações padrões da distribuição.

Gerenciador	Memória livre (Kb)
Modo texto	30764
BlackBox 0.40.9	23540
IceWM 0.9.42	23416
WindowMaker 0.60.0	20904
KDE 1.1.1	8376
Gnome 1.0.7	4660



Com a tabela acima, podemos verificar a diferença entre os ambientes, desde o BlackBox, ocupando 40 Mb, até o Gnome, ocupando 60 Mb. Uma boa opção é o Window Maker, que totaliza quase 45 Mb, mas tem uma interface muito interessante.

A parte de interfaces gráficas termina por aqui, mas antes de concluir o capítulo, falaremos de alguns programas para interfaces gráficas.

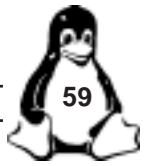
Programas

A variedade de programas para **Linux**, com ou sem interface gráfica, é realmente impressionante. Todos os dias surgem mais programas, feitos pela comunidade **Linux**, seguindo o movimento Open Source, que tem o código aberto e pode ser adquirido sem custos financeiros.

Comentaremos, agora, sobre alguns programas para escritórios, como editores de texto, planilhas, apresentações, etc.

Temos como Offices vários programas, como Gnome Office, KOffice, StarOffice (Open Office), Siag Office, Word Perfect, CorelDraw, Corel Photo Paint, Applix Office, entre outros. Dentro destes, temos vários softwares de edição de textos, planilhas, banco de dados, editores gráficos (GIMP), criador de diagramas, calendários, controle de finanças pessoais (Gnucash), editores HTML, enfim, tudo que é preciso e um pouco mais.



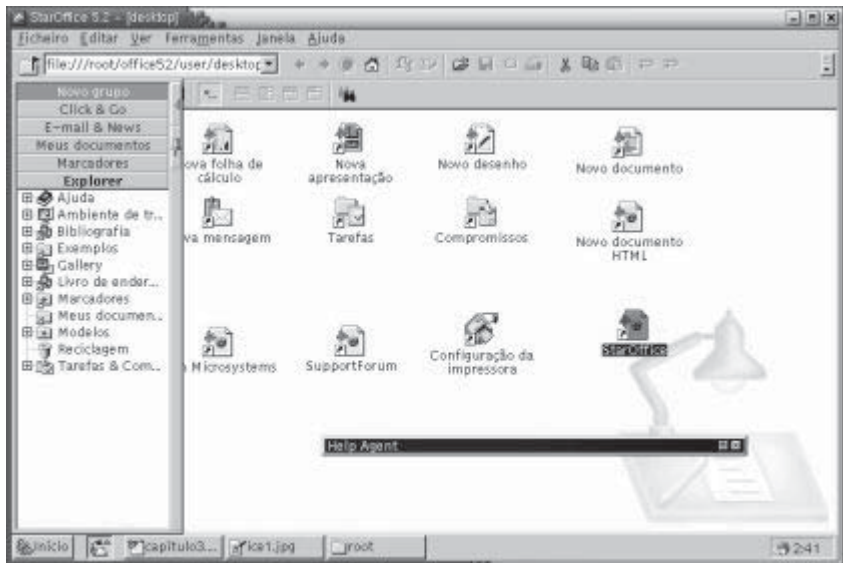


Gimp

Esses softwares, além de serem “free”, são muito bons, como o Gimp, que pode tratar imagens em 2 e 3D.

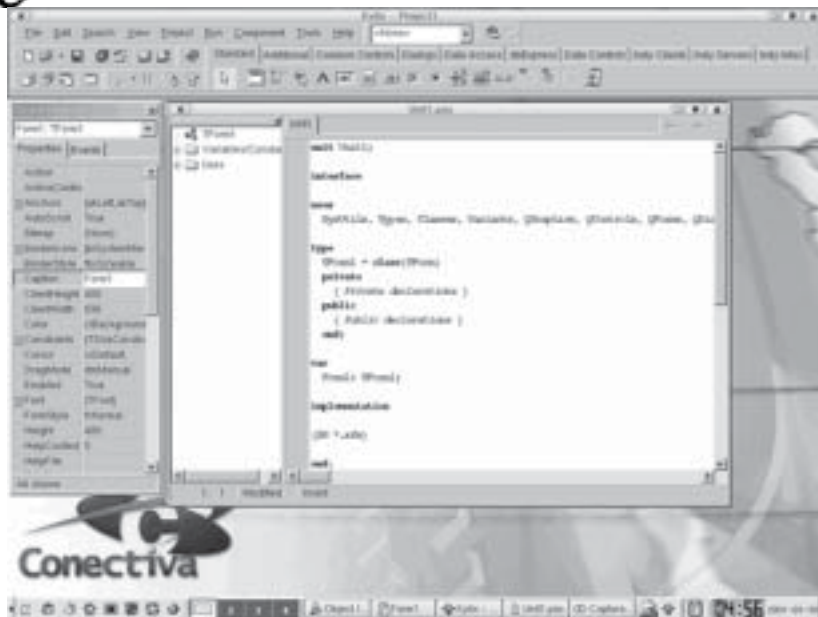
O Star Office da Sun (www.sun.com) pode salvar arquivos com extensões do Microsoft Office, como **doc**, **xls**, etc.

Abaixo, uma figura do Star Office:



Star Office

Para o pessoal da programação, o mais novo integrante da família **Linux** é a Borland, que lançou o Kylix. O Kylix é o Delphi para **Linux**, com uma interface para criação de programas muita parecida com a do Delphi. Na realidade, mudam apenas algumas ferramentas em sua paleta e, logicamente, o modo como trabalha. Ela é uma ferramenta RAD - Aplicação de Desenvolvimento Rápido. É um ambiente de desenvolvimento visual, o qual permite desenvolver aplicações tanto gráficas, como em modo texto.



Interface do Kylix

Também é possível ouvir música sem nenhum problema com o Xmms, que é um dos players de MP3 para **Linux**, mas nada impede que você escute no modo texto.



XMMS

Jogos

Além destes, existem softwares gráficos para FTP, download de MP3, áudio para CD, instalação de pacotes, games, etc.



Games? Sim, Games. Com certeza já existem muitos games, do Doom aos Quakes, Unreal, Rune, Civilization, Tribes, etc.

Aliás, a Lokigames disponibilizou uma interface para instalação automática dos jogos disponíveis em seu site. Neste caso, são versões beta ou demo que podem ser adquiridas. Mas, a interface faz tudo automaticamente pela Internet: download e instalação. Veja sua interface:



www.lokigames.com

Você pode escolher o jogo ou fazer updates de novos jogos clicando em *Update Demos*.

Com isso, desmistificamos a idéia de que o **Linux** é difícil (e não softwares ou interfaces gráficas), ao contrário de muitos softwares, o que torna a escolha até mais difícil por haver tantos softwares bons e sem custo financeiro.

No próximo capítulo, veremos como é feita a instalação de novos pacotes (softwares), updates, tanto no modo texto como no modo gráfico.



Instalação de Softwares

Forma de Instalação

Resolvi falar sobre instalação de novos pacotes pelo fato de ser uma grande dificuldade para usuários que migraram para o **Linux**. Felizmente, este processo nas novas distribuições tornou-se muito fácil, sendo que, com apenas alguns cliques, já se é possível instalar novos softwares.

Há várias formas de se instalar novos pacotes, como através do próprios códigos-fonte, pacotes *rpm*, *pkg* (Distribuição Debian), que podem ser instalados através de comandos ou interfaces gráficas.

A atualização do **Linux** é extremamente recomendável, tendo em vista a correção de novos bugs descobertos e para manter a própria segurança do sistema.

Até pouco tempo atrás usava-se muito os códigos-fonte dos softwares, compilando-os na própria máquina. Normalmente, estes pacotes vêm no formato *tar.gz* e, conseqüentemente, as extensões de arquivos vêm neste formato. E é por esse tipo de instalação que vamos começar. Por quê?

Atualmente, este recurso ainda é usado. Muitas vezes o produtor do software ao invés de fazer vários programas para várias distribuições, ele faz apenas uma, e cada um copia em sua própria distribuição. Acontece este fato muitas vezes com módulos (driver) de hardware, pois normalmente os responsáveis são grupos da comunidade **Linux** que se prontificaram a fazê-lo.

Vamos começar pelos códigos fontes. São disponibilizados no formato *tar.gz* ou *tar.bz*.



Instalando Pacotes do Tipo Tar

O *tar* é um aplicativo muito utilizado em backup, que estaremos falando algo posteriormente.

O *tar* é usado para compactar arquivos, ou então, como no nosso caso, para armazenar todos os arquivos e diretórios envolvidos no processo em um só lugar e guardar o path dos arquivos e diretórios para que quando estes forem descompactados, ocupem o mesmo local de quando compactados. Os arquivos e diretórios compactados ou não pelo comando *tar*, têm extensão **.tar**.

É usado juntamente com o *tar* outro aplicativo, como o *gunzip* ou o *bunzip2*, que são os verdadeiros responsáveis pela compactação do arquivo.

Para entendermos melhor todo conteúdo, vamos supor que temos um arquivo *opera.tar.gz*.

Vamos descompactá-los da seguinte forma:

Abra um terminal, e digite o comando abaixo no diretório onde se encontra o arquivo:



```
Terminal
Ficheiro Editar Settings Ajuda
[root@dominiolinux linux]# tar -xzf opera.tar.gz
```

Onde **x** corresponde a *extract*, **z** indica que é um arquivo *gunzip*, **v** (verbose) mostra todos os arquivos e diretórios tratados na tela terminal e **f** (file) significa *ARQUIVE* e descompacta no diretório original. Se fosse um arquivo *B72* no lugar de *gunzip*, utilizaríamos o **y** ao invés do **z**.

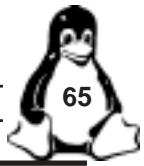
Será descompactado e criado um subdiretório com o mesmo nome do arquivo:

Acesse-o com o comando:

```
cd opera <enter>
```

Esta foi a parte mais fácil. Agora temos que copilá-lo. Mas antes disso, verifiquemos se os pacotes necessários para podermos copilar estão instalados. Para obtermos esta informação, devemos digitar o seguinte:

```
rpm -qa | grep automake <enter>
rpm -qa | grep make <enter>
```

- rpm será o nosso próximo tópico.

No terminal, deverá aparecer a versão dos pacotes *make* e *automake*. Caso não responda nada, é necessário instalar os pacotes que se encontram no primeiro CD da distribuição.

Agora sim, estamos prontos para copilar o código-fonte. Acesse o diretório que foi criado no momento da descompactação. Procure pelo arquivo *README*, pois todo processo de instalação está descrito neste arquivo. Alguns pacotes seguem padrões diferentes dos usuais. Por via das dúvidas, sempre dê uma olhada neste arquivo.

Dentro do diretório, digite os seguintes comandos:

```
./configure <enter>  
make <enter>  
make install <enter>
```

P.S.: Todos os arquivos executáveis que não estão no path do **Linux**, devem ser executados sempre com um *ponto e barra* na frente, pois isso informa para o **Linux** que é um arquivo para ser executado.

Neste procedimento, copilamos e instalamos o aplicativo no sistema. Deve-se prestar bastante atenção, verificando se o procedimento foi completado ou em algum momento ele parou e abortou.

A causa mais freqüente da compilação parar é a falta de librares. Por exemplo, no capítulo anterior, instalei o BlackBox e ele abortou pois não encontrava o X. Na realidade, quando ele fala do X, não está falando do aplicativo em si, mas sim das suas librares. Tive que instalar o pacote *Xfree-devel.*.rpm* que é o correspondente as librares do XFree para que então fosse possível continuar a instalação. No mundo **Linux**, nos chamamos tudo isso de dependências, que indica que um software necessita de alguns pré-requisitos para poder funcionar. Todos os pacotes que estiverem faltando devem ser instalados para que o software possa prosseguir até o final da instalação.

Reforço que a maioria das informações sobre a instalação estão no arquivo *README*, por isso sempre leia este arquivo.

Este é um dos mais complexos modo de instalação, comparando como os modos automáticos, como *apt-get*.



Instalando Pacotes RPM

A próxima forma de instalarmos também é no modo terminal, do tipo RPM, que significa Red Hat Package Manager. Originalmente, o projeto foi desenvolvido pela Red Hat e adotado por várias distribuições. A Red Hat, pensando em facilitar a vida do usuário final, desenvolveu o sistema RPM, que consiste em uma base de dados com a lista de todos os pacotes instalados, suas localizações, dependências e locais onde estão instalados.

Com o RPM, é possível instalar, atualizar, remover e verificar, para que haja uma manutenção total dos pacotes que a máquina contém.

Logicamente, para podermos utilizá-lo, o mesmo deve estar instalado na máquina. Digite *rpm* no console que ele dará várias opções de instalação. Por padrão, o RPM já é instalado pela distribuição no momento da instalação do sistema. Caso não esteja instalado, vá até o site www.rpm.org e pegue os códigos-fonte do *rpm*, algo do tipo *rpm-2.1.*.tar.gz*. Descompacte-os e instale conforme mostrado anteriormente.

Depois de instalado, é só usar. Vamos a formas de usar.

Se quisermos saber se um pacote está instalado, o comando é:

```
rpm -qa | grep nome_do_pacote <enter>
```

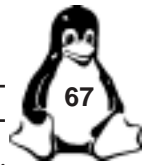
Onde **q** significa **query** e **a** significa **all**. Assim, você está pedindo para ele fazer uma consulta em todos os pacotes. O símbolo **|** (pipe), no **Linux**, serve para canalizar um comando dentro de outro (veremos isso no capítulo sobre modo texto). O comando *grep* localiza pelo nome neste caso. Então, o que aconteceu?

O *rpm* procura no banco de dados o aplicativo *nome_do_pacote*.

Se ele não der nenhum retorno, significa que este pacote não existe.

Como ele não existe, temos que achá-lo na Internet, fazer download do mesmo, para depois podermos instalá-lo.

O site www.rpmfind.net possui uma grande quantidade de pacotes *rpm*, divididos por nome e tipo de distribuição. Normalmente é possível localizar quase todos os necessários. Mas se mesmo assim não achar, procure no repositório e FTP da sua distribuição. E, por último, no próprio site do produtor desse pacote.



Feito download do pacote, *nome_do_pacote.i386.rpm*, precisamos instalá-lo. Acesse o diretório onde o pacote se encontra e digite:

```
rpm -ivh nome_do_pacote.i386.rpm <enter>
```

Onde **i** significa **install**, **h** para mostrar uma série de caracteres do tipo # (sustenido), indicando o progresso da instalação; **v** torna mais descritivo o item **h**, informa quando está verificando as dependências, quando realmente está instalado, etc. Somente com o parâmetro **-i**, já seria o suficiente para instalá-lo, mas como queremos ver o progresso da instalação, utilizamos os outros parâmetros.

Como no caso do código-fonte, os pacotes podem depender de outros para poder funcionar. Neste caso, o RPM informa que existe um pacote faltando no **Linux** para que este novo possa funcionar corretamente. O que fazer?

Procurar este pacote no *rpmfind* ou no site da distribuição, instalá-lo e, depois disso, instalar o pacote principal. Deve-se instalar todos os pacotes que o RPM solicitar.

Outro fato ocorre quando se tenta instalar um pacote que tem dependências e essas dependências não podem ser alteradas. Vamos supor que tenhamos na máquina o KDE 2.0 (Interface gráfica) e resolvemos instalar o KDE 2.2. Ocorreu, porém, que quando fomos instalá-lo, o RPM informou que não existe uma librare (por exemplo, *libxxxx.2.2*). Fizemos, então, uma consulta, como *rpm -qa | grep libxxxx*, e verificamos que existe uma versão desta librare que seria algo do tipo *libxxxx.2.1*. É necessário que se faça um upgrade. O sinal **-u** faz update desta versão, assim teremos que fazer download do *libxxxx.2.2* e um update do pacote antigo:

```
rpm -uvh libxxxx.2.2 <enter>
```

Infelizmente, quando o fomos instalar, ele informa que vários pacotes do Gnome (Interface gráfica) dependem do *libxxxx.2.1* para funcionar corretamente. E agora?

Temos algumas opções. Podemos forçar a atualização com o atributo *—force*, que pode não funcionar enquanto você não desinstalar os pacotes do Gnome que precisam da librare antiga. Complicado né? Você quer o Gnome e o KDE 2.2, o que fazer?

Existe a possibilidade de verificarmos se há alguma versão nova do Gnome para que possamos atualizá-la também de forma que não atrapalhe a instalação da nova versão do KDE, ou forçamos a instalação dessa nova librare, para ficarmos com as duas. Se você usar o atributo



`-ivh`, ele informará que já existe um pacote instalado com aquele novo. Então, façamos:

```
rpm -force -ivh libxxxx.2.2 <enter>
```

Se agora fizermos uma consulta, verificaremos que constam as duas librares como instaladas.

Não é recomendado usar o atributo `—force`, a menos que você saiba bem o que está fazendo.

Temos uma opção muito importante, que é a remoção dos pacotes. De repente, foi instalado um programa que quando foi iniciado você não gostou. Então, remova-o:

Primeiro consultamos :

```
rpm -qa | grep libx
```

O RPM informará que existem dois, o *libxxxx.2.1* e *libxxxx.2.2*.

Então, digite:

```
rpm -e libxxxx.2.1
```

Feito isso, o pacote é desinstalado.

Para facilitar a vida dos usuários, as duas interfaces mais conhecidas criaram ferramentas gráficas para o RPM. São elas GnoRPM, do Gnome, e Kpackage, do KDE.

Instalando pacotes RPMs através do Linuxconf

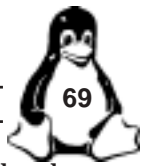
Ainda há a possibilidade de gerenciar os pacotes *rpm* através do Linuxconf. Digite em um terminal ou, no próprio menu, escolha executar e digite:

```
linuxconf <enter>
```

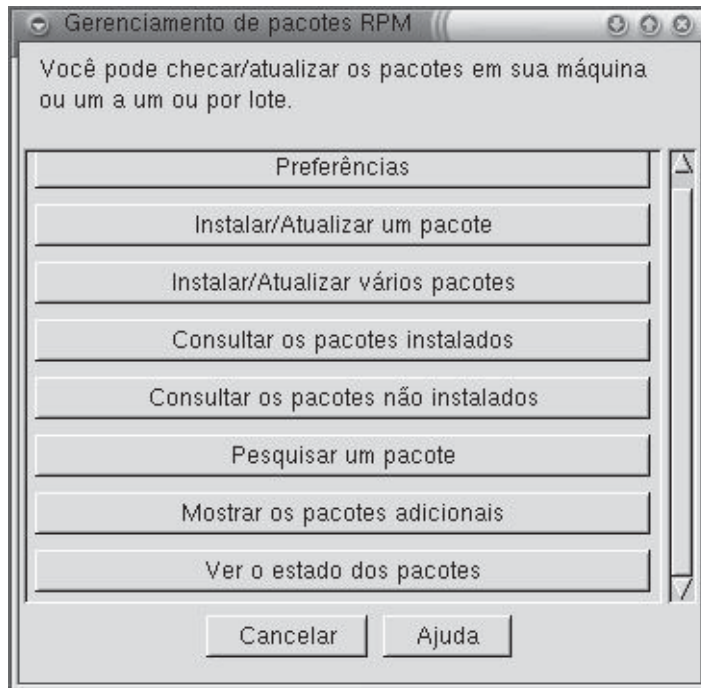
Será iniciado o aplicativo Linuxconf, muito usado para várias outras opções para quem se sentir mais confortável na interface gráfica. Através do Linuxconf é possível administrar vários tipos de servidores, regras para firewall, serviços, etc.

Neste momento, iremos utilizá-lo para gerenciar pacotes *rpm*.

Depois de aberto o Linuxconf, vá à guia *Controle/Gerenciamento de Pacotes RPM*. A partir daqui fica fácil administrar a instalação, atualização e remoção de pacotes; temos botões onde são indicadas atualizações de um ou vários pacotes, consulta de pacotes instalados



e não instalados. É muito importante que você coloque o caminho dos pacotes quando for instalar. Por exemplo, se estiver no `/root`, deverá indicar o caminho `/root`. Caso queira instalar através do CD, digite o caminho `/mnt/cdrom/conectiva/RPMS` (É onde ficam os pacotes da distribuição Conectiva), logicamente com o CD-ROM montado (veremos a respeito de montagem de sistemas no capítulo sobre o modo texto).



Mas mesmo através do Linuxconf, enfrentamos os problemas de dependências, pois caso falte um pacote, este deve ser instalado manualmente.

Para facilitar ainda mais a vida do usuário foram criados alguns aplicativos que verificam a dependência, buscam na internet os pacotes e atualizam automaticamente, de forma que o usuário final nem necessite tomar parte neste processo.

Instalação Automática - Apt-get

Uma das mais famosas é o *apt-get* (Advanced Package Tool).

O APT, inicialmente, foi feito pela Debian, uma outra distribuição do **Linux**. Porém, não dava suporte a pacotes *rpm*. Em pouco tempo,



foram adicionados em um projeto a parte, que anda paralelamente ao original APT da Debian, onde este dá suporte a pacotes RPM. Com isso, é possível instalar, atualizar, desinstalar, resolver dependências de forma transparente e tudo isso com a facilidade do uso da internet.

Isso mesmo!!! O *apt-get*, quando é solicitado a instalar um novo pacote através do modem ou rede, procura no repositório de FTP da distribuição. Localizando-o, verifica as dependências. Se houver necessidade, faz download não só do pacote escolhido para instalar, mas de suas dependências também. Para se ter uma idéia, é possível instalar o StarOffice sem problemas através do *apt-get*. Não tente via modem pois 247 Mb pelo telefone pode ser demorado, a não ser que você tenha um pouco de paciência.

Então, vamos ver como funciona.

O *apt* pode ser configurado para procurar os *rpm* no CD da distribuição ou na internet, nos próprios sites da mesma distribuição.

Primeiramente, precisamos saber se está instalado:

```
rpm -qa | grep apt-get <enter>
```

A resposta deve ser algo do tipo :

```
apt-0.3.19cnc50-2cl.
```

Esse é o pacote que precisamos que esteja instalado. Pode ser que apareçam outros pacotes, mas esse é o principal. Se não existir instale-o através do CD-ROM ou download no site da sua distribuição.

APT através do CD-ROM

Por padrão, o *apt-get* vem configurado para utilizar os FTP do site da distribuição. O arquivo que informa isso ao aplicativo é o *sources.list*, que se encontra em */etc/apt/*. Edite este arquivo e comente (coloque na frente da linha o sinal de suspenso - #) todas as linhas que possuem entradas referentes ao FTP. (Para aprender mais sobre como editar em modo texto, procure no capítulo referente ao assunto pelo editor VI).

Salve e saia do arquivo.

No Conectiva Linux 7.0, é necessário alterar outro arquivo o *apt.conf*, que também se encontra em */etc/apt*. Edite-o e acrescente a seguinte linha:

```
Acquire::CDROM::Copy "true";  
    apt::cdrom::thorough "true";    #- insira esta  
linha
```



Salve e feche o arquivo novamente.

E, por último, precisamos adicionar uma entrada nos arquivos do APT para o CD-ROM. Então, entre no console e digite o comando abaixo:

```
[root@localhost]# apt-cdrom add -d /mnt/cdrom  
<enter>
```

Agora, só é necessário usá-lo, por exemplo, para instalarmos um telnet-server. Digite:

```
[root@localhost]# apt-get install telnet-server <enter>
```

Automaticamente, ele procurará o arquivo no CD-ROM e irá instalá-lo. A única limitação é que se o arquivo ou as dependências não estiverem no CD ele não procurará na internet e, conseqüentemente, não instalará.

APT através de FTP Conectiva

Esta é a forma mais usual do *apt-get*, pois através da internet é possível instalar os últimos pacotes com bugs para correção do **Linux** e estarmos sempre atualizados com a nossa distribuição e os novos softwares que são inclusos no repositório.

Precisamos novamente editar o arquivo *sources.list* no diretório */etc/apt/*.

Comente todas as linhas e acrescente a seguinte:

```
rpm [cncbr] ftp://atualizacoes.conectiva.com.br 6.0/  
conectiva updates
```

Salve e saia do arquivo.

A linha que foi inserida, utiliza a “chave” **gpg** da Conectiva, que é uma chave de criptografia que garante que o pacote utilizado foi montado pela Conectiva.

É necessário que importemos a chave para a nossa instalação. Pode-se usar o CD-ROM, pois o mesmo contém esta chave. Caso não possua o cd da distribuição procure instruções em <http://distro.conectiva.com.br/seguranca/chave/> e instale a chave.

Para você que possui o CD da distribuição, proceda da seguinte forma:



Monte o primeiro CD da Conectiva :

```
[root@localhost]# mount /dev/cdrom /mnt/cdrom <enter>
```

Em seguida, instale a chave pública:

```
[root@localhost]# gpg --import /mnt/cdrom/RPM-GPG-KEY
<enter>
```

Pronto! Já podemos utilizá-lo. Antes, porém, precisamos efetuar um pequeno procedimento:

Se conecte a internet, abra um console e digite:

```
[root@localhost]# apt-get update <enter>
[root@localhost]# apt-get upgrade <enter>
```

O *apt-get update* vai atualizar e sincronizar sua lista de pacotes com a do site, enquanto o *apt-get upgrade* trará a lista dos pacotes que podem ser atualizados e os instalará, pois são mais recentes.

Caso deseje somente ver o que pode ser atualizado, digite:

```
[root@localhost]# apt-get -S upgrade <enter>
```

Agora, podemos instalar pacotes via internet, o comando é *install*. Veja no exemplo abaixo: estaremos instalando o MySQL, um banco de dados muito usado no mundo **Linux**.

```
[root@localhost]# apt-get -install MySQL <enter>
```

Para remover um determinado aplicativo use *apt-get remove nome_do_pacote*.

Ocorre, às vezes, do site de FTP da Conectiva estar um pouco sobrecarregado, o que retorna para o APT um erro, informando que não é possível estabelecer uma conexão e que se deve tentar mais tarde. Resolvi isso pegando o arquivo *sources.list* dos mirrors da Conectiva. Faça download do mesmo e salve em */etc/apt/*. Com isso, temos mais opções para atualizarmos o sistema. É possível fazer download do arquivo *source.list* em <http://distro.conectiva.com.br/apt/sources.list>. É necessário que você retire os comentários (excluindo os # em frente as linha) com referência aos vários FTP, para que funcione o aplicativo. Após o download, siga todo o procedimento descrito, incluindo a importação da chave-pública da Conectiva.

Instalação Automática no Modo Gráfico - Synaptic

Deixei para o final uma surpresa. Foi desenvolvido a pouco uma interface gráfica para o APT.

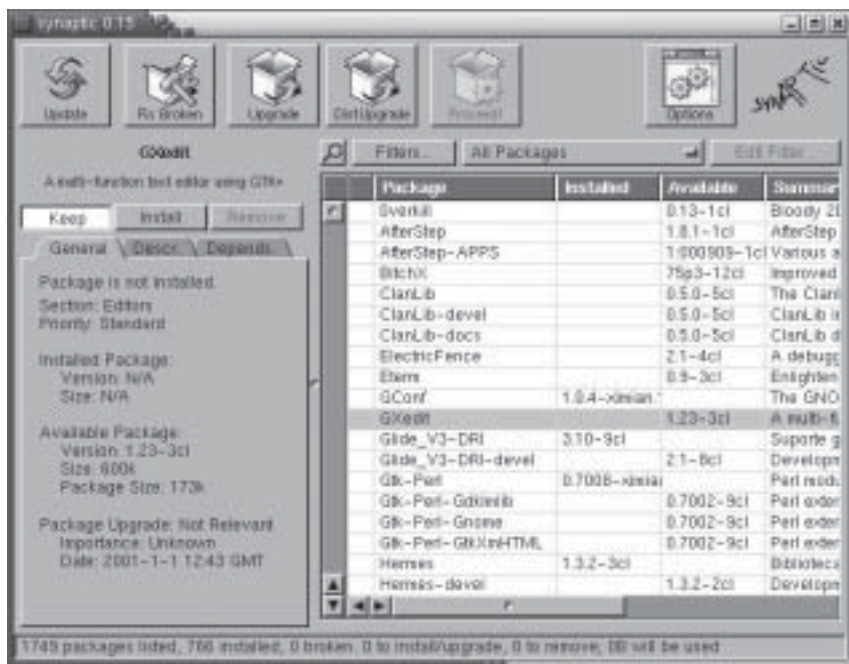


Chama-se Synaptic, que na versão 0.13 da Conectiva já vem no CD. É muito simples de usar: possui um painel do lado esquerdo sobre a descrição do pacote e, do lado direito, todos os pacotes instalados ou não. Alguns botões servem para instalar, remover, etc. Ou seja, não é necessário ficar digitando o comando no modo console para quem não gosta. Simplesmente, selecione o pacote a ser instalado, removido, etc.

Clique no botão referente à ação.

Devo lembrar que o Synaptic é apenas um frontend (Interface gráfica) para o APT, quem faz tudo é *apt-get*, por isso ele deve estar instalado e funcionando para que você possa aproveitar o Synaptic.

Dê uma olhada abaixo:



Interface do Synaptic

Red Carpet - Instalação Automática do Ximian

Para quem instalou o Ximian Gnome, pode-se usar também o Ximian Red Carpet. Um ótimo software, que além de poder instalar aplicativos da Red Hat (tome muito cuidado se sua distribuição não for a Red Hat, pois dependendo dos arquivos que forem atualizados podem danificar o sistema permanentemente), pode fazer instalações dos próprios



aplicativos Ximian, Wmware (máquina virtual), StarOffice e jogos da Loki games.

Para usá-lo, vá ao menu *Ximian/Programs/Internet/Red Carpet*.

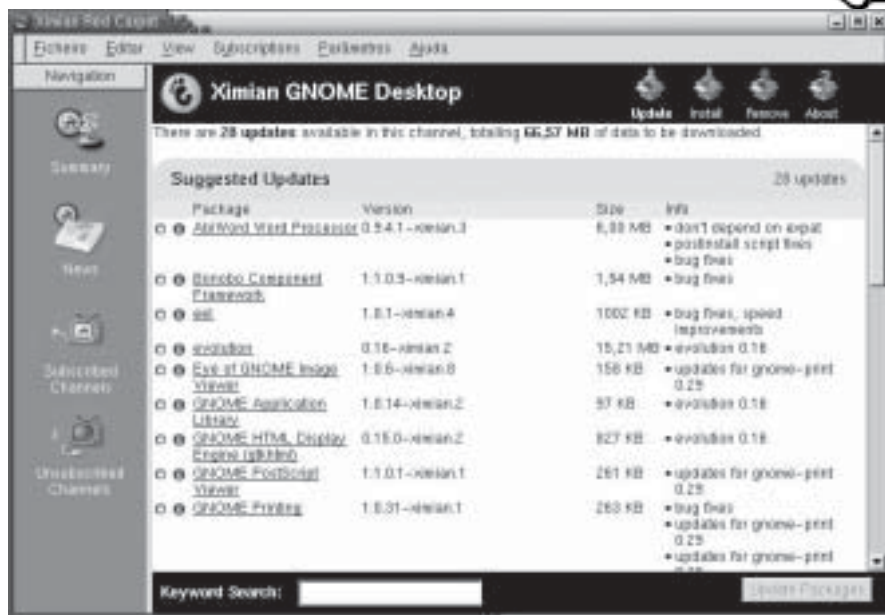
A interface será aberta. Procure do lado esquerdo o botão *Unsubscribed Channels*, como na figura abaixo. Serão mostrados os itens que você pode se inscrever.

Para inscrever-se, clique no próprio link do produto e, posteriomete, no canto direito inferior, aperte o botão *Subscribe* .

Agora, clique no botão *Subscribed Channel* para ver os canais que você está inscrito. A princípio, serão mostradas sugestões de updates (se você você já tiver algo instalado), caso contrário, você deve clicar no botão *Install*, que logo aparecerão os pacotes que podem ser instalados. Selecione os pacotes e clique no botão *Instalar*.



Nesta próxima figura, é possível ver os pacotes do Ximian:





5

Modo Texto

Introdução

Com certeza, este é um dos capítulos mais interessante do livro. Mesmo que o usuário não goste do modo texto e prefira o modo gráfico, recomendo que leia todo este capítulo. Quando ver o poder do que pode ser feito no modo texto, não deixará de usá-lo.

Para começar, devemos saber o que é o console. Precisamos escolher um tipo de console.

O console nada mais é que um interpretador de comandos. O mais usual é o BASH (Bourne Again Shell), conhecido como *Shell*. Para satisfazer um usuário, além do Shell, é necessário uma quantidade de programas para console que satisfaça todas as necessidades do usuário, tais como editar textos, ler e enviar e-mail, tocar música, jogar, calcular, navegar na internet, etc. Tudo isso é possível ser feito no modo texto.

Normalmente, quando estamos no modo gráfico, utilizamos o *xterm*, *wterm*, *eterm*, entre outros. Estes aplicativos simulam um console. De certa forma, não há nenhum problema em utilizá-lo, mas seria mais aproveitável se estivessemos em modo texto, pois poderíamos ter vários consoles. Por exemplo, em um você está criando um script, em outro vendo logs e, num terceiro, arquivos de configuração, alternando-os através das teclas *Alt + F1* para o primeiro, *Alt+ F2* para o segundo, e assim, sucessivamente.

Vou separar em vários subcapítulos para que o leitor identifique em futuras dúvidas, de modo mais rápido, onde encontram-se as respostas.



Vim - Um Editor de Texto

O VIM, também conhecido como VI (lê-se *vi ai*), é um dos editores do **Linux**. Existem muitos outros, como Emacs, Tex, etc.

Existe, inclusive, uma disputa entre o VI e o Emacs. Muitos dizem que o Emacs é melhor, outros que o VI. Particularmente, uso o VI, mas não testei o Emacs, o que me deixa na situação de expectador. Como não estou familiarizado com o Emacs, vamos falar do VI. Desculpem-me os Emacnianos.

O VI é um poderoso editor de texto, capaz de ter todas as funções de um gráfico, como copiar, colar, recortar, replace. Ele também pode ser usado como ferramenta de programação para linguagem C com próprio Shell Script, pois tem a qualidade de estar alterando cores conforme os comandos dados. Por exemplo, uma linha comentada sempre fica em azul claro, uma variável do tipo Shell fica em azul escuro, e assim por diante.

Para quem está acostumado somente no modo gráfico, ou até com o *edit* do DOS, vai estranhar bastante. Mas com um pouco de treinamento, logo estará familiarizado com o VI.

A sintaxe para iniciá-lo é bem simples:

```
vi nome_do_arquivo
```

Existindo o arquivo, ele abrirá o mesmo. Caso não exista, ele criará um novo.

Para movermos, inicialmente usamos a teclas de seta, ou se preferir, pode usar as teclas **h**, **l**, **j** e **k**, que representam esquerda, direita, abaixo e acima, respectivamente. Talvez a ilustração abaixo explique melhor:

```
      ^  
  
      k  
  
    < h      l >  
  
      j  
  
      v
```



O VI trabalha praticamente de duas formas: no modo *normal* e no modo *comando*.

O modo normal é esse em que estamos ao entrar no VI, onde algumas teclas informam ao VI o que deve ser feito. Por exemplo, quando queremos inserir algum texto, digitamos a letra *i*, então, o VI entrará em modo de edição. Para voltarmos ao modo normal, a tecla *Esc* é a responsável. Se quisermos apagar uma letra, podemos editar novamente com a letra *i* e usar o *Del* normalmente. Ou ainda, no modo normal, vamos até o caractere usando as setas, posicionado o cursor em cima da tecla que desejamos deletar e teclamos *x*. Quando teclamos *x*, o VI entende que ele deve deletar o caractere que estiver sobre o cursor.

O modo comando é usado para salvarmos e sairmos do VI. Estando no modo normal (para certificar-se tecle *Esc*), digite “:”. Esta letra fará entrar em modo comando, onde aparecerá um prompt na parte inferior esquerda do console. Digitamos a letra *a* qual queremos utilizar e teclamos *Enter*. Por exemplo, para sairmos, digite:

```
: q <enter>
```

Ela saíra do arquivo caso você não tenha alterado nada, pois ao contrário, ele informará que o arquivo não foi salvo. Para salvá-lo, digite:

```
:w <enter>
```

Agora é possível sair do arquivo com **:q**. Mas, por que não fazemos tudo de uma vez? Digite, então:

```
:wq <enter>
```

Através desse comando, o arquivo será salvo e você sairá dele automaticamente.

Mas, se você deseja sair do arquivo sem salvar as alterações feitas, digite:

```
: q! <enter>
```

Estes são quase todos os comandos do modo comando. O normal tem muito mais opções.

Vamos iniciar pela decepção.

Para deletar uma letra, mantenha o cursor sobre ela e aperte *x* ou *Del*.



Para debilitarmos uma palavra inteira, colocamos o cursor em frente à palavra e digitamos *dw*. Se vamos deletar a linha inteira, digite *d\$*. Não é necessário digitar *Enter*.

A sintaxe do delete é a seguinte:

[número] d objeto

Ou:

d [número] objeto

Onde:



d: é o comando deletar.



número: indica quantas vezes será executado o comando (opcional, default=1).



objeto: é a forma como ele vai deletar; *w* para o final da palavra incluindo espaço; *e* para o final da palavra não incluindo espaço; *\$* para a linha inteira.

Pensado na frequência em que deletamos linhas, os projetistas colocaram mais uma forma de excluí-las, que é digitando *dd*. Com certeza, torna-se mais fácil digitar *dd* do que *d\$*.

Este número que consta na sintaxe é usado quando queremos deletar mais de uma linha. É comum este tipo de sintaxe com os comandos do modo normal. Vamos deletar três linhas diretas a partir de onde o cursos se encontram: *3dd* ou *dd3*. E se quisermos deletar somente a metade da linha? Devemos posicionar o cursor em frente ao texto a ser deletado e digitar **D**. Assim, será excluído o texto da linha localizado somente a partir do cursor. No caso anterior, indiferente do posicionamento na linha, esta era completamente deletada.

Na verdade, se compararmos com os editores mais usuais, quando deletamos estamos recortando, ou seja, o texto é apagado e gravado no buffer do VI, podendo, ainda, ser reutilizado.

Temos, também, outra forma de deletar, que é selecionando o que deve ser excluído e teclando *del* ou *d*. Para selecionar, digite *v* e mova o cursor até onde desejar. Depois disso, aperte *del* ou *d* para deletar.

O delete é usado também no esquema copiar/colar. Selecionando ou digitando *d*, você está deletando, mas gravando no buffer, podendo colá-lo com a letra *p*.



A letra *p* (put) é usada para colar o que está no buffer no VI. Para colar a palavra anteriormente deletada, coloque o cursor no local onde deseja colar e tecle *p*. Caso queira colar a palavra mais de uma vez no mesmo local, digite *2p* para colar duas vezes; mas se deseja colar em outro local, mova o cursor até o mesmo.

Falando nisso, existem formas mais rápidas para nos movermos.

Para irmos para uma linha específica, usamos *nG*, onde *n* é o número da linha que queremos ir e *G* o comando que nos levará até lá. Por exemplo, se queremos ir para a linha 110, basta digitar *110G* ou podemos ir através do modo comando e digitarmos o número da linha para onde desejamos mover o cursor, ou seja, digite *:110*, que o VI te levará até a linha 110.

Podemos usar o *Page Up* e o *Page Down* para nos movermos de página em página e as teclas *Ctrl + b* e *Ctrl + f*, que vão para as telas anterior e posterior, respectivamente. Use *w* para ir para a próxima palavra e *b* para voltar a palavra anterior. O *0* (zero) move o cursor para a primeira letra da linha e o *\$* move para a última, mas nada impede de usarmos as teclas *Home* e *End*. Se quisermos ir para primeira linha da tela, digitamos *H*; para a última, *L*; para movermos o cursor para o meio da tela, usamos *M*. Ufa! Muita coisa, né? Não, ainda tem mais... Mas não se preocupe, pois com o tempo você vai preferir usar o VI e as interfaces gráficas.

Com isso, já podemos nos mover no texto sem problemas.

Para copiarmos algo, utilizamos o *y*. Por default, ele copia a linha inteira, como se usado com *\$*. Lembra *d\$* ou *dd* para deletar uma linha. Pois é !!! Podemos usar *yw* para copiarmos uma palavra, somente *y* para copiar a linha inteira e o *ny* para que várias linhas sejam copiadas. Para copiarmos quatro linhas, digitariamos *4y* ou *y4*.

Ainda não falei de um comando muito importante, o *Desfazer*? Muito simples. Tecle a letra *u*, que automaticamente será desfeita a última ação.

Para acharmos uma palavra, como o *search* ou *find* de alguns programas, temos um comando para isso: */ palavra*. Quando for digitado */* (barra), entrará em modo de comando, aparecendo */* no canto esquerdo inferior, como quando queremos gravar ou salvar. Para continuarmos a busca para no texto, digite *n* e para procurar nas páginas anteriores, digite *Shift + n*.



Uma função muito utilizada em editores é a troca automática de palavras. Por exemplo, substituir *copilar* por *compilar*. A sintaxe ficaria assim:

```
: s/palavra_antiga/palavra_nova
```

Dessa forma, o nosso comando ficaria da seguinte forma `:/s/copilar/compilar`.

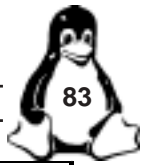
Se for usado dessa maneira, ele alterará somente a primeira ocorrência desta palavra. Para estendermos pela linha inteira, colocamos a letra *g* no final, assim, ficaria `:/s/copilar/compilar/g`. Agora ele verificará todas as ocorrências na linha atual. Enfim, para alterar o arquivo inteiro, trocamos a “/” inicial por “%”, ficando da seguinte forma: `%s/copilar/compilar/g`, ainda podemos especificar a linha que queremos alterar, se fôssemos alterar somente nas linhas 5 e 15, digitariamos: `5,20s/copilar/compilar/g`.

Esta vai para programadores: Quando estamos programando, usamos muitos `()`, `{}` e `[]`. Às vezes, pode ficar difícil identificar onde começa um parêntese e onde fecha. Mas com o VI isso ficou fácil. Não precisamos pedir para ele localizar todos os parentes, simplesmente colocamos o cursor sobre o parêntese, colchete ou chave e teclamos `%`. Com isso, iremos para o `(`, `[` ou `{` que casam com ele, ou seja, que o fecha. Digitando `%` novamente, voltaremos para o anterior).

Muito interessante para debugar códigos de programas.

Outra função muito interessante do VI é podermos editar vários arquivos simultaneamente. É possível abrir dois ou mais arquivos, que são divididos na tela. Vamos supor que temos o *arquivo1* e o *arquivo2* no diretório */root*. Abrimos um com VI `/root/arquivo1 <enter>`. Agora, para abrímos o outro, digitamos `:new /root/arquivo12` e a janela será dividida em duas. Para alternar entre uma e outra, usamos o conjunto de teclas `Ctrl + w + p` ou `Ctrl + w + k`. Fechamos da forma normal `:q` ou vamos à janela que queremos que continue aberta e digitamos `:only <enter>`.

Não estenderei mais este assunto, o VI. O que já foi dito é suficiente para que possamos usar o VI sem muitas complicações. Abaixo, deixarei uma pequena tabela com o que foi visto e mais alguns comandos. Para saber mais, vá em www.dominiolinux.com.br.



Modo	Entra em Modo de Edição
i	Insere texto antes do cursor.
r	Insere texto no início da linha.
a	Insere texto depois do cursor.
A	Insere texto no fim da linha.
o	Adiciona linha abaixo da linha corrente.
O	Adiciona linha acima da linha corrente.
Ctrl+h	Apaga o último caractere.
Ctrl+w	Apaga última palavra minúscula.
Esc	Volta para o modo de comando.
rx	Substitui o caractere sob o cursor pelo x (replace).
Modo	Não Entra em Modo Edição
Rtexto	Substitui o texto corrente pelo texto indicado.
cw	Substitui a palavra que está sob cursor (change).
c\$	Substitui a linha que esta sob o cursor.
cc	Substitui a linha que está sob o cursor.
C	Substitui a linha a partir do cursor.
Ctrl + f	Passa para a tela seguinte.
Ctrl + b	Passa para a tela anterior.
h	Move o cursor para a esquerda.
j	Move o cursor para baixo.
k	Move o cursor para cima.
l	Move o cursor para direita.
H	Move o cursor para a primeira linha da tela.
M	Move o cursor para o meio da tela.
L	Move o cursor para a última linha da tela.
w	Move o cursor para o início da próxima palavra.
b	Move o cursor para início da palavra anterior.
0	Move o cursor para início da linha.
\$	Move o cursor para o fim da linha.
nG	Move o cursor para a linha n.
G	Move para a última linha do arquivo.
x	Deleta o caractere que está sob o cursor.
dw	Deleta a palavra que está sob o cursor.
Modo	Não Entra em Modo Edição
dd	Deleta uma linha.
d\$	Deleta uma linha.
D	Deleta uma linha a partir do cursor.
u	Desfaz a última ação.
U	Desfaz todas ações feitas na linha corrente.
J	Une a linha corrente à próxima.
/palavra	Procura pela palavra.
n	Procura pela próxima palavra.
Shit + n	Procura pela palavra anterior.
Ctrl + g	Mostra o nome do arquivo, número de linhas e número de
%	Sobre [, (e { procura seu fechamento ,) e } vice-versa.
v	Entra em modo seleção.



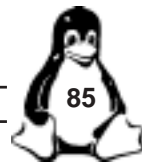
Modo	Comando
:q	Sair do arquivo.
:q!	Sair do arquivo sem salvar alterações.
:wq	Salva e sai do arquivo.
:w	Salva arquivo.
:n	Vai para a linha n.
:s/velho/novo	Muda a palavra novo por velho na primeira ocorrência. Se usado com /g após o /novo, faz alterações em uma linha inteira e, se usar %s em vez de /s faz alterações no arquivo inteiro.
:set number	Numera todas as linhas do arquivo.
Modo	Comando
:set number	Retira a opção de numerar linhas.
:set ic	Ignora cases (maiúscula/minúscula).
:set noic	Retira a opção case.
:help	Abre o arquivo de ajuda do VIM.
!:comando	Executa comando externo. Por exemplo, !ls listará o diretório de atualização; :r arquivo insere o conteúdo do arquivo dentro do arquivo atual a partir do cursor.

Estes são somente alguns dos comandos mais usuais. Qualquer dúvida pode-se usar o help online do VIM, digitando *F1* ou *:help*. Por exemplo, na opção *set*, que é usada para ativar e desativar opções do VI, digite *:help set* para que seja mostrada uma explicação do *set*. Agora, para ver todas as opções do *set* (que são muitas), digite *:set all*. Para saber algo específico, como no caso do *set number*, você terá que digitar *:help number*. Um arquivo bastante completo, na versão 5.8.4, possui um help com 1298 linhas, ou seja, tem realmente bastante coisa. Para sair do help, usa-se o modo de comando, *:q*.

Comandos

Neste momento, falaremos de muitos outros comandos que não foram citados no capítulo 2, onde foi feita uma comparação com os comandos do DOS, da Microsoft. Também falaremos de alguns arquivos de configuração que facilita muito a vida do usuário.

Sabemos que para listar um diretório, podemos usar o *ls*, mas podemos usá-lo, ainda, com outras opções, como *-a*, *-l*, *-M*, onde *a* mostra os arquivos ocultos; *l*, como já foi dito, lista o diretório com seus donos, grupos, permissões; e o *M* mostra o tamanho de arquivo em Mb. Os arquivos ocultos são identificados no **Linux** por um “.” (ponto). Por exemplo, se o leitor possui o Netscape instalado, provavelmente terá



um diretório chamado *.netscape*. Para acessá-lo, é usado o comando normal (*cd*), mas não podemos esquecer do ponto antes do nome, ou seja, *cd .netscape*. Há muitas outras opções do comando, mas aqui só falaremos do mais importante.

Todos já sabem usar o comando *cd*, mas quem utiliza o DOS está acostumado a digitar *cd..* (onde *cd* e *pontos* estão juntos). Lembre-se que desta maneira o **Linux** não executa. Por isso, temos que digitar *cd ..* (*cd*, *espaço*, *ponto* e *ponto*). Mas temos uma forma de alterar isso para que o **Linux** aceite o *cd* com ou sem espaço.

Existe um arquivo em */etc* chamado *bashrc*. Vamos editá-lo:

```
vi /etc/bashrc
```

Neste arquivo, constam várias linhas iniciadas por *alias*. Esses aliases são apenas nomes, ou melhor dizendo, atalhos para os comandos verdadeiros. Então, vamos adicionar mais uma linha no final do arquivo. Digite *i* para entrar em modo de inserção e adicione a seguinte linha:

```
alias cd.. = "cd .."
```

Com isso, sempre que digitarmos *cd..* (sem espaço), ele buscará o *cd..* (com espaço), que é o comando que o **Linux** entende. Se você usar comandos muito grandes, é possível fazer este tipo de atalho para agilizar quando for necessário executar tal comando.

O *pwd*, como já foi dito, serve para dizer em que diretório estamos. Da mesma forma, *mkdir* e *rmdir* são utilizados para criar e remover diretórios.

O comando *rm*, usado para deletar arquivos, pode ser muito interessante. Quando vamos deletar arquivos, ele pergunta se realmente desejamos removê-los. Para que a pergunta não seja feita, use *rm -f* e para que os arquivos e diretórios sejam deletados, use *rm -rf*.

O *cat*, é um comando muito usado no **Linux**. Ele mostra o conteúdo do arquivo em binário ou texto. Por exemplo, naquele caso do arquivo */etc/bashrc*, se quisermos apenas ver o seu conteúdo, digitaríamos:

```
cat /etc/bashrc
```

Ele é muito usado como entrada para outros comandos, ou seja, você executa um comando, a saída ou o resultado do mesmo, e vai direto para outro através do símbolo “|” (pipo). Normalmente,



chamamos esse tipo de comando de *pepilines*, onde veremos mais a frente como funciona.

Uma pergunta que deve estar na mente do leitor desde o começo é como usar o CD-ROM e disquete. Para isso, temos um comando, o *mount*. No Windows, isso é feito automaticamente quando acessamos o driver de CD ou disquete, assim como em outras interfaces gráficas. No caso do Ximian Gnome, quando montamos um CD-ROM, ele cria um ícone com o nome do CD-ROM. Para desmontar o driver, precisamos jogar o CD-ROM ou o floppy na lixeira. Mas no modo texto, não temos ícones. Dessa forma, é necessário montar e desmontar manualmente os drivers. A sintaxe é a seguinte:

```
mount /local_de_origem /ponto_de_montagem -t filesystem
```

Onde *local de origem* é o driver onde contém as informações como:

```
floppy = /dev/fd0  
cdrom = /dev/cdrom  
hd local = /dev/hd (hda, hda1, hdb, hdc, etc.)  
hd remoto = endereço_ip:/caminho/interno
```

Ponto de montagem é o local onde os arquivos estarão disponíveis localmente. Por padrão, existe um diretório chamado *mnt* na raiz do **Linux**. Já existem pontos de montagem para o CD-ROM e floppy. Para montar o CD-ROM, usaríamos a sintaxe abaixo:

```
mount /dev/cdrom /mnt/cdrom -t iso9660, o iso9660 é  
opcional
```

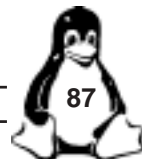
Se fossemos montar um disquete para leitura formatado no Windows, usaríamos a seguinte sintaxe:

```
mount /dev/fd0 /mnt/floppy -t vfat
```

Devemos, ainda, esclarecer o último item que é o *file system*. O *file system*, como o nome sugere, informa ao comando *mount* qual tipo de sistema de arquivos são usados na montagem. Para o CD-ROM, temos *iso9660*; para Windows, o *vfat*; para o **Linux**, temos *ext*, *ext2*, *ext3*; para arquivos de rede (*network file system*), usamos o *nfs*. No caso do CD-ROM, pode ser omitido este parâmetro.

Depois de montando, é só acessar o ponto de montagem como */mnt/cdrom* ou */mnt/floppy* e usar dados normalmente.

É possível montar uma partição Windows no **Linux** sem nenhum problema. Vamos imaginar que seu HD tenha três partições: uma Windows */dev/hda1* e outras duas **Linux** e *swap*. Para montar a



partição Windows dentro de um diretório, a primeira coisa a ser feita é criarmos o diretório onde ficarão disponíveis os dados montados. Como já temos */mnt/*, onde já utilizamos para montar sistemas, criemos dentro deste diretório um novo diretório chamado *Win*:

```
cd /mnt/  
mkdir Win
```

Agora, já podemos montá-lo para uso:

```
/mount /dev/hda1 /mnt/Win -t fat  
mount /dev/hda1 /mnt/windows -t fat
```

É muito importante não esquecer o tipo de arquivo no caso do Windows *vfat*.

Mas sempre que eu quiser acessar meus arquivos do Windows tenho que fazer tudo isso?

Não. Existe um arquivo que monta os volumes automaticamente quando a máquina é ligada.

Edite o arquivo */mnt/fstab*, nele existem seis colunas.

Colocaremos o *device* (*/dev/hda1*), o ponto de montagem, o tipo de arquivo, se será montado automaticamente ou não. Ficará dessa forma:

```
/dev/hda1 /mnt/windows vfat defaults 0 0
```

Para finalizar, salve o arquivo. Na próxima vez que reiniciar o micro, será montada automaticamente a partição do Windows.

Em outras ocasiões, precisaremos montar arquivos *iso*. É isso mesmo, com extensão *iso*, que seria a imagem de um CD-ROM. Muitas versões do **Linux** disponibilizadas para download vêm no formato *iso*.

Crie o diretório onde ficará montado o arquivo *iso*. Por exemplo, */mnt/iso*. Vá ao diretório onde se encontra o arquivo *o*. Mostre o local onde ele se encontra e use *-o*. Veja abaixo:

```
mount -o loop /root/cl60-1.iso /mnt/iso
```

É muito útil. O Open Kylix (a versão “free” da Borland para o Kylix) vem neste formato. Montamos a imagem, instalamos, e a desmontamos. Podemos até deletar o arquivo ou gravar em um CD-ROM.

Agora, já estamos craques em montagens. Vamos partir para outros comandos.



Quando queremos localizar um arquivo, usamos o *find*, sua sintaxe é a seguinte:

```
find [diretório] [opções / expressões]
```

Entre as opções, temos *-name /*, expressão que procura por um nome e que processa os subdiretórios antes de processar o diretório principal. Vamos procurar a palavra *conectiva* no diretório */etc*.

```
find /etc -name "conectiva"
```

O diretório acima procurará exatamente a palavra *conectiva*. Acho um pouco limitado, pois se queremos localizar uma palavra que contenha *conectiva* no meio, ele não achará. Neste caso, usamos o *pepiline*, que lembra a saída de um comando ligado para outro.

```
find /etc/ | grep conectiva
```

Neste outro acima, o *find* gera uma lista dos arquivos em */etc* e o *grep* procura na saída a palavra *conectiva*. Com isso, acharemos pelo menos um arquivo, o *versão-conectiva*.

Já que citamos o *grep*, vamos verificar sua utilidade. O *grep* será usado para procurar um texto dentre os arquivos ou dispositivos de entrada padrão. Para usá-lo sozinho, especifique a palavra e o arquivo onde deseja procurá-la. A saída deste comando será mostrar as linhas que contêm esta palavra.

```
grep "palavra procurada" texto
```

Quando estamos procurando mais de uma palavra no texto, devemos colocá-las entre aspas. Ainda há várias opções interessantes, como *-A*, que mostra o número de linhas encontradas; *-i* (ignore-case), que ignora a diferença entre maiúsculas e minúsculas. Pode-se, ainda, usar o comando *zgrep* para procurar diretamente em arquivos compactados do tipo *gzip*.

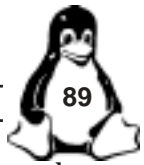
É muito usado em conjunto com outros comandos, como no caso do *find*.

Para visualizar, temos também o *head* e o *tail*.

O *head*, usado para visualizar o começo de um arquivo por default, verifica as dez primeiras linhas. Para alterar, use a opção *-n*.

```
head -n 20 arquivo.txt
```

Já o *tail*, ao contrário do *head*, lê o final do arquivo. Muito útil para verificação de *log*. Por exemplo, instalamos o *squid*, um servidor proxy,



mas queremos ver o que as pessoas estão tentando acessar ou acessando. Digitamos:

```
tail -f /var/log/squid/http.access
```

Com isso, ele mostrará as dez últimas linhas do arquivo e quando houver um nova inserção, ele atualizará a tela automaticamente. A opção *-f* traz esse atributo. Para sairmos, temos que digitar o *Crtl + C*.

Às vezes, a tela está um pouco suja de caracteres, então, limpamos com o *clear*. Caso esteja acostumado com o *cls*, edite uma entrada no arquivo */etc/bashrc* novamente e coloque o seguinte:

```
alias cls = "clear"
```

Feito isso, as alterações estarão habilitadas no próximo login.

Mas mesmo quando queremos visualizar um arquivo muito grande, não podemos usar somente o *cat*, precisamos de um auxiliar. Podemos utilizar diretamente o *more* ou *less* (como no DOS), ou em conjunto. Como no caso abaixo:

```
cat arquivo.txt | more
more arquivo.txt
```

O *less* aceita *Page Down* e *Page Up* para rolar as telas, já com *more* temos que utilizar o *Enter*.






É interessante sabermos, agora, o tamanho do nosso diretório, partição, etc.

Usamos o comando *df* para verificar o tamanho da partição ou partições. Logicamente, as partições devem estar montadas. Usando a opção *-h* será mostrado a quantidade de espaço livre e ocupado em Kb, Mb e Gb. A opção *-k* mostra em Kb e *-m* em Mb.

```
df -h /dev/hda - para verificar todo hd.
```

E os arquivos e diretórios?

Para listar o tamanho dos arquivos e diretórios, temos o *du*. Informamos qual o diretório e a opção que queremos verificar. Temos:





-  **-a:** Verifica ocultos;
-  **-c:** Faz uma soma total de todo o espaço ocupado;
-  **-D:** Não conta links simbólicos;
-  **-h:** Como o *df*, retorna os valores em Kb, Mb e Gb;
-  **-S:** Não calcula o espaço ocupado por subdiretórios.



```
du -hacD /root
```

Neste caso acima, estamos verificando o tamanho do diretório */root* na forma (Kb, Mb e Gb), contando arquivos ocultos - exceto links simbólicos - e totalizando no final.




Um comando muito importante no mundo **Linux** é o *ps*, que exhibe os status dos processos que estão rodando na máquina. Os comandos abaixo verificam:

-  **-A:** Todos os processos que estão rodando;
-  **-a:** Os processos utilizando *tty* (bash);
-  **-C:** O nome do processo;
-  **-U:** O userID do usuário.

Realmente, tem muitas opções. Dê uma olhada no *help*.

Digite *ps —help*.

Vamos a alguns exemplos:

-  **ps -A:** Mostra todos os processos.
-  **ps -C:** Netscape. Mostra os processo que estão rodando o Netscape.
-  **ps -AU root:** Mostra todos os processos que estão rodando, sendo o root o dono.

Pode ser utilizado com outros comandos, por exemplo:

-  **ps -ax | grep netscape:** Mostrará todos os processos com o Netscape.

Na listagem, está o PID, o número do processo, o *tty*, o tempo e onde está o executável.

Utilizamos o comando *kill* para matar o processo.

Podemos usar a sintaxe *kill -n* PID, onde:

- n = 1 (HUP) = Reiniciar;
- n = 15 (TERM) = Finalizar;
- n = 9 (KILL) = Matar.

Ou seja, através do *ps*, descobrimos o PID e, com o *kill*, matamos ou reiniciamos os processos.

É interessante utilizá-lo quando algum aplicativo trava para reiniciá-lo ou finalizá-lo.



Darei um exemplo de como usar o *killall -n* ou o *Status nome_do_processo* para ficar mais claro:


```
killall -kill netscape
```

Ou :

```
killall -9 netscape
```

Ambos fazem a mesma coisa: matam o processo. Dessa forma, você utilizará o nome (HUP, TERM ou KILL) e seu respectivo número 1, 15 ou 9.

Você também pode querer saber quantos processos estão sendo rodados por um aplicativo. O comando *pidof* retorna o número de processos e o PID de cada processo que está rodando:

 *pidof httpd*: retornará os PID que estão rodando o Apache.

Com isso, já é possível administrar os processos que estão rodando na máquina, mas tome cuidado, pois matando processos errados, sua sessão, por exemplo, pode ser fechada ou algo pára de funcionar corretamente.

Temos mais alguns comandos de sistema que podem ser muito úteis.

Por exemplo, o comando *who*, verifica quem está conectado na máquina. Em caso de invasões será possível ver quem está conectado à máquina. Ou até no trabalho, para verificar a utilização da máquina, compartilhando recursos de rede.

O comando *free*, mostra detalhes sobre a utilização de memória *ram* e *swap*. Utilize-a com a opção *-m*, que mostra a quantidade em Mb.

```
free -m
```

Outro recurso interessante é o comando *su*. Permite que o usuário troque de identidade sem precisar fazer *logoff* e *logon*. É usado quando precisamos fazer algo com o usuário *root*, como alterar permissões, mas estamos logados com um usuário normal. Digitamos:

```
su root
password:
```

Após digitarmos *su root*, para mudarmos para o usuário *root*, ele pedirá a senha. Para retornar, use o *su* novamente.



Existe o comando *time*, que verifica quanto demora para ser executado um comando, ou seja, ele conta o tempo desde o início até a finalização do comando.

```
time ls -l
```

Útil para testarmos novos aplicativos e scripts.

Já o comando *uptime*, mostra o tempo de execução do sistema desde que o computador foi ligado.

Como em qualquer outro sistema, a data e hora são muito importantes.


Para vermos a hora, usamos o comando *date*.


```
Qui Mar 23 02:46:30 BRT 2000
```

Para alterá-lo usamos o formato *MMDDhhmmAA*, onde:

 *MM* = mês;

 *DD* = dia;

 *hh* = hora;

 *mm* = minuto;

 *AA* = ano.

Veja um exemplo de como alterar a data para 20 de janeiro de 2001, às 14 horas e 20 minutos:

```
date 0120142001 <enter>
```

É necessário, então, sincronizar o relógio da BIOS. Digite *clock -w <enter>* que ele sincronizará.

Bom, eu tenho um pouco de dificuldade para acertar a hora, nunca me lembro do formato, por isso criei um script que deixa mais fácil essa tarefa. Veja abaixo as instruções:

Crie um arquivo em */usr/bin*. No meu caso, chamei-o de relógio.

```
vi /usr/bin/relogio
```

Adicione estas linhas:

```
#!/bin/bash
# $1 = dia $2 = mês $3 = ano $4 = hora e 45
=minuto
```

Isto é apenas um comentário:

```
date $2 $1 $4 $5 $3
```



```
clock -w  
echo "Operação realizada"
```

Salve e saia do arquivo. Digite :

```
chmod 755 /usr/bn/relogio
```

Agora, você já pode alterar uma data. Veja:

```
relogio 20 01 01 14 20 <enter>
```

Colocamos no formato *dia/mes/ano/hora/minuto* e o script altera a ordem e grava corretamente.

O comando *echo*, da mesma forma que o DOS, serve para exibir mensagens. Como no caso anterior, pode ser usado na criação de scripts para mostrar mensagens na tela enquanto acompanhamos a execução.

Outro comando interessante, é o *cal*, de calendário. Ele exibe o calendário do mês por default.

MARÇO 2000						
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	

Mas, caso você queira ver o calendário do ano inteiro ou de outro ano, é só digitar:

```
cal ano - ex: cal 1975 , cal 2001
```

Comandos para Administração de Usuários e Grupos

Agora, precisamos dar uma olhada na administração de usuários. Esta tarefa pode ser feita no modo visual através do Linuxconf (veja no capítulo referente à ele).

Veremos, agora, como se dá a criação e remoção de usuário e grupos no modo texto.



O comando *adduser* é usado para criar usuário. Automaticamente, cria o diretório do usuário, o subdiretório de e-mail, adiciona um novo grupo para este usuário. Todo o trabalho que o *adduser* faz poderia ser feito manualmente:

```
adduser Nicole
```

Adicionamos um usuário chamado *Nicole*. Agora, precisamos dar uma senha para ele. Para isso, temos o comando *passwd*:

```
passwd Nicole
```

```
Enter new Unix password:
```

```
Enter new Unix password:
```

Em resposta, ele pedirá para você digitar a senha, e redigitá-la.

O arquivo que contém os usuários é o */etc/passwd*. As linhas têm entrada do tipo:

```
root:x:0:0:root:/root:/bin/bash
```

Neste arquivo, os campos são separados por : (dois-pontos). Partindo da esquerda para direita, temos o nome do usuário, a senha ou token de senha (informando que a mesma está criptografada em outro arquivo, normalmente no */etc/shadow*), *userid*, *groupid*, nome verdadeiro ou comentário, diretório do usuário e se é usuário do Shell. Caso tire esta entrada, o usuário não poderá logar. É útil para criação de usuário somente para e-mail.

Quando for adicionar manualmente um usuário, insira com *Id* acima de 500, pois até 500 são usuários utilizados pelo sistema.

A remoção pode ser feita manual ou graficamente. Deve-se retirar a entrada no arquivo */etc/passwd* e remover o arquivo do usuário *rm -r /home/Nicole*.

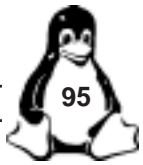
Para adicionarmos um grupo no modo texto, será manualmente, editando o arquivo */etc/group* (no modo gráfico use o *Linuxconf*). A entrada desse arquivo é similar a esta:

```
root:0
```

```
administradores:admix5674:509:0,500,501,502,509
```

As entradas também são divididas com : (dois-pontos).

Da esquerda para direita, temos o nome do grupo, senha, *id* do grupo e usuários do grupo.



No primeiro caso do root, temos a entrada de nome e *id* do grupo. No segundo, temos todas as entradas, desde o nome, até os usuários de *id* 0, 500, 501, 502 e 509.

Permissões de Arquivos e Diretórios

As permissões em arquivos e diretórios são dadas através dos comandos *chmod* e *chown*.

O *chmod*, como foi dito anteriormente, altera o tipo de permissão do arquivo, como escrita, leitura e execução para o dono, grupo e demais usuários.

Já o *chown*, altera o dono e o grupo do arquivo ou diretório.

Por exemplo, a permissão do arquivo abaixo está como o *root* e o grupo *root* sendo donos.

```
-rwxr--r-- 1 root root 8325   Abr 23 01:15 arquivo.txt
```

Vamos alterar o dono. O novo dono será a usuária *Nicole*.

```
chown Nicole.Nicole arquivo.txt
```

O primeiro *Nicole* diz respeito ao dono e o segundo respeito ao grupo, que é dono do arquivo.

Então, por que não darmos permissão para o grupo administradores? Simples:

```
chown .administradores arquivo.txt
```

Note que há um . (ponto) antes de *amestradores*. Este ponto é referente ao dono, que não será alterado. Se dermos novamente um *ls*, veremos o seguinte:

```
-rwxr--r-- 1 Nicole administradores 8325   Abr 23  
01:15 arquivo.txt
```

Para remover um grupo, é só editar o arquivo */etc/group* e remover a entrada.

Com esses comandos já é possível administrar usuários, grupos e suas permissões no modo texto.

Desligando o Sistema

Depois de tudo isso, precisamos descansar, desligando o sistema. Como?



O comando é o *shutdown*. Perfeito, né?

Temos algumas opções do *shutdown*, sua sintaxe é a seguinte:

```
shutdown -[opção] time message
```

As opções acima podem ser: *h* para parar o sistema; *r* para rebootar; *-k* para só avisar e *c* para cancelar o shutdown. Esses são os mais usuais.

No *time*, você pode designar a hora de ativar o shutdown. Se for imediato, use *now*. Mas também é possível especificar um tempo em minutos. Por último, a mensagem será informada ao usuário que está conectado na sua máquina. Para garantir que o Shell não entenda sua mensagem como um comando, use sempre entre ' ' (aspas simples).

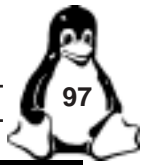
```
shutdown -k now 'asta la vista!!!!' <enter>
```

Isso irá informar ao usuário que o sistema será desligado, mas a opção *-k* só mostrará a mensagem e cancelará o boot.

Temos muitos outros comandos, como o *tar*, que será visto em *backups*; o *cut*, o *awk* e *sort*, que serão expostos em Shell Scripts.

Lista de Comandos e suas Funcionalidades

Comando	Função
awk	Procura por palavras em um texto. É uma linguagem de programação.
cat	Mostra calendário.
cat	Exibe conteúdo de um arquivo.
cc	Compilador de linguagem c (gcc).
cd	Acessa um diretório.
chmod	Altera permissões de arquivos e diretórios.
chown	Altera proprietários e grupo de arquivos e diretórios.
clear	Limpa tela.
cmp	Compara dois arquivos e mostra a localização da primeira diferença.
cp	Copia arquivo.
comm	Compara dois arquivos e verifica quais linhas são comuns.
Crontab	Agenda tarefas de sistema.
bsf	Procura arquivos grandes.
date	Retorna e altera data e hora do sistema.
diff	Mostra diferença entre dois arquivos ou diretórios.



Comando	Função
diff3	Mostra a diferença entre três arquivos ou diretórios.
df	Exibe informações sobre as partições, uso, quantidade livre,
du	Relatório de arquivos do diretório, uso, quantidade livre,
echo	Exibe mensagens.
find	Procura palavras em um texto.
finger	Mostra o usuário associado a certa chave.
fd	Formata disquetes (fd0H1440).
free	Exibe quantidade de memória e swap, livres e usadas.
grep	Procura palavras em um texto.
head	Mostra um número determinado de linhas do início do arquivo.
help	Ajuda no console.
kill	Reinicia e termina processos.
ln	Criar arquivos de link.
lpr	Imprime.
mail	Recebe e envia mensagens.
mkdir	Cria diretório.
more	Pagina texto.
mv	Move e renomeia arquivos e diretórios.
ps	Exibe processos do sistema.
pwd	Exibe o diretório atual.
reboot	Reinicia o computador.
rm	Deleta arquivo.
rmdir	Deleta diretório.
Comando	Função
sort	Ordena um texto ou saída de comando.
sleep	Coloca um processo em estado inativo por tempo determinado.
shutdown	Desliga ou reinicia o sistema.
stty	Exibe ou escolhe parâmetros do terminal.
su	Permite a troca de usuário sem fazer logoff.
tail	Exibe o fim do arquivo (10 linhas).
touch	Cria um arquivo.
tr	Troca letras.
tset	Escolhe o tipo de terminal.
uptime	Mostra o tempo de execução de um comando.
uniq	Elimina linhas repetidas seguidas em um texto
wc	Conta letras, palavras e linhas de um arquivo.
who	Mostra que está online na máquina.
whoami	Mostra que é você (Com que usuário você está logado)
exit e	Sai da sessão atual.



Esses são os comandos mais usados na administração e no dia-a-dia do **Linux**.

Antes de entrarmos na parte avançada de comandos, onde estaremos trabalhando com encadeamento de comandos através do pipe, vou mostrar como é possível sobreviver no modo texto, desde a navegação na internet até escutar música.

Para começarmos a usufruir da rede, temos que configurá-la (veremos no próximo capítulo).

Para conectar-se à internet via modem no modo texto, é necessário um programa que faça esta parte. Um dos mais famosos é o *Minicom*.

Para conectar-se à internet, todos os parâmetros da rede, como os DNSs e hosts, devem estar corretamente preenchidos (verifique *Redes* no próximo capítulo).

Instale o software Minicom. Para usá-lo, vá ao console e digite *minicom*. Na próxima tela, teremos que digitar o número de telefone do provedor da seguinte forma: uma string de inicialização do modem e o número. Veja:

```
ATDP22222222 ou ATDT22222222 <enter>
```

Onde ATDP é usado no caso da linha ser do tipo *pulse* e ATDT usado quando linha é *tone*.

Estabelecida a conexão, aparecerá uma tela solicitando username e senha. Digite os dados para efetivar uma conexão discada.

O processo de conexão via Minicom está finalizado, mas ainda precisamos conectar com o provedor. Para isso, precisamos sair do Minicom sem finalizá-lo. Para isso, digite *ALT + A + Q*. Assim, o console ficará liberado, mas sem finalizar o programa de conexão.

Agora, vá ao console e digite:

```
pppd /dev/modem defaultroute <enter>
```

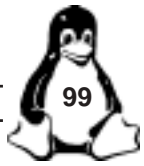
Pronto. A conexão está feita. Para desligar, é só digitar *ATH0*.









Já estamos conectados. Mas como vamos navegar? Existe um software só para isso, o *lynx*.

Para usá-lo, digite *lynx url*. Ou seja, supomos que queiramos entrar no site *dominiolinux*. Então digitaríamos:

```
lynx www.dominiolinux.com.br <enter>
```

Abaixo, os comandos para se navegar:



-  **Avançar:** barra de espaço;
-  **Voltar:** seta para a esquerda;
-  **Próximo link:** seta para baixo;
-  **Link anterior:** seta para cima;
-  **Acesso ao link:** Enter;
-  **Adicionar item ao bookmark:** A;
-  **Remover item do bookmark:** R;
-  **Ver bookmarks:** V.

Ainda podemos verificar nossos e-mails usando o *pine* ou *mutt* em conjunto com o *fetchmail*.

Para baixar as mensagens, use:

```
fetchmail -p POP3 pop.dominio.com.br -u usuario
```

Onde *pop.dominio.com.br* é o endereço de pop do servidor, e *usuário*, logicamente, seu usuário do servidor pop.

No console, digite *pine* para abrir o software. O *pine* é bastante intuitivo. Teclando *S*, você entrará em configurações. Para enviar mensagens, use a letra *C*. Não há muito mistério.

Mas não estaria completo sem o ICQ. Software que troca mensagens instantâneas entre usuários. Existe, também, o ICQ para modo texto, chamado *zicq*. Existem muitas outras opções, mas só falaremos do *zicq*.

Faça download dele no site *dominiolinux* e instale-o. Depois de instalado, é só executá-lo:

```
zicq <enter>
```

Quando executado a primeira vez, ele solicita seu UIN. Caso não seja cadastrado, digite *0* para se cadastrar e preencha as informações solicitadas.

Após o cadastro ser iniciado, ele dividirá a tela em três partes. No lado direito, estão os usuários cadastrados no seu contato. No lado esquerdo, as mensagens recebidas e, no canto inferior, as mensagens enviadas.

Para mudar o status para online, é só digitar *online*.

Agora, adicione o usuário aos seus contatos.



```
serch stato@domain.com <enter>
```

Este comando procurará o usuário no servidor. Caso ache, mostrará as informações do mesmo.

Para adicioná-lo, digite:

```
add uin apelido
```

Onde *uin* é o número do usuário no ICQ e *apelido* é o nome de exibição que você deseja para o usuário.

Para enviar, também é muito fácil:

```
msg [uin/apelido] <mensagem> <enter>
```

Mais informações sobre software podem ser obtidas no documento *commands.doc*, que está localizado no mesmo diretório do *zicq*.

E quando você quiser ver alguma imagem? No modo texto, também é possível visualizar imagens com o software *zgv*. O pacote de instalação está formatado e compactado. Então, é necessário compilá-lo. Mas não há muito problema quanto a isso. Dê uma olhada no capítulo anterior referente à instalação.

Depois de instalado, chame o software:

```
zgv <enter>
```

Os diretórios de seu winchester serão abertos. Navegue com as setas, selecione o arquivo que deseja visualizar e tecele *Enter*. É possível ver vários tipos de arquivos, como *bmp*, *jpg*, *gif*, etc.

Mas não estaria totalmente completo se não pudéssemos ouvir músicas, como *mp3*, *wav*.

Para escutar uma música *wav*, por exemplo, é só dar o comando:

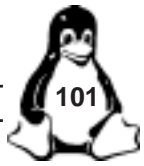
```
play musica <enter>
```

Para *mp3*, temos o *mpg123*. Instale-o e depois o chame no prompt:

```
mpg123 musica.mp3
```

Com tudo isso, você pode navegar, mandar e receber e-mail, visualizar figuras, mandar mensagens via ICQ, escutar música, enfim, no modo texto, é possível trabalharmos como no modo gráfico.

Não existem desculpas para não usar o modo texto.



Comandos Avançados

Falaremos do redirecionamento de comandos. O que é isso? É justamente o que o nome diz. Ele pega a saída de uma comando e joga para entrada de outro. O responsável por isso é o pipe (significa cano em inglês, já que ele encana uma saída na entrada de outro comando), simbolizado por uma barra vertical “|”.

Mas para que serve? Vamos supor que você listará todos os arquivos de seu diretório.

```
ls
```

Mas você quer listar apenas arquivos do tipo *txt*.

```
ls -l *txt
```

Até aí, sem problemas, né? Mas, se você tem uma quantidade “infinita” de arquivos *txt* e precisa somente dos que contém a palavra *lista*, seja no início, fim ou meio, digite:

```
ls -l *txt | grep lista
```

Fizemos a primeira canalização de comandos, onde o *ls* cria uma lista dos arquivos *txt* e o *grep* procura nessa lista os arquivos que tenha a palavra *lista* contida no nome. Ainda podemos ordenar tudo isso com o comando *sort*:

```
ls -l *txt | grep lista | sort
```

Feito isso, você salva o resultado em um arquivo chamado *totalista*:

```
ls -l *txt | grep lista | sort > totalista
```

Vamos a outro exemplo.

Você gostaria de saber quantos arquivos *txt* você tem? Fácil:

```
ls -l *txt | wc -l
```

Simplesmente, o *ls* fez um lista dos arquivos e o *wc* contou quantas linhas tinham nesta lista.





Agora, um exemplo interessante publicado pela revista do **Linux**, do Edison Figueira Jr.

Trabalhando em cima do arquivo *passwd* (arquivo onde estão os usuários criados no **Linux**), observamos que possui vários campos no seguinte formato cada linha:

```
login:senha:Uid:Gid:Descrição:HOME: Shell
```



Onde:

-  **Uid:** é o número que identifica o usuário;
-  **Gid:** é o número do grupo;
-  **HOME:** representa o diretório padrão do usuário;
-  **Shell:** o interpretador que será usado pelo usuário.

Então, todos os campos são separados por dois-pontos (:). O comando inicial, logicamente, é mostrar o conteúdo do arquivo */etc/passwd*:

```
cat /etc/passwd
```

O que queremos é fazer um relatório do tipo:

```
Login -> Descrição.
```

Precisamos do primeiro e do quinto campo. Através do *cat*, exibimos o conteúdo do arquivo e redirecionaremos para o *awk*, que tem a propriedade de tirar somente os campos solicitados:

```
cat /etc/passwd | awk -F ":" '{print $1,$5}'
```

Temos uma novidade: o comando *awk*. O que fizemos foi informar para o *awk* que os campos são separados por `:` (`-F ":"`) e, por fim, imprimir na tela os campos 1 e 5 do conteúdo (`print $1,$5`). Mas, ao final, temos uma saída do tipo “usuário descrição” e não é exatamente o que queremos. Faremos o seguinte:

```
cat /etc/passwd | awk -F ":" '{print $1 "\t -> \t" , $5}'
```

O que fizemos foi adicionar o item `\t -> \t` junto ao primeiro campo. O Shell, quando esbarra com `\` (barra), entende que a próxima letra deve ser desconsiderada, sendo assim, ele dará um espaço em branco e colocará o sinal `->`.

Certamente, necessitamos ordená-lo por ordem alfabética. Só adicionamos ao final comando o *sort*.

```
cat /etc/passwd | awk -F ":" '{print $1 "\t -> \t" , $5}' | sort
```

Assim, conseguimos indexar, mas, na realidade, era para indexar pelo segundo campo. Sem problemas:

```
cat /etc/passwd | awk -F ":" '{print $1 "\t -> \t" , $5}' | sort +2
```

Para finalizar este relatório, jogamos para um arquivo *txt*.



```
cat /etc/passwd | awk -F ":" '{print $1 "\t -> \t" , $5}' | sort +2 > relatório.txt
```

Para complementar esse exemplo, queremos que na saída final as letras estejam em maiúscula. Para isso, usamos o *tr*.

```
cat /etc/passwd | awk -F ":" '{print $1 "\t -> \t" , $5}' | tr a-z A-Z | sort +2 > relatório.txt
```

Nessa mesma matéria, temos outro exemplo não utilizando pipes, propriamente, mas um recurso do *find*. Iremos alterar os diretórios (somente diretórios) de um determinado usuário.

```
find / -type d -user edison -exec chmod u-w {}
```

Com a opção *find*, estamos tentando localizar todos os diretórios (-type d) do usuário *edison* (-user edison). E por último, com o parâmetro -exec podemos executar qualquer outro comando. No nosso caso, executamos *chmod*, alterando as permissões de usuário para escrita (w) para qualquer retorno do find ({}).

Podemos também utilizar variáveis. Por exemplo:

```
TODOSAS = cat /etc/passwd | grep ^a
```

A variável *TODOSAS* guardará a informação do comando, que exibe o conteúdo do arquivo *passwd* e manda para *grep*, que guarda somente as linhas que começam com a letra *a* (minúscula).

Para visualizar o conteúdo da variável, fazemos o seguinte:

```
echo $TODOSAS
```

O **Linux** entende que quando um nome é antecipado por \$, ele é uma variável, a não ser que esteja entre acentos. Experimente:

```
echo ` $TODOSAS `
```

Para ilustrar, vamos a mais um exemplo:

```
who | wc -l
```

Com isso, ele contará quantos usuários estão conectados.

```
echo Existem `who | wc -l` usuários conectados <enter>
```

Existem 8 usuários conectados

O Shell executa o comando até encontrar a ' (crase). Quando encontra a crase, executa este como outro comando dentro do inicial e retorna o valor para o primeiro comando.



Poderíamos criar um script que executaria tudo isso com apenas um comando nosso. Veremos nesse mesmo capítulo como fazer scripts.

Outra opção interessante é a utilização de resultados de contas, como por exemplo soma, divisão, etc. As opções, neste caso, são o *bc* e o *expr*. Para cálculos de inteiros, use o *expr*. Como para resultados não é inteiro, é necessário utilizar o *bc* e informá-lo das casas decimais.

```
soma : expr 2 + 5
subtração : expr 2 - 5
multiplicação : expr 2 \* 5
divisão: expr 10 / 2
modulo: expr 7 % 3
```

No caso do módulo, ele retornará o resto da divisão entre 7 e 3.

Mas se precisarmos fazer uma conta onde o resultado não seja inteira, usaremos o *bc*. Através do comando *echo*, veja o exemplo abaixo:

```
echo "13 / 2 " | bc
```

Ele retornará o valor 6. Por isso, precisaremos informar a escala do ponto decimal:

```
echo "scale=2 ; 13 / 2 " | bc
```

Ele pode fazer contas de adição, multiplicação, etc.

Uma outra forma de executarmos comando seguidos, seria usando “;” (ponto-e-vírgula), ou seja, ele executará um comando, depois outro, sem passar a saída de uma para outro, como no caso do pipe.

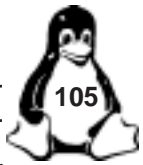
Vamos a um exemplo prático. Supondo que o diretório atual seja o */root*, acessaremos o */etc*, verificaremos se é mesmo o */etc* e em seguida listaremos o conteúdo do diretório:

```
cd /etc; pwd ; ls -l <enter>
```

Feito esse comando, ele acessará o */etc*, mostrará que estamos realmente no diretório citado e, por fim, listará todo o conteúdo. Agora, vamos executar o mesmo comando entre parênteses:

```
(cd /etc; pwd ; ls -l ) <enter>
```

O que aconteceu?



Ele acessou o diretório */etc*, mostrou que estávamos lá e listou o conteúdo. Mas, no final da execução, voltou para o diretório original. Por que isso?

Quando usamos os parênteses, ele chama um novo Shell, uma espécie de filho, executa os comandos, finaliza o novo Shell e retorna para o anterior.

Estes comandos são muito úteis na utilização de loops ou comandos que precisam anteriormente de alguma dependência. Se voltarmos para o capítulo de interface gráfica, temos um script que chama a interface gráfica *blackbox*. Logo abaixo, vemos o conteúdo do script:

```
#!/bin/bash
echo "exec blackbox" >/$HOME/xinitrc
startx
```

Poderíamos sem problema digitar no console:

```
echo "exec blackbox" >/$HOME/xinitrc ; startx <enter>
```

Certamente, o *blackbox* seria aberto. Agora fica mais fácil de entender o que foi feito. O comando *startx* executa a interface gráfica setada no arquivo *xinitrc*. O que fizemos foi gravar no arquivo *xinitrc* a linha *exec blackbox* e depois chamamos o *startx*, responsável por abrir uma interface.

Mesmo assim, fica mais prático termos toda essa linha diretamente em um script para podermos executá-lo através de um comando único.

O Shell ainda é um pouco mais poderoso do que vimos até agora, pois podemos executar instruções de loop, como *for*, *if*, *case*, entre outras, fazendo do Shell uma verdadeira linguagem de programação que nos auxiliará na administração do ambiente.

Vamos nos aprofundar um pouco mais neste assunto, onde veremos os loops acima citados e faremos scripts com o Shell.






Shell Script

O que é Shell Script?

Shell Script é um conjunto de comandos gravados em um arquivo-texto que são executados um após o outro.



Como sabemos, o Shell é o interpretador de comando do **Linux**. Temos vários Shells, mas o que normalmente é usado é o *bash* (Bourne-Again Shell). Existem outros, como o *sh* (Posix Shell), *ksh* (Korn Shell), *cs*h (C shell), etc.

-  Algumas características do Shell:
-  Uso de alias (apelido) para comandos;
-  Complemento de nome de arquivo ou comando;
-  Recuperação de comandos;
-  Uso de variáveis locais ou globais.

Alias

Já falamos sobre o alias num capítulo anterior, mas daremos uma relembração.

sintaxe : `alias nome = "comando"`

O arquivo `/etc/bashrc` é o responsável por guardar essas informações. Quando instalamos o **Linux** por default ele já cria alguns alias variando de distribuição para distribuição.

No **Linux**, por exemplo, não existe o comando *dir* (lista diretório) e *cls* (apaga digitação na tela). Porém, muitas vezes migramos para eles... temos certa dificuldade para lembrar de seus similares, o *ls* e *cls* (Mas se fizemos uma forcinha não vamos esquecer, com certeza :-)). Para amenizar este tipo de esquecimento em novos usuários, usamos um alias.

Como?

Façamos o seguinte:

Edite o arquivo:

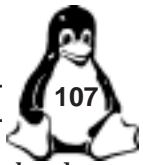
`/etc/bashrc`

Acrescente as seguintes linhas ao final do arquivo:

```
alias dir="ls -l"
alias cls="clear"
```

Salvamos o arquivo e executamos um logoff :

```
exit <enter>
```



Ao voltarmos para o Shell, quando executarmos *dir* ou *cls*, ele executará o similar gravado no *bashrc*. É interessante usarmos ele em comandos longos, como para ver logs de um determinado servidor, por exemplo. Teríamos que digitar *tail -f /var/log/Servidor*. Mas poderíamos criar um alias para esse comando:

```
alias Servidor="tail -f /var/log/Servidor"
```

Acredito que nesse caso seja mais interessante o uso do alias.

Autocompletar - Comandos

Dando continuidade as características, temos o complemento de nome de arquivo ou comando. Como funciona?

Realmente é como o nome diz: ele completa automaticamente o comando ou o arquivo.

Vamos fazer um teste?

Digite no console:

```
ifc <aperte a tecla tab>
```

Como vemos, ele completou para *ifconfig*.

E se digitarmos só *if*. Nesse caso, teclamos o *Tab*, ele retornará um som. Este som significa que existe mais de um comando começando com as letras *if*. Tecle novamente o *Tab* e ele mostrará todos os comandos que estão no *path*.

E quanto aos arquivos? Ele completa também?

Digamos que temos dois arquivos */root* chamados: *Talarico.txt* e *Talheres.txt*. Suponhamos que queiramos editar o *Talarico*. Observe que até a letra *l* seus nomes são iguais e, para obtermos sucesso, teremos que digitar ao menos mais uma letra após o *l*. Com isso, teremos:

```
vi /root/Tala <tecle TAB>  
vi /root/Talarico
```

Após teclarmos *Tab*, ele completará o nome do arquivo, *Talarico.txt*. Depois disso, tecle *Enter* para executar o comando.

No começo, você irá estranhar um pouco, mas logo se adaptará e, certamente, irá gostar muito, pois ele agiliza o processo de digitação.

Falamos também da recuperação de comandos, que nada mais é do que utilizar os últimos comandos digitados sem ter que digitá-los novamente. Usando as teclas *seta para baixo* <½> e *para cima* <¿>, você



poderá navegar pelos últimos comandos realizados no Shell. É similar ao *doskey*, do DOS, com um pouco mais de capacidade, pois quando desligamos a máquina ele não perde o que foi digitado anteriormente.

Váriáveis

Por característica, temos as variáveis.

A variável *HOME*, por exemplo, guarda o diretório do usuário atual no sistema. Basta digitar *echo \$HOME* <enter> para poder visualizar seu conteúdo. Essa variável já é predefinida pelo sistema, assim como *PATH*, *PS1*, *MAIL*, *TZ*, *LOGNAME*, *TERM*, etc.

Veremos mais a frente como podemos trabalhar com ela.


Mas para aqueles que não sabem o que é uma variável, iremos explicar e é a partir daqui que iremos começar a entender o Shell Script.


Variável, como o nome diz, é um lugar onde o Shell guarda informações para serem usadas posteriormente, podendo ser modificadas conforme seu uso. Em programas compilados, a variável só existe enquanto o programa estiver rodando, pois ela fica armazenada na memória RAM.

Para se criar uma variável, não existe segredo. A sintaxe é bem simples:

```
variável=conteúdo
```

Existem dois tipos de variáveis: as locais e as ambientais ou globais.

 **Variável Local:** Disponível somente para o Shell atual, ou seja, este que estamos usando. Se abrirmos outro Shell, estas variáveis não estarão disponíveis para outros processos ou subprocessos.

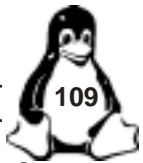
 **Variáveis Globais ou Ambientais:** Sempre estará disponível para o Shell atual com outros, podendo ser utilizado em subprocessos e outros Shells.

É possível fazer uma variável local tornar-se global. Deve-se utilizar o comando *export*.

```
export variável
```

Pode-se criar ao mesmo tempo em que se exporta o conteúdo:

```
export nome=Pedro
```



Para listar as variáveis locais, usa-se o comando *set*, e para listar as variáveis globais, usa-se o comando *env*.

Muitas vezes queremos que uma variável tenha um valor fixo, que não possa ser alterada ou apagada, para isso temos um comando muito simples:

```
readonly variável
```

E, finalmente, para apagar um variável, usa-se o comando:

```
unset variável
```

Uma variável protegida pelo *readonly* não pode ser deletada, ela só será finalizada com logout ou terminar o processo que a criou.

Como foi dito anteriormente, para visualizarmos o conteúdo de uma variável, devemos usar o comando *echo*, mas diante da variável usamos o símbolo *\$*. Sendo assim, devemos executar o comando da seguinte forma:

```
echo $variável <enter>
```

No Shell, existem alguns tipos de substituição onde sinais significam mais que isso. É possível passar saída de comandos para variáveis, entre outras coisas.

Utilizemos como exemplo o nome José da Silva Pereira. Gravamos nas variáveis *nome1*, *nome2* e *nome3* para *Jose*, *Silva* e *Pereira*.

```
nome1=Jose
nome2=Silva
nome3=Pereira
echo ${nome1}da$nome2$nome3 <enter>
```

Usamos as chaves em *nome1*, pois o Shell certamente iria interpretar o *\$nomeda* como sendo uma variável e esta variável nem existe.

Você pode colocar um *path* na variável, por exemplo:

```
arquivo=/root/arquivo.txt <enter>
cat $arquivo
```

Outra forma de usarmos as variáveis é para substituir um comando por seu resultado.

A sintaxe é *\$(comando)* ou *'comando'*.

Podemos armazenar o diretório corrente em uma variável:



```
diretório_atual=(pwd)
diretório_atual='pwd' (este acento é uma crase)
```

Podemos mover alguns tipos de arquivos para um diretório específico:

```
mv $(ls *gif) /home/user/gifs
mv `ls *gif` /home/user/gif
```

Existe também uma substituição por sinais, usando-se o til (~).

O til sozinho é substituído pelo conteúdo da variável HOME.

```
echo ~ <enter>
/root
```

O til seguido do + (sinal de mais) é substituído pelo valor da variável PWD (diretório atual).

```
echo ~+
```

E o til seguido de - (sinal de menos) é substituído pelo valor da variável OLDPWD (diretório corrente anterior).

```
echo ~-
```

Vamos verificar algumas variáveis já configuradas no sistema, por exemplo:

 **PATH:** Guarda o caminho.

As alterações definitivas desse arquivo devem ser feitas em *.bash_profile* dentro do diretório do usuário. Para tornar essa configuração globalizada, deve-se alterar o arquivo */etc/profile*. Inclusive, muitas outras variáveis estão configuradas nesses arquivos. Por exemplo, a variável *PS1*. Quem?

Essa variável é a responsável pelas informações do prompt de comando. Não ficou muito claro?

Lembre-se desse prompt:

```
[root@localhost /root]
```

Lembrou? É isso aí!!! Ele é o responsável por essas informações.

Vamos brincar um pouco colocando algo do tipo:

```
nome_do_usuario@superman - data - diretório_atual >>>>
```

Ficariamos com algo como:

```
PS1 ``$LOGNAME`@superman - `date + %d/%m/%y` - $PWD
>>>` <enter>
```



Logo que digitarmos o *enter*, o prompt automaticamente será alterado para o padrão que foi solicitado. Deve-se colocar no arquivo *bash_profile*, no diretório do usuário, ou */etc/profile* para alteração ser global.

Mas devemos tomar cuidado com alguns sinais ou letras que o Shell interpreta como sinais especiais, como o \$, que informa para o Shell mostrar conteúdo da variável. Por vezes, queremos apenas o caractere sem que o Shell interprete como um sinal especial, como para mostrar valores monetários (R\$ 200,00, por exemplo). Para isso, o sistema oferece um mecanismo que remove o significado especial de alguns caracteres. Chamamos isso de Quoting.

Os caracteres usados (neste caso, para omissão de caracteres especiais) são: barra invertida (\), aspas simples (') e aspas duplas (").

A barra invertida remove o significado do próximo caractere:

Exemplos:

```
nome=Andre  
echo \$nome
```

```
$nome
```

```
echo ~  
/home
```

```
echo \~  
~
```

Já as aspas simples omitem o significado especial dos caracteres entre as aspas.

Exemplo:

```
nome=Joao  
echo 'Meu nome é $nome'  
Meu nome é $nome  
  
echo 'Menu nome é '$nome  
Menu nome é Joao
```

Não podemos confundir as aspas com a crase. A crase normalmente é usada para dizermos que existe um comando entre elas.



E enfim, as aspas duplas são usadas também para suprimir o conteúdo que está entre as aspas. Mas ele é menos exigente, pois entende quando estamos querendo passar o conteúdo de uma variável, quando usamos a crase para passar comandos e os outras *quoting*, como a barra invertida e a aspa simples.

Exemplos:

```
nome=Joao
echo "Meu nome é $nome"
Meu nome é Joao

echo "A lista do diretório é a seguinte ls -l"
A lista de diretório é a seguinte ls -l

echo "A lista de diretório é `ls -l`" (uso em conjunto
crase)

echo A lista de diretório é total 4 arquivos
....

...
...

echo "A variável \${nome} é $nome"
A variável $nome é Joao
```

Criando Arquivos Shell

Com tudo o que foi passado, podemos dar o primeiro passo para construção de nosso Shell Scripts.

Vamos partir de comandos até instruções e funções, onde serão passadas novas informações sobre o ambiente em si.

Digamos que você sempre execute o comando *find / -name arquivo*.

Crie um arquivo utilizando o editor VI, por exemplo.

```
vi procura
```

O primeiro parâmetro que passamos é em qual Shell queremos executá-lo. O *#!/bin/bash* fará com que o Shell execute no *bash*. Caso queiramos que seja executado em outro, basta trocar: *#!/bin/csh*, etc. O padrão utilizando normalmente é o *bash*, por isso vamos utilizá-lo.



Logo depois de colocado o Shell na primeira linha do arquivo em que devemos passar o comando, teremos o conteúdo abaixo:

```
#!/bin/bash  
find / -name arquivo
```

Salve o arquivo. Agora é preciso transformá-lo em um arquivo executável. Como?

Simples: dando permissão de execução sobre o mesmo. Lembre-se: pode ser através de números com valores de permissão ou diretamente na letra.

```
chmod a+x procura  
chmod 755 procura
```

Esses dois comandos transformam o arquivo em executável com permissão, podendo ser executado pelo dono, pelo grupo e por outros usuários.

Agora é só executá-lo:

***procura**

Ih... não funcionou. Por quê?

Simples, como o arquivo não está no path para executá-lo, devemos colocar `./` (ponto e barra) antes do nome do arquivo. Isso informa para o Shell executar esse arquivo. Pode-se, ainda, editar o arquivo `/etc/profile` e acrescentar o caminho na variável `PATH`, ou colocá-lo em um path que já exista, por exemplo `/usr/bin`.

Assim, podemos executar:

```
./procura
```

Note que não ficou tão bom, pois só podemos consultar um arquivo. Seria mais interessante se pudéssemos escolher qual arquivo queremos que o comando procure.

Exatamente o que faremos.


No Shell, existem variáveis posicionais que, como o nome diz, informam a posição desta variável em uma linha. Com isso, podemos alterar nosso código para que capturem a primeira variável posicional após o comando.

A forma que é passada para o Shell é através das posições, que variam de 1 até x. Logicamente, quanto menores forem os valores que o usuário precisar entrar, melhor. Supondo que você escreva em uma linha:




```
./comando    nome sobrenome etc
```

Sendo que:

 *Nome* = variável 1;

 *Sobrenome* = variável 2;

 *Etc* = variável 3.

Não devemos esquecer que para pegarmos o conteúdo, sempre usaremos o valor \$, que mostra conteúdo de variáveis.

Agora ficou fácil!

Alteremos de :

```
#!/bin/bash
find / -name arquivo
```

Para:

```
#!/bin/bash
find / -name $1
```

Para procuramos, por exemplo, um arquivo chamado *conectiva*, faremos:

```
./procura conectiva
```

Podemos saber quantas entradas temos na linha usando a variável \$#. Para listá-las, usamos \$*.

Outra forma de passar o nome do arquivo para o Shell, o que é, aliás, mais interessante, pois o usuário não necessita saber como funciona o programa nem passar instruções antecipadas.

Passamos essa informação através do comando *read*. Devemos iniciar o programa com uma mensagem, informando ao usuário o que desejamos. Depois, com o comando *read*, delimitamos as variáveis (caso exista mais de uma). O *read* dá um pause no script até ser digitado o *enter*.

Se o usuário digitou algo, o *read* pegará o que foi digitado, absorverá as informações e redistribuirá em variáveis, desde que o usuário tenha separado por espaço cada entrada de valor.

Vamos para um exemplo:

Peguemos o arquivo que procuramos e o editamos, fazendo as seguintes alterações:

```
#!/bin/bash
```



```
echo Digite o nome do arquivo a ser procurado:
read arquivo
find / -name $arquivo
```

Salve e execute:

O programa dará a mensagem e aguardará que seja informada uma entrada e pressionado o *Enter*.

Posteriormente, podemos implementar para que, se o usuário não digitar nada, o script dê uma mensagem e saia, sem dar aqueles erros padrões. Veremos isso mais adiante.

```
./procura
```

Digite o nome do arquivo a ser procurado:

```
$|
```

Muitas vezes, aparecem mensagens de erros do tipo:

```
find: /root: Permissão negada
```

Não queremos que o usuário veja as mensagens de erro. Podemos trabalhar com isso, evitando mensagens desnecessárias. É bem simples! O que devemos fazer é redirecionar a saída do *fd* (file descriptor) do comando para uma saída nula, ou qualquer outro arquivo de erros.

Por padrão, a saída de comando é a número 1 e a de erro é a 2.

Podemos, por exemplo, redirecionar a saída de erro de um comando para um arquivo, por exemplo:

```
find / -name conectiva 2>erros.txt
```

Ou então, vamos direcionar a saída de erro para que esta não apareça.

```
find / -name conectiva 2>/dev/null
```

Com isso, serão omitidas todas as mensagens de erro.

Vamos supor que queiramos listar todos os arquivos com extensão *.txt* de nossa máquina, mas não teremos permissões em todos os diretórios, principalmente um usuário normal. Mas esse comando trará mais de 1000 arquivos, o que seria impossível visualizar. Então, queremos fazer o que? Visualizar devagar os arquivos e omitir as mensagens de erro.

```
find / -name "*.txt" 2>/dev/null | more
```

Agora, podemos visualizar todos os arquivos sem mensagens de erro, mas mesmo assim ficou difícil gerenciar tantos arquivos só com



a visualização. Vamos armazená-los em um arquivo e as mensagens de erro em outro.

```
find / -name "*.txt" 1>lista 2>erros
```

Ficou quase bom. Mas queremos, ainda, mandar o resultado para um arquivo e vê-lo na tela. Para isso, temos um comando *tee*. Veja seu modo de uso abaixo:

```
find / -name "*.txt" 2> erros | tee lista
```

O comando *tee*, além de possibilitar a visualização da saída de um arquivo, manda para outro (e sem as mensagens de erros, já que os erros estão sendo direcionados para o arquivo *erros*).

Agora realmente ficou bom.

É sempre bom utilizarmos este tipo de comando, ou até mesmo com instruções, pois quanto menos exigido do usuário, menor a chance de dar algo errado.

Agora, suponhamos que você queira mover um arquivo de 20 Mb para outro diretório diferente do original, ou então que queiramos escutar uma música mp3. A princípio, ele prenderá o nosso prompt e seremos obrigados a usar outro Shell. Isto se chama *foreground*, ou *execução em primeiro plano*.

Para liberarmos o prompt, podemos fazer eles executarem em segundo plano, assim devolvendo o prompt de comando para ser usado.

Simplesmente digite o comando seguido do sinal *&*. Ele retornará algo do tipo *[1] 1853*, onde *1* é o número do *job*. Se tivéssemos executando uma cópia em *background* do tipo *cp /home/arquivo /diretório &*, ele retornaria *[1] 1245*.

Logo depois, vamos escutar uma mp3, *mpg123 musica.mp3*. Agora, ele retornaria *[2] 1684*. Então, ficou claro que temos 2 *jobs* rodando em *background*. O número seguinte é referente ao *pid* do processo rodando no Kernel.

Legal, não é?

O arquivo foi copiado com sucesso e a música continua a tocar. E se você quiser trocar de música?

Temos dois comandos: o *fg* e o *bg*. Para trazermos de volta o mp3, teríamos que digitar *fg 1*, onde *fg* é *foreground* (primeiro plano) e *1* é o



job. Já que o arquivo terminou de ser copiado, a música passou a ser o primeiro *job* rodando. Fácil!

Agora, para que serve o comando *bg*?

Supondo que você começou a tocar as músicas e esqueceu do sinal *&*. Ou melhor, você copiou outro arquivo de 50 Mb para um novo diretório e esqueceu de colocar o sinal *&*, mas ele já copiou 5 Mb. Isto é, se você parar, vai perder o trabalho já executado. Temos, então, um pequeno truque para o *bash*.

Digite *Ctrl+Z*, que ele parará a execução do comando.

```
cp /home/user/arquivo50MB /diretório
Digitamos Ctrl + Z
[1]+ Stopped cp /home/user/arquivo50MB /diretório
bg 1
[1]+ cp /home/user/arquivo50MB /diretório
```

Para consultarmos os jobs que estão sendo executados, precisamos usar o comando *jobs*. Ele mostrará todos os jobs que estão sendo executados, o número, o status e o comando.

Com isso, podemos executar vários comandos e programas. Quando quisermos alterar suas formas de executar em *foreground* e *background*, executaremos os comandos *fg* e *bg* ou até o sinal *&*.

Comando test - Retorno de Variáveis

Tudo isso é muito útil, mas adiante precisaremos de variáveis com valores *true* e *false* para execução de instruções, como *while*, *for*, etc. Para isso, temos o retorno da variável *\$?*. Se o resultado do comando ou script for true, essa variável guardará no conteúdo o valor *0*, caso contrário, guardará *1*.

Veja o exemplo abaixo:

```
cp
cp : faltam argumentos dos ficheiros
Tenta 'cp -help' para mais informação
```

Digite:

```
echo $?
1
```

O valor retornado foi *false*, por isso a variável guardou o valor *1*. Agora, digitemos outro teste:



```
ls -l
echo $?
0
```

Como o comando executado foi correto, o retorno foi true, ou seja a variável guardou o valor 0.

Com o retorno de variável, podemos executar várias instruções, mas existe um comando que recupera o valor da \$? Esse comando é o *testa*, que pode ser usado na forma de []

Sua sintaxe é a seguinte:







```
teste expressão ou [ expressão ]
```

O comando *testa* pode avaliar condições de inteiros, strings e arquivos.

Criaremos alguns exemplos, iniciando com testes numéricos. Sua sintaxe está logo abaixo:

```
[ numero parâmetro numero]
```

Onde os parâmetros são:

-  **lt** : menor que;
-  **le**: menor igual que;
-  **gt**: maior que;
-  **ge**: maior igual que;
-  **eq**: igual;
-  **ne**: diferente.

Exemplos:

```
VALOR= 10
```

```
[ $VALOR lt 7 ]
echo $?
1
```

```
[ $VALOR gt 7 ]
echo $?
0
```

Devemos sempre prestar atenção para não passarmos valores nulos. Caso não seja atribuído um conteúdo à variável *VALOR*, na hora da verificação, ocorreria algo do tipo [= 7], o que gera um erro no Shell. Para evitarmos isso, sempre que houver variáveis, vamos



colocá-las entre aspas (“”), pois caso a variável não tenha valor atribuído, o *testa* comparará [“” -lt 3], retornando como false.

Agora, passaremos para os testes de string. Veja a sintaxe abaixo:

```
[ string parâmetro string ] ou [ parâmetro string ]
```

Onde parâmetro :

=	Igual.
!=	Diferente.
=-z string	Se o comprimento for 0, retorna true.
=-n string	Se o comprimento for maior que 0, retorna true.
string	Se o comprimento for maior que 0, retorna true.

Em caso da string ter mais de uma palavra, é obrigatório o uso do parêntese para não gerar erro no Shell.

Exemplos :

```
A="Bom Dia"
```

```
[ $A = "Bom Dia" ]
Erro
```

```
[ "$A = "Bom Dia" ]
echo $?
0
```

```
[ $A ]
Erro
```

```
[ "$A" ]
echo $?
0
```

```
[ -z $A ]
Erro
```

```
[ -z "$A" ]
echo $?
1
```

Quando forem usados parâmetros numéricos (*lt*, *gt*, *eq*, etc.), o Shell tratará todos os valores da comparação como números e quando



os parâmetros usados forem de string (=, !=, etc.), tratará apenas como string.

Exemplo :

```
X=1
Y=01
```

```
[ " $X" = "$Y" ]
echo $?
1
```

Já que a string 3 é diferente da string 03.

```
[ $X eq $Y ]
echo $?
0
```

Nesse caso, o retorno foi *true* por que o número 3 é igual ao número 03.

Ainda temos o teste de arquivos, onde a sintaxe está logo abaixo:

```
[ -opção arquivo ]
```

Onde opção é :

-c	Se o arquivo existir e for do tipo caractere, o retorno será verdadeiro.
=-h	Se o arquivo existir e for do tipo link simbólico, o retorno será verdadeiro.
=-p	Se o arquivo existir e for do tipo named pipe (rede), o retorno será verdadeiro.
=-e	Se o arquivo existir, o retorno será verdadeiro (= 0).
=-f	Se o arquivo existir, o retorno será verdadeiro (= 0).
=-r	Se o arquivo existir e puder ser lido, o retorno será verdadeiro.
=-s	Se o arquivo existir e seu tamanho for maior que 0 Kb, o retorno será verdadeiro.
=-w	Se o arquivo existir e puder ser gravado, o retorno será verdadeiro.
=-x	Se o arquivo existir e puder ser executado, o retorno será verdadeiro.
=-x	Se o arquivo existe e pode ser executado, o retorno será verdadeiro.
=-d	Se o arquivo é um diretório, o retorno será verdadeiro.



Podemos utilizar um exemplo para ver se um determinado serviço está ativo. No **Linux**, os serviços, quando são iniciados, criam um arquivo em determinado diretório, e quando são “stopados”, apagam este arquivo. Vamos tomar como exemplo o Samba, um servidor que serve para comunicação entre máquinas **Linux** e **Windows**, dentre outras coisas que veremos em outros capítulos.

O Samba, quando é iniciado, cria um arquivo *smb* no diretório */var/lock/subsys*.

Para vermos se o Samba está sendo executado, só digitamos:

```
[ -f /var/lock/subsys/smb ]
echo $?
1 ou 0
```

Se estiver sendo executado, o retorno será 0, caso contrário, 1. Veja a utilidade se colocarmos essa comparação dentro de uma instrução *if* para verificar se o Samba está sendo executado. Se a resposta for positiva, ele sai do script, se não, ele inicia o serviço.

Existem, também, os valores booleanos para adicionarmos ao comando *test*, são eles:

<code>= -o</code>	<code>-or (ou).</code>
<code>= -a</code>	<code>-and (e).</code>
<code>!</code>	<code>-not (negação).</code>
<code>\(... \)</code>	<code>agrupamento de condições.</code>

Exemplos:

```
[ "$RESP" = y -o "$RESP" = Y ]
```

Verdadeiro, caso a variável *RESP* seja igual a *y* ou *Y*.

```
[ "$X" -gt 30 -a "$X" -lt 50 ]
```

Verdadeiro, caso a variável seja maior ou igual a 30 e menor ou igual a 50.

```
[ ! -f /var/lock/subsys/smb ]
```

Verdadeiro, caso o arquivo *smb* não exista, ou seja, o serviço Samba esteja parado.

```
[ \( -f $1 \) -a \( "$2" = y -o "$2" = Y \) ]
```

Se a primeira entrada é um arquivo e a segunda é igual a *y* ou *Y*.



Creio que agora já estamos preparados para entender instruções e funções.

Vamos iniciar por uma das mais conhecidas: o *if*, posteriormente, veremos outras instruções.

Instruções if, elif e else

Usada para desvios, faz com que uma determinada condição execute determinado passo dentro da instrução.

Sintaxe:

```
if      [ condição ] ; then
    comandos
elif    [ condição ] ; then
    comandos
elif    [ condição ] ; then
    comandos
fi
```

Exemplo:

X=10

```
if      [ $X -lt 5 ] ; then
    echo "O valor da variável é $X, que é menor que
5"
elif    [ $X -gt 5 ] ; then
    echo "O valor da variável é $X, que é maior que 5 "
elif    [ $X -eq 5 ] ; then
    echo "O valor da variável é $X, que é igual a 5"
else
    echo "O valor é igual a 0"
fi
```

Instrução Case

O *case* é um comando de fluxo tal como o *if*. É usado para suportar interfaces de menu para decisão em várias opções disponíveis.

Sintaxe:

```
case variável in
    caso 1) comandos
;;
```



```
    caso 2) comandos
    ;;
    caso 3) comandos
    ;;
esac
```

Exemplo:

```
case $VALOR in
    [dD] ) date
    ;;
    [ cC ] ) cal
    ;;
    [wW] )who
    ;;
    *) echo opção invalida
esac
```

Como você pode ver, é bem simples. Mas existe, um fato, aqui, um pouco estranho: Como o case pegará o valor da variável? Para isso, podemos usar o *read*, lembra-se? Com ele, pegamos o valor que o usuário digitou e podemos mostrar um menu com todas opções. Vamos mudar algumas coisas no Script:

```
echo "      MENU      "
echo "d mostra data e hora"
echo "c mostra calendário"
echo "w mostra com que usuário você está logado"
read VALOR
case $VALOR in
    [dD] ) date
    ;;
    [ cC ] ) cal
    ;;
    [wW] )who
    ;;
    *) echo opção invalida
esac
```

Ficou muito melhor... mas podemos melhorar ainda mais com o comando *select*.

Sintaxe:

```
Select  variável in <LISTA>;
do
    comando 1;
    comand 2;
done
```



Em vez de usarmos o *case* usaremos o *select*, que já cria, automaticamente, um menu que facilita a vida do usuário.

```
Select $X in date cal who exit ; do clear ; $X; done;
```

- 1) Veja a data de hoje.
- 2) Veja o calendário do mês.
- 3) Com que usuário você está logado.
- 4) Sair.

O que fizemos foi criar uma variável *X*, que fica vinculada aos números de um a quatro, que, por sua vez, ficam vinculados à lista, que é *date*, *cal*, *who* e *exit*. Quando digitarmos 3, ele atribui o valor que está na lista para a variável *X*. Dessa forma, o conteúdo de *X* seria *who*, logo em seguida, o *select* executa o que está depois de *do*, que no nosso caso é *clear* (usado para limpar a tela) e, em seguida, executa o conteúdo da variável *X* (*\$X*), que neste caso é *who*.

Um método interessante, já que o usuário final simplesmente chamará o script. Em seguida, escolha no menu uma das opções, evitando, com isso, algum conhecimento prévio do usuário.

Agora, está na hora de vermos os comandos de looping, como *while*, *for*, etc.

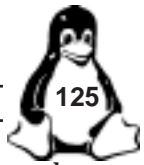
Antes de começarmos propriamente, temos que falar do *let* ou *(())*. Em geral, usamos o comandos de looping incrementando-os. Usamos, na maioria das vezes, algo do tipo *variável = variável + 1*. Com o uso do *let*, podemos usar expressões aritméticas, tornando possível o uso dos loopings no Shell Script.

Podemos usar tanto o *let* como o *(())*. Veja abaixo a demonstração:

```
let "$X =$X + 1" ou (( $X = $X + 1 ))
```

Os operadores lógicos estão listados abaixo:

!	Negação (not).
==	Igual.
!=	Diferente.
*	Multiplicação.
/	Divisão.
%	Módulo.
< <=	Menor que, menor igual.
> >=	Maior que, maior igual.



O uso do *()* torna-se indispensável, pois o Shell poderia entender como caracteres especiais, como o */* (para omitir a próxima letra) e o *>* (que redireciona de comando).

Visto isso, iniciaremos pelo *for*.

Instrução For

O *for* funciona como um contador. Você terá que fornecer a lista para ele, sendo que o mesmo atribuirá cada valor da lista a uma variável executando os comandos. Sua sintaxe está logo abaixo:

```
for <var> in <lista>;
do
    comandos
done;
```

Podemos trabalhar de várias formas com o *for*, ele é muito útil para realizarmos *backups*, renomeação de arquivos, etc.

Seguem abaixo alguns exemplos:

Neste primeiro exemplo, mostraremos o funcionamento básico do *for*:

```
for i in 1 2 3 4 5 6 7 8 9 10 ; do
    echo $i
done;
```

Neste caso acima, a cada loop ele vai mostrar o conteúdo da variável *i*. Mas, pode ser mais fácil fazer algo do tipo:

```
for i in $(seq 10); do
    echo " $i x 4 = " $(( $i * 4))
done;
```

Resultado:

```
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4x 5 = 20
4 x 6 = 24
4x 7 = 28
4 x 8 = 32
```



$$4 \times 9 = 36$$

$$4 \times 10 = 40$$

Já no segundo exemplo, o comando *seq* preencheu uma sequência de números, o que facilitou o preenchimento da nossa lista. Em seguida, usamos apenas o *echo* com mais um comando, que simplesmente multiplica o valor da variável por quatro, mostrando na tela uma saída do tipo $4 \times 1 = 4$, $4 \times 2 = 8$, etc.

Quando usamos o asterisco (*), ele trabalha como o comando *ls*:

```
for i in * ; do
    echo $i
done;
```

Com o exemplo acima, ele pegará cada arquivo do diretório e mostrará na tela. Com isso, temos uma ótima forma de alterarmos nomes de arquivos e backups. Para fazermos um backup, usaríamos a expressão abaixo:

```
for i in *; do
    mv "$i" "/usr/backup/$i.bkp"
done
```

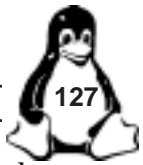
Para renomear, não seria muito diferente. Vamos renomear todos os arquivos com extensão *gif* para *.gif.bak*:

```
for i in *.gif ; do
    if [ -f "$i" ]; then
        mv "$i" "${i}.bak"
    fi;
done
```

Mas para voltar para uma forma normal, por exemplo, supondo que o arquivo ficou com uma extensão parecida com *arquivo.gif.bak*, teremos que fazer um procedimento um pouquinho diferente:

```
for i in *.bak; do
    cp $i /diretório/$(echo $i | awk -F"." '{print $1"."$2}')
done;
```

Neste último caso, o que fizemos foi o seguinte: o *for* irá contar apenas os arquivos *.bak*. Dessa forma, copiamos todos os *gif.bak* para o diretório que queremos. Mas para alterarmos o nome, tivemos que executar um conjunto de comandos. O primeiro passo foi passar o conteúdo da variável *i* para o comando *awk*, que em seguida separa o conteúdo onde houver "." (ponto) e imprime o primeiro e o segundo



conjunto de caracteres entre os pontos; mas se usássemos vírgula, os arquivos saíam separados como *gif*, por isso utilizamos o *awk* para imprimir o primeiro conjunto *\$1* e colocamos “.” (ponto). Depois, ele imprimirá o segundo conjunto de caracteres *\$2*, formando, assim, uma palavra com extensão *arquivo.gif*. No início desta instrução, sua sintaxe falava sobre lista, e literalmente pode ser uma lista como a seguinte:

```
for i in Banana Maça Goiaba ; do
    echo $i
done;
```

Ou:

```
OPCAO="Banana Maça Goiaba"
for i in $OPCAO; do
    echo $i
done;
```

O resultado será:

```
Banana
Maça
Goiaba
```

Este exemplo abaixo altera o nome de arquivo de letras maiúsculas para minúsculas. Pode ser muito útil, pois pouparia muito tempo. Também podemos fazer o contrário, incrementando mais ainda. Seria possível fazer a primeira letra maiúscula e o restante minúscula, mas no nosso caso, iremos mostrar o exemplo de maiúsculas para minúsculas:

```
for padrão in * ; do
    resultado=$(echo $original | tr [:upper:]
[:lower:])
    if [ $original != $resultado ]
        mv $original $resultado
    fi;
done;
```

Neste caso acima, listamos o diretório jogando cada loop na variável *original*. O primeiro comando atribui à variável *resultado* o valor da variável *original* passado pelo comando *tr*, que transforma tudo que está em maiúsculo para minúsculo. Na instrução *if*, ele compara o valor da variável *original* com a *resultado*, que caso seja diferente, subentende-se que a *original* possui pelo menos um caractere maiúsculo. Sendo assim, ele renomeará o arquivo *original* com o valor da variável *resultado*.



Tivemos uma boa visão do comando *for*, mas agora veremos um dos mais usados no esquema de laços ou loops: o *while*.

Instrução While

A comando é muito simples de ser usado. O *while* testará uma condição até que ela seja verdadeira. Executará os comandos dentro do *while* e voltará a testar a condição. Se for atingido, ele sai do loop, caso contrário, ele executará novamente o comando. A condição pode ser uma um resultado positivo [*-f arquivo*] ou de comparação [*\$X -qt \$Y*].

Sua sintaxe está logo abaixo:

```
while [ condição ];  
do  
    comando 1;  
    comando 2;  
    ...  
done;
```

O *while* pode ser usado para infinitas tarefas. Vamos exemplificar algumas:

```
while [ $contador -lt 10 ]; do  
    echo O valor de contador = $contador  
    (( contador = contador + 1 ))  
done
```

Resultado:

```
O valor de contador = 0  
O valor de contador = 1  
O valor de contador = 2  
O valor de contador = 3  
O valor de contador = 4  
O valor de contador = 5  
O valor de contador = 6  
O valor de contador = 7  
O valor de contador = 8  
O valor de contador = 9
```

Note que o script conta apenas até nove. Por que ?



Quando fizemos a comparação, solicitamos que, enquanto a variável *contador* fosse menor que 10, ele realizaria os comandos de *echo* e *soma*, mas não falamos nada de igual a 10. Para ele contar até 10, deveríamos, no lugar de *lt*, usarmos o *le*.

O comando *while* é melhor utilizado com outros comandos em conjunto, testando condições (com o *if*, por exemplo). Existe também a opção de utilizarmos a recursividade, ou seja, loop dentro de loop. Mas, devemos tomar cuidado para que não entre em loop eterno, o que fará com que o Shell não tenha fim.

```
while [ condição ]; do
    comandos
    while [ condição ]; do
        comandos
        comandos
        ...
    done;
    ...
done;
```

Veja o exemplo abaixo:

```
x=0
while [ "$x" -lt 10 ];
```

O primeiro loop será executado enquanto *x* for menor que 10.

```
y="$x"
while [ "$y" -ge 0 ];
```

Haverá execução até que *y* seja menor igual a 0.

```
let y=y-1
done
let (( x = x + 1 ))
echo
```

O valor de *x* é *\$x* e o valor de *y* é *\$y*.

```
done;
```

Note que, no começo do segundo *while*, fará um loop aumentando na medida em que o valor de *x* aumenta. No primeiro loop, o valor de *x* é 0. Portanto, *y* = 0-1, que é -1 e menor que 0, fará apenas um loop, voltando para o loop inicial, onde o *x* ainda vale 0 e é menor que 10. Já quando chegamos no *x*=4, o *y* obviamente também valerá 4. No primeiro loop do segundo *while*, *y* = 4 -1, que vale 3, ainda não é menor ou igual 0; um loop *y*=3-1, que vale 2, ainda não é menor ou igual a 0



e assim por diante, até chegar ao valor de 0. Quando estivermos com $x = 10$, o segundo *while* fará 10 loops até chegar a 0 (zero).

Para visualizar melhor, você pode inserir no segundo *while* um comando *echo* para mostrar o valor da variável *y*.

Outro fato interessante e proposital foi o uso da expressão *let* no primeiro *while* e de *(())* no segundo, pois é indiferente utilizar um ou outro. Anteriormente, falamos a respeito disso: o *let* e os parentes têm o mesmo significado.

Neste próximo exemplo, verificaremos a entrada de um valor nulo como pré-requisito para execução do *while*:

Veja o exemplo abaixo:

```
#!/bin/bash
RESPONSE=
while [ -z "$RESPONSE" ]; do
    echo "Entre com o nome do diretório "
    read RESPONSE
    if [ ! -d "$RESPONSE" ]; then
        echo "ERROR, diretório não existente neste
path"
        RESPONSE=
    fi
done
```

Igualamos a variável *RESPONSE* a um valor nulo. O *while* vai verificar se esta variável tem valor nulo -z. Ele continuará executando enquanto o valor *RESPONSE* não for nulo, por isso colocamos no *if*, novamente, um valor nulo, para que o *while* ainda possa continuar. No *if*, simplesmente verificamos a existência de um diretório que vai contido também em *RESPONSE*, depois do comando *read*, que é responsável por gravar o que for digitado na variável.

Não é muito difícil! Poderíamos implementar para que ele procure num path especificado. Dessa forma, se ele não encontrasse o diretório, ele solicitaria o path, mas isso quem vai fazer é você.

Dica: Use o comando *read* para gravar o path em outra variável. Com os comandos normais *cde* *pwd*, você pode verificar em que diretório está.



Instrução Until

Uma outra forma de usarmos loop, é com o comando *until*, que trabalha de forma similar ao *while*, veja a sintaxe abaixo:

```
until [condição ]; do
    comando
    comando
    comando
    ...
done
```

Diferente do *while*, que faz enquanto a condição não for verdadeira, o *until* faz até que condição seja verdadeira. Abaixo, trarei um exemplo simples do uso do *until*:

Exemplo:

```
#!/bin/bash
X=1
until [ "SX" -gt 10 ]; do
    echo $X
    let X=X+1
done
```

Como podemos verificar, seu uso é muito simples, não requerendo muito mais estudo sobre o comando.

Faremos mais alguns exemplos somente para ilustrar seu uso com variáveis numéricas como strings.




```
#!/bin/bash
RES="sim"
until [ "$RES" == "não" ]; do
    echo Digite sim ou não para continuar
    read RES
done
```

No exemplo acima, enquanto não for digitado *não*, ele ficará em loop.

```
#!/bin/bash
until (( $# == 0 ))
do
    if [ -d $1 ]; do
        echo "O conteúdo de $1 é "
        ls $1
    fi
    shift
    Echo "Existem $#item na linha de comando"
done
```



O que ocorre neste script é muito interessante. Ele será executado até a variável `$#` não conter nenhum parâmetro. Vamos lembrar.

-  `$1` - É referente ao primeiro argumento na linha comando. Logicamente, ele ignorará o próprio nome do script.
-  `$2` - Diz respeito ao segundo argumento.
-  `$*` - Guarda o valor de todos os argumentos. Se na linha de comando estiver escrito *amarelo azul*, seria `$1 = Amarelo`, `$2 = azul` e `$* = amarelo azul`. Por último, o `$#`, que conta a quantidade de argumentos. No caso acima são dois, então `$# = 2`.

Voltando ao script, o *until* só parará quando `$#` for igual 0.

No entanto, temos o comando *shift*. Este comando atribuirá os valores de `$1`, `$2`, `$n`, desconsiderando o primeiro, neste caso. Suponhamos que foram colocados os seguintes argumentos na linha de comando: *banana laranja morango*, (no primeiro loop, `$# = 3`). O *shift* desconsiderará o valor *banana* e reorganizará os `$n` a partir de *laranja*, então, teremos *laranja morango*. Neste caso, `$# = 2`, o *shift* retirará novamente o valor *laranja* e deixará somente o argumento *morango*. No caso de `$# = 1`, no próximo *shift* não teremos argumentos na linha comando e o *until* irá parar.

O *shift*, por default, reordena o valor a partir do segundo item. O padrão dele é 1, ou seja, retira somente um argumento. Se colocássemos *shift 2*, ele reordenaria a partir de *morango*, fazendo apenas um loop.

Podemos, também, alterar a sequência do loop. Talvez não queiramos que ele execute totalmente, a não ser que satisfaça um item. Poderíamos usar funções, mas veremos isso adiante.

No caso de querermos parar e iniciar uma sequência, usamos o *continue* e o *break*.

O *continue* interrompe o loop atual e volta para primeira sequência de comandos. Não ficou muito claro? Se tivermos, por exemplo, um *while*, dentro dele uma instrução *if* e, nela, colocamos o *continue*. Por default, vale 1, então ele retornará à instrução inicial do *while*. Se usássemos *continue 2*, ele procuraria outro *while* acima deste ou outra instrução, como *for* ou *until*. Já o *break*, pára a instrução e pula para o primeiro *done* que encontrar. Se usarmos *break 2*, ele saltará do local que está diretamente para o segundo *done* que encontrar. Vamos para mais um exemplo, que com certeza deixará tudo mais claro:



```
#!/bin/bash
while true; do
    echo "Digite um nome de arquivo "
    read arquivo
    if [ ! -f $arquivo ]; then
        echo "$arquivo não é um arquivo válido"
        continue
    fi
    echo "Arquivo encontrado"
    cat $arquivo
    break
done
```

Este script ilustra bem o uso do *continue* e do *break*. Enquanto não digitado um valor válido para o *if*, ele retornará para o início do *while*. O que seria um valor válido? No nosso *testa ([! -f \$arquivo])*, estamos verificando se o valor passado para variável *arquivo* através do *read* é mesmo um arquivo, pois poderia ser um diretório, etc.

Quando realmente digitamos um nome de arquivo válido, nem entraremos no *if*, pois ele só entra em execução se o valor de arquivo não for um arquivo. Partimos para a instrução após o *if*, onde encontramos o *break*, que por sua vez não continua o loop, simplesmente vai para o *done* da instrução e sai dela.

Com isso, terminamos de falar a respeito dos loops e iremos ver um outro item muito interessante: as funções.

Funções

Muito fácil! Para quem já conhece o conceito de módulo, fica mais fácil ainda.

As funções nos ajudam a organizar melhor e poupar-nos do trabalho de redigitar seqüências de comandos quando usados várias vezes.

Sua sintaxe é bem simples:

```
function nome_da_função
{
    instruções
        comandos
        comandos
    instruções
    comandos
}
```



Ou:

```
nome_da_funcao  ()
{

    instrução
    instrução
    comandos
}
```

Nesta sintaxe, quis deixar bem claro que você pode usar instruções, tais como *while*, *for*, etc. Mas também pode utilizar comandos diretamente, como o *echo*, por exemplo.

Abaixo, deixei um exemplo de função que procura por um nome e um código dentro de um arquivo. Foi incrementado para que solicite o nome e se é necessário continuar a procura por outros nomes dentro do arquivo. O arquivo, neste exemplo, se encontra no diretório */home* com o nome *arquivo.txt*, os campos são separados por “;” (ponto-e-vírgula), como em um arquivo com extensão *csv*:

Andre;2001

Caio;2454

Maria;4547

...

...

Script de exemplo:

```
#!/bin/bash
```

```
#Esta é a primeira Função
function Primeira_funcao
{
    while [ "RES" != "não" ]; do
        clear
        echo
```

Digite um nome a ser procurado no arquivo:

```
read NOME
cat /home/arquivo.txt | awk -F";" '{ print $1,
"codigo: $2"}' | grep -i $NOME
```

```
Segunda_funcao
done
```



```
}

# Esta é a segunda Função
Segunda_funcao ()
{
    echo Deseja continuar procurando nomes
    RES="sim não"
    select opt in $RES; do
        if [ $opt = "sim" ]; then
            RES="sim"
            Primeira_funcao
        elif [ $opt = "não" ]; then
            RES="não"
            Primeira_funcao
        else
            echo Opção inválida
            sleep 1
            clear
            continue
        fi
    exit
done
}
```

Primeira_funcao

Vamos analisar as funções.

Na primeira função, fazemos um *while*, que executará enquanto o valor de *RES* for diferente de *não*. Usamos um conjunto de comandos entre *cat* para mostrar todo conteúdo do arquivo, *awk* para separar os campos do arquivo e, enfim, *grep -i* (torna o *grep*, não *case sensitive*, ou seja, é indiferente se forem maiúsculas ou minúsculas) para procurar nos campos o valor passado para a variável *NOME* pelo usuário. Feito todo o processo, ele chama a segunda função, que é um pouco maior, mas em compensação, mais simples de ser feita. Fizemos um menu para usuário, onde é perguntado se o mesmo quer continuar procurando nomes no arquivo ou deixar o programa. Usamos o *select* e dentro dele a instrução *if*. Nas duas primeiras etapas do *if*, a instrução simplesmente seta o valor da variável *RES* como *sim* ou *não* e chama a primeira função. Note que, se o valor for *sim*, entraremos novamente no *while*, onde ele pedirá um nome a ser procurado. Se o valor for *não*, *while* não será executado e sairemos do script. Já na última etapa do *if*, caso não seja escolhido 1 (sim) ou 2 (não), ele informará que o valor é inválido.



O comando *sleep* fará o script parar por um segundo, o *clear* limpará a tela, o *continue* voltará para o começo da função e, enfim, o *exit* para sair desta função.

Ainda existe a possibilidade de usar scripts já prontos em outros arquivos, facilitando ainda mais execuções de vários scripts. Para chamar um arquivo de script dentro de outro script, usa-se *bash nome_do_arquivo_shell* ou somente */path/./arquivo*.

Exit e Return

O comando *exit*, usado dentro Shell, é utilizado para terminar a execução do programa e ajustar o código de saída para *0* (*true*) ou outra saída não padrão (*false*). Caso não seja passado nenhum código de retorno, irá ajustar para o último comando realizado dentro do script. Podemos ajustar para qualquer código de saída da seguinte forma:

 *exit n* - Onde *n* é o número ajustado para o código de saída.

Código de saída?

Não acredito, já esqueceu ???!

Sim. O código que é gravado na variável ? (interrogação).

Usamos para verificar se uma sentença era falsa ou verdadeira:

```
echo $?
```

Caso verdadeiro, o valor de ? é *0* (zero), caso contrário *1* (um) ou qualquer outro número.

A outra forma de passar um valor de retorno para um certa função é usando o comando *return*.

Veja o exemplo abaixo:

```
#!/bin/bash
echo $?
return 51
echo $?
```

Com isso, teremos o valor *51* como retorno da saída do script, gravado na variável ?.

Comando trap

Muitas vezes são passados sinais para o script.



Quando teclamos *CTRL + C*, ele envia um sinal para o Shell terminar esse processo que está rodando, fazendo, assim, a finalização do script.

Imagine que você fez um script que grava informações em um arquivo temporário e, quando solicitado sair do script, ele finalmente grava todo o conteúdo em um outro arquivo onde é realmente considerado o banco de dados. O que aconteceria se digitássemos *CRTL + C* no meio da execução do script? Ele simplesmente não apagaria o arquivo temporário e não gravaria as informações no arquivo principal.

Por isso, existe o comando *trap*.

Ele captura esse sinais e executa uma instrução pré-estabelecida no script. Sua sintaxe está logo abaixo:


```
trap comandos sinais
```

Exemplo:

```
trap ``2
```

Este comando acima simplesmente ignora o sinal *INT* (^C). Portanto, se alguém digitar *CRTL + C*, nada acontecerá.

Mas podemos fazer melhor. Baseado no esquema dito anteriormente do arquivo temporário e do banco de dados, poderíamos enviar uma mensagem ao usuário. Veja abaixo:

 *trap* ' Você digitou *CRTL+C*, só é permitido sair do programa depois que foram atualizados o arquivo do bando de dados' 2

Neste formato acima, sempre que o usuário tentar sair com o sinal ^C, terá esta mensagem no console.

Abaixo, veja os sinais mais utilizados:

0	Saída do Shell.
1	Hangup.
2	INT (^C).
3	Quit (Sair).
4	Kill.
15	Term.

Com este comando, com certeza não iremos correr o risco de um usuário danificar uma base de dados com teclas problemas.



Getopts

Este comando é usado para verificarmos argumentos passados através das linhas de comando.

Tenho certeza que o leitor já viu algo do tipo *comando -u, usuário -p*, etc. Justamente isso que o *getopts* faz o script. Vamos usar aquele mesmo arquivo que usamos para o item *função*, que tem um formato “nome;código”, localizado em */home - arquivo.txt*.

```
#!/bin/bash
nome=Andre
codigo=444
isdef=0
x=1

while [ $X = 1 ]; do
if [ $# -lt 2 ]; then
    echo Faltam argumentos
    exit 10
else
    X=0
fi
done

while getopts n:c:d opt; do
    case $opt in
        n) nome="$OPTARG"
           ;;
        c) cod="$OPTARG"
           ;;
        d) isdef=1
           ;;
    esac
done

if [ $isdef -eq 0 ]; then
    echo "Nome = $nome e código = $cod"
    echo "$nome;$cod" >> /root/script/arquivo
    cat /root/script/arquivo
else
    exit
fi
```

Vamos comentar, agora, a respeito de todo procedimento tomado neste script.



No item *while [\$X = 1]*, é verificada a quantidade de parâmetros que é passada para o comando. Se for menor que 2, ele dá uma mensagem de erro e sai do script dando um código de retorno de valor 10.

Já no segundo *while*, é onde realmente ele usa o comando *getopts*, onde será passado para o comando os valores *n nome* e *c código*. A próxima instrução gravará os valores de *nome* e *código* no *arquivo.txt*.

Para usá-lo, deve-se seguir o padrão:

```
script -n nome -c código
```

Caso algo esteja faltando, ele apontará o erro. Seria ideal implementar o código, que em vez de notificar um erro, mostra suas opções e como deve ser usado. É um tipo de *help* para ajudar o usuário.

Conhecendo o AWK

Já foi dito anteriormente que o *awk* não é apenas um comando, mas uma linguagem de programação muito usada para se trabalhar com arquivos. Mostrarei o básico desta linguagem, dando noções essenciais para uso desta ferramenta. Não será totalmente aborto, pois existem livros que falam só do *awk* para se ter um noção da extensão do assunto.

A linguagem *awk* foi inventada em 1977, seus criadores foram Aho, Kernighan e Weinberger. Com a inicial de cada nome deram origem ao programa.

A sintaxe do AWK está logo abaixo:

```
awk '{ faça algo }' arquivo
```

Onde *faça algo* pode ser usando com inúmeras funções, como imprimir, procurar ,etc.

Abaixo, alguns exemplos:

```
cat /etc/passwd | awk -F":" '{print $1,$3}'
```

Este comando imprime a primeira e a terceira coluna do arquivo */etc/passwd*, que é onde ficam os nomes dos usuários cadastrados nesta máquina. A primeira coluna é referente ao nome e a terceira ao código.

```
awk '/#/ { print $2, $3 }' /etc smb.conf
```

No exemplo acima, é verificado no arquivo */etc/smb.conf* (Arquivo de configuração do Samba) todas as linhas que contenham o caractere



#. Tão logo ele ache, já imprimirá na tela os valores da primeira e da segunda variável após o caractere.

É possível trabalhar com script, da mesma forma que o próprio Shell, ou trabalhar com arquivos para entrada de comandos e entrada de dados. Daremos um pequeno exemplo para ficar mais claro:

Crie um arquivo chamado *Hello* e dentro do arquivo coloque a seguinte linha:

```
Hello, Word
```

Agora, execute o comando:

```
awk '{print}' Hello
```

O resultado:

```
$Hello, Word
```

O que fizemos foi pedir para o *awk* imprimir o arquivo *Hello*. Simples, não? Mas vamos para um exemplo mais complexo:

Faremos um arquivo da seguinte forma: nome e sobrenome, código e horas.

Estamos simulando um arquivo de horas trabalhadas mês:

```
Andre Stato;2001;186
```


```
Carlos Chagas;555;200
```

```
San Goku;1000;220
```

```
Elvis Presley;0001;120
```

Copie este trecho acima e salve com o nome de *horas*. Agora, criaremos outro arquivo e chamaremos de *pouca_hora*. Seu conteúdo será o seguinte:

```
$3 < 150 {
```

```
     OFS="      " # Define qual será o separador de campos
    na saída no nosso caso # Espaço
```

```
print $1 , $3
```

```
}
```

Agora, só resta o comando a ser dado:

```
awk -F";" -f pouca_hora horas
```

O resultado será :



Elvis Presley 120

Podemos trabalhar com variáveis no linguagem *awk*, um pouco diferente do padrão, mas não é difícil.

Para somarmos a quantidade de linha, usamos algo como *variável++*. Se quisermos somar um campo numérico, podemos usar algo do tipo *numero=numero + \$X*, onde *\$X* é o campo propriamente dito (\$1, \$2) dentro do arquivo.

Vamos fazer um script para melhor exemplificar.

Iremos ainda trabalhar com o arquivo *horas*.

Veja abaixo o script, crie-o e salve com o nome *contagem*, por exemplo:

```
$3 < 150
```

```
{
usuarios++ # mostrará a quantidade de usuários que o
awk gerou da
           # solicitação de $3 < 150, campo $3 menor que
150
}
{
campo2=campo2 + $2
}
{
campo3=campo3 + $3
}
END {
    if (usuarios > 0) {
        print "Temos ",usuarios," com o Campo 3 menor
que 150"
    }
    print "O campo 2 tem um total de ",campo2
    print "O campo 3 tem um total de ",campo3
}
```

Agora, execute o comando:

```
awk -F":" -f contagem horas
```

O resultado será algo do tipo:

Temos 1 usuário com o Campo 3 menor que 150

O campo 2 tem um total de 726

O campo 3 tem um total de 3557



No primeiro item, está o total de usuários que se enquadram na requisição $\$3 < 150$. No segundo, ele somou o segundo campo e no terceiro somou todos os valores de cada linha do campo $\$3$.

O *awk* é muito extenso. Vou estar mostrando algumas outras formas de trabalhar com ele.

Você pode procurar uma sequência ou um caractere específico dentro de um arquivo. Vamos usar o arquivo */etc/hosts*, supondo que existe pelo menos uma linha comentada inteira, ou seja, com o símbolo de sustenido no início da linha (#) e uma com o sinal de sustenido no meio da linha, comentando apenas parte dela.

Executemos :

```
awk '/#/ {print
```

“Esta linha está comentada no início”

```
} /^#/ {print
```

“Esta linha está comentada no meio”

```
} ' /etc/hosts
```

Não é muito útil sabermos somente isso. Precisamos saber qual linha do arquivo está comentada e qual o seu conteúdo.

Podemos usar duas variáveis para isso. Essas variáveis são fixas do *awk*. Para mostrar o número da linha, usaremos o *NR* e para verificar o seu conteúdo, usamos o *\$0*.

Dessa forma, o comando ficará assim:

```
awk '/#/ {print "A linha" ,  
NR, "está comentada no início,"  
    conteúdo: "$0"} /^#/ {print "A linha",  
NR, " está comentada no meio conteúdo : ", $0}' /etc/  
hosts
```

Com isso, ele dará a seguinte resposta:

```
A linha 4 está comentada no início, conteúdo : #  
comentário  
A linha 7 esta  comentada no meio , conteúdo :  
192.168.200.2 Localhost #Maquina local  
...  
...
```

Abaixo, observe algumas das variáveis usadas no *awk*:



FS	Separador de campos na entrada.
OFS	Separador de campos na saída.
ORS	Separador de registros de saída.
ARGC, ARGV	Descritores de parâmetros de chamada.
ENVIRON	Vetor contendo as variáveis de ambiente.
ERRNO	Número do erro ocorrido na última chamada ao sistema.
NF	Número de campos na linha corrente.
NR	Número do registro corrente.

Com o *awk*, podemos trabalhar com expressões regulares do Shell script.

Crie um arquivo e copie o conteúdo abaixo para ele:

```
/^$/ {print "Esta linha está em branco"}
/[0-9]+/ {print "Este é um numero"}
/[A-Za-z]+/ {print "Esta é uma string"}
```

Salve o arquivo e digite o comando:

```
awk -f arquivo
```

Quando digitar uma letra, ele informará:

```
Está é uma string
```

Quando digitar um número:

```
Este é um número
```

E quando não digitar nada:

```
Esta linha está em branco
```

Outra boa característica do *awk* é que se pode usar funções numéricas.

Vamos a um exemplo para ficar mais claro:

Criemos uma arquivo chamado *notas*, que terá como conteúdo as notas de alguns alunos:

```
Andre 85 79 65 89
```

```
Pedro 85 75 64 80
```

```
João 90 76 84 57
```

Agora, vamos criar um arquivo para trabalhar em cima do arquivo *notas* e obtermos a média de cada aluno:

```
{ total = $2 + $3 + $4 + $5
```



```
avg = total / 5
print $1, avg }
```

Salve o arquivo como o nome *avg*.

Agora, é só chamar os arquivos com o comando *awk*:

```
awk -f avg notas
```

Resultado:

Andre 79,5

Pedro 76

João 76,75

Abaixo, algumas das expressões que podem ser usadas com o *awk*:

^	Potência.
*	Multiplicação.
/	Divisão.
%	Módulo.
+	Soma.
-	Subtração.

Mas além das expressões do Shell, o *awk* possui suas próprias funções numéricas. Digite os exemplos abaixo no Shell:

```
awk 'BEGIN {print 100/3}'
33,3333
```

```
awk 'BEGIN {print int(100/3)}'
33
```

Nestes dois exemplos, é mostrado o uso do *awk* com expressões e funções. O primeiro exemplo usa uma expressão do Shell e o segundo é uma função. O *int* despreza as casas decimais.

```
awk 'BEGIN {print rand()}'
0,08482
```

Esta segunda função, o *rand*, gera um número aleatório entre 0 e 1.

Abaixo, estão algumas das funções numéricas do *awk*:



int(x)	Executa a operação em x e elimina a parte decimal.
sin(x)	Seno de x (x em radianos).
cos(x)	Cosseno de x (x em radianos).
sqrt(x)	Raiz quadrada de x.
exp(x)	Exponencial de x.
log(x)	Logaritmo de x.
atan2(x,y)	Arco tangente de y e x em radianos.
rand ()	Gera um número aleatório entre 0 e 1.
srand ()	Inicializará o gerador de número aleatório.

Agora, falaremos das funções de string. Supomos que tenhamos uma string em variável e desejamos retirar dela alguma informação contida, ou pelo menos saber em que parte da string uma ocorrência acontece. Considere os exemplos abaixo:

```
awk 'BEGIN      '{print index("Valor Toral da
string","Total")}'
7
```

```
awk 'BEGIN '{print substr("011 4444-4444",1,3"
011
```

Essas duas funções podem ser de grande utilidade para uso com arquivos agindo como banco de dados. Existem muitas outras funções que descrevo abaixo:



<code>gsub(expr,valor, string)</code>	Procura por todas as ocorrências de <code>expr</code> dentro da <code>string</code> , que pode ser uma variável ou a própria <code>string</code> e substitui por <code>valor</code> .
<code>index(string,valor)</code>	Informa a posição de <code>valor</code> dentro da <code>string</code> .
<code>length(string)</code>	Informa o comprimento da <code>string</code> .
<code>math(string,expr)</code>	Verifica se a expressão <code>exp</code> ocorre dentro de <code>string</code> .
<code>split(string,array,[separador])</code>	Separa os componentes da <code>string</code> em arrays com uma opção de separador opcional. Retorna o número de elementos.
<code>sprintf(formato, expressão)</code>	Trabalha de forma análoga ao <code>printf</code> , só que um pouco melhor. Veja no <code>help</code> do <code>sprintf</code> .
<code>sub(expr,valor,string)</code>	Trabalha de forma similar ao <code>gsub</code> , só que ao invés de substituir todas, substitui apenas a primeira ocorrência.
<code>substr(string,início,tamanho)</code>	Retorna uma substring dentro da <code>string</code> , pegando o caractere especificado em início até o seu tamanho (veja o exemplo exemplo acima).
<code>tolower(string)</code>	Converte a <code>string</code> para minúscula.
<code>toupper(string)</code>	Converte a <code>string</code> para maiúscula.

Ainda existe a possibilidade de se trabalhar com array, não com o comando *splt* mas de forma similar ao Pascal, dando o nome array e seu conteúdo.

Vamos criar um arquivo chamado *array.awk*, seu conteúdo será o seguinte:

```
array[1]="Andre"
array[2]="Stato"
{print array[1],array[2]}
```

Agora digite no Shell:

```
awk -f array.awk
```

```
#Andre Stato
```

Logicamente, isso não é um manual do *awk*, mas abrange muita coisa sobre a linguagem. Era para ser apenas uma leve introdução, mas acabei me empolgando um pouco. Um ótimo livro sobre o *awk* é o *Effective awk Programming, 3rd Edition Text Processing and Pattern Matching*, da Editora Oreilly.



Uma Introdução ao SED

O SED é um editor de texto que não se utiliza da interação direta no arquivo, mas de um comando passando diretivas, ou seja, não é necessário editar o texto em si, basta passar ao SED que se deseja editar, substituir ou remover dentro de um texto, mandando o resultado para uma saída padrão.

A saída padrão do SED é o próprio Shell, ou seja, ele pegará o arquivo, executará as instruções e imprimirá na tela todo o resultado.

Quando as tarefas forem simples, usar o SED torna tudo muito mais rápido, pois em uma linha você pode fazer todas as alterações que queira em um texto inteiro. Mas tratando-se de tarefas mais complexas, logicamente pode-se usar um editor como o VI.

O SED funciona como um filtro que vai lendo uma linha, executa as tarefas que foram solicitadas, vai para próxima linha e executa novamente até o final do arquivo.

Abaixo, se encontra sua sintaxe:

```
sed [opções] regras [arquivo]
```

Onde as regras são:

```
endereço1 [, endereço2]] comando [argumento]
```

Vamos a um exemplo simples. Temos um arquivo chamado *alunos*.

Coloque o seguinte em *alunos*:

60% dos alunos desta escola repetiram.

Dos alunos que repetiram, 60% eram do sexo feminino.

A escola tem um total de 500 alunos.

Agora, vamos alterar *alunos* por *estudantes*.

```
sed -e 's/alunos/estudantes/g' alunos.txt
```

O que fizemos?

O parâmetro *-e*, serve para passar a instrução. Por exemplo, se quisermos alterar também os 60% para 70%, colocaríamos ao final da instrução “.../estudantes/g” -e ‘s/60/100/g’alunos.txt”, ou seja, ele serve para separar as instruções do SED. Já o parâmetro *s*, funciona como um tipo de *search* ou *replace*, localiza e substitui. Já no final do comando, a letra *g* informa para o SED que é para alterar todas as



ocorrências que encontrar no arquivo, caso contrário, ele alteraria só a primeira ocorrência.

O SED, por padrão, imprime a saída do arquivo na tela, ou seja, naquele arquivo anterior, ele imprimiria na tela todo o texto já alterado. Isso pode ser evitado com o parâmetro `-n`. Vamos fazer um teste. Digite:

```
sed 'p' alunos.txt
```

60% dos alunos desta escola repetiram.

60% dos alunos desta escola repetiram.

Dos alunos que repetiram, 60% eram do sexo feminino.

Dos alunos que repetiram, 60% eram do sexo feminino.

A escola tem um total de 500 alunos.

A escola tem um total de 500 alunos.

O `p`, comando do SED, faz com que a saída imprima duas vezes cada linha. Mas agora use:

```
sed -n 'p' alunos.txt
```

Você verá que será impressa apenas uma linha, pois o parâmetro `-n` faz com que o SED suprima da saída padrão a duplicação das linhas.

Há várias opções de uso do SED, não somente a troca de palavras, mas a própria deleção. Mas preste atenção, pois esse parâmetro deleta toda a linha:

```
sed '/60/d'alunos.txt
```

A escola tem um total de 500 alunos.

Com isso, ele apagará as duas primeiras linhas que contém a string 60.

Outra forma de deletar é apagando diretamente o arquivo dando o número da linha:

```
sed '3d'aluno.txt
```

Caso quiséssemos apagar só a palavra, poderíamos usar algo como:

```
sed 's/alunos//g'alunos.txt
```

60% dos desta escola repetiram.

Dos que repetiram, 60% eram do sexo feminino.

A escola tem um total de 500.



Podemos dar intervalos de linhas:

```
sed '10,20d' aluno.txt
```

```
sed '1,/feminino/d' aluno.txt
```

No primeiro caso, ele deletaria da linha 10 à 20. No segundo exemplo, ele deletaria a primeira linha até encontrar a palavra feminino.

Outra forma de referência também é usando sinais, por exemplo da ocorrência da palavra feminino mais duas linhas:

```
sed '/feminino/,+3d'aluno.txt
```

Podemos utilizar a inversão, utilizando o sinal “!” (exclamação). Alguém lembra do comando *head*?

É usado para ler um número de linhas de um arquivo.

```
head -n20 arquivo.txt
```

Poderíamos, facilmente, utilizar o SED:

```
sed -n '1,20p' arquivo.txt
```

Com inversão, podemos fazer com que o SED execute todos os comandos, menos para o que foi designado. No caso acima, ele leria todas as linhas, menos as de 1 à 20.

```
sed -n '1,20!p'arquivo.txt
```

Nota-se que, em todos os casos, usamos a “,” (vírgula) como separador. Isso mesmo! A vírgula é o que informa ao SED que estamos fazendo um intervalo.

Em geral, queremos alterar um arquivo definitivamente, pois podem ocorrer alguns problemas.

O caro leitor iria querer fazer algo como:

```
sed 's/alunos//g'alunos.txt > alunos.txt
```

Não, não. Realmente não iria funcionar, pois quando tentar gravar o arquivo *alunos.txt* nele mesmo, o Shell, de antemão, já abrirá o arquivo e o truncará. Depois disso, antes mesmo de executar o SED, o arquivo já estará vazio e SED nada executará. Para finalizar, ainda terá perdido todo o conteúdo do arquivo. O mais seguro é gravarmos em um outro arquivo e depois sobrescrevê-lo:

```
sed 's/alunos//g'alunos.txt > alunos_novo.txt  
mv alunos_novos.txt alunos.txt
```



Um fato interessante é quanto ao uso dos campos *endereço* (entrada, saída) separados. Normalmente, usamos “/” (barra). O que aconteceria se quiséssemos alterar, por exemplo, um endereço de diretório, como `/usr/local/bin` para somente `/usr/bin`? Temos um problema, pois o SED entenderia a / (barra) como um separador de endereços. Temos duas formas de resolver. Primeiro a mais complicada, que é usar um parâmetro antes da barra, que é justamente um escape, para que o comando não entenda como um parâmetro e simplesmente como uma barra. Esse escape é a barra invertida.

Ficaria dessa forma o comando (meio sinistro):

```
sed 's/\ /usr\/local\/bin\/usr\/bin/g' arquivo.txt
```

Um pouco complicado, não?! Mas podemos usar a segunda opção que substitui o separador por outro símbolo, como vírgula, sustenido, etc. Veja como fica mais fácil agora:

```
sed 's#/usr/local/bin#/usr/bin#g' arquivo.txt
```

O importante é que, de ambas as formas, o SED vai funcionar, ficando a gosto do usuário a forma como será utilizado.

É possível parar uma instrução SED com a letra *q*:

```
sed '20q'
```

Com isso, podemos usá-lo de forma similar ao *head*, que lê um número especificado de linhas.

O SED é quase uma linguagem de programação, tendo em vista que não há variáveis e instruções loops. Mas, mesmo assim, ele é muito extenso e poderíamos, com certeza, fazer um livro somente falando sobre o assunto.

Mas, de qualquer forma, vamos ver mais algumas características do SED e alguns exemplos comentados.

Vou começar citando uma característica muito interessante do SED, que é quebrar as palavras em novas linhas com o parâmetro adicional, que é a letra *n*.

```
sed 's/ /\n/g' arquivo.txt
```

Neste caso acima, podemos verificar que é solicitado ao SED que procure todos os itens com espaço (/ /) e troque por uma quebra de linha `/n` em todas as ocorrências.



Como o *awk*, o SED também pode ler arquivos onde contenham as instruções para o comando *sed*, evitando, assim, aqueles onde serão passados muitos parâmetros.

Vamos para um exemplo mais prático:

Primeiro, vamos criar um arquivo muitas linhas, em torno de 25. Como poderíamos criar um arquivo de forma que escrevamos as 25 linhas? Iremos inserir em um arquivo 25 linhas, cada linha terá apenas o seu respectivo número:

```
for i in $(seq 25); do echo $i >> arquivonovo.txt ;  
done;
```

Feito isso, podemos, agora, trabalhar realmente com o SED sobre esse arquivo:

```
cat arquivonovo.txt
```

Agora, vamos criar um arquivo chamado *execsed*, com o seguinte conteúdo:

#Troca o valor de 2:

```
s/2/50
```

apaga a linha 10:

```
10d
```

apaga a linha 5,6,7:

```
5,7d
```

#Para de ler o arquivo na linha 20:

```
20q
```

Salve os arquivos e execute o seguinte comando no Shell:

```
sed -f execsed arquivonovo.txt
```

Como você pode ver, ele realmente fez o que queríamos mas podemos fazer melhor, tornando-o um executável.

Na primeira linha de *arquivonovo.txt*, adicione a seguinte linha:

```
#!/bin/sed -f
```

Salve o arquivo,e digite:

```
chmod 755 execsed
```

Pronto! É só chamar o comando:

```
./execsed arquivonovo.txt
```



Ou:

```
cat arquivonovo.txt | ./excsed
```

Ambos terão o mesmo efeito.

Vamos parar por aqui com o SED, mais informações podem ser obtidas em:

```
man sed
```

```
http://verde666.org/sed/sed-HOWTO/sed-HOWTO.html
```

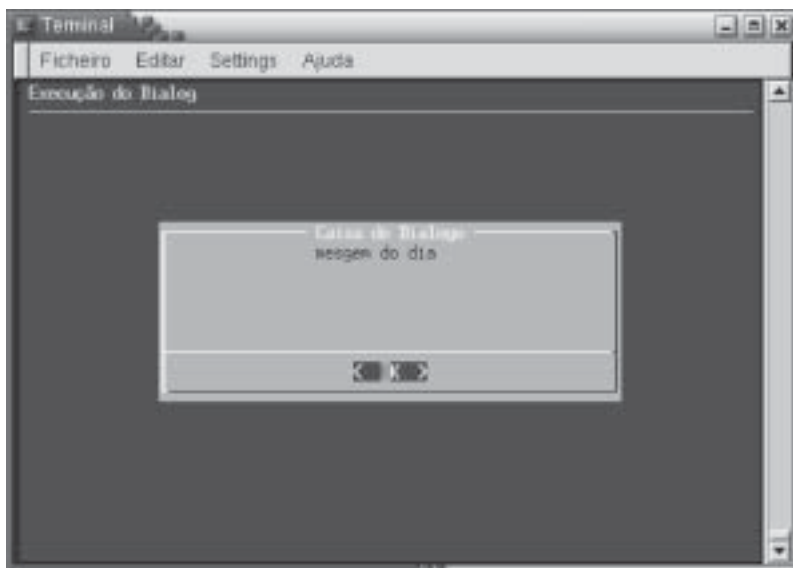
```
http://www.gnu.org/manual/sed-3.02/sed.html
```

E para finalizar, temos um último recurso, que é o *dialog*.

Apresentando o Dialog

O *dialog* é utilizado para criar scripts com menus, apresentando telas “gráficas” (em ANSI, tipo o estilo do Clipper que roda sobre o DOS).

Veja abaixo uma figura na tela:



Caixa de menu Dialog

Vamos começar com um exemplo bem simples, chamaremos de menu:

```
#!/bin/bash
```

```
dialog --title "Caixa de Dialogo" --backtitle "Execução do
```




```
Dialog"\
-msgbox "mensagem do dia" 9 50
```

Como qualquer outro arquivo script, devemos alterar os tipos de permissões.

Salve o arquivo, mude suas permissões para poder executá-lo (chmod 755), e chame-o no Shell:

```
./menu
```

O que foi feito?

O item *title*, como o nome diz, imprime o título do *dialog*; *backtitle* imprime um *title* na caixa de diálogo, no caso, uma *messageBox*. O item *msgbox* abre uma caixa de diálogo com um botão OK e imprime a mensagem “Mensagem do dia”. Por último, o item *9 50* são as dimensões do *Box*.

A barra serve para que o comando possa continuar em outra linha, interrompendo o seu fluxo, de forma que as informações fiquem mais claras.

No *dialog*, podemos trabalhar de várias formas, como *msgbox*, *inputbox*, *infobox* e menus. Abaixo, uma pequena descrição de como funciona:

- - msgbox “Mensagem” altura largura
- - yesno “Mensagem” altura largura
- - infobox “Mensagem” altura largura
- - inputbox “Mensagem” altura largura
- [comandos]
- - textbox “Mensagem” altura largura
- - menu “Mensagem” altura largura
- menu altura largura tag
- item1
- item2

Abaixo, um exemplo de *yesno*. Para isso, crie um arquivo chamado *yesno* com o seguinte conteúdo:

```
#!/bin/bash
dialog \
- -title "Teste yes ou no" \
- -yesno "\nVocê quer lista o diretório /root" 7 50 \
```



```
valor=$?
case $valor in
0) ls /root;;
1) echo "você selecionou não" ;;
255) echo "Você teclou ESC" ;;
esac
```

Os traços em - *-title*, por exemplo, são dois hífen colados. A sintaxe tem que estar totalmente correta, pois caso contrário, o script não será executado. E não esqueça de alterar as permissões.

Podemos notar que foi usado uma instrução *case* dentro do *dialog*. Podemos trabalhar, juntamente com o *dialog*, tanto com instruções de loop, como com o próprio script Shell. Vejam que o próprio *dialog* faz parte do Shell script quando invocado sob o Shell (*/bin/shell*).

Agora, veremos outro exemplo com o *inputbox*:

```
#!/bin/bash
dialog \
- -title "Teste InputBox" \
- -backtitle "Titulo" \
- -inputbox "\nQual diretório você quer lista" 8 60 2>
/tmp/input.$$
valor=$?
comando= `cat /tmp/input.$$`
case $valor in
0) "ls $comando" ;;
1) echo "Cancelado" ;;
255) echo "Você teclou ESC" ;;
esac
```

No caso acima, o *inputbox* terá um campo editável, onde o usuário entrará com um valor, no caso um diretório a ser listado, que por sua vez jogará o conteúdo deste campo no arquivo *input.\$\$* que será gerado. Depois disso, ele verifica se foi escolhido *OK*, *Cancel* ou *ESC* para executar sua respectiva instrução dentro do *case*.

Enfim, iremos trabalhar diretamente com menus:

```
#!/bin/sh
dialog \
-title "Exemplo Menu" \
-backtitle "Menu de opções" \
-menu "Selecione a opção desejada" 13 50 3 \
Data/Hora "Mostra Data e hora" \
Arquivos "Lista o diretório do user" \
```



```
Calendario "Mostra um calendário" \  
sair "Sai deste menu" 2>/tmp/arquivo.tmp  
conteúdo='cat /tmp/arquivo.tmp'  
opt=$?  
case "$conteudo" in  
Ajuda) ./help ;;  
Data/Hora) date ;;  
Arquivos) ls $HOME ;;  
Calendario) ./cal ;;  
Sair) exit 0 ;;  
esac
```

Neste último exemplo, quando executado, abre-se um tela similar às anteriores (*inputbox*, *msgbox*, etc.). Mas, nesse caso, existe um menu onde o usuário irá navegar para escolher uma das opções do menu.

Existem outras funções dentro *dialog*, como *checklist*, *radiolist*, *yesno*, *infobox*, etc. Cabe agora a vocês aprofundar seus conhecimentos no *dialog*, pois o capítulo sobre programação terminará por aqui. Mas não fique triste, pois iniciaremos as configurações de rede e servidores, que provavelmente usará scripts para podermos administrar melhor nossos servidores. Sendo assim, ainda veremos algo sobre script.

Nosso próximo passo é a rede.



6

Redes

Introdução

Neste capítulo, falaremos sobre as instalações física e lógica das redes.

Mas antes de entrarmos propriamente nestes aspectos, veremos algumas noções básicas de rede, como protocolo, encapsulamento, endereços e máscara de rede, classes de IP, etc.

Não será abordado profundamente o protocolo TCP/IP (Transmission Cotrol Protocol/Internet Protocol), por exemplo, pois a intenção é dar ao usuário o mínimo necessário de conhecimento para que possa montar sua rede, seja ela pequena, como dois micros ponto-a-ponto, ou maior, com a utilização de hubs.

Também não será abordado roteamento com equipamentos específicos, como roteadores Cisco, e outros. Em outros capítulos, falaremos do **Linux** atuando com roteador de duas ou mais redes, em nível de software.

Tenho certeza que após ler este capítulo, o usuário estará apto a montar uma rede e verificar possíveis problemas com as conexões.

Noções Básicas de Rede

O que é uma rede?

Uma rede de computadores é caracterizada por dois ou mais micros que estão interligados de forma que possam trocar informações entre si, bem como compartilhar recursos de software ou hardware. Por exemplo, um programa de banco de dados como uma impressora.

A interligação física entre os computadores podem ser dar através de cabos de rede conectados à placa, linha telefônica, satélite, rádio, etc. Já a comunicação lógica, é realizada por um protocolo, que

nada mais do que um conjunto de regras que regula a circulação de dados na rede.

Existem vários protocolos de rede, como a Rede Novell, que tem como responsável pelo transporte o IPX, e o NPC, responsável pelo gerenciamento. Temos também o TCP/IP, onde TCP é responsável pelo transporte e o IP pelo gerenciamento.

Falaremos do TCP/IP, por ser o mais usado na rede de Internet. A Internet é uma rede mundial pública de comunicação de dados, com controle descentralizado, que utiliza como padrão o protocolo TCP/IP. O TCP é, ainda, o protocolo nativo do Unix, sendo assim do **Linux**, mas isso não quer dizer que não é possível comunicação com outros protocolos, existe a possibilidade de encapsulamento.

Imagine um cano grande por onde passa várias informações, OK? Esse cano é o TCP.

Temos outros protocolos responsáveis por alguns serviços importantes para Internet, como o HTTP (protocolos para páginas [www](#)), SMTP e POP3 (para envio e recebimento de e-mail). Esses protocolos são canos menores que estão dentro do cano grande, que é o TCP. Isto é, o TCP tanto encapsula estes protocolos, como também é encapsulado em outros casos por outros protocolos.

Veja a figura abaixo para compreender melhor:

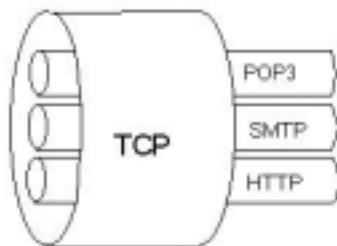
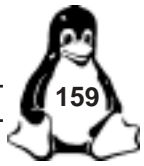


Figura 6.1 - Protocolo TCP/IP

O protocolo TCP/IP é muito extenso. Sua arquitetura é dividida em funções de sistema em estruturas de camadas, como Aplicação, Transporte, Inter-rede e Rede, que por sua vez tem suas responsabilidades dentro do protocolo, dando assunto para um livro inteiro. Dessa forma vamos passar por cima da teoria e ir direto ao foco: a utilização.



Endereço IP

Em uma rede TCP/IP, cada interface de rede (placa de rede), também chamada de host, tem um identificador único chamado de endereço IP. É como se fosse um endereço, ou telefone, único para cada interface de rede.

Este endereço consiste na formação de 4 octetos de 8 bits, totalizando 32 bits (4 bytes), onde cada octeto é separado por ponto, sendo que uma parte do conjunto é destinada a identificação da rede e a outra destinada a identificação do host.

Veja o exemplo abaixo:

11000000 . 10101000.11001000.00000001

Normalmente, os bits são representados em decimais, no caso acima, 192.168.200.1.

Alguém lembra como se converte bits para decimais?

Vamos relembrar.

Cada octeto, logicamente, tem 8 números entre 1 e 0, onde um quer dizer verdadeiro e 0 falso.

Partindo da esquerda para direita, temos:

1	1	1	1	1	1	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

Então, dessa forma, onde houver o valor 1 no octeto, soma-se seu valor decimal, onde houver 0, a parte decimal será desconsiderada.

Partindo deste princípio, temos como valor mínimo 1 (00000000) e valor máximo 255 (11111111). No exemplo de rede acima, 192.168.200.1, para calcular, usaríamos:

192

1	1	0	0	0	0	0	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	0	0	0	0	0	0



168

1	0	1	0	1	0	0	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	0	32	0	8	0	0	0

200

1	1	0	0	1	0	0	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	0	0	8	0	0	0

1

0	0	0	0	0	0	0	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
0	0	0	0	0	0	0	1

Neste momento, não é tão importante este cálculo, mas quando utilizarmos máscara para sub-rede, será de grande valia. Antes de entrarmos neste aspecto, devemos verificar outro item do endereço IP, que são as classes. Elas são divididas para melhor acomodação de redes muito grandes ou muito pequenas.

0 bits 7 bits 15 bits 23 bits 31

Octeto 1	Octeto 2	Octeto 3	Octeto 4
----------	----------	----------	----------

Classe A	0	REDE	HOST
----------	---	------	------

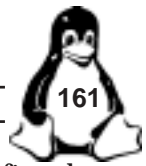
Classe B	1 0	REDE	HOST
----------	-----	------	------

Classe C	1 1 0	REDE	HOST
----------	-------	------	------

Classe D	1 1 1 0	ENDEREÇO MULTICAST	
----------	---------	--------------------	--

Como podemos ver, as classes usuais são A, B e C, sendo que a classe D é usada para multicast, que é utilizado para envio de informação em massa de forma controlada, por exemplo, as rádios na Internet.

Com a classe A, o primeiro octeto é para identificar a rede e o restante para os hosts. Dessa forma, temos como valor mínimo para



rede 00 00 00 00 e máximo 01 11 11 11, que em decimais significa de 0 a 127. Possui endereços suficientes para endereçar 128 redes (primeiro octeto de 0 a 127) e 16.777.216 hosts cada rede.

Já na classe B, o primeiro e o segundo octeto são designados para rede e o restante para os hosts. Temos como valor mínimo 10 00 00 00 . 00 00 00 00 e máximo 10 11 11 11 . 11 11 11 11 para redes. Em decimais, 128.0 a 191.255. Possui endereços suficientes para endereçar 16.284 redes com até 65.536 hosts cada rede.

Na classe C, os três primeiros designam a rede e apenas o último designa os hosts, com o valor mínimo de 11 00 00 00 . 00 00 00 00. 00 00 00 00 a 11 01 11 11 . 11 11 11 11 . 11 11 11 11, em decimais 192.0.0 até 223.255.255. É capaz de suportar 2.097.152 redes com até 256 hosts.

Com essa divisão, foi possível acomodar redes grandes e pequenas, mas supondo que tenhamos uma rede local (Intranet), com apenas 100 hosts, provavelmente iríamos usar a classe C, que suporta 256 hosts. Nesta divisão, temos um desperdício de 153 hosts, sendo que o primeiro host designa a rede e o último o endereço de broadcast (veremos mais a frente). Por isso, esta forma de divisão não é mais empregada sozinha, agora temos uma variação que é a sub-rede e o mascaramento dela.

Máscara de Sub-Rede

A divisão de redes em classes foi causa de muitos problemas em relação à distribuição de endereços IP. Uma rede, indiferente da quantidade de máquinas, deve ser compatível com uma das classes de rede. Se tivesse 30 máquinas, por exemplo, receberia um endereçamento de classe tipo C, causando desperdício de 226 endereços IPs. Já uma rede com 246 endereços, receberia um endereçamento de classe do tipo B, causando um desperdício de 62000 endereços.

A partir de 1988, o número de redes interligadas à Internet aumentou muito, agravando o problema de disponibilidade de endereços.

Para reverter essa situação, foi necessário criar uma forma de aumentar os números de endereços de redes disponíveis sem afetar o funcionamento já existente.



A solução encontrada para esses problemas foi a implementação da máscara de rede atuando juntamente e sobre o endereçamento IP de forma que os bits para rede aumentassem e os bits para os hosts diminuíssem.

A máscara de sub-rede não aceita misturas de 1's e 0's, por isso deve-se preencher primeiramente os números 1's, que são os bits designados para a rede, e, posteriormente, os 0's que são os bits do host.

Tem-se o padrão de máscaras para sub-redes que varia da seguinte forma:

Classes	Endereçamento	Máscara
Classe A	0-127	255.0.0.0
Classe B	128-191	255.255.0.0
Classe C	192-223	255.255.255.0

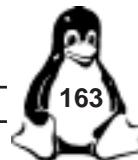
Vamos supor que tenhamos uma rede com 60 micros. Logicamente, iríamos usar um endereçamento de classe tipo C 192.168.200.0, com 256 hosts disponíveis. Com isso, temos um desperdício de 196. Nesta ocasião é que devemos aplicar a máscara de rede, com algumas alterações em relação a padrão.

bits 0	7	15	23	31
Octeto 1	Octeto 2	Octeto 3	Octeto 4	
192	168	200	0	
11000000	10101000	11001000	0	
REDES				HOSTS

No caso acima, mostra a situação atual, ou seja, os três primeiros octetos para rede e o último octeto para os hosts, sendo que o último octeto é flexível e pode ser alterado conforme o número e endereço IP atribuído.

Temos uma rede que vai de 192.168.200.1 (11000000.10101000.11001000.000000) até 192.168.200.255 (11000000.10101000.11001000.11111111), tendo 254 IPs disponíveis, utilizando apenas dois para endereços privados, número da rede e endereço de broadcast (será visto adiante).

Nesta situação, teremos um grande desperdício, por isso vamos aplicar a máscara de rede, de forma que possa pegar dois bits do quarto octeto. Sendo assim, a máscara será 255.255.255.192.



Redes

bits 0	7	15	23	31
Octeto 1	Octeto 2	Octeto 3	Octeto 4	
192	168	200	00000000	
11000000	10101000	1100100	00	000000
REDES				HOSTS

255	255	255	192
11111111	11111111	11111111	11 000000
REDE			HOST

Com isso, o endereço IP sofre uma pequena alteração:

bits 0	7	15	23	31
Octeto 1	Octeto 2	Octeto 3	Octeto 4	
192	168	200	0	
11000000	10101000	11001000	00	000000
REDES				HOSTS
255	255	255	192	
11111111	11111111	11111111	11	000000
REDE				HOST

Enfim o que foi feito ?

Você não entendeu ?

Vamos ver o que foi feito:

Anteriormente, tinha a rede 192.168.200.0 (11000000.10101000.11001000.00000000) com máscara 255.255.255.0 (11111111.11111111.11111111.00000000).

A máscara simplesmente aumenta o número de bits da rede, que era de 23 (3 octetos), para 25 (3 octetos e para do 4º octeto), deixando apenas 6 bits para os hosts (máquinas). Enfim, a máscara só vai permitir atribuir os últimos 6 bits para o host, como se ele travasse os dois primeiros bits do 4º octeto.

A rede só poderá alterar os seis últimos bits do 4º octeto, ou seja, irá de 00000000 até 00111111 (em decimal 0 até 63).

O endereçamento IP ficará da seguinte forma:



De 192.168.200.0 (11000000.10101000.11001000.00000000).

Até 192.168.200.63 (11000000.10101000.11001000.00111111).

Terá endereço de rede 192.168.200.0, broadcast 192.168.200.3 e 62 endereços de IP disponíveis para a rede. No total, são 64 endereços.

Entendeu agora?

Com o uso da máscara de sub-rede, aumentamos o número de bits para a rede e diminuimos o número de bits para o host, deixando de desperdiçar, assim, o restante dos endereços da classe C.

Espera aí!!

Quer dizer que se eu aumentar mais um bit na máscara de rede, vou diminuir ainda mais o número de hosts?

Isso mesmo!!! Vamos fazer um teste?

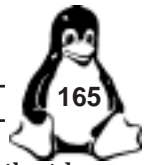
Vamos usar a mesma rede 192.168.200.0. Anteriormente, a máscara era 255.255.255.192 (em bits 11111111.11111111.11111111.11000000). Adicionando um bit 11111111.11111111.11111111.11100000. Calculando isso para decimais, teremos 255.255.255.224.

Nos itens anteriores, referentes ao 4º octeto (00000000), somente os 6 últimos bits podiam ser alterados para atribuir valor ao host. Agora, isso diminui um pouco. Como o 4º octeto da máscara é 224 (11100000), faz com que só possam ser alterados os cinco últimos bits do 4º octeto do endereçamento.

Com isso, a variação do 4º octeto do endereçamento IP irá de 00000000 até 00011111, que em decimal significa de 0 até 31, fazendo um total de 32 hosts, onde 192.168.200.0 é atribuído à rede, 192.168.200.31 ao broadcast e trinta valores disponíveis para as máquinas com máscara 255.255.255.224

Provavelmente, você não usará tudo isso, principalmente se for uma rede pequena ou caseira, com poucos micros. Na realidade, não precisa se preocupar, pois em uma rede caseira, estes endereços de IP são internos, não afetando demais hosts da Internet.

Entramos, agora, em outros aspectos do protocolo. Neste capítulo, mencionei algo sobre endereço de rede, broadcast, etc. Estes endereços são conhecidos como endereços privados, ou seja, que tem um uso específico para eles.



O primeiro endereçamento IP de uma classe deve ser atribuído à identificação do endereço de rede, por exemplo 192.168.200.1 200.18.240.0, representada por todos os bits referentes ao host com valor 0.

A rota default, como é conhecida, usa o endereço 0.0.0.0 e é utilizada para operações com DHCP.

Endereço de loopback, usa a rede 127.0.0.0. Para cada placa de rede local, é atribuído um valor, como por exemplo 127.0.0.1, que é muito útil para teste. Mas é atribuído automaticamente, dessa forma não pode ser usado por atribuição manual.

Temos também o endereço de broadcast, que identifica todas as máquinas da rede específica, representada por todos os bits do host com valor 1. Por exemplo, 192.168.200.255/255.255.255.0 (Endereçamento IP/Máscara), 192.168.200.31/255.255.255.224.

Todos esse endereços não podem ser atribuídos a nenhuma máquina por terem seu uso já especificado no protocolo.

Então, posso atribuir qualquer outro endereçamento de IP em minha máquina sem problemas?

Não.

Os IPs válidos na Internet são fornecidos por provedores ou empresas de telecomunicações e outros órgãos. São utilizados para uso privado dentro de um Intranet, pois não são roteadas pela Internet, ou seja, não são vistas, e são essas redes que poderemos usar em ambiente interno.

Internet e Roteamento





Como sabemos, o protocolo usado na Internet é o TCP/IP, que encapsula o WWW, SMTP, POP3, etc.

De qualquer forma, quando acessamos uma página da Internet, usamos o protocolo HTTP, que roda sobre o TCP. Se for TCP, temos, então, um endereçamento IP para as páginas que acessamos. Já pensou se todas as vezes que precisássemos acessar um site tivéssemos que digitar o seu IP? Seria muito mais trabalhoso. Para isso, temos um servidor de nomes conhecido como DNS (Domain Name Service), que nada mais é que um tradutor. Traduz o nome que digitamos para seu



respectivo número IP. Isto também pode ser levado em conta no caso de SMTP e POP, que é envio e recebimento de mensagens.

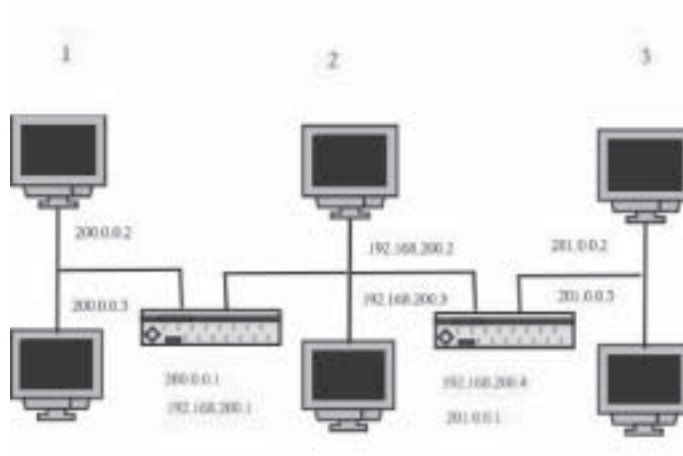
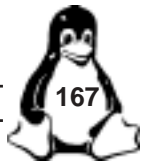
Então, todas as vezes que digitamos www.tal.com.br, nosso micro vai ao servidor de nomes e verifica qual o IP para este host. Temos já alguns domínios prontos que podem ser usados em caso específicos. Por exemplo, para achar um host ou página no Brasil, normalmente, o final é *br*, no Reino Unido, seria *uk*, e assim por diante. Segue abaixo uma pequena lista:

-  com= Comercial
-  gov= Governamental
-  edu= Educacional
-  mil= Militar

Para se registrar um domínio, logicamente temos que ter endereços IP válidos, ou seja, que não sejam privativos para uso doméstico ou privado e, posteriormente, registrar um nome em <http://registro.br>. São registrados os domínios que vinculam seu IP a um nome. Pode ser obtida muita informação sobre domínios, como registrá-los, quais suas extensões, etc.

Um fato interessante no protocolo IP, é a comunicação entre redes diferentes. Essa comunicação se dá por Gateway e roteadores. O Gateway consiste em uma máquina que tem acesso às duas redes, tendo duas interfaces de rede (placas de rede) que controlam o acesso de uma a outra rede.

A seguir, temos a figura de três redes e a forma como se comunicam entre si:

*Rede exemplo*

Na figura acima, podemos ver que existem três redes, 200.0.0.0, 192.168.200.0 e 201.0.0.0, designadas como 1, 2, 3. Para a rede A comunicar-se com a rede C, terá, obrigatoriamente, que passar pelo 1º e pelo 2º roteador.

Na rede 1, temos duas estações 200.0.0.2 e 200.0.0.3.

Na rede 2, temos mais duas estações 192.168.200.2 e 192.168.200.3.

Na última rede, a 3, existem também duas estações 201.0.0.2 e 200.0.0.3.

Além das seis estações, temos dois Gateway's (Roteadores), que servem de elos entre a rede 1, 2 e 3. Vamos chamá-los de A e B.

Para cada Gateway, deve haver uma tabela de roteamento, de forma que, quando solicitado a entrega de um pacote a outra rede, esta prontamente entregue.

Roteador A

Destino	Gateway
200.0.0.0	eth0 (rota direta)
192.168.200.0	eth1 (rota direta)
201.0.0.0	201.0.0.1
Gateway default	201.0.0.1



Roteador B

Destino	Gateway
200.0.0.0	200.0.0.1
192.168.200.0	eth0 (rota direta)
201.0.0.0	eth1 (rota direta)
Gateway Default	201.0.0.1

Talvez eu não tenha falado anteriormente, mas os itens designados como *eth* são as próprias placas de rede. Para a primeira placa, temos a device *eth0*; para a segunda, *eth1*; para a terceira, *eth2*; e assim sucessivamente.

Falamos em roteador, mas se a máquina 200.0.0.2 tentar acessar a máquina 192.168.200.2, o que seria necessário fazer?

A máquina 200.0.0.2 tentaria mandar o pacote. Em suas configurações, teria um Gateway padrão, comumente chamado de default Gateway. Se esse Gateway tiver acesso e a máquina remetente tiver permissão, ele repassará o pacote para a máquina destinatária. Caso ele não acesse a rede destino (por exemplo, se o pacote fosse para a rede 201.0.0.0), ele repassa para o outro Gateway, que verifica as permissões, para então repassar o pacote finalmente.

Vamos analisar melhor:

- 📁 Máquina Remetente: 200.0.0.2 (1).
- 📁 Máquina Destinatária: 192.168.200.2 (2).
- 📁 Caminho: Máquina 1 envia pacote e Gateway recebe pacote.
- 📁 Gateway verifica se a máquina 1 tem permissão de envio para rede da máquina 2.
- 📁 Gateway verifica se tem acesso a rede da máquina 2.
- 📁 Gateway envia o pacote para máquina 2.
- 📁 Máquina 2 recebe o pacote.

Com isso, temos a seguinte configuração da máquina 1:

IP : 200.0.0.2

Máscara de IP: 255.255.255.0

Default Gateway: 200.0.0.1



Máquina 2:

IP: 192.168.200.2.

Máscara de IP: 255.255.0.0.

Default Gateway: 192.168.200.1.

Agora, suponhamos que a máquina 200.0.0.2 mande uma pacote para 201.0.0.2.

Máquina Remetente: 200.0.0.2 (1).

Máquina Destinatária: 201.0.0.2(2).

Gateway 1: 200.0.0.1 - 192.168.200.1.

Gateway 2: 201.0.0.1 - 192.168.200.4.

Caminho:

☞ Máquina 1 envia pacote.

☞ Gateway 1 recebe (200.0.0.1) o pacote.

☞ Gateway 1 verifica se a máquina 1 tem permissão de envio para rede da máquina 2.

☞ Gateway 1 verifica se tem acesso a rede da máquina 2.

☞ Gateway 1 envia para o seu Gateway Default (Gateway 2).

☞ Gateway 2 (201.0.0.1) verifica se máquina 1 tem permissão de envio para rede da máquina 2.

☞ Gateway 2 envia o pacote para máquina 2.

☞ Máquina 2 recebe o pacote.

Neste outro exemplo, veja que, como o primeiro Gateway não tinha acesso a rede 3, ele repassou o pacote para o Gateway 2, que por sua vez repassou o pacote para a máquina destinatária.

É desta forma que a Internet, e grandes redes se comunicam, sempre repassando pacotes de um roteador para outros do mundo inteiro.

Depois dessa introdução, estamos prontos para configurar a rede **Linux**.



Rede Linux

Separaremos a montagem da rede **Linux** em duas partes: uma referente a parte física e outra a parte lógica.

Na parte física, será vista a configuração da placa rede na máquina, seus aspectos em relação a módulos e carregamento dos mesmos. Já na parte lógica, serão atribuídos o endereçamento IP, a máscara de sub-rede, o Gateway, além de serem vistos aspectos do DNS, DHCP, etc.

Configurações da Rede Linux

Módulos

O primeiro item que deve ser verificado, é a interface de rede, isto é, sua placa. Normalmente, ela é configurada na instalação. Algumas placas podem não ser compatíveis com o sistema, mas são muitas raras, pois até as placas onbord funcionam com o **Linux**.

Para verificar se sua placa é suportada pelo **Conectiva Linux**, acesse o site <http://www.conectiva.com.br/hardware>.

Para verificar se sua placa foi configurada, acesse o arquivo */etc/modules.conf*:

```
vi /etc/modules.conf
```

Neste arquivo, deve ter uma entrada do tipo:

```
alias eth0 modulo
```

Onde o módulo é o da sua placa. Por exemplo, da 3COM, um dos módulos é o 3c509, portanto, a entrada estaria dessa forma se sua placa fosse deste tipo:

```
alias eth0 3c509
```

Este arquivo carrega os módulos da sua placa na inicialização do sistema.

Se sua placa for do tipo NE 2000, você pode usar o módulo *ne*:

```
alias eth0 ne
```

O *ne* seria o módulo para sua placa NE 2000.



Provavelmente, existirá outras entradas iniciadas pela palavra *alias*, que se refere a outros hardwares. Não se preocupe com eles.

Se você possui mais de uma placa de rede, deve adicioná-la a este arquivo. As interfaces de rede são chamadas *eth*, para primeira placa *eth0*, para segunda *eth1* e assim por diante.

Existem formas bem simples de configurar as placas, mas para começar, configuraremos da forma mais difícil, acessando os arquivos de configuração e editando-os.

Para sabermos se o módulo está carregado, existe uma forma de verificá-lo através do comando *lsmod*, digite-o no prompt:

```
lsmod
```

A resposta deve ser algo do tipo:

dmfe	9216	1	(autoclean)
vmnet	16192	3	
vmppuser	5520	0	(unused)
parport_pc	7504	0	[vmppuser]
parport	7360	0	[vmppuser parport_pc]
vmmon	18176	0	(unused)
8139too	12768	1	(autoclean)
agpgart	18768	0	(unused)
supermount	12944	2	(autoclean)
es1371	26448	0	
soundcore	2608	4	[es1371]
usb-ohci	12576	0	(unused)
usbcore	42992	1	[usb-ohci]

No exemplo acima, temos dois módulos carregados: o *dmfe*, referente à placa Davicom e a *8139too*, referente a placa Realtec.

Isso mostra que elas estão carregadas. A segunda coluna é referente ao tamanho do módulo alocado.

Mas se ela não estiver neste lista, o que devemos fazer?

Existe o conjunto de comandos *mod*. No caso, o *lsmod* lista, *rmmod* descarrega da memória e *insmod* carrega. Mas antes, tem que estar configurado no arquivo *modules.conf*. Após configurar corretamente o arquivo, você poderá carregar sem problemas. Vamos



supor que sua placa é uma *sis900*. Para carregá-la, adicionamos uma linha no arquivo *modules.conf*:

```
alias eth0 sis900
```

Posteriormente, a carregamos:

```
insmod sis900
```

A resposta será:

```
Using /lib/modules/2.2.19-15cl/net/sis900.o
```

Isto quer dizer que o módulo foi carregado, mas se por algum motivo você notou que o módulo correto não é esse, é só digitar:

```
rmmmod sis900
```

Automaticamente, o módulo será descarregado da memória.

Configurações de Rede - Arquivo Network

Com esses passos, configuramos a placa de rede. Agora, precisamos configurar a rede propriamente dita. O primeiro arquivo a ser visto é o *network*... bem sugestivo, não ?!!!

Esse arquivo é responsável por ativar a rede, e está localizado em */etc/sysconfig/network*.

Acesse através de qualquer editor:

```
vi /etc/sysconfig/network
```

Abaixo, um arquivo de exemplo:

```
NETWORKING=yes  
FORWARD_IPV4=yes  
HOSTNAME="dominiolinux.com.br"  
GATEWAY = 200.0.0.1
```

A primeira linha, *Networking*, é responsável por ativar a rede. Se caso estiver *no*, a máquina não estará ativa na rede.

O segundo item, *Forward_ipv4*, é muito importante no caso da máquina ser responsável por algum roteamento, ou seja, se ela for elo entre uma rede e outra. Um caso muito comum é quando ela será Gateway de outras máquina para acessar Internet através de modem, ADSL, ISDN, etc. Se não for o caso da sua máquina, deixe-o desabilitado com o valor *false*.



A linha iniciando com *Hostname*, refere-se ao nome e domínio da máquina. Se no DNS (Domain Name Service) ela estiver cadastrada com outro nome, logicamente, esse valor só valerá localmente.

A linha *Gateway* é muito importante, pois é através dela que teremos acesso à outra rede, inclusive a Internet. Caso não saiba qual é o Gateway da sua rede, informe-se com o administrador da rede ou seu provedor de acesso a Internet. Se suas configurações forem feitas através de DHCP (veremos a frente), pode deixá-lo em branco, pois será repassado para máquina automaticamente.

Essa são as configurações iniciais da rede, mas ainda temos algum trabalho pela frente: iremos fazer a configuração da rede na placa.






Configuração da Placa - Arquivo *ifcfg-ethx*

O arquivo responsável é o *ifcfg-ethx*, onde *x* é referente ao número da placa. Por exemplo, se for a primeira placa, será *ifcfg-eth0*; se for a segunda, *ifcfg-eth1*; e assim por diante.

Edite o arquivo para configurarmos a rede. Caso não exista, crie um novo:

```
vi /etc/sysconfig/network-scripts/ifcfg-eth0
```

Os itens que devem ser preenchidos no arquivo são:

-  **DEVICE:** "ethx". Esta linha informa qual é o adaptador de rede que está sendo configurado, similar ao nome do arquivo você deve designar o número da sua placa, *eth0*, *eth1*, etc.
-  **IPADDR:** "192.168.200.1". Endereço IP designado a máquina.
-  **NETMASK:** "255.255.255.0". Máscara de sub-rede atribuída a máquina.
-  **ONBOOT:** "yes". este item especifica se o adaptador será carregado no boot ou não, caso queira carregar o adaptador posteriormente.
-  **BOOTPROTO:** "none". Especifica se o adaptador receberá suas configurações via DHCP (leia abaixo). Se for esse o caso, o valor deve ser "dhcp".



BROADCAST: "192.255.255.255". Endereço de broadcast da rede.

Com esses arquivos, já temos a configuração para o adaptador de rede. Mas caso você use DHCP, deve deixar todos os campos em branco, ou seja, somente as aspas. Mas o conteúdo de BOOTPROTO deve ser igual a DHCP.

DHCP, ou *Dynamic Host Configuration Protocol*, é uma máquina na rede que gerencia o endereçamento IP, distribuindo por todas a rede configurações de IP, máscara de sub-rede, endereços de DNS. Com isso, não é necessário configurar manualmente cada máquina, e, dependendo do tamanho da rede, isso pode ser muito trabalhoso.

Abaixo, temos dois exemplos: um de uma máquina configurada manualmente e outro em que as configurações são feitas via DHCP.

Via DHCP:

```
DEVICE="eth0 "  
IPADDR=" "  
NETMASK=" "  
ONBOOT="yes"  
BOOTPROTO="dhcp"
```

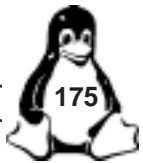
Configuração manual:

```
DEVICE="eth1 "  
IPADDR="192.168.200.1 "  
NETMASK="255.255.255.0 "  
ONBOOT="yes"  
BOOTPROTO="none"
```

Outros arquivos importantes são os *resolv.conf*, de cada host. Neles devem constar os valores dos servidores DNS (só para lembrar, DNS é um servidor que serve para resolver nomes, ou seja, quando for digitado um endereço - seja ele WWW ou o nome de uma máquina da rede - para não ditarmos o endereçamento IP, este servidor recebe o valor digitado, transforma para endereçamento IP de 32 bits e, posteriormente, envia para a máquina. Veja, abaixo, um exemplo de um arquivo *resolv.conf*:

```
search servidor.dns.com.br  
nameserver 200.0.0.1  
nameserver 200.0.0.2
```

Na linha *search*, deverá constar o nome do servidor e, nas linhas *nameserver*, deve ser colocado o DNS primário e secundário.



Devo lembrar que, se for o caso, usaremos um servidor DHCP e deve-se deixar em branco todas as linhas.

Configurando DNS

Agora, se por acaso você usar um servidor de DNS na Internet e não possuir um servidor DNS na sua própria rede, para encontrarmos uma máquina temos que procurar pelo endereçamento IP. Muito trabalhoso, não?

Podemos colocar o endereçamento IP e seu nome no arquivo `hosts`. Veja o exemplo abaixo:

```
vi /etc/hosts
127.0.0.1      localhost.localdomain  localhost
192.168.200.1 Stato.dominiolinux        dominiolinux
```

Dessa forma, podemos localizar as outras máquinas na rede através do nome.

Para iniciar a rede, devemos acessar o diretório `/etc/rc.d/init.d/` ou digitarmos `cds`, que é um atalho para esse diretório.

Neste diretório, ficam os serviços que rodam no **Linux**, chamados de *daemon*. Neste diretório, iremos iniciar o serviço de rede, digite:

```
./network restart
```

Se tudo ocorreu bem, ela dará várias mensagens de serviços sendo iniciados e um deles, em especial, é o referente a inicialização da *eth0*, que deve responder OK. Se houver uma falha na inicialização do adaptador de rede, ele exibirá uma mensagem. Devemos rever todos esses arquivos e verificar se existe algo errado em alguma configuração.

Comandos de Rede

Após verificar se há algum problema nos arquivos, a placa pode ser carregada e descarregada (somente a placa) sem necessidade de reiniciar todo o serviço de rede.

Podemos fazer isso através dos comandos *ifup* e *ifdown*.

Como o próprio nome diz, *up* para levantar, ou seja, inicializar as configurações de placa:

```
ifup eth0
```



Ifdown serve para parar, descarregar e, enfim, descarregar as configurações da placa de rede:

```
ifdown eth0
```

Dessa forma, podemos carregar e descarregar as configurações da placa conforme a necessidade, principalmente quando estamos configurando pela primeira vez, como é o caso, onde pode ocorrer algum problema nesta configuração.

Similar a este comando é o *ifconfig*:

```
ifconfig eth0 up  
ifconfig eth0 down
```

Mas o *ifconfig* serve também para verificar as configurações e alterá-las.

Digite no prompt:

```
ifconfig
```

Será mostrada a configuração da rede, conforme o exemplo abaixo:

```
eth0  Encapsulamento do Link: Ethernet  Endereço de  
HW 00:E0:7D:A0:7E:78
```

```
inet end.: 192.168.200.1  Bcast:192.168.200.255  
Masc:255.255.255.0
```

```
UP BROADCASTRUNNING MULTICAST  MTU:1500  
Métrica:1
```

```
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
```

```
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
```

```
colisões:0 txqueuelen:100
```

```
RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
```

```
IRQ:10  Endereço de E/S:0x7e00
```

```
lo  Encapsulamento do Link: Loopback Local  inet  
end.: 127.0.0.1  Masc:255.0.0.0
```

```
UP LOOPBACKRUNNING  MTU:3924  Métrica:1
```

```
RX packets:330 errors:0 dropped:0 overruns:0 frame:0
```

```
TX packets:330 errors:0 dropped:0 overruns:0 carrier:0
```

```
colisões:0 txqueuelen:0
```

```
RX bytes:65908 (64.3 Kb)  TX bytes:65908 (64.3 Kb)
```




Podemos ver, no exemplo acima, alguns dados interessantes. Primeiro temos um endereço para interface *eth0* e outro para *lo*, ou seja, *loopback*. A *eth0* com endereçamento IP, 192.168.0.1, com seu endereço de broadcast e máscara de sub-rede.

Outra forma de utilizar o comando *ifconfig* é para carregamento da interface:

```
ifconfig eth1 192.168.200.1 255.255.255.0
```

Adicionando o *up* ao final configuramos e carregamos.

Uma forma de verificar se a interface está funcionando, é através do comando *ping*, similar ao da Microsoft:

```
ping 192.168.200.1
```

```
64 bytes from 192.168.200.1: icmp_seq=0 ttl=255 time=0.4
ms
64 bytes from 192.168.200.1: icmp_seq=1 ttl=255 time=0.2
ms
64 bytes from 192.168.200.1: icmp_seq=2 ttl=255 time=0.2
ms
```

Se o resultado for como o exemplo acima, parabéns! Sua placa está configurada. O último item é o tempo para ele chegar ao destino. Com certeza é bem baixo, pois estamos *ping* na própria máquina, mas quando trata-se de outros hosts fora rede local, é um ótimo analisador: quanto maior o tempo, mais longe a rede está, ou com maior tráfego.

Agora, ao contrário deste valor, você obter algo como:

```
PING 192.168.200.1(192.168.200.1): 56 data bytes
```

```
ping: sendto: Network is unreachable
```

```
PING 192.168.200.1((192.168.200.1): 56 data bytes
```

```
ping: sendto: Network is unreachable
```

Deve-se verificar as configurações, pois não está funcionando. A máquina não consegue “pingar” este endereçamento IP, que significa algo de errado na placa de rede ou nas configurações. Neste caso, amigo, volte ao começo do capítulo e comece novamente.

Tratando-se agora de Gateway e roteamento, quando queremos, por exemplo, acessar uma outra rede ou a própria Internet, temos que ter o endereço IP do Gateway do provedor configurado.

Ele fica configurado no arquivo */etc/sysconfig/network*.



Mas, através de alguns comandos, é possível adicionar e remover rotas dentro do **Linux**. Vejamos o *route*.

O primeiro item que é somente uma consulta, digite no terminal:

```
route -n
```

Podemos ver o resultado abaixo:

Tabela de Roteamento IP do Kernel

Destino	Roteador	Máscara Geen	Opções	Métrica	Ref	Uso	Iface
192.168.200.1	*	255.255.255.0	U	0	0	0	eth0
192.168.200.0	*	255.255.255.0	U	0	0	0	eth1
127.0.0.0	*	255.0.0.0	U	0	0	0	lo
default	192.168.200.1	0.0.0.0	UG	0	0	0	eth0

Nesta tabela, podemos verificar o seguinte: temos duas placas *eth0* e *eth1*. A rede da *eth1* é a mesma da *eth0*, ficando bem visível que nosso Gateway é a *eth0*, onde a linha default mostra qual o roteador padrão da rede na linha iniciada por default e na coluna *Opções* da mesma linha preenchida com o valor *UG*, onde *G* significa Gateway.

Mas descobrimos que o endereço de Gateway não é esse e sim um endereço válido de Internet do tipo *200.xxx.xxx.xxx*, diferente de endereço privado, como vimos anteriormente.

Neste caso, teremos que remover a configuração dos arquivos *network*, adicionar a nova e reiniciar a rede.

Muito trabalho, não?

Para isso, temos o comando *route*. É só digitar o seguinte:

```
route del default gw 192.168.200.1
route add default gw 200.xxx.xxx.xxx
```

Com o *route del*, retiramos da configuração do roteador e com o *route add*, colocamos a nova sem a necessidade de reiniciar toda a rede.

Temos outras opções de uso do *route*. Veja abaixo:




```
route [add/del] -net {ip da rede destino} netmask
{máscara da rede destino} <adaptador>
```

Ex: route add -net 192.168.100.0 netmask 255.255.255.0
eth1

Com isso estamos roteando tudo que entra no adaptador *eth1* para uma rede específica.



Seguem abaixo os comandos. Vou chamar a rede destino de *rd*, adaptador de *adap*, host destino de *hd*, máscara da rede destino de *mask*.

-  `route [add/del] -net <rd> netmask <mask> {adap}` - Já foi falado antes.
-  `route [add/del] -host <hd> {adpt}` - roteia um adaptador de rede (*eth0*, *eth1*,...) para um host de destino.
-  `route [add/del] gw <ip do roteador>` - Designa um gateway padrão.

Se, por exemplo, você não quiser que ele roteie no primeiro e no segundo caso (*add net* e *host*), pode-se acrescentar, ao final do comando, o item *reject*, mas isso não fará com que o host ou a rede destino rejeite, e sim, que o adaptador simplesmente não percorra essa rota (poderá ser roteado por outra qualquer).

Em relação a configuração, já está terminada. Veremos como pode ser feita graficamente. Mas antes, quero mostrar alguns comandos interessantes para uso em rede.

O primeiro é o *netstat*, que fornece informações sobre as portas TCP, UDP (entre outras) abertas e que estão em uso:

O comando *netstat* sozinho mostrará a lista de portas abertas, as que estão ocupadas ou ouvindo o tipo de conexão, o host, etc.

Mas caso queira ver o endereçamento IP em suas respectivas portas, use o *netstat*, seguindo de *-n*:

```
netstat -n
```

Outro comando muito interessante, é o *traceroute*, ele mostra o caminho percorrido da máquina local até o host destino:

```
traceroute www.dominoliinux.com.br
 1  192-168-200-1 (192.168.200.1)  1.235 ms  1.031 ms
3.159 ms
 2  200-xxx-xxx-xxx .domino.com.br (200.xxx.xxx.xxx]
17.510 ms  16.786 ms  19.784 ms
...
...
```

E assim por diante até o destino final, ideal para verificar os roteadores da uma rede interna até saírem para Internet.

Também temos o comando *nslookup*, que resolve o endereçamento IP nome através do DNS usado.



```
nslookup 200.250.58.152
Server:      200.204.0.10
Address:     200.204.0.10#53
```

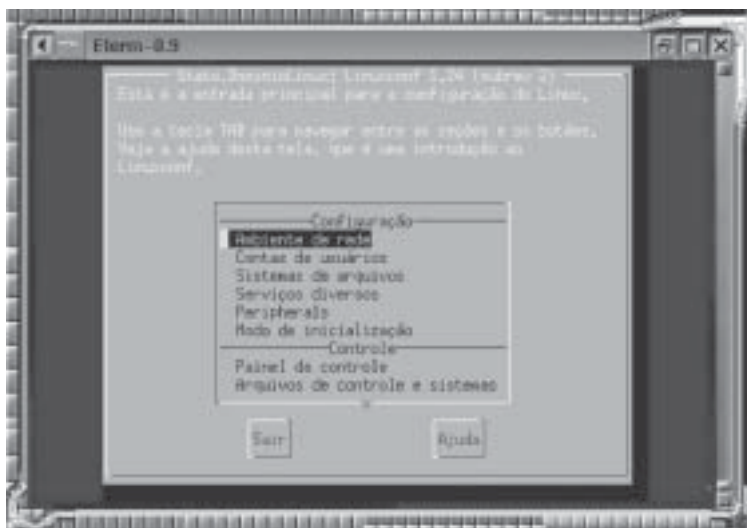
```
152.58.250.200.in-addr.arpa    canonical name =
152.128-191.58.250.200.in-addr.arpa.
152.128-191.58.250.200.in-addr.arpa    name =
dino.conectiva.com.br.
```

Neste exemplo acima, procuramos o host para o IP 200.250.58.152, onde ele informa que pertence a Conectiva. Este comando é muito útil para testar DNS Masters, que ficam na rede local para resolver nomes da própria rede.

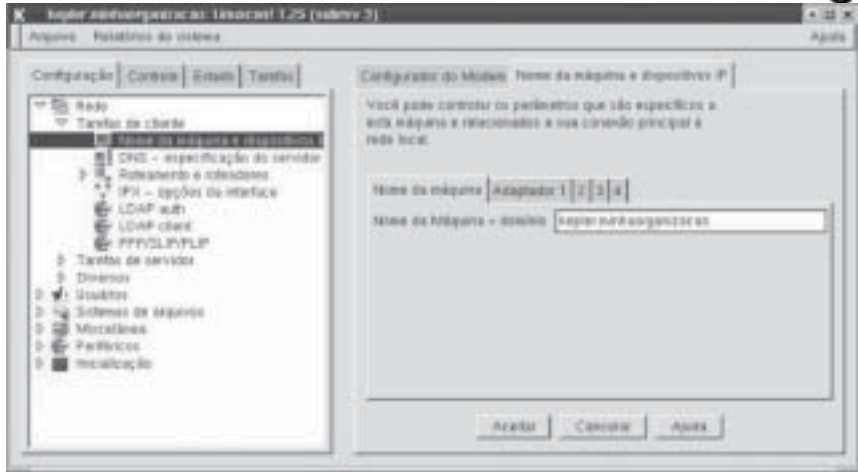
Configurando a Rede através do Linuxconf

Bom, chegamos ao momento de fazer as coisas de um modo mais fácil e simples através de interface gráfica.

Em modo texto ou gráfico, podemos usar o Linuxconf. Veja abaixo as duas figuras :











Linuxconf no modo texto



Linuxconf modo Gráfico

Para configurar em modo texto, digite o *linuxconf* e siga a rota *Ambiente de Rede/Nome da máquina e dispositivos IP de rede*. O primeiro item a ser preenchido é o nome da máquina e domínio. Abaixo dessa configuração, encontra-se as configurações de adaptadores, começando por *Adaptador 1* até o *Adaptador 7*. Cada configuração consta os seguintes itens:

-  **Modo de configuração:** Marque o tipo de inicialização da interface conforme sua rede, manual, DHCP ou boot.
-  **Primeiro nome + domínio:** Nome da máquina e domínio para esta placa.
-  **Apelidos (opc):** Apelido para máquina é opcional.
-  **Endereço IP:** Número do IP designado para a máquina.
-  **Máscara (opc):** Máscara de sub-rede.
-  **Dispositivo de rede:** Dispositivo de rede (*eth0*, *eth1*, *eth2*, etc.).
-  **Módulo do Kernel:** Módulo usado para carregar a placa (3c509, 8139too, sis900, etc.).
-  **Porta E/S (opc):** Porta I/O, opcional também.

Teclando *Tab*, alterne até o item *Aceitar*, assim será feita a configuração da placa. Voltemos, então, a tela inicial para configurar o servidor DNS - especificação dos servidores de nomes. No item do servidor padrão, coloque o nome dos servidores. Logo abaixo terá o item IP do servidor de nomes 1 e IP do servidor de nomes 2, onde serão



colocados os endereços IP de Servidor DNS primário e secundário. Feito isso, da mesma forma que você acessou o item *Aceitar* anteriormente, você também o fará agora.

Novamente na tela inicial, você verá o item *Roteamento e roteadores/Definir Padrão*. Neste item, será colocado o endereçamento IP do Gateway.

Quando acabar, selecione as opções *Aceitar/Fechar/Fechar*. Será informado que algumas alterações serão realizadas. Acesse o item *Faça isso*. Serão realizadas as alterações necessárias nos arquivos de configuração e serviços para que a rede esteja disponível imediatamente.

Como foi visto, é bem fácil configurar através desta ferramenta, mas caso não funcione, verifique os arquivos manualmente.

No Linuxconf para X ou desktop, seu visual é mais bonito, mas sua utilidade é a mesma. Seguindo o item *Configuração/Rede/Tarefas de cliente/Nome da máquina e dispositivos IP de rede*, da mesma forma como foi dito anteriormente, você irá configurar sua placa de rede. Só um item que leva vantagem em relação ao modo texto, que é a escolha do módulo (em vez de você preenchê-lo com o nome do módulo, escolhe dentro de várias possibilidades). Mas isso não descarta que você, de antemão, deva saber qual o nome da placa e seu módulo.

Nó próximo capítulo, veremos como transformar sua máquina Linux em um Gateway para rotear duas ou mais redes. Também veremos como montar um firewall **Linux**, usando as ferramentas *ipchains* e *iptables*.

Firewall com Linux




Firewall com Linux

Falaremos de alguns conceitos e preparemos o Kernel para atuar como nosso servidor.

O que é um firewall? Firewall é um dispositivo de hardware ou software que tem como principal função filtrar pacotes na rede, com objetivo de proteger negando informações ou repassar informações.

No firewall, trabalharemos praticamente com três tipos de protocolo: TCP, UDP e ICMP.

Basicamente, o firewall funciona da seguinte forma:

-  Endereço de origem porta;
-  Endereço de destino porta;
-  Aceita, rejeita ou nega.

O firewall pode, ainda ter outras funções: roteador, servidor NAT, etc.

Para habilitar o firewall no **Linux**, alguns itens devem estar funcionando no Kernel.

Para acessar, vá em `/usr/src/linux`.

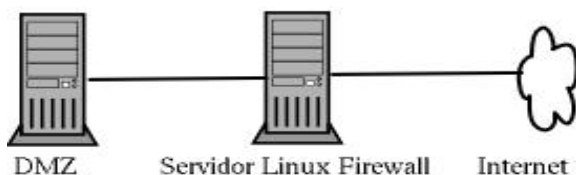
Gateway

Antes de iniciarmos nosso firewall com *ipchains* ou *iptables*, vamos configurar ele como um Gateway para uma rede externa.

Configuraremos a DMZ (Delimited Zone), que é a rede interna para acessar a Internet.



Veja a figura abaixo:



A figura abaixo representa bem o que queremos fazer. Temos de um lado uma rede interna, com endereçamento IP privativo do tipo 192.168.200.0; um servidor **Linux** com duas placas de rede; e, por último, a Internet.

Para iniciarmos este processo, deve estar habilitado o repasse de pacotes. Para habilitá-lo, digite no Shell o seguinte:

```
echo 1 > /proc/sys/net/ipv4/forward
```

Ou se achar melhor, pois toda vez que der um boot na máquina terá que digitar novamente, podemos criar um script ou editar o arquivo `/etc/sysconfig/network` e alterar a seguinte linha:

De:

```
FORWARD_IPV4= "false"
```

Para :

```
FORWARD_IPV4= "true"
```

Ou acessando o arquivo `/etc/sysctl.conf` e alterando a linha, deixando-a da seguinte forma:

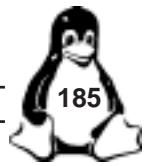
```
net.ipv4.ip_forward = 1
```

Posteriormente reiniciar a rede ou o *daemon network*. Com isso, habilitamos o repasse de pacotes pela máquina servidora, mas devemos fazê-la de Gateway.

E o que vem a ser um Gateway? Em termos bem leigos, é a ponte entre duas redes distintas. O elo de comunicação entre as duas.

Aqui, iremos configurar uma rede que tem acesso a Internet através de uma máquina conectada a ela.

Mas vamos entender melhor o que é Gateway, também conhecido com roteador.



Veja a figura abaixo:

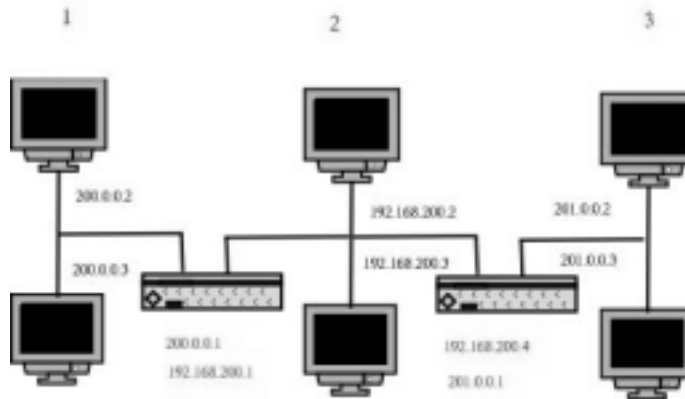


Figura 7.2 - Gateway Linux

A primeira coisa que devemos fazer é instalar duas placas de rede ou uma placa de rede e um fax modem. Por quê? O modem terá uma conexão com a Internet, enquanto a placa de rede terá acesso e dará acesso a rede Internet. Pode-se trabalhar com duas placas de rede, sendo que uma terá acesso.

No exemplo acima, temos três redes distintas:

200.0.0.0

192.168.200.0

201.0.0.0

Todos os micros estão representados pelos computadores e as outras figuras representam os Gateways das redes.

Vamos começar com a rede 200.0.0.0. Suponhamos que o micro 200.0.0.2 queira acessar um recurso que está em 192.168.200.2. Como ocorrerá essa conexão? Neste momento, entra o Gateway. Veja os passos abaixo:

Micro1 = 200.0.0.2

Micro2 = 192.168.200.2

Gateway da rede 192.168.200.0 = 192.168.200.1

Gateway da rede 200.0.0.0 = 200.0.0.1

Micro1 envia pacote para micro 2

Gateway Recebe o pacote



Gateway verifica se o Micro1 tem acesso à rede do Micro2
Micro1 tem permissão de acesso
Gateway envia pacote para Micro 2
Micro2 recebe o pacote

Notaram como quem fez todo o processo de roteamento foi o próprio Gateway.

Certo, agora vamos supor que o *Micro1* (200.0.0.1) queira mandar um pacote para o *Micro3* (201.0.0.2).

Ocorre que o *Gateway 1* (192.168.200.1 - 200.0.0.1) não tem acesso direto à rede 201.0.0.0, por isso é obrigado a repassar para o *Gateway2* (192.168.200.4 - 201.0.0.1), a não ser que o *Gateway1* tivesse três placas e uma fosse configurada com o endereço da rede destino, que é a 201.

Vamos ver o que acontece:

Micro1 manda pacote para Micro3
Gateway 1 recebe o pacote
Gateway um verifica que não tem acesso à rede 3(201.0.0.0)
Gateway um repassa pacote para Gateway2
Gateway 2 recebe o pacote
Gateway 2 verifica se Micro1 tem acesso à rede3
Micro1 tem permissão de acesso
Gateway2 envia pacote para Micro3
Micro3 recebe pacote.

Vamos ver como seria a tabela de roteamento dos Gateways.

O primeiro Gateway está configurado da seguinte forma:

eth0 - 192.168.200.1
eth1 - 200.0.0.1

O segundo, da seguinte maneira:

eth0 192.168.200.4
eth1 201.0.0.1

Tabela para o primeiro roteador:

Rede Destino	Roteador (Gateway)
200.0.0.0	Eth0 (Direto)
192.168.200.0	Eth1 (Direto)
201.0.0.0	201.0.0.1

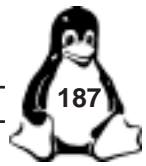


Tabela para o segundo roteador:

Rede Destino	Roteador (Gateway)
201.0.0.0	Eth1 (Direto)
192.168.200.0	Eth0 (Direto)
200.0.0.0	200.0.0.1

Como visto acima, um roteador depende do outro. Este é um exemplo utilizando duas redes privadas.

Na nossa configuração, utilizaremos uma rede interna e um Gateway que acessa a Internet.

O que deve ser feito? Vamos supor que a rede interna seja 192.168.200.0 e a externa 200.254.153.203.

Temos que configurar o micro com duas interfaces de rede, uma para o endereço 192.168.200.1 e outra para 200.254.153.203. Temos que fazer mais uma configuração no arquivo `/etc/sysconfig/network`.

Teremos nesse arquivo algo do tipo:

```
NETWORKING=yes
FORWARD_IPV4=yes    # not used anymore. see /etc/
sysctl.conf
GATEWAYDEV=eth0
```

A linha mais importante é referente ao `GATEWAYDEV`. Com esta linha informamos que o nosso servidor poderá repassar pacotes para Internet através da interface `eth0`.

Neste mesmo arquivo, colocamos o valor do nosso `GATEWAY`. Se for DHCP o tipo de rede externa, sem problemas... é pego automaticamente. Mas se não for o seu caso, adicione a linha e esse será o nosso Gateway:

```
GATEWAY=200.254.153.101 #
```

Agora o toque final: permitir que o firewall repasse pacotes TCP. Se seu Kernel for 2.2 ou inferior, utilizará o `ipchains`. Veja abaixo a regra:

```
Ipchains -A forward -s 192.168.200.0/24 -d 0/0 -j MASQ
```

Agora, se o seu Kernel for 2.4, usará o `iptables`:

```
iptables -t nat -A POSTROUTING -s 192.168.200.0/24 -d 0/0 -j MASQUERADE
```



Neste momento, você não tem necessidade de entender o que aconteceu, tendo em vista que veremos o dois tipos de firewalls detalhadamente.

Para efetivar as alterações rede, é necessário reiniciar.

```
cds
./network restart
```

Para verificar se as configurações estão OK, primeiro você deve conseguir pingar na rede interna e externa.

Quando você digitar *ipchains -L* ou *iptables -L*, mostrará a regra de mascarar.

E, por último, digite o comando *route -n*.

```
[root@Stato bin]# route -n
```

Tabela de Roteamento IP do Kernel:




Destino	Roteador	MáscaraGen.	Opções	Métrica	Ref	Usa	Iface
200.254.153.192	0.0.0.0	255.255.255.192	U	0	0	0	eth0
192.168.200.0	0.0.0.0	255.255.255.0	U	0	0	0	eth1
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	200.204.153.101	0.0.0.0	UG	0	0	0	eth0

Note que a rede 200.254.153.192 não é um endereço e sim uma rede inteira. Ela é escrita desta forma por causa da máscara de sub-rede 255.255.255.192.









Bom, o mais importante é a quarta linha, que informa sobre o roteador padrão da rede 20.204.153.101, que é o Gateway do servidor **Linux**.

Comando route

Para adicionar ou retirar dados da tabela de roteamento, veja abaixo as formas de usá-lo:

-  **route**: Mostra tabela de roteamento.
-  **route add -net <rd> netmask <mdr> <adaptador>**: Roteia um adaptador local para uma rede de destino.
-  **route add -host <hd> <adaptador>**: Roteia um adaptador local para um host de destino.



-  **route add default gw <rot>:** Estabelece um Gateway default.
-  **route add -net <rd> netmask <mdr> reject:** Não roteia desta rede para a rede mencionada.
-  **route add -host <hd> reject:** Não roteia desta rede para o host mencionado.
-  **rd:** Rede destino.
-  **mdr:** Máscara de rede destino.
-  **adaptador:** *eth0, eth1, ppp0, etc.*
-  **hd:** Host destino.
-  **rot:** Roteador ou Gateway (endereço IP).

Para remover qualquer configuração, use *del* em vez de *add*.

Para o nosso exemplo, poderemos adicionar uma Gateway:

```
route add default gw 200.204.153.101
```

Com isso, adicionamos nosso Gateway padrão.

Se tudo estiver OK, as máquinas cliente já podem se conectar com a Internet ou a rede onde estamos fazendo NAT (falaremos mais a frente sobre isso).

Cliente Windows e Linux - Configurando Gateway

Para configurar os clientes, devemos seguir algumas regrinhas descritas a seguir.

No **Linux**, através do *Linuxconf/Rede/Informações básicas da máquina*, faremos as configurações básicas da rede, que são os DNSs no *etc/resolv.conf*, o router em */etc/network*, onde deve ser informado o **GATEWAY**, no caso 192.168.200.1, que é o nosso servidor **Linux**. Maiores detalhes você encontra no capítulo referente a redes.

Já no Windows, vá ao *Painel de Controle/Redes/TPC/IP/propriedades*:

O primeiro item a configurarmos é o endereçamento IP e a máscara de sub-rede (no nosso caso, será fixo). Posteriormente, teremos um capítulo para implantação de um servidor DHCP, com isso torna-



se desnecessário colocarmos todas configurações do TCP/IP no Windows e Redes no Linuxconf., pois o servidor distribuirá todas as configurações de rede, tanto para as máquinas cliente **Linux** como Windows.

Mas, de qualquer forma, neste capítulo faremos manualmente para que haja um melhor entendimento do que acontece.



Figura 7.3 - configurações de Rede do Windows

Neste caso, colocamos o endereço de IP da máquina cliente. Lembre-se que este IP deve fazer parte da rede liberada no servidor 192.168.200.0. Neste caso, o IP do cliente deverá ir de 192.168.200.2 até 192.168.200.254. Se fossemos usar o DHCP, marcaríamos o item *Obter um IP automaticamente*.

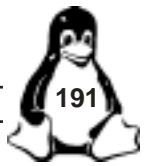


Figura 7.4 - configurações do Gateway

Esta segunda parte é onde informaremos quem é o Gateway da nossa rede, que é o nosso servidor **Linux**, com endereçamento IP 192.168.200.1.

Um fato muito importante, é não esquecer de colocar os DNSs na guia *Configuração do DNS*. Caso contrário, coloque os do seu provedor de acesso, existem alguns gratuitos que podem ser usados sem problemas.

O Windows pedirá para reiniciar. Reinicie a máquina e, quando voltar, tente abrir uma página da Internet. Caso não abra, reveja as configurações. Tente pingar pelo prompt do DOS no servidor e endereços externos.







Firewall

A primeira pergunta é: O que é firewall? No sentido da palavra, firewall é parede de fogo. Seu significado é mais abrangente. Firewall é um dos sistemas de segurança usado para proteger a rede.

O firewall pode ser um software, hardware ou ambos, sendo híbrido. A sua principal função é filtrar pacotes que entram e saem da rede e proteger as informações que trafegam nela.






Além da função de filtrar pacotes no **Linux**, os firewalls (*ipchains* e *iptables*) também são usados para outros serviços, tais como:

-  Mascaramento (Fizemos anteriormente no nosso Gateway);
-  Redirecionamento de portas (Muito útil no proxy transparente);
-  NAT;
-  Balanceamento de carga.

No **Linux**, trabalhamos praticamente com dois firewalls, um para versões de Kernel 2.2 e outra para versão de Kernel 2.4. Na versão 2.2, usamos o *ipchains* e na 2.4, usamos o *iptables*.

Neste primeiro momento, estaremos trabalhando com filtros de pacotes. O filtro de pacotes é um dos principais mecanismos de segurança, mediante regras estipuladas pelo administrador da rede, para impedir ou liberar acessos em portas TCP, UDP e ICMP.

Para criarmos uma regra, basicamente teremos que ter as seguintes informações:

-  Endereço de origem;
-  Endereço de destino;
-  Tipo de protocolo (TCP, UDP e ICMP);
-  Porta;
-  Acesso (Permite, não permite).

Em alguns casos, partes destes dados podem ser omitidas.

Vamos supor que você não queira que sua rede inteira acesse serviços de FTP externo. O que faremos? Primeiramente, os dados:

1. **Endereço de origem:** Minha Rede (192.168.200.0);
2. **Endereço de destino:** Internet (0.0.0.0);



3. Tipo de Protocolo: TCP e UDP;

4. Portas: 20 e 21;

5. Acesso: Negado.

Com este exemplo acima, temos já uma regra que não permitirá que nossa rede acesse servidores FTP externos.

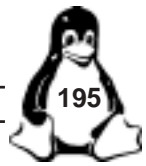
Para verificar quais portas existem, dê uma olhada em `/etc/services`. Abaixo, coloco as principais:

Portas TCP e UDP

Serviço	Porta
tcpmux	1/tcp
echo	7tcp
echo	7/udp
ftp-data	20/tcp
ftp	21/tcp
ssh	22/tcp
ssh	22/udp
telnet	23/tcp
smtp	25/tcp
whois	43/tcp
domain	53/tcp
domain	53/udp
gopher	70/tcp
gopher	70/udp
finger	79/tcp
www	80/tcp
www	80/udp
kerberos	88/tcp
kerberos	88/udp
linuxconf	98/tcp
rtelnet	107/tcp
rtelnet	107/udp
pop-2	109/tcp
pop-2	109/udp
pop-3	110/tcp
pop-3	110/udp
nntp	119/tcp
netbios-ns	137/tcp



Serviço	Porta
netbios-ns	137/udp
netbios-dgm	138/tcp
netbios-dgm	138/udp
netbios-ssn	139/tcp
netbios-ssn	139/udp
imap2	143/tcp
imap2	143/udp
xdmcp	177/tcp
xdmcp	177/udp
irc	194/tcp
irc	194/udp
ipx	213/tcp
ipx	213/udp
imap3	220/tcp
imap3	220/udp
https	443/tcp
https	443/udp
exec	512/tcp
biff	512/udp
login	513/tcp
who	513/udp
Shell	514/tcp
syslog	514/udp
printer	515/tcp
route	520/udp
klogin	543/tcp
kshell	544/tcp
kerberos-adm	749/tcp
swat	901/tcp
radius	1812/tcp
radius	1812/udp
mysql	3306/tcp
mysql	3306/udp
kerberos4	750/udp
kerberos4	750/tcp
kerberos_master	751/udp
kerberos_master	751/tcp
passwd_server	752/udp
krb_prop	754/tcp
krbupdate	760/tcp



Serviço	Porta
kpasswd	761/tcp
ssl-ldap	636/tcp
socks	1080/tcp
postgres	5432/tcp
postgres	5432/udp
fax	4557/tcp
hylafax	4559/tcp
webcache	8080/tcp
webcache	8080/udp
tpoxy	8081/tcp
tpoxy	8081/udp
amanda	10080/udp





Estes são os mais importantes e usados normalmente, mas a lista real é bem maior. Para conhecê-los, acesse o arquivo mencionado anteriormente.

Configurando o Kernel para Suporte à Firewall

Antes de tudo, devemos verificar se o nosso Kernel tem suporte para o firewall propriamente dito.

Vá a um Shell, acesse diretório `/usr/src`. Se estiver em modo gráfico, digite “make xconfig” e, caso esteja em modo texto, “make menuconfig”.

Caso não exista o diretório **Linux**, deverão ser instalados os pacotes abaixo:

-  Automake;
-  Make;
-  Kernel-headers;
-  Kernel-source.








Acesse o item *Network Options* e marque os seguintes itens:

- 1) Packet socket - y.
- 2) Network firewalls - y.
- 3) Unix domain socket - y.
- 4) TCP/IP network - y.



- 5) IP Advanced router - y.
- 6) IP firewalling - y.
- 7) IP Transparent Proxy - y.
- 8) IP masquerading - y.
- 9) IP : ICMP masquerading - y.
- 10) IP: masquerading special modules support - y.
- 11) IP: alias support - y.
- 12) IP: TPC syncookie support - y.
- 13) IP: Allow large windows -y.

Após marcar *yes* em todos os itens, precisamos compilar o Kernel. Para compilá-lo, digite os comandos abaixo um após o outro:

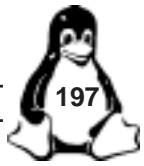
```
 make dep;  
 make clean;  
 make bzImage;  
 make bzlilo;  
 make modules;  
 make_install;  
 shutdown -r now.
```

Deve-se ressaltar que esse processo é um tanto demorado. Por isso, quando for fazê-lo, certifique-se de que não tenha nenhum compromisso...

Outros aspectos a respeito de segurança devem ser vistos. Tais aspectos serão vistos no final deste capítulo. Lembre-se de que um **Linux** mal configurado pode ser comparado com outros sistemas com mínima segurança para o usuário. Dessa forma, sempre configure seu **Linux** de modo correto, primeiro para assegurá-lo quanto a invasões, segundo para não desmerecer o nome **Linux** no mundo dos Sistemas Operacionais.

Muitos usuários instalam e pronto. Não fazem nada a respeito da segurança e, após algum tempo, notam que estão sendo invadidos ou derrubados constantemente.

Mas tenho certeza que se tal usuário tomasse as precauções necessárias, tornaria muito mais difícil tal invasão.






Ipchains

O comando *ipchains* resume-se na linha abaixo:














```
ipchains (tipo) [chain] destino origem ação {opções}
```

As *chains* estão descritas abaixo:

-  **input:** Verifica os pacotes que entram na rede.
-  **output:** Verifica os pacotes que estão saindo da rede.
-  **forward:** Verificam pacotes que entram e saem da rede interna para externa e vice-versa.

Agora, vamos dar uma olhada no tipo de regra, que são as políticas adotadas pelo administrador.

Este item vem antes das *chains*.

-  **-A (append):** Adiciona uma nova regra ao sistema.
-  **-D (delete):** Exclui uma regra já existente no sistema.
-  **-R (replace):** Substitui uma regra existente no sistema.
-  **-I (insert):** Substitui uma regra na mesma posição no *chain*.
-  **-L (list):** Lista as regras existentes no sistema ou do *chain* específico.
-  **-F (flush):** Este comando zera as regras do sistema ou na *chain* específica.
-  **-Z (zero):** Zera uma regra específica.
-  **-N (new-chain):** Cria uma nova regra com nome específico.
-  **-X (delete-chain):** Exclui uma regra com nome específico.
-  **-P (policy):** Torna a regra padrão. Caso não exista uma regra para um pacote em particular, ele usará esta regra padrão.
-  **-M (masquerading):** Esta opção permite ver as conexões correntes mascaradas, usando em conjunto com **-L**.
-  **-S (set tcp tcpfin udp):** Altera os valores de tempo máximo do *masquerade*.
-  **-C (check):** Checa as regras básicas do firewall.



-h (help): Mostra uma pequena ajuda.



-V (version): Mostra a versão do *ipchains*.

Bastante coisa, não é? Mas não se preocupe... é difícil, mas quando é utilizado constantemente se torna habitual, e tudo que se torna hábito fica fácil.

Lembra da regra do FTP. Então, como faríamos? Primeiro invocamos o *ipchains* e o tipo de regra, saída entrada e forward.

```
ipchains -A output
```

Nesta regra acima, será adicionada uma regra ao *chain output*, que é o que sai do sistema. Ficaria algo do tipo:

```
ipchains -A output destino origem porta NEGAR
```

Onde *NEGAR* é a própria ação.

Mas, ainda precisamos de outros atributos para conhecer e poder passar para o firewall que é destino, a origem, qual a ação a ser tomada e que porta ou serviço.

As ações possíveis no *ipchains* são as seguintes:



ACCEPT: Permite a passagem do pacote



REJECT: Não permite a passagem do pacote pelo Firewall e manda um aviso ao endereço que requisitou o acesso.



DENY: Não permite a passagem do pacote pelo Firewall e não manda um aviso ao endereço que solicitou o acesso.



MASQ: Permite a passagem do pacote, mas mascara o IP interno para o externo.

Bem, nossa regra ficaria parecida com a linha abaixo:

```
ipchains -A output origem destino porta REJECT
```

Sim, ficou boa, mas estão faltando os parâmetros, veja-os abaixo:



-p (protocol): Define qual protocolo será tratado pelo firewall.



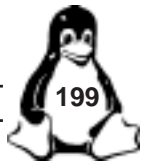
-s (sourcei): Dados de origem , endereço de origem.







-d (destination).








—icmp-type [!] (typename): Permite especificar se o tipo ICMP pode ou não passar pelo firewall. No final destas descrições, temos uma tabela com os tipos, códigos e mensagens ICMP.



-  **-h (ICMP):** Mostra a tabela ICMP.
-  **-j (jump target):** Especifica o destino de uma regra. Redireciona para ação a ser tomada.
-  **-f (fragment):** Trata datagramas fragmentados.
-  **! [regra]:** Inversão. Inverte a regra. Por exemplo, `-s 192.168.200.1`, com o argumento `-s ! 192.168.200.1`. Serão incluídos na regras, todos os endereço diferentes de 192.168.200.1 ou, ainda, `-p ! tcp` (todos os protocolos, exceto o TCP).

Outras opções:

-  **-b (bidirectional):** Permite criar uma regra bidirecional entre origem e destino, evitando criar duas regras, uma para entrada e outra para saída.
-  **-v (verbose).**
-  **-n (numeric):** Visualiza os IPs e portas de endereço.
-  **-l (log):** Toda vez que a regra que contém a opção `-l` for usada, o `ipchains` registra no log a ação. No nosso exemplo, usaremos muito ele.
-  **-o (output [maxsize]):** Opção utilizada para cópia de datagramas.

Ipchains - Tipos de ICMP

Mensagem	Tipo	Código
echo reply(ping)	0	0
destination-unreachable	3	
network-unreachable	3	0
host-unreachable	3	1
protocol-unreachable	3	2
port-unreachable	3	3
fragmentation-needed	3	4
source-route-failed	3	5
network-unknown	3	7
host-unknown	3	7
network-prohibited	3	9
host-prohibited	3	10



Mensagem	Tipo	Código
TOS-network-unreachable	3	11
TOS-host-unreachable	3	12
communication-prohibited	3	13
host-precedence-violation	3	14
Precedence-cutoff	3	15
source-quench	4	0
redirect	5	
network-redirect	5	0
host-redirect	5	1
TOS-network-redirect	5	2
TOS-host-redirect	5	3
echo-request (ping)	8	0
router-advertisement	9	0
router-solicitation	10	0
time-exceeded	11	
Ttl-zero-during-transit	11	1
ttl-zero-during-reassembly	11	1
Parameter-problem	12	
Ip-header-bad	12	0
required-option-missing	12	1
timestamp-request	13	0
timestamp-reply	14	0
address-mask-request	17	0
address-mask-reply	18	0

Com isso, já temos uma boa noção de como fazer no firewall, lembra da regra? Como ele ficaria agora?

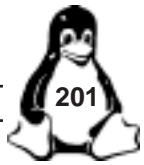
```
ipchains -A input -p tcp -s 192.168.200.1/255.255.255.0  
-d 0.0.0.0/0.0.0.0 21 -j ACCEPT  
ipchains -A input -p udp -s 192.168.200.1/255.255.255.0  
-d 0.0.0.0/0.0.0.0 20 -j ACCEPT
```

Ficou um pouco grande, acho que pode ser melhorada.

```
ipchains -A input -p tcp -s 192.168.200.1 -d 0/0 21 -j  
ACCEPT  
ipchains -A input -p udp -s 192.168.200.1 -d 0/0 20 -j  
ACCEPT
```

Ou ainda:

```
ipchains -A input -p tcp -s 192.168.200.1 -d 0/0 ftp -j  
ACCEPT  
ipchains -A input -p udp -s 192.168.200.1 -d 0/0 ftp-  
data -j ACCEPT
```

Mas temos um problema, nossas regras são adicionadas enquanto a máquina está ligada. Depois de desligada, perdemos toda informação, pois o *ipchains* não guarda as regras adicionadas. Dessa forma, faz-se necessário criar um script, assim, podemos invocá-lo quando quisermos. Crie um arquivo chamado *regras_ipchains* em */usr/bin*.

```
touch /usr/bin/regras_ipchains
chmod 755 /usr/bin/regras_ipchains
```

Agora, adicione na primeira linha do arquivo:

```
#!/bin/bash
```

Todas as regras serão adicionadas neste arquivo e, posteriormente, o arquivo será executado para finalmente adicionar as regras na tabela do firewall.

Antes de começarmos a fazer as regras, vou sugerir trocar de local o arquivo de log do Kernel para que as mensagens que serão trazidas do firewall fiquem mais claras.

Edite o arquivo:

```
/etc/rc.d/init.d/syslog
```

Procure por este trecho dentro do arquivo:

```
gprintf "Starting %s: " "syslogd"
daemon syslogd -m 0 $LOGSERVER $SOCKETS
daemon klogd -f
echo
touch /var/lock/subsys/syslog
```

E altere da seguinte forma :

```
gprintf "Starting %s: " "syslogd" /
daemon syslogd -m 0 $LOGSERVER $SOCKETS
daemon klogd -f /var/log/error
echo
touch /var/lock/subsys/syslog
```

Adicionando a "/" e o path */var/log/error*, todos os erros serão registrados neste arquivo, *error*.

Reinicie o *syslog*.

```
cds
./syslog stop
./syslog start
```

Agora, podemos iniciar nossas regras de firewall. Para iniciar nosso firewall, o primeiro passo é zerar todas as regras existentes anteriormente:



```
#Limpando as regras  
ipchains -F
```

E o segundo passo é colocar as regras que são padrões. É natural que fechemos todas as portas para ir abrindo aos poucos conforme necessidade.

```
#Fechando todas as portas  
ipchains -P input REJECT  
ipchains -P output REJECT  
ipchains -P forward REJECT
```

Agora, como dito anteriormente, iremos usar muito o log, por quê? Porque através dele poderemos aprender mais, quando os erros forem surgindo no arquivo, para posteriormente podermos usá-lo para arrumar nossas regras.

```
#Logando o sistema  
ipchains -A input -j REJECT -l  
ipchains -A output -j REJECT -l
```

Este tipo de regra servirá apenas como aprendizado. Tome muito cuidado quando usar em produção. Álias, não use.

Estas regras estão todas em nosso arquivo. Podemos, agora, rodá-las.

```
./regras_ipchains
```

Suponhamos que sua rede é do tipo classe C, 192.168.200.0, seu servidor 192.168.200.1 e o cliente 192.168.200.3.

Abra um terminal e digite:

```
tail -f /var/log/error
```

Este comando mostrará as últimas dez linhas do arquivo, atualizando automaticamente sempre que houver alterações no arquivo.

Abra o terminal e de um *ping* do servidor para o cliente:

```
ping 192.168.200.3  
ping: sendto: Operation not permitted  
ping: wrote 192.168.200.3 64 chars, ret=-1
```

Com certeza você estará recebendo esta mensagem acima.

Dê uma olhada no terminal onde está o arquivo *error*.

```
<6>Packet log: output REJECT eth1 PROTO=1  
192.168.200.1:8 192.168.200.3:0 L=84 S=0x00 I=1661  
F=0x0000 T=64 (#1)
```



Vamos analisar a primeira coisa informada, que é um log de um pacote *Packet log*. O segundo item informa que é na *chain output*, onde foi rejeitado o pedido pela interface *eth1*, um pacote ICMP.

Dê uma olhada no arquivo */etc/protocols*:

ip	0	IP	# Internet Protocol, Pseudo Protocol Number
icmp	1	ICMP	# Internet Control Message Protocol
igmp	2	IGMP	# Internet Group Management
ggp	3	GGP	# Gateway-Gateway Protocol
ipencap	4	IP-ENCAP	# IP Encapsulated In IP (officially "IP")
st	5	ST	# ST Datagram Mode
tcp	6	TCP	# Transmission Control Protocol
egp	8	EGP	# Exterior Gateway Protocol
pup	12	PUP	# PARC Universal Packet Protocol
udp	17	UDP	# User Datagram Protocol
hmp	20	HMP	# Host Monitoring Protocol
xns-idp	22	XNS-IDP	# Xerox NS IDP
rdp	27	RDP	# "Reliable Datagram" Protocol
iso-tp4	29	ISO-TP4	# ISO Transport Protocol Class 4
xtp	36	XTP	# Xpress Transfer Protocol
ddp	37	DDP	# Datagram Delivery Protocol
idpr-cmtp	39	IDPR-CMTP	# IDPR Control Message Transport
rspf	73	RSPF	# Radio Shortest Path First
vmtp	81	VMTP	# Versatile Message Transport
ospf	89	OSPF	# Open Shortest Path First IGP
ipip	94	IPIP	# Yet Another IP Encapsulation
encap	98	ENCAP	# Yet Another IP Encapsulation

Os mais usados são 1 ICMP, 6 TCP e o 17 UDP.

Agora já sabemos que a regra de saída (*output*) rejeitou o protocolo ICMP.

Vamos olhar o resto do log:

```
192.168.200.1:8 192.168.200.3:0
```

O pacote ICMP, do tipo 8, partindo da máquina 192.168.200.1, com destino para 192.168.200.3, com ICMP do tipo 0.







Qual tipo é esse? Vamos dar uma olhada naquela tabela de tipo de mensagens ICMP.



Segundo a tabela, o tipo 0 é *echo reply* e o tipo 8 *echo request*, ou seja, o *ping* de saída é respondido.

O que fazer neste momento? Não consigo pingar minha rede! E agora? Acalme-se, só precisamos adicionar uma regra para este caso.

Tenho certeza de que você já entendeu qual regra que devemos liberar. Então, vamos fazê-la:

-  **ipchains** (A output): a regra *input* que não deixou o pacote sair.
-  **-p** (icmp): É o protocolo que estamos usando.
-  **-s** (192.168.200.1 8): Endereço e porta de origem.
-  **-s** (192.168.200.3 0): Endereço e porta de destino.
-  **-j** (ACCEPT): Aceitar.
-  **ipchains -A output -p icmp -s 192.168.200.1 8 -d 192.168.200.3 0 -j ACCEPT**

Agora, *ping* novamente:

```
ping 192.168.200.3
```

O que ocorreu? O *ping* ficou parado, não?

Vamos olhar o log:

```
<6>Packet log: input REJECT eth1 PROTO=1 192.168.200.3:0  
192.168.200.1:0 L=84 S=0x00 I=26113 F=0x0000 T=32 (#1)
```

Agora parece que a regra *input* (o que entra) rejeita o pacote, partindo de 192.168.200.3.

icmp do tipo 0 para a máquina 192.168.200.1, protocola icmp do tipo 0.

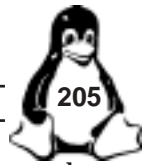
Então, vamos adicionar a regra:

```
ipchains -A input -p icmp -s 192.168.200.3 0 -d  
192.168.200.1 0 -j ACCEPT
```

Adicione esta linha ao script e rode-o em seguida.

Para completar, tente pingar novamente:

```
ping 192.168.200.3  
64 bytes from 192.168.200.3: icmp_seq=17 ttl=32 time=0.6 ms  
64 bytes from 192.168.200.3: icmp_seq=18 ttl=32 time=0.5 ms  
64 bytes from 192.168.200.3: icmp_seq=19 ttl=32 time=0.5 ms
```



Parece tudo normal agora. Vamos fazer outro teste: pingar da estação para o servidor.

```
Ping 192.168.200.1
Request timeout
```

O que aconteceu? Nós havíamos permitido somente saída e a entrada da resposta para o servidor.

Olhemos o log.

```
<6>Packet log: input REJECT eth1 PROTO=1 192.168.200.3:8
192.168.200.1:0 L=60 S=0x00 I=61704 F=0x0000 T=32 (#2)
```

A regra *input* rejeita, pacotes ICMP do tipo 8 da máquina 192.168.200 3 para a máquina 192.168.200.1, protocolo ICMP do tipo 0.

Adicionemos a regra:

```
ipchains -A input -p icmp -s 192.168.200.3 8 -d
192.168.200.1 0 -j ACCEPT
```

Mesmo assim, quando pingamos, volta o erro. Analisemos o log novamente:

```
<6>Packet log: output REJECT eth1 PROTO=1
192.168.200.1:0 192.168.200.3:0 L=84 S=0x00 I=1661
F=0x0000 T=64 (#1)
```

Agora, o problema é quanto à resposta do servidor, pois a regra *output* não previa tal resposta.

Teremos que adicionar mais uma regra. No total, teremos quatro regras para permitir o ping entre uma e outra máquina:

```
ipchains -A output -p icmp -s 192.168.200.1 8 -d
192.168.200.3 0 -j ACCEPT

ipchains -A input -p icmp -s 192.168.200.3 0 -d
192.168.200.1 0 -j ACCEPT

ipchains -A input -p icmp -s 192.168.200.3 8 -d
192.168.200.1 0 -j ACCEPT

ipchains -A output -p icmp -s 192.168.200.1 0 -d
192.168.200.3 0 -j ACCEPT
```

Podemos criar, também, uma regra bidirecional:

```
ipchains -A input -b -p icmp -s 192.168.200.1 -d
192.168.200.3 -j ACCEPT

ipchains -A output -b -p icmp -s 192.168.200.1 -d
192.168.200.3 -j ACCEPT
```



Agora, vamos começar a liberar outras portas do nosso firewall. No cliente, tente pingar o site da Conectiva.

```
ping www.conectiva.com.br
```

Agora, olhe o log:

```
<6>Packet log: input REJECT eth1 PROTO=17
192.168.200.3:1031 200.204.0.138:53 L=60 S=0x00 I=5385
F=0x0000 T=32 (#3)
```

Neste caso, a rede interna tenta acessar um endereço externo, protocolo UDP (17) na porta 53.

O que é isso? A porta 53 é por onde passam as solicitações para o DNS. A primeira coisa que ele faz é tentar resolver o nome www.conectiva.com.br. Posteriormente, ele tentará pingar com o protocolo ICMP. De antemão, sabemos, então, que deverá existir duas regras de *input* para ele poder pingar na rede.

```
ipchains -A input -p udp -s 192.168.200.0/24 -d 0/0 53 -
j ACCEPT
```

```
ipchains -A input -p icmp -s 192.168.200.0/24 -d 0/0 -j
ACCEPT
```

Mas desse jeito vai ser um pouco complicado.

Vamos fazer o seguinte: é necessário que a placa de rede que possui acesso à Internet tenha permissão para sair e pingar externamente. Você pode tratar diretamente pelo IP, ou talvez queira tratar pela interface. Vou supor que a interface que possui acesso à Internet seja *eth0.m*.

```
ipchains -A output -i eth0 -d 0/0 53 - j ACCEPT
```

```
ipchains -A input -s 0/0 53 -i eth0 -j ACCEPT
```

Agora, é necessário que a rede interna possa ter acesso a porta 53 também:

```
ipchains -A input 192.168.200.0/2+4
```

Enfim, vão existir vários passos para que a rede interna possa pingar externamente. Olhe as regras abaixo como ficarão, seguindo o nosso padrão. Mas antes de começar, note que no exemplo abaixo estou usando uma variável *\$eth0*. Isso mesmo, é possível usar variáveis em vez do próprio endereço IP.

No começo do script, temos que informar seu conteúdo:

```
eth0="200.204.159.001"
```



E depois usá-la. Veja abaixo:

```
ipchains -A input -p udp -s 192.168.200.0/24 -d 0/0 53 -j ACCEPT
ipchains -A input -p udp -s 0/0 53 -d 192.168.200.0/24 -j ACCEPT
ipchains -A input -p udp -s 0/0 53 -d $eth0 -j ACCEPT
ipchains -A output -p udp -s $eth0 -d 0/0 53 -j ACCEPT
ipchains -A output -p udp -s 0/0 53 -d 192.168.200.0/24 -j ACCEPT
ipchains -A input -p icmp -s 192.168.200.0/24 -d 0/0 -j ACCEPT
ipchains -A input -p icmp -s 0/0 -d $eth0 -j ACCEPT
ipchains -A output -p icmp -s 0/0 -d 192.168.200.0/24 -j ACCEPT
ipchains -A output -p icmp -s $eth0 -d 0/0 -j ACCEPT
```

O primeiro passo foi liberar o DNS tanto para a rede interna como para a o endereço real.

Os passos foram os seguintes:

- 1) Informamos o conteúdo da variável *eth0*;
- 2) Liberamos a entrada de pacotes da rede interna para externa na porta 53;
- 3) Liberamos a entrada de pacotes da rede externa na porta 53 para a rede interna;
- 4) Liberamos a saída de pacotes do IP real para rede externa na porta 53;
- 5) Liberamos a saída de pacotes da rede externa na porta 53 para a rede interna;
- 6) Liberamos a entrada de pacotes do tipo ICMP da rede interna para a externa;
- 7) Liberamos a entrada de pacotes do tipo ICMP da rede externa para o IP real;
- 8) Liberamos a saída de pacotes ICMP da rede externa para rede interna;
- 9) Liberamos a saída de pacotes ICMP do IP real para a rede externa.



Não necessariamente ocorrem nesta ordem. Você pode liberar uma regra de cada vez, verificando o log ocorrido com cada regra adicionada.

Para ficar um pouco mais claro, vamos liberar o acesso à porta 80 WWW para rede interna e para o servidor, mas não liberaremos nossa porta 80, a não ser que tenhamos um servidor de WEB.

Vá a uma estação e digite em um browser:

www.conectiva.com.br

O log nos mostra, que:

```
Packet log: input REJECT eth1 PROTO=6 192.168.200.3:1073
200.250.58.152:80 L=44 S=0x00 I=27658 F=0x4000 T=32 SYN
(#8)
```

As regras de entrada estão rejeitando pacotes TCP (16) de origem 192.168.200.3 para 200.250.58..152 na porta 80, então, vamos liberar:

```
ipchains -A input -p tcp -s 192.168.200.0/24 -d 0.0 www
-j ACCEPT
```

Note que usa a origem como 192.168.200.0/24, ou seja, a rede inteira, e no lugar de WWW pode-se colocar a porta 80 sem problemas.

Adicionado, rode o script e tente acessar a página novamente:

Vejamos o log:

```
>Packet log: output REJECT eth0 PROTO=6
200.204.159.001:61019 200.250.58.152:80 L=44 S=0x00
I=30474 F=0x4000 T=31 SYN (#7)
```

Parece que nosso endereço real não consegue sair para acessar a porta 80, então, liberemos também:

```
ipchains -A output -p tcp -s $eth0 -d 0/0 www -j ACCEPT
```

Olhando novamente o log, vemos o seguinte:

```
6>Packet log: input REJECT eth0 PROTO=6
200.250.58.152:80 200.204.153.203:61021 L=44 S=0x00
I=27762 F=0x4000 T=53 (#9)
```

Os pacotes que estão partindo da porta 80 de um endereço externo não conseguem entrar em nosso endereço real. OK! Mais uma vez... Vamos abrir esta porta nesta *chain*:

```
ipchains -A input -p tcp -s 0/0 www -d $eth0 -j ACCEPT
```

Ainda não foi desta vez, não é? Olhe o log.



O que ocorre desta vez, é que a rede externa, ou Internet, não tem acesso à rede externa, o firewall não deixa sair os pacotes para a rede interna.

```
<6>Packet log: output REJECT eth1 PROTO=6
200.250.58.152:80 192.168.200.3:1079 L=44 S=0x00 I=40588
F=0x4000 T=52 (#8)
```

Vamos criar uma última regra:

```
ipchains -a output -p tcp -s 0/0 www -d 192.168.200.0/24
-j ACCEPT
```

Agora sim, a página foi aberta. O que ocorreu:

```
192.168.200.3 ----->http Servidor -----> WWW
WWW-----> Servidor -----> 192.168.200.3
```

Em resumo é isto que está acima. Fazemos a solicitação para o servidor, ele abre, passamos para a Internet e, posteriormente, a Internet retorna para nós, passando pelo servidor.

As quatro regras são essas:

```
ipchains -A input -p tcp -s 192.168.200.0/24 -d 0/0 www
-j ACCEPT
ipchains -A output -p tcp -s $eth0 -d 0/0 www -j ACCEPT
ipchains -A input -p tcp -s 0/0 www -d $eth0 -j ACCEPT
ipchains -A output -p tcp -s 0/0 www -d 192.168.200.0/24
-j ACCEPT
```

Estas regras estão ordenadas conforme foi solicitado pelo firewall. Agora, ficou bem fácil, não é?

Para liberarmos qualquer outro serviço, como POP e SMTP, devemos fazer o mesmo procedimento, liberando apenas as porta 110 e 25.

Para liberarmos um servidor de web, por exemplo, supondo que temos um servidor Apache rodando, teremos que fazer algo do tipo:

```
ipchains -A input -p tcp 0/0 -d $eth0 80 -j ACCEPT
ipchains -A output -p tcp -s$eth0 80 -d 0/0 -j ACCEPT
```

Aqui, na realidade, abrimos uma porta do nosso lado para acesso da rede externa.

No final deste capítulo, existe um script com as portas que são necessárias não para um rede corporativa, mas para uma rede caseira, onde queremos fazer download de MP3, softwares, etc.



Ipchains - Script Caseiro

Existem módulos que gerenciam melhor algumas tarefas, tais como FTP, ICQ. Para carregá-lo, deve-se usar o comando *modprobe ip_masq_ftp*. Abaixo, segue uma lista dos módulos:

FTP	=- ip_masq_ftp
Real Audio	=- ip_masq_raidio
IRC	=- ip_masq_irq
Quake	=- ip_masq_quake
Quake Word, II e III	=- ip_masq_quake 2600, 2700, 27910, 27960
Cuseeme	=- ip_masq_cuseeme
Video Live	=- ip_masq_vcolive

Você pode estar utilizando tais módulos, mas não esqueça de liberar cada porta para cada serviço. Abaixo, coloco um arquivo para regras de firewall, que é usado caseiramente, ou seja, não-corporativo.

```
#!/bin/bash
eth0="200.204.000.001" #Este ip é fictício
Ipint="192.168.200.0/24"

##ativar ip_v4
echo "1" >/proc/sys/net/ipv4/ip_forward

#Protegendo contra Spoofing
if [ -e /proc/sys/net/ipv4/conf/all/rp_filter ]; then
    echo "Carregando proteção contra Ataques
    Spoofing"
    for f in /proc/sys/net/ipv4/conf/*; do
        echo 1 > $f/rp_filter;
        echo 0 > $f/accept_redirects
        echo 0 > $f/accept_source_route
        echo 1 > $f/log_martians
        done
        echo "Carregado"
    else
        echo "Problemas com os arquivos de
configuração"
        /sbin/sulogin $CONSOLE
    fi

echo 1 > /proc/sys/net/ipv4/
icmp_ignore_bogus_error_responses
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```



#Limpendo as regras anteriores

```
ipchains -F
```

#Fechando todas as portas

```
ipchains -P input REJECT
ipchains -P output REJECT
ipchains -P forward REJECT
```

```
echo "Habilitando DNS"
```

#Abrindo DNS

```
ipchains -A input -p tcp -s 0/0 53 -d 192.168.200.0/0 -j ACCEPT
ipchains -A input -p udp -s 0/0 53 -d 192.168.200.0/0 -j ACCEPT
ipchains -A output -p tcp -s 192.168.200.0/0 -d 0/0 53 -j ACCEPT
ipchains -A output -p udp -s 192.168.200.0/0 -d 0/0 53 -j ACCEPT
```

Abrindo DNS para placa externa

```
ipchains -A input -p tcp -s 0/0 53 -d $eth0 -j ACCEPT
ipchains -A input -p udp -s 0/0 53 -d $eth0 -j ACCEPT
ipchains -A output -p tcp -s $eth0 -d 0/0 53 -j ACCEPT
ipchains -A output -p udp -s $eth0 -d 0/0 53 -j ACCEPT
echo "Compartilhando Internet"
```

#Compartilhando a Internet

```
ipchains -A forward -d 0/0 -j MASQ
echo "Liberando icmp"
```

#Liberando icmp

```
ipchains -A input -p icmp -s 192.168.200.0/24 -d 0/0 -j ACCEPT
ipchains -A output -p icmp -s 0/0 -d 192.168.200.0/24 -j ACCEPT
ipchains -A input -p icmp -s 0/0 -d 0/0 -i eth0 -j ACCEPT
ipchains -A input -p icmp -s $eth0 -d $eth0 -j ACCEPT
ipchains -A output -p icmp -s $eth0 -d $eth0 -j ACCEPT
ipchains -A output -p icmp -s $eth0 -d 0/0 -j ACCEPT
```



#Liberando Samba

```
ipchains -A input -p tcp -s 192.168.200.0/24 -d
192.168.200.0/24 138 -j ACCEPT -i eth1

ipchains -A input -p udp -s 192.168.200.0/24 -d
192.168.200.0/24 138 -j ACCEPT -i eth1

ipchains -A input -p tcp -s 192.168.200.0/24 -d
192.168.200.0/24 139 -j ACCEPT -i eth1

ipchains -A input -p tcp -s 192.168.200.0/24 -d
192.168.200.0/24 137 -j ACCEPT -i eth1

ipchains -A input -p udp -s 192.168.200.0/24 -d
192.168.200.0/24 137 -j ACCEPT -i eth1

ipchains -A input -p tcp -s $eth0 -d 200.204.153.255 138
-j ACCEPT

ipchains -A input -p udp -s $eth0 -d 200.204.153.255 138
-j ACCEPT

ipchains -A output -p udp -s 192.168.200.0/24 -d
192.168.200.0/24 137 -j ACCEPT

ipchains -A output -p udp -s 192.168.200.0/24 -d
192.168.200.0/24 138 -j ACCEPT

ipchains -A output -p udp -s $eth0 -d 200.204.153.255 -j
ACCEPT

echo "Liberando www"
```

#Liberando HTTP

```
ipchains -A input -p tcp -s 192.168.200.0/24 -d 0/0 80 -
j ACCEPT

ipchains -A input -p tcp -s 0/0 80 -d $eth0 -j ACCEPT

ipchains -A output -p tcp -s $eth0 -d 0/0 80 -j ACCEPT

ipchains -A output -p tcp -s 0/0 80 -d 192.168.200.0/0 -
j ACCEPT
```

Liberando Ssl

```
ipchains -A input -p tcp -s 0/0 443 -d $eth0 -j ACCEPT

ipchains -A input -p tcp -s 192.168.200.0/24 -d 0/0 443
-j ACCEPT
```

#Carregando módulos

```
modprobe ip_masq_ftp 1 >/dev/null
echo "Liberando FTP"
```



```
ipchains -A input -p tcp -s 192.168.200.0/24 -d 0/0 21 -  
j ACCEPT  
ipchains -A input -p tcp -s 192.168.200.0/24 -d 0/0 20 -  
j ACCEPT  
ipchains -A input -p tcp -s 0/0 21 -d $eth0 -j ACCEPT  
ipchains -A input -p tcp -s 0/0 20 -d $eth0 -j ACCEPT  
ipchains -A output -p tcp -s $eth0 -d 0/0 20 -j ACCEPT  
ipchains -A output -p tcp -s $eth0 -d 0/0 21 -j ACCEPT  
ipchains -A output -p tcp -s 0/0 21 -d 192.168.200.0/24  
-j ACCEPT  
ipchains -A output -p tcp -s 0/0 20 -d 192.168.200.0/24  
-j ACCEPT
```

Liberando VNC

```
ipchains -A input -p tcp -s 192.168.200.2 5800 -d  
192.168.200.1 -j ACCEPT  
ipchains -A input -p tcp -s 192.168.200.2 5900 -d  
192.168.200.1 -j ACCEPT  
ipchains -A output -p tcp -s 192.168.200.1 -d  
192.168.200.2 5800 -j ACCEPT  
ipchains -A output -p tcp -s 192.168.200.1 -d  
192.168.200.2 5900 -j ACCEPT
```

#Liberando Correio

```
ipchains -A input -p tcp -s 0/0 110 -d $eth0 -j ACCEPT  
ipchains -A input -p tcp -s 192.168.200.0/24 -d 0/0 110  
-j ACCEPT  
ipchains -A input -p tcp -s 0/0 25 -d $eth0 -j ACCEPT  
ipchains -A output -p tcp -s $eth0 -d 0/0 110 -j ACCEPT  
ipchains -A output -p tcp -s $eth0 -d 0/0 25 -j ACCEPT  
ipchains -A output -p tcp -s $Ipint -d 192.168.200.3 110  
-j ACCEPT  
ipchains -A output -p tcp -s $Ipint -d 192.168.200.3 25  
-j ACCEPT  
ipchains -A input -p tcp -s 192.168.200.3 -d 0/0 110 -j  
ACCEPT  
ipchains -A input -p tcp -s 192.168.200.3 -d 0/0 25 -j  
ACCEPT
```



```
ipchains -A output -p tcp -s 0/0 -d 192.168.200.3 25 -j  
ACCEPT
```

#Abrindo ICQ

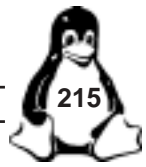
```
ipchains -A output -p tcp -s $eth0 -d 0/0 4000 -j ACCEPT  
ipchains -A output -p udp -s $eth0 -d 0/0 4000 -j ACCEPT  
ipchains -A input -p tcp -s 0/0 4000 -d 0$eth0 -j  
ACCEPT  
ipchains -A input -p udp -s 0/0 4000 -d $eth0 -j ACCEPT
```

Abrindo Morpheus

```
ipchains -A input -s 192.168.200.0/24 -d 192.168.200.1 -  
i eth1 -j ACCEPT  
ipchains -A input -p tcp -s 192.168.200.0/24 -d 0/0 1214  
-j ACCEPT  
ipchains -A output -p tcp -s $eth0 -d 0/0 1214 -j  
ACCEPT  
ipchains -A input -p tcp -s 0/0 1214 -d $eth0 -j ACCEPT  
ipchains -A output -p tcp -s 0/0 1214 -d 192.168.200.0/  
24 -j ACCEPT  
ipchains -A input -p tcp -s 192.168.200.0/24 -d 0/0 1075  
-j ACCEPT
```

#Abrindo WinMX

```
ipchains -A input -p tcp -s 192.168.200.0/24 -d 0/0  
7729:7732 -j ACCEPT  
ipchains -A output -p tcp -s $eth0 -d 0/0 7729:7732 -j  
ACCEPT  
ipchains -A input -p tcp -s 0/0 7729:7732 -d $eth0 -j  
ACCEPT  
ipchains -A output -p tcp -s 0/0 7729:7732 -d  
192.168.200.0/24 -j ACCEPT  
ipchains -A input -p tcp -s $Ipint -d 0/0 6699 -j ACCEPT  
ipchains -A output -p tcp -s $eth0 -d 0/0 6699 -j ACCEPT  
ipchains -A input -p tcp -s 0/0 6699 -d $eth0 -j ACCEPT  
ipchains -A output -p tcp -s 0/0 6699 -d $Ipint -j  
ACCEPT
```



#Abindo WinMx para fora

```
ipchains -A input -p tcp -s 0/0 -d $eth0 6699 -j ACCEPT
ipchains -A output -p tcp -s $eth0 6699 -d 0/0 -j ACCEPT
```

#Abrindo APT-GET

```
ipchains -A output -s $eth0 -d 131.211.28.48 -j ACCEPT
ipchains -A input -s 131.211.28.48 -d $eth0 -j ACCEPT
```

#utilizando -l para logar tentativas no sistema

```
ipchains -A input -j REJECT -l
ipchains -A output -j REJECT -l
ipchains -A forward -j REJECT -l
```

Logo no começo do arquivo temos o item *#Protegendo contra Spoofing*. Este item será visto mais à frente, no item de segurança.

Temos nossas regras. Agora é só aplicá-las?

Ipchains - Criando um Daemon

Pode ser ainda melhor. Abaixo, tenho um script simulando um *daemon*, ou seja, um serviço onde pode ter as opções *start*, *stop* e *status*.

Primeiramente, crie um arquivo chamado *firewall*, por exemplo. Algumas novas distribuições já possuem um *daemon* chamado *firewall*, dessa forma, será necessário criar um com outro nome.

O local onde deve ser criado o arquivo é em:

```
/etc/rc.d/init.d
#! /bin/sh

#
# pidfile: /var/run/iptablesless.pid

. /etc/rc.d/init.d/functions
. /etc/sysconfig/network

if [ ${NETWORKING} = "no" ]
then
    exit 0
fi

case "$1" in
    start)
```



```
gprintf "Iniciando o serviço de %s: " "Firewall"
    daemon ipchains_regras
    touch /var/lock/subsys/firewall
    echo
        /usr/bin/ipchains_regras

    ;;
stop)
gprintf "Parando o serviço de %s: " "Firewall"
    daemon ipchains_regras
    rm -r /var/lock/subsys/firewall
    echo
        /sbin/ipchains -F
        ipchains -P forward ACCEPT
        ipchains -P input ACCEPT
        ipchains -P output ACCEPT
    ;;
status)
    if [ -e /var/lock/subsys/firewall ]; then
        echo "O firewall está rodando"
    else
        echo "O firewall está parado"
    fi
    ;;
*)
    gprintf "Uso: firewall (start|stopi|status)"
    echo
        ;;
esac
exit 0
```

Vamos analisar o arquivo:

A primeira coisa a ser feita é a análise de uma variável de ambiente, que nos informa se a rede está OK ou não. Logo após a análise, entramos no case, caso *start*, *stop* e *status*. Caso seja escolhido *start*, será criando um arquivo em:

```
var/lock/subsys/firewall
```

E o nosso script será executado.

Caso seja escolhido *stop*, será deletado o arquivo:

```
/var/lock/subsys/firewall
```

Em seguida, todas as regras `/sbin/ipchains -F` devem ser limpas e, por último, mudado o padrão de regras para aceitar tudo:

```
ipchains -P forward ACCEPT
ipchains -P input ACCEPT
ipchains -P output ACCEPT
```




Caso seja escolhido *status*, será verificada a existência do arquivo:

```
/var/lock/subsys/firewall
```

Se existir, informará que o firewall está rodando, caso contrário, informará que o firewall está parado. Por isso, nas rotinas *start* e *stop* criamos e deletamos este arquivo.

Por último, temos o item “*”, que se refere a todos os outros itens que forem utilizados com este script. Se não for *start*, *stop* e *status*, ele entrará neste procedimento, que informará ao usuário que coloque *start*, *stop* ou *status*.

Com este exemplo, agora é possível iniciar e parar quando quisermos utilizar o firewall.

Entraremos, agora, em um outro item que é muito necessário na rede: o redirecionamento de servidores ou NAT.

NAT

O que é NAT? Netware Address Translation. Extremamente útil quando desejamos colocar um serviço, seja ele WEB, FTP ou outros que estão localizados em endereços falsos atrás do firewall, disponibilizando para a Internet.

Podemos ter, por exemplo, um servidor WEB no endereço 192.168.200.10 e disponibilizarmos para Internet através de um endereçamento IP válido com 200.254.153.003. Para isso, podemos utilizar vários softwares, por exemplo, o *rinetd*, *ipmasqadm*, *redir*, etc.

No nosso caso, falaremos do *ipmasqadm* e o *rinetd*.

NAT com Ipmasqadm

O *ipmasqadm* é a atualização do *portfw*, utilizado no Kernel 2.0.x. No Kernel 2.2, o mesmo foi substituído e acrescentado sob a forma de um módulo experimental, que não o torna dispensável, podendo ser usado sem problemas.

Vamos começar pelo *ipmasqadm*. Verifique se o mesmo está instalado em sua máquina:

```
rpm -qa | grep ipmasqadm
```

Caso não esteja:

```
apt-get install ipmasqadm # faça download dois pacotes  
rpm no site da conectiva ou no rpmfind.
```



Verifique se o comando está disponível para o Shell:






```
ipmasqadm portfw -h
Usage: portfw -a -P PROTO -L LADDR LPORT -R RADDR
RPORT [-p PREF] add entry
       portfw -d -P PROTO -L LADDR LPORT [-R RADDR
RPORT]      delete entry
       portfw -f
clear table
       portfw -l
list table
       portfw <args> -n
no names
PROTO is the protocol, can be "tcp" or "udp"
LADDR is the local interface receiving packets to be
forwarded.
LPORT is the port being redirected.
RADDR is the remote address.
RPORT is the port being redirected to.
PREF  is the preference level (load balancing,
default=10)
```

Teremos uma resposta, como mencionado acima.




Sua sintaxe é como a descrita abaixo:

```
ipmasqadm portfw -[o] -P PROTO -L LADDR LPORT -R RADDR
RPORT [-p PREF] add entry
```




Onde temos:

-  **-a:** Adiciona uma entrada.
-  **-d:** Exclui uma entrada.
-  **-f:** Exclui todas as entradas.
-  **-l:** Lista as entradas existentes.
-  **-n:** Não utiliza uma tabela de nomes (DNS hosts) no sistema.

Os comandos estão abaixo descritos:

-  **-P PROTO:** Define qual o protocolo deverá ser utilizado (TCP ou UDP).
-  **-L LADDR:** Define qual interface de rede deverá ser redirecionada internamente (endereço IP).
-  **LPORT:** Define qual porta TCP ou UDP será redirecionada.



-  **-R RADDR:** Define qual interface de rede responderá pelo roteamento (endereço IP inválido).
-  **RPORT:** Define qual porta TPC ou UPD será redirecionada a porte de origem.
-  **-p PREF:** Nível de preferência (padrão 10).

Vamos supor, como dito anteriormente, que tenhamos um servidor WEB no endereço 192.168.200.10 e o disponibilizamos para Internet através de um endereçamento IP válido, com 200.254.153.003.

```
ipmasqadm portfw -a -P tcp -L 200.254.153.003 80 -R  
192.168.200.10 80
```

Agora, podemos adicionar ao nosso script já criado. Vamos supor que tenhamos um servidor WEB já citado e um servidor de SMTP.

```
echo "Iniciando NAT"
```

#Limpa regra

```
ipmasqadm portfw -f
```

#Redireciona o a porta 80 para 192.168.200.3

```
ipmasqadm portfw -a -P tcp -L 200.254.153.003 80 -R  
192.168.200.10 80
```

#Redireciona o servidor SMTP na porta 25 para 192.168.200.4

```
ipmasqadm portfw -a -P tcp -L 200.254.153.003 25 -R  
192.168.200.4 25  
echo "Regra ipmasqadm ok"
```

NAT com ritined

Agora, veremos como funciona o *ritnetd*. O *rinetd* é um *daemon* de único processo que roda no *server/firewall/router*. Usando o *I/O nonblocking*, pode administrar um grande número de conexões sem um grande consumo dos recursos do Server. Este NÃO redireciona FTP, pois o serviço de FTP necessita de mais de um soquete.

Para instalá-lo, faça download do RPM em sua página oficial: <http://www.boutell.com/rinetd> ou em <http://www.rpmfind.net>.

A configuração é bem simples e baseada em um único arquivo localizado em */etc/rinetd.conf*.



COMEÇO DO ARQUIVO DE CONFIGURAÇÃO -

```
#
# Arquivo de Configuração /etc/rinetd.conf
#
# Ex Default:

IP_VERDADEIRO 80 IP_FALSO 80

# OU

200.xxx.xxx.xxx 80 192.168.200.3 80
#
#####Fim do Arquivo
#####
```

Veja como é fácil utilizar o *rinetd*. Colocamos o IP verdadeiro, a porta que queremos redirecionar o IP da máquina que receberá este redirecionamento e a porta que será utilizada.

Veja abaixo configurações de outras portas:

```
# Redirecionamento do serviço TELNET
# IP na porta do TELNET == 23

0.0.0.0 23 192.168.1.10 23
```

Pode-se colocar o nome do serviço ao invés da porta. Veja o exemplo:

```
# no caso 80 == www
200.xxx.xxx.xxx www 192.168.200.3 www
```

Para iniciar o serviço, vá em:

```
/etc/rc.d/init.d
./rinetd start
```

Os arquivos de log ficam em:

```
/var/log/rinetd.log.
```

Para torná-lo automático, ou seja, iniciá-lo automaticamente quando a máquina for bootada, use o aplicativo:

```
ntsysv
```

E marque o item *rinetd* para que possa ser inicializado junto com a máquina. Tanto nesse caso como no *ipmasqadm*, será necessário tratar isso no firewall, liberando o repasse dessas informações para que tudo funcione bem.



Outra observação muito importante é que, se você, por exemplo, for redirecionar o servidor Apache para um IP inválido, a máquina que redireciona, o firewall, não deve ter o Apache startado, pois causará conflito com o *rinetd* anulando assim suas configurações.

Com esses dois softwares, já é possível fazer NAT de um IP válido para um IP inválido.

Segurança no Firewall

Veremos, a seguir, o que significa aquela rotina no começo do script de exemplo do firewall. Isso fará com que vejamos outros aspectos de segurança para melhorá-lo ainda mais.

```
#Protegendo contra Spoofing
if [ -e /proc/sys/net/ipv4/conf/all/rp_filter ]; then
    echo "Carregando proteção contra Ataques
    Spoofing"
    for f in /proc/sys/net/ipv4/conf/*; do
        echo 1 > $f/rp_filter;
        echo 0 > $f/accept_redirects
        echo 0 > $f/accept_source_route
        echo 1 > $f/log_martians
        done
        echo "Carregado"
    else
        echo "Problemas com os arquivos de
configuração"
        /sbin/sulogin $CONSOLE
    fi

echo 1 > /proc/sys/net/ipv4/
icmp_ignore_bogus_error_responses
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```


Uma forma de protegermos nossa rede, primeiramente, é desabilitando serviços desnecessários, como Telnet, FTP, Finger, RPC, Shell, login, Apache, Sendmail, Imap, POP3, rshell, rlogin, Auth, etc. Eles podem se tornar muito perigosos em determinados ataques.


Posteriormente, podemos setar os arquivos que existem no script que serão explicados abaixo.


Temos como objetivo proteger nosso servidor contra ataques DoS, ou Negação de Serviço, IP Spoofing ou Exploit, Sniffer.



Segue abaixo uma breve explicação do que é cada item:

 **DoS:** Negação de Serviço (Denial of Service): Quando há ataques de negação de serviço, não é necessariamente com objetivo de ganhar acesso privilegiado à máquina ou acesso a dados pessoais, mas para evitar que usuários que podem usar serviços oferecidos por tais máquinas o utilizem. Pode acontecer de várias formas, mandando uma grande quantidade de dados pela rede (lixo, flood), causando a exaustão do número de processos através da requisição intensa de serviços ou mesmo causando danos físicos a partes da rede. Enfim, consiste, basicamente, em atacar certo serviço, de forma a, geralmente, travar todo o sistema ou evitar certo privilégio de alguns usuários, como o root de logar no sistema. Um ataque muito comum é o envio de solicitações de páginas WEB em um servidor para um IP falso, ou melhor, um IP inválido, do tipo 192.168.200.1.

 **Explorando Confiança (Exploiting of Trust):** Componentes da rede sempre criam uma relação de confiança entre eles. Por exemplo, antes de executar algum comando, um computador checa em outro se determinado atributo está setado para poder continuar sua tarefa. Aparecendo como sendo esse segundo computador e “simulando” seu comportamento, você pode obter acesso não privilegiado ao sistema.

 **Packet Sniffer:** Um Sniffer de pacotes é um programa que “escuta” os dados que trafegam pela rede. Tais dados podem incluir nomes, senhas, informações secretas, como contas bancárias e dados mais pessoais e não públicos.

Quanto ao script, vamos começar analisando a primeira linha *for*: `for f in /proc/sys/net/ipv4/conf/*`. O que significa tudo isso?

Se olharmos dentro do diretório `/proc/sys/net/ipv4/conf`, teremos outros tantos diretórios, tais como *all*, *default*, *eth0*, *eth1* e *lo*. Em cada diretório, existem os arquivos a serem configurados como *rp_filter*. Sem *for*, teríamos que entrar em cada um para setar a nossa opção.

Nossa primeira opção é o *rp_filter*, posteriormente o *accept_redirects* e assim por diante.

```
echo 1 > $f/rp_filter;
```



Quando digitamos `echo 1>`, estamos enviando o valor `1` para o arquivo `rp_filter`. Esta opção verifica o endereço de origem do pacote, prevenindo contra IP Spoofing. Para ativar o serviço, devemos setar com `1` e, para desativá-lo, setamos com `0`.

```
echo 0 > $f/accept_redirects
```

Esta opção decide se o Kernel aceita redirecionar mensagem ICMP ou não.

O `1` ativa e `0` desativa.

```
echo 0 > $F/accept_source_route
```

Esta opção permite estabelecer o caminho que um pacote segue até chegar a seu destino, e conseqüentemente, o caminho de volta desse pacote. O mesmo esquema: o `1` ativa e `0` desativa.

```
echo 1 > $f/log_martians
```

Esta opção permite que pacotes de origem suspeita ou desconhecida sejam logados pelo próprio Kernel.

As outras opções abaixo existem apenas em um diretório, por isso não estão dentro do *for*.

```
echo 1 > /proc/sys/net/ipv4/  
icmp_ignore_bogus_error_responses
```

Esse arquivo é responsável por ignorar mensagens falsas de *icmp_error_responses*.

```
echo 1 > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
```

Esse arquivo é responsável por rejeitar todas as requisições de ICMP echo, ou apenas aquelas destinadas a endereços broadcasting ou multicasting.

```
echo 1 > /proc/sys/net/ipv4/tcp_syncookies
```

Esse arquivo habilita no Kernel a opção *config_syscookies*, evitando ataques com *syn flood*.

Existem outros arquivos que podem ser configurados, mas não serão abordados nesta primeira edição do livro. Na próxima edição, estaremos vendo a parte de segurança mais a fundo.

Dando continuidade a implementação da segurança, devemos nos ater a outros fatos também de grande importância, como desativar todos os serviços desnecessários (como terminais virtuais do arquivo */etc/inittab*), desativar o root para conexão direta no arquivo */etc/security*.



Deste modo, logando como usuário normal e depois alterando para o root com o comando *su*. Também existem vários programas que ajudam na administração da segurança, como TCP Wrapper ou TCPD, Tripwire, Tiger, Swatch, Strobe e muitos outros.

Alguns guardam as datas de último acesso dos arquivos do servidor, evitando trojans; outros listam portas, procuram de certo tipo de ocorrência dentro do sistema, controle de conexões. Neste momento, abordaremos somente o TCP Wrapper.

TCP Wrapper - TCPD








O TCP Wrapper ou TCPD controla os acessos aos serviços de seu servidor, bloqueando ou permitindo de forma segura as conexões e logando todas as entradas para uma monitoração preventiva.

Esta ferramenta permite um grande controle das conexões TCP solicitadas a um sistema. A lógica de ativação é muito simples: ao receber uma solicitação de serviço, o *super-daemon* (ou *inetd*) executa um programa de controle (Wrapper ou TCPD) em vez do servidor original.

Esse recurso foi implementado pela Eindhoven University, depois de vários ataques sofridos em seu servidor.

O TCPD trabalha entre a conexão e o *daemon*, exemplificando com um sessão Telnet, o cliente solicita a conexão e o *inetd* chama o TCPD, que por sua vez chama o Telnetd.

Ao ser chamado, ele toma uma ou mais ações, dentre as seguintes:

-  Exibe um banner para o solicitante do serviço.
-  Executa uma dupla verificação da origem da conexão (ip->nome,nome->ip).
-  Compara o endereço do solicitante com o serviço desejado contra uma lista de controle de acesso.
-  Utiliza o protocolo *identd* para determinação da identidade do solicitante (dentro dos limites desse protocolo).
-  Registra os resultados via *syslog*.
-  Executa um comando associado à lista de controle de acesso.
-  Passa o controle para o programa *real*.



A implementação é feita através do par de arquivos `hosts`.

`/etc/hosts.allow`

(permite acesso controlado)

`/etc/hosts.deny`

(não permite acesso)

Se o par *serviço/host* cliente não for encontrado nesses arquivos, o acesso será permitido. Cada linha obedece ao seguinte formato:

`lista_de_serviços: lista_de_clientes : comando`

A configuração dos arquivos `/etc/hosts.allow` e `/etc/hosts.deny` é bem simples.

A configuração desses arquivos se baseia nos endereços IP, no username e no nome do serviço oferecido. A consulta é então feita em dois arquivos. Após encontrar a primeira ocorrência, ela pára imediatamente.

É possível usar coringas como domínio, subdomínio, parte de endereços, *ALL*, *LOCAL*, *UNKNOWN*, *EXCEPT*, etc. Veja abaixo:



* **ALL:** Significa tudo ou todos.



* **LOCAL:** Os endereços da rede local que tiverem resolução pelo DNS da máquina.



* **UNKNOWN:** Serve para identificar quando não se consegue resolver o endereço IP da conexão.



* **KNOWN:** Serve para identificar quando se consegue resolver o endereço IP da conexão.



* **PARANOID:** Automaticamente, *DENY* todas as conexões que não resolverem o endereço IP.



* **EXCEPT:** É usado para *lista_de_clientes* no exemplo: *list_1 EXCEPT list_2*. Isso vai servir para identificar os endereços da *list_1*, exceto os endereços da *list_2*.

Um exemplo simples de controle pode ser a seguinte cláusula no arquivo `/etc/hosts.deny`, este exemplo negará todo o Telnet, exceto para a rede local ou IPs resolvidos localmente.









`in.telnetd : ALL EXCEPT LOCAL`

Outra opção de uso é o *shel_command*. Muito útil quando você está sendo atacado e precisa fazer um sistema de prevenção mais completo.



Se a regra contém a opção de *shell_command*, e o *lista_de_serviço* e a *lista_de_cliente* combinarem com a conexão recebida, é, então, executado o *shell_command*, usando o Shell *'/bin/sh'*, e *stdin/stdout* e *stderr* são direcionados para */dev/null*. É recomendado colocar o caractere *'&'* para não ficar esperando o Shell script acabar antes de completar ou recusar a conexão. O processo *"/bin/sh ... &"* ficará em background. Não esqueça que esses scripts não vão ter a variável de sistema *PATH* definida. É necessário que você use paths completos nele.

Esses recursos são precedidos pelo caractere *'%'*:

-  * **%a (%A)**: O endereço da máquina que fez a requisição de conexão.
-  * **%c**: Informação dos clientes *user@address*, *user@host*, *host*, *address*, dependendo da quantidade de informação que há disponível.
-  * **%d**: O nome do **daemon** que vai ser executado.
-  * **%h (%H)**: O hostname da máquina que fez a requisição de conexão.
-  * **%p**: O número ID do processo.
-  * **%s**: Informação do servidor, mesmo recurso do **%c**.
-  * **%u**: O nome do cliente (ident), ou 'unknown' caso não consiga pegar.
-  * **%%**: Para usar o caracter *'%'*.

Para ficar mais claro, usaremos alguns exemplos.

É comum termos máquinas com vários nomes ou IPs, como o uso de virtual hosts para hospedagem de homepages, etc. Para esse tipo de situação, existe um recurso a mais no TCPD que lhe permite fazer regras de acesso de acordo com o *hostname* desejável:

```
nome_do_processo@host_servidor : lista_de_clientes  
...[:shell_commands]
```

Vamos ao exemplo mencionado:

```
etc/hosts.allow:  
wu.ftp: LOCAL, .dominiolinux.com.br  
/etc/hosts.deny:  
wu.ftp: ALL: (/usr/sbin/in.fingerd -l @%h | \ /bin/  
mail -s %d-%h root) &
```



Nessa configuração, iremos aceitar as conexões feitas pela rede local, pela rede *dominiolinux.com.br* ou por qualquer máquina desses domínios. Qualquer outra conexão que vier e não seja dos domínios que constam no *hosts.allow* será proibida. Mandaremos um finger para a máquina que faz listar os usuários conectados nela e um e-mail para o seu root.

Iptables

A partir do Kernel 2.4 do **Linux**, o firewall padrão é o *iptables*. O *iptables* é de fácil utilização, tanto quanto o *ipchains*, dessa forma, não abordaremos da mesma forma que foi abordado o *ipchains*, aliás, seu uso é muito similar, diferindo em alguns comandos e recursos que o *ipchains* não possui.

No *iptables*, temos uma diferença inicial muito visível: as tabelas. Existem três tabelas distintas no *iptables*, mas falaremos somente das duas mais utilizadas (a terceira como não é usada, ou usada raramente, não será abordada).

A primeira tabela é a *filter*, a tabela padrão que controla o tráfego de dados, sem a ocorrência de NAT, utilizando as *chains INPUT*, *OUTPUT* e *FORWARD*.




A segunda tabela, a *NAT*, é usada quando existe tráfego de uma rede para outra. Admite as *chains PREROUTING*, *OUTPUT* e *POSTROUTING*.

A sintaxe do *iptables* está logo a seguir:

```
ipchains [tipo] (-t tabela) [chain] destino origem ação  
{opções}
```

As tabelas são *filter*, que é a padrão, *NAT* e, ainda, temos a *mangle*, que não será abordada neste livro.

As *chains* são:

-  **INPUT:** Verifica os pacotes entrantes na rede pelo firewall.
-  **OUTPUT:** Verifica os pacotes saindo da rede, passando pelo firewall.
-  **FORWARD:** Verificam pacotes que entram e saem do firewall entre a rede interna.



PREROUTING: Verifica os pacotes que estão entrando através de NAT (outras redes). Faz ações de NAT com o endereço de destino, DNAT.



POSTROUTING: Verifica os pacotes que estão saindo através de NAT (outras redes). Faz ações de NAT com endereço de origem do pacote, DNAT.



-A (append): Adiciona uma nova regra ao sistema.



-D (delete): Exclui uma regra já existente no sistema.



-R (replace): Substitui um regra existente no sistema.



-I (insert): Substitui uma regra na mesma posição no *chain*.



-L (list): Lista as regras existentes no sistema ou do *chain* específico.



-F (flush): Este comando zera as regras do sistema ou na *chain* específica.



-Z (zero): Zera uma regra específica.



-N (new-chain): Cria uma nova regra com nome específico.



-X (delete-chain): Exclui uma regra com nome específico.



-P (policy): Torna a regra padrão. Caso não exista uma regra para um pacote em particular, ele usará a regra padrão.



-h (help): Mostra um pequena ajuda.

Seguem abaixo as possibilidades:



-p (protocol): Define qual protocolo será tratado pelo firewall.



-s (source): Dados e endereço de origem.



-d (destination).









icmp-type [!] (typename): Permite especificar se o tipo ICMP pode ou não passar pelo firewall. No final destas descrições, temos uma tabela com os tipos, códigos e mensagens ICMP.








-j (jump target): Especifica o destino de uma regra e redireciona para ação a ser tomada.






-  **-f (fragment):** Trata datagramas fragmentados.
-  **! [regra]:** Inverte a regra. Por exemplo, `-s 192.168.200.1`, com o argumento `-s ! 192.168.200.1`, serão incluídos na regras todos os endereços diferentes de `192.168.200.1`, ou ainda, `-p ! tcp` (todos os protocolos menos o TCP).
-  **-i (interface de entrada):** Especifica a interface de entrada. O `-i` não pode ser utilizado com a `chain OUTPUT`, já que se trata da interface de entrada.
-  **-o (interface de saída):** Especifica a interface da saída. Similar a `-i`, mas também não pode ser usada com a `chains INPUT`.
-  **—sport (Source Port - Porta de Origem):** Funciona juntamente e somente com a opção `-p`, de *protocol*.
-  **—dport (Destination Port - Porta de Destino):** Similar a *sport*, para pacotes que saem.






Outras opções :

-  **-t (tabela):** Informa qual tabela será usada. A tabela padrão é a *filter*, caso seja omitido este parâmetro. Usaremos a NAT para mascaramento.
-  **—to:** Utilizado para definir IP e porta de destino, após um DNAT, ou de origem, após um SNAT. Deverá ser usado depois da ação **-j DNAT** ou **SNAT**.
-  **—to-port:** Define uma porta de destino, após um REDIRECT.
-  **—syn:** Especifica o uso dos bits ACK e FIN em requisições SYN TCP.
-  **—mac-source (endereço):** Especifica qual a placa de rede, através do MAC Address, que irá transmitir ou não o pacote pelo firewall.

Abaixo, a tabela de ações possíveis:

-  **ACCEPT:** Aceitar o pacote. Permite a passagem do pacote.
-  **DROP:** Negar. Não permite a passagem do pacote e não avisa a origem sobre o ocorrido.
-  **REJECT:** Rejeita. Não permite a passagem do pacote e avisa a origem.



-  **LOG:** Registra uma entrada no arquivo de Log.
-  **SNAT:** Utilizado com o POSTROUTING para fazer ações de mascaramento da origem.
-  **DNAT:** Utilizado com o PREROUTING E OUTPUT para fazer ações de redirecionamento de portas e servidores, balanceamento de carga e Proxy.
-  **MASQUERADING:** Faz mascaramento da saída de pacotes.
-  **REDIRECT:** Redireciona uma requisição para uma porta local do firewall.

Não iremos fazer como *ipchains* e demonstrar todo um complexo de regras, teremos exemplos que podem ser utilizados e desenvolvidos para uso, pois o *iptables* é muito similar ao *ipchains*.

Como todo bom firewall, o padrão é fechar tudo e depois ir abrindo conforme necessidade:

```
iptables -F
iptables -P FORWARD DROP
iptables -P INPUT DROP
iptables -P OUTPUT DROP
```

Devemos carregar alguns módulos para que o firewall funcione perfeitamente:

```
modprobe ip_table
modprobe iptable_nat #Usado para fazer o nat
insmod ip_conntrack_ftp
insmod ip_nat_ftp    #Como vemos usado para o ftp
```

Para começarmos a “brincar”, vamos usar regras simples de *forward*.

```
iptables -A FORWARD -s 192.168.200.0/24 -d
www.sexo.com.br -j DROP
```

Este exemplo nega pacotes de origem da rede 192.168.200 com destino ao host www.sexo.com.br.

```
iptables -A FORWARD -s www.sexo.com.br -j DROP
iptables -A FORWARD -s 200.231.004.156 -j DROP
```



Nega todos os pacotes do host www.sexo.com.br e da rede 200.231.004.153.

```
iptables -A FORWARD -p tcp -s 192.168.200.0/24 -d 0/0--  
dport 80 -j ACCEPT
```

Pacotes de origem da rede 192.168.200.0/24 com destino a qualquer host na porta 80 serão aceitos. Neste caso, em especial, como estamos falando de uma rede interna para outra rede, seria necessário fazer uma regra de NAT para que se torne possível que a saída de rede interna.

```
iptables -A FORWARD -p tcp -s 192.168.200.4 --sport 80 -j  
LOG
```

Todos os pacotes de origem no endereçamento 192.168.200.4, na porta 80, com destino a qualquer lugar, terão registros no log.

```
iptables -A FORWARD -mac-source 00:D0:09:FE:A3:6B -d 0/  
0 -j REJECT
```

Pacotes oriundos da placa de rede acima, com destino a qualquer lugar, serão rejeitadas.

```
iptables -A FORWARD -i eth0 -j DROP
```

Pacotes que entrem pela interface *eth0* serão rejeitados.

```
iptables -A OUTPUT -o eth0 -j ACCEPT
```

Pacotes que saem de qualquer lugar destinados a *eth0* são aceitos.

```
iptables -A forward -s 192.168.200.0/24 -d 10.1.0.0/8 --  
dport 80 -j ACCEPT
```

```
iptables -A forward -s 10.1.0.0/8 --sport 80 -d  
192.168.200.0/24 -j ACCEPT
```

Este caso permite a passagem e o retorno de pacotes da rede 192.168.200.0 para 10.1.0.0 na porta 80. É muito importante esse tipo de regra, pois se for previsto o retorno, provavelmente teremos problemas.

Não podemos esquecer o NAT, que tenho certeza será usado na sua rede, principalmente quando estamos falando de compartilhar a Internet com a rede interna. Com o comando *iptables -t nat -L*, veremos quais regras existem na tabela NAT.

Da mesma forma que a tabela filter, é necessário zerá-la para podermos carregar nossas regras:

```
iptables -t nat -F
```



Agora, o mascaramento em si:

```
iptables -t nat -A POSTROUTING -o ppo0 -j MASQUERADE
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
iptables -t nat -A POSTROUTING -s 192.168.200.0/24 -d 0/0 -j MASQUERADE
iptables -t nat -A POSTROUTING -s 192.168.200.0/24 -j MASQUERADE
iptables -t nat -A POSTROUTING -d 0/0 -j MASQUERADE
```

Temos várias regras: umas mais seguras, outras menos. A primeira, mascara tudo que sair pela interface *ppo0*; a segunda, da mesma forma pela interface *eth0*; a terceira, tudo que sair da rede 192.168.220.0 destinado a qualquer lugar será mascarado, assim como a quarta regra. Note que, na quinta regra, a segurança já não está sendo levada tão em conta, pois qualquer pacote saindo da rede interna para qualquer lugar será aceito.

Uma grande vantagem do *iptables* sobre o *ipchains* é que não existe a necessidade de se obter outro software para fazer redirecionamento de servidores e balanceamento de carga em alguma interface de rede.

No exemplo abaixo, estaremos redirecionando primeiramente portas, como é usado no proxy transparente.

```
iptables -t nat -A PREROUTING -o eth0 -p tcp -dport 80 -j REDIRECT --to-port 3128
iptables -t nat -A PREROUTING -p tcp -s 192.168.200.0/24 -d 0/0 -dport 80 -j REDIRECT --to-port 3128
```

Neste caso, todos os pacotes que forem sair pela porta 80 na interface *eth0* serão redirecionados para a porta 3128. No segundo caso, todo pacote que sair da rede 192.168.200.0 com destino à porta 80 será redirecionado para porta 3128.

```
iptables -t nat -A PREROUTING -p tcp -d 200.253.104.102 -dport 80 -j DNAT --to 192.168.200.4
```

Todos os pacotes enviados à porta 80 do endereço 200.253.104.102 serão redirecionados para 192.168.200.4. Como não foi mencionado, a porta que irá receber por padrão será a mesma de entrada, ou seja, 80. Com isso, o redirecionamento trafega de um servidor de WEB para uma máquina interna com IP inválido.



```
iptables -t nat -A PREROUTING -i eth0 -j DNAT -to  
192.168.200.3-192.168.200.5
```

Aqui, todos os pacotes que entrarem pela *eth0* serão distribuídos entre as máquinas 192.168.200.3, 192.168.200.4 e 192.168.200.5. Também conhecido como Load Balance.

Estudando as regras colocadas aqui, já é possível construir um firewall. Em relação à segurança, podemos utilizar os mesmos arquivos utilizados no *ipchains* para que não ocorram ataques do tipo DoS, entre outros. Também podemos trabalhar com o próprio firewall, veja abaixo algumas regras que podem fazer diferença.

```
iptables -A FORWARD -p icmp-type echo-request -m limit --  
limit 1/s -j ACCEPT
```

Este caso acima, aceita e repassa pacotes ICMP do tipo *echo-request* com um limite máximo por segundo, mas só faria sentido se todos os pacotes *echo-request* (ping) estivessem sendo barrados como essa regra: *iptables -A FORWARD -p icmp-type echo-request -j DROP*. A regra com limite pode barrar ataques Ping of Death, ou DoS.

Mas, trancarmos portas ICMP pode não ser tão interessante em certos casos, dessa forma, podemos negar todas as requisições com ICMP, usando a regra padrão *-P* e abri-la conforme necessidade

```
iptables -A FORWARD -p icmp -s 0/0 -icmp-type 0 -j  
ACCEPT
```

```
iptables -A FORWARD -p icmp -s 0/0 -icmp-type 3 -j  
ACCEPT
```

```
iptables -A FORWARD -p icmp -s 0/0 -icmp-type 5 -j  
ACCEPT
```

```
iptables -A FORWARD -p icmp -s 0/0 -icmp-type 11 -j  
ACCEPT
```

```
iptables -A FORWARD -p tcp -m limit --limit 1/s ACCEPT
```

Da mesma forma que o anterior, ele trata pacotes que estão sendo repassados, mas com um limite máximo em todas as portas do protocolo TCP, como a 80, de 21, que é WEB e FTP.

```
iptables -A FORWARD -p tcp --tcp-flags SIN, ACK, FIN,RST  
-m limit --limit 1/s -j ACCEPT
```

Nesta regra, são tratados os vários estados da conexão, evitando, como isso, Scanners.



```
iptables -A FORWARD -m unclean -j DROP
```

```
iptables -A FORWARD -p tcp ! --syn -m state --state NEW -j  
LOG --log-prefix "New not syn:"
```

```
iptables -A FORWARD -p tcp ! --syn -m state --state NEW -j  
DROP
```

Deleta pacotes danificados ou suspeitos.

Com isso, terminamos esse capítulo do firewall. Em edições posteriores, serão adicionados mais materiais visando à segurança em todos os sentidos, mas, certamente, isto é assunto para um livro inteiro.

Parte deste material foi recolhido no site www.iptablesbr.cjb.net. Recomendo uma boa leitura do material sobre *ipchains* contido neste site (no Iptables How To, é claro).

8

Servidor Proxy

Introdução

No capítulo anterior, implementamos um firewall que permite ou não acessos e filtra pacotes da Internet como o TCP.

Neste capítulo, veremos o que é um proxy e implementaremos o proxy *squid*, um dos mais famosos do mundo.

A primeira pergunta é: O que é um proxy? O proxy tem várias funções que, se trabalhadas junto com o firewall, podem trazer ótimos resultados em relação ao controle e segurança de acesso a Internet.

Uma das funções mais importantes do proxy é o *cache*. O que é cache? Primeiro devemos entender como funciona o acesso a uma página qualquer.

Quando acessamos uma página, fazemos uma requisição ao servidor WEB que armazena o conteúdo. Após a solicitação ser processada, a nossa máquina começa a fazer download da página solicitada. Imagine aquele logo da empresa que você está acessando no site e que tem um 100 Kb, pouco não é?. Mas, se muitas máquinas estiverem fazendo o download deste logo ao mesmo tempo, já pode ocasionar uma demora. Imagine, então, que este site fique no Reino Unido, do outro lado do oceano.

Entendeu? Tudo isso acarreta e uma demora natural para carregar as páginas WEB, sem falar no FTP, etc.

Com a implantação do proxy, temos, como dito anteriormente, o famoso cache, que nada mais é do que um depósito do sites acessados pela rede.

Vejam como funciona. Uma máquina da rede solicita acessar um site, obviamente com o proxy instalado em um servidor. Esta requisição primeiramente passará pelo proxy, que, por sua vez, verificará no diretório de cache se tal página está armazenada. Estando,



ele devolve a página armazenada para o cliente local, caso contrário, irá buscar esta página, fará o download, entregará a solicitação para o usuário e guardará a página em cache. Dessa forma, quando qualquer usuário acessar a página, seja o primeiro que solicitou a página WEB ou um novo usuário que quer conhecê-la, não necessitará ir até o Reino Unido para buscar esta página novamente, pois o proxy a guardou.

Mas o que acontece com proxy? Ele guardará tudo? Como ele armazena? Logicamente, existe um limite, que é estabelecido pelo Administrador da Rede, de qual o tamanho do diretório de cache para uma rede pequena, com uns dez micros. É sugerido algo em torno de 100 Mb, conforme o tamanho da sua rede, este valor vai sendo aumentado. No caso, é interessante fazer testes, pois um diretório muito grande causa demora também pelo fato do próprio proxy passar muito tempo procurando a página em seu cache.

Delimitado o tamanho, o servidor trabalha sozinho. Ele guarda as informações mais recentes e, quando o diretório está cheio, ele apagará os documentos mais antigos, ou seja, aqueles que raramente são acessados, deixando, assim, os sites mais visitados em cache.

Outra função interessante do proxy são as políticas de controle de acesso, conhecidas por ACL (Access Control List).

As ACLs permitem especificar endereços de origem ou destino, domínios, horários, usuários, portas ou métodos de conexão ao proxy, que serão utilizados para permitir ou negar acessos. Com isso, podemos, por exemplo, especificar quais endereços podem ser acessados, em que horário, qual o usuário, que um usuário somente pode acessar a partir de uma máquina específica, que um protocolo pode ou não ser utilizado, ou qualquer combinação dessas permissões/restrições.










Isto torna bastante interessante do ponto de vista da empresa que quer ter controle sobre o que os empregados estão acessando e, na realidade, o que eles podem ou não podem acessar durante expediente.

Vamos falar da máquina ideal. Na realidade, o termo máquina ideal para proxy é muito vago. O que necessariamente faz a diferença, é a memória RAM. Então, neste caso, o processador é relevante, desde que haja bastante memória RAM. Outra configuração que pode trazer muitos benefícios é a do HD. Um HD SCSI, por exemplo, tem melhor performance que o IDE pelo fato de o proxy estar sempre consultando as páginas guardadas no HD.



Bom, agora que sabemos um pouco mais sobre proxy, falaremos do *squid*. O *squid* é um proxy mundialmente conhecido, ele é o proxy padrão da Conectiva. Existem muitos outros, mas falaremos apenas do *squid*.

Algumas características do *squid*:

-  Proxy e cache para HTTP, FTP e outros protocolos baseados em URL.
-  Proxy para SSL (páginas criptografadas).
-  Cache hierárquico, que traz uma melhora significativa de performance.
-  Suporte para proxy transparente (veremos a frente).
-  Políticas de controle de acesso extremamente flexíveis.
-  SNMP.
-  Logs Avançados.
-  DNS cache (guarda informações sobre o mapeamento entre endereços IP e nomes de máquinas de Internet, acelerando a procura de máquinas).
-  Cache em RAM, que mantém os objetos mais utilizados na memória RAM.

Instalação de Squid

A instalação é bem simples. Como todos os pacotes que queremos instalar, podemos usar *rpm*, *tar*, *apt-get*, etc.

No exemplo, mostrarei a instalação do pacote *rpm* contido no CD do Conectiva:

```
mount /dev/cdrom /mnt/cdrom
cd /mnt/cdrom/conectiva/RPMS
```

```
rpm -ivh squid-*
```

Vamos iniciar a *squid*, para que ele possa criar os arquivos de configuração:

```
squid -z
cds
./squid start
```



Feito isso, podemos começar a configurar o *squid* do jeito que quisermos.

O arquivo de configuração principal fica em */etc/squid.conf*.

Para limitar a memória RAM, procure pela linha que contém a seguinte palavra:

```
cache_ram MEM 8 MB
```

No caso da máquina ter sido feita somente para o proxy, este valor pode ser consideravelmente aumentado, mas não esqueça de deixar para os outros processos que rodam na máquina, como o próprio *squid*, já que este *cache_ram* serve somente para guardar os objetos mais usados em memória RAM, não incluindo, por exemplo o daemon do *squid*.

Altere conforme sua necessidade e quantidade de memória:

```
cache_ram MEM 128 MB
```

Vamos configurar o diretório de cache do *squid*.

Procure pela linha *cache_dir*:

```
cache_dir ufs /var/spool/squid 100 16 256
```

Onde *ufs* é o tipo de sistema de armazenamento que o *squid* utilizará. Importante ressaltar que, esses diretórios não são criados automaticamente pelo *squid*. Após isso, temos *100*, que se refere ao tamanho máximo que pode ser utilizado pelo *squid* para armazenar arquivos.

Logo após, temos dois outros números que querem dizer o seguinte: o primeiro, *16*, representa os diretórios de primeiro nível que *squid* pode criar, ou seja, no diretório */var/spool/squid*, poderão ser criados 16 diretórios; o segundo número, *256*, refere-se aos diretórios de segundo nível, que nada mais são do que a quantidade de subdiretórios que os primeiros 16 podem ter.

Fazendo uma conta rápida, chegamos à conclusão que dentro de */var/spool/squid*, teremos um total de 4096 diretórios e subdiretórios dentro do diretório *squid*.

A porta padrão do proxy *squid* é o 3128, caso for necessário, altere-a na linha que tem o início igual *http_port*. Muitos dos proxys trabalham com a porta 8080. Fica a gosto do usuário.



Controle de Acesso - ACL

Como foi dito anteriormente, o Servidor Proxy Squid, permite um controle de acesso baseado em ACL.

Por padrão, o servidor nega todo e qualquer acesso, dessa forma, é necessário liberar acesso para as máquinas da rede.

Mude de:

```
http_access DENY all
```

Para:

```
http_access allow all
```

Dessa forma, você liberará todas as páginas para todos os usuários. Caso não seja isso que pretende, deveremos fazer algo mais complexo.

Existem várias formas de se usar ACL, desde configurações simples, como permitir ou negar, até data, hora, etc.

Funciona da seguinte forma:

```
acl [nome_da_acl] [tipo_da_acl] {argumento}
```

Veja o exemplo abaixo:

```
acl minharede src 192.168.200.0/255/255/255/0
http_access allow minharede
```

Neste exemplo acima, criamos uma regra do tipo origem que incluiu toda a rede. Posteriormente, damos permissão à rede. É desta forma que as ACLs funcionam.

Abaixo, temos os tipos de ACLs possíveis:

Origem/Destino Endereço IP

```
src - Origem
```

Foi como a que usamos anteriormente para liberar uma rede. Veja o exemplo abaixo:

```
acl redenegada src 192.168.200.0/255.255.255.0
acl diretor src 192.168.200.10/255.255.255.0
http_access DENY redenegada
http_access allow diretor
dst- Destino
```



Neste outro exemplo, a diferença que negamos o endereço de destino. Similar ao SRC, mas ao invés de usá-lo, optamos pelo DST.

```
acl DestRej dst 192.168.200.0/255.255.255.0
acl IpDest Perm dst 192.168.100.0/24
http_access deny DestRej
http_access allow DestPerm
```

Origem/Destino domínio

```
srcdomain - domínio origem
```

Este tipo de ACL é usada para tratar diretamente nomes de domínio, sem a necessidade do IP.

```
acl MeuDominio srcdomain dominiolinux.com.br
http_access allow MeuDominio
```

O domínio de origem - dstdomain

Similar ao SRC, mas trará os sites que serão acessados.

```
acl DomainReject dstdomain dominiorecusado.com.br
acl IPReject dst 10.255.1.2
http_access deny DomainReject
http_access deny IPReject
http_access allow minharede
http_access deny all
```

Palavras:

```
url_regex
```

Este tipo é um dos mais úteis, barra ou libera tudo que tenha uma palavra em particular.

Por default, o *squid* é case-sensitive, ou seja, distingue maiúsculas de minúsculas. Isso pode ser um problema, pois Sexo, SEXO e sexo, são diferentes para o *squid*, por isso deve-se sempre que possível utilizar o *-i*, que ignora o case-sensitive.

```
acl minharede src 192.168.200.0/255.255.255.0
acl rej_URL url_regex -i sexo
http_access deny rej_url
http_access allow minharede
http_access deny all
```




Este exemplo acima, um tipo clássico, cria a ACL, rejeitando a palavra *sexo*; cria outra ACL para a rede interna; rejeita a ACL do tipo *url_regex*, criada em *rejURL*; libera a rede interna e nega para o restante.

Outro fato que pode ser explorado nesta ACL é o uso de uma variável.

Por exemplo, muitos sites usam arquivos do tipo *avi* (vídeo, ou outro qualquer). Para evitar que o browser abra tais arquivos através de outros programas, é só impedir que o usuário faça download.

No exemplo abaixo, permitimos que os usuários até consigam navegar, mas não baixar arquivos de sites que tenham a palavra *sexo* no domínio.

```
acl minharede src 192.168.200.0/255.255.255.0
acl rej_URL url_regex -i sexo.*<avi$
http_access deny rej_URL
http_access allow minharede
http_access deny all
```

Data hora:

```
time
acl nome time [lista-de-dias]
[hora_inicial:minuto:hora_finalminuto]
```

Esta ACL refere-se, como o nome diz, à data e hora. Você pode criar uma ACL para que em certo horário negue alguns tipos de sites ou, durante alguns dias, não liberar acesso, como sábado e domingo. Enfim, é só ter criatividade.

Na lista de dias, temos:

-  S - Sunday (Domingo)
-  M - Monday (Segunda-feira)
-  T - Tuesday (Terça-feira)
-  W - Wednesday (Quarta-feira)
-  H - Thursday (Quinta-feira)
-  F - Friday (Sexta-feira)
-  A - Saturday (Sábado)



Exemplos:

```
acl nao_comercial time 17:00-6:00
acl noite time 17:00-24:00
acl madrugada time 00:00-6:00
acl final_semana time AS
```

Um exemplo melhor:

```
acl MinhaRede src 1192.168.200.0/24
acl FinaisdeSemana time AS
http_access deny FinaisdeSemana
http_access allow MinhaRede
```

Porta de Destino

ports

Este tipo de ACL é usada com portas TCP (por exemplo, SSL, que é a 443). Por padrão, são trazidas várias portas liberadas, que devem ser alteradas conforme necessidade.

```
acl SSL_ports port 443 563
acl Safe_ports port 80 21 443 563 70 210 1025-65535
acl Safe_ports port 280 # http-mgmt
acl Safe_ports port 488 # gss-http
acl Safe_ports port 591 # filemaker
acl Safe_ports port 777 # multiling http
acl CONNECT method CONNECT
```

É interessante fazer uma ACL para as portas 80, 21, 443, 563. E uma outra para 1025-65535, onde ocorrem outros serviços que podem trazer perigo ao servidor.

Note que *Safe_ports* é uma regra só que engloba todas as portas acima, ou seja, não são várias regras *Safe_ports*, mas apenas uma, como o exemplo abaixo:

```
acl minharede src 192.168.200.0/255.255.255.0
acl minharede src 192.168.100.0/255.255.255.0
```

Seria a mesma coisa que escrevêssemos :

```
acl minharede src 192.168.200.0/255.255.255.0
192.168.100.0/255.255.255.0
```

Posteriormente, temos que liberar ou negar tais portas:

```
http access allow SSL_ports
```



```
http_access deny Safe_ports ou
http_access deny !Safe_ports (Nega todas que não estejam
incluídas em Safe_ports)
```

Protocolo

```
proto
```

É possível restringir o acesso pelo prefixo usado na página, como HTTP, FTP, etc.

```
acl FTP proto ftp
acl HTTP proto http
alc minharede src 192.168.200.0/255.255.255.0
acl all src 0.0.0.0/0.0.0.0
http_access Deny ftp
http_access allow http
http_access allow minharede
http_access Deny all
```

Um fato muito importante, é que o *squid* lê as regra de cima para baixo, parando na primeira que coincidir. Veja no exemplo acima uma solicitação HTTP. Primeiro ele veria que está liberado o HTTP, mas para quem? Desceria lendo mais algumas linhas até encontrar a linha referente a *minharede*.

Existem muitas outras configurações que podem ser vistas diretamente em <http://www.squid-cache.org/> que é site oficial do *squid*.

Neste momento, iremos falar de algo mais interessante, a autenticação por usuário.

Autenticação

Existem várias formas de fazer esta autenticação, como Pam, NCSA, Samba e por aí vai.

Neste nosso servidor, falarei sobre o NCSA. O NCSA já vem com o *squid*, sendo assim, fica fácil configurá-lo, não tendo necessidade de fazer download de nada e muito menos compilar novamente o *squid* por não ter suporte a esse módulo.

A primeira coisa a ser feita, é localizar o arquivo *ncsa_auth*. Na versão 7.0 do Conectiva, ele fica em */usr/lib/squid/ncsa_auth*.



Localizado, teremos que criar os usuário que terão acesso:

```
htpasswd -c /etc/squid/squid_passwd usuário  
New password:  
Re-type new password:
```

Este arquivo que usei foi criado na hora. Usa-se o comando quando o primeiro usuário for criado, pois o arquivo *squidpasswd* não existe ainda. Nas próximas criações de usuários, não haverá a necessidade de usar o *-c*.

Quanto ao arquivo, fica de livre escolha o local onde ficará o nome. Usei o próprio diretório do *squid* para tornar o manuseio mais rápido e um nome bem sugestivo para não esquecer.

Necessitamos alterar alguns dados em */etc/squid/squid.conf*. Descomente a linha:

```
acl password proxy_auth REQUIRED
```

Esta outra linha deve ser criada ou alterada, caso esteja comentada ainda:

```
authenticate_program /usr/lib/squid/ncsa_auth /etc/  
squid/squid_passwd
```

Aqui, colocamos o path (caminho do arquivo *ndsc_auth* e o caminho do nosso arquivo de password) Lembra daquela linha *http_access allow all*? Ela será também alterada para:

```
http_access allow password
```

Depois disso, pode ir ao diretório dos daemon e reiniciar o serviço do *squid*.

```
./squid restart ou start
```

Com estas alterações, quando o usuário tentar acessar uma página, primeiramente será necessário colocar o nome e a senha em uma janela que se abre. Veja abaixo:



Janela do solicitação de senha do Netscape



Não esqueça de alterar as configurações do firewall, pois como fizemos anteriormente, todas as máquinas têm acesso. Então, bloqueie o acesso das estações na porta 80, liberando a do servidor proxy.

Para configurar o cliente em qualquer browser é bem fácil. Procure as configurações. No Netscape, por exemplo, está localizado em *Edit/Preferences/Advanced/Proxies*. No Internet Explorer, *Ferramentas/Opções da Internet/Conexões/Configuração da Lan*.

Nestas guias, terão os itens *HTTP proxy*, *FTP proxy*, etc. Preencha com o nome da máquina servidor proxy, se você tiver um servidor DNS, lógico. Caso contrário, coloque o IP na porta 3128, se você alterou a original do *squid*.

Para outros browsers, como Unix, como Lynx, algumas variáveis de ambiente devem ser editadas:

```
http_proxy="http://192.168.200.1:3128"  
ftp_proxy="http://192.168.200.1:3128"  
export http_proxy ftp_proxy
```

Com isso, o Lynx também pedirá nome de usuário e senha.

Assim ficou fácil controlar os passos de cada usuário, não é? Quase! Um ponto importante que deve ser visto, são os logs. Normalmente, localizado em */var/log/squid*, onde encontramos os arquivos *Access.log*, *cache.log* e *store.log*.

Não é nem preciso dizer que o mais importante é o *Access.log*. Nele, estão contidas as informações a respeito do usuário, data, hora e página acessada.

Mas fica muito difícil trabalhar com diretamente com os logs. Para isso, temos duas ferramentas que geram página HTML a partir dos *Access.log*: *WebAlizer* e o *Sarg*. Ambos são ótimos. Mas, neste momento, falaremos do *Sarg*. O *Sarg* (Squid Analysis Report Generator) gera a partir de alguns dados fornecidos uma página *index.html* em um diretório padrão *usr/local/etc/httpd/htdocs/squid-reports/ "Data" / index.html*.

No diretório *data*, refere-se realmente a data em que o *Sarg* utilizou para gerar o arquivo.

A página para download do *Sarg* está em <http://web.onda.com.br/orso>.

Baixe o arquivo *sarg.xxxx.tar.gz* e faça todo processo contido no arquivo *Readme* na própria página, que são:



```
./configure  
make  
make install
```

Depois disso, acesse o arquivo `/usr/local/sarg/sarg.conf`. Neste arquivo, vários itens podem ser alterados conforme a necessidade do administrador.

Para obter mais informações, use `sarg -h`.

Abaixo, uma figura da página gerada pelo Sarg:



Página gerada pelo Sarg

Proxy Transparente

Nos dias de hoje, é muito comum usarmos proxy transparente, para facilitar a vida do usuário. O que é isso? Como o nome diz, é um proxy que fica totalmente transparente para o usuário, ou seja, ele não notará que existe um proxy até alguma regra não permitir que ele acesse determinado site.

A maior vantagem de se trabalhar com o proxy transparente é o fato de não ser necessário configurar todas as máquinas para acessar o proxy, pois o mesmo trabalhará na porta 80.



Como funciona? Muito simples, aliás, uma ótima sacada!! Simplesmente redirecionamos tudo que vier da rede interna com destino na porta 80 para a porta 3128 do servidor proxy.

O que deve ser analisado: Primeiro, serão usadas regras de *ipchains* e *iptables* para redirecionamento de porta.

Caso o servidor proxy seja em outro, devemos usar algum software que redirecione as entradas de um determinado range de IP para o IP do servidor proxy na porta 3128.

Este item é muito importante, pois o *ipchains* só faz redirecionamento de portas na mesma máquina. Se for em outra máquina, terá que usar um software para redirecionar. Dê uma olhada nos próximos capítulos que falam do Rinetd e Redir.

Outro detalhe muito importante é o funcionamento da autenticação com o proxy transparente. Por padrão, não funcionam juntos, mas, logicamente, usam algumas técnicas brasileiras para “quebrar o galho”. Não entraremos neste aspecto, mas caso se interesse, dê uma olhada na página da lista de discussão *linux-br*, que provavelmente você encontrará alguém que fez funcionar.

Finalmente, iremos configurar o proxy transparente. No Kernel versão 2.4, com *iptables*, use o seguinte:

```
iptables -t nat -A PREROUTING -o eth0 -p tcp -dport 80 -
j REDIRECT --to-port 3128
```

Neste caso, o *squid* e firewall estão na mesma máquina. No Kernel 2.2, com *ipchains*.

```
ipchains -A input -p tcp -s 192.168.200.0/24 -d 0/0 80 -
j REDIRECT 3128 (ou 8080 conforme o squid)
```

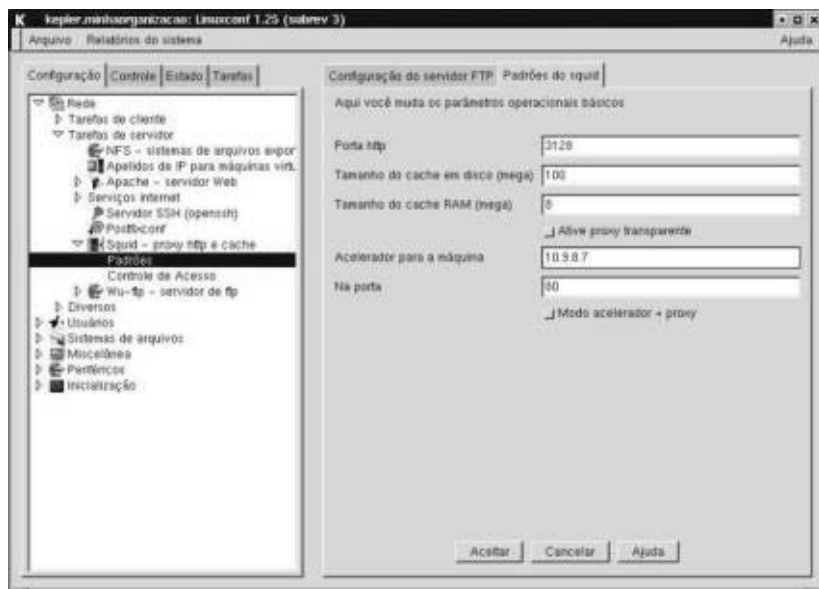
Lembre-se que se usar outra máquina sendo proxy, terá que tratar no firewall as diferenças. Por exemplo, se a máquina proxy for de endereço 192.168.200.10, no firewall, você não pode pedir para que a mesma seja redirecionada para ela. Uma boa opção é diferenciar apenas ela:

```
ipchains -A input -p tcp -s ! 192.168.200.10 -d 0/0 80
-j REDIRECT 3128
```

Ou seja, redirecionar todas as entradas com destino à porta 80, menos 192.168.200.10, para a porta 3128. Neste caso, a máquina 192.168.200.10 terá acesso a Internet através da porta 80.



Outra forma de fazer as configurações é pelo Linuxconf, tudo é visual. Agora que você já aprendeu a fazer manualmente, não vai sentir a mínima dúvida com o Linuxconf.



Configuração do squid via Linuxconf

Aqui, temos todas as opções vistas anteriormente, com tamanho de cache, memória RAM e até as ACLs para controle de acesso.

Pode-se usar o item proxy transparente para que, automaticamente, a regra seja inserida na tabela de filtros, seja *ipchains* ou *iptables*.

Adiante, veremos outras formas de administrar tanto proxy, DHCP (nosso próximo capítulo), servidor web, etc.

Introdução

O que é um DHCP? Seu significado é Dynamic Host Configuration Protocol.

Imagine sua rede com cem micros, onde cada um deve ter um endereçamento IP, servidores DNS, Gateway configurado, etc. De repente, seu provedor informa que o endereço IP do DNS mudou. O que acontece? Você terá que ir micro a micro configurando o DNS, certo? Errado, se você tiver um servidor DHCP.

O DHCP serve justamente para passar estas informações para as máquinas clientes. Na realidade, ele aluga um endereçamento IP por um tempo, quando esse tempo expira, dependendo da ocasião, ele renova ou revalida.

Como funciona? Isso nós veremos daqui a pouco.

Primeiramente, vamos instalar o DHCP.

```
rpm -ivh dhcp-*.i386.rpm      (Servidor DHCP)
rpm -ivh dhcpcd-*.i386.rpm   (Cliente DHCP)
```

O daemon do DHCP será instalado no diretório */usr/sbin/*, representado pelo programa DHCP.

Os arquivos do DHCP:

/etc/dhcp.conf

Arquivo principal, onde serão informadas as faixas de IPs, máscara de rede, entre outras informações que serão passadas para o cliente.

O arquivo abaixo guardará as informações das máquinas que estarão utilizando o DHCP no momento:

/etc/dhcpd.leases





Caso esse arquivo não seja criado na instalação, deve-se criá-lo com o comando *touch*:

```
touch /etc/dhcpd.leases
```

Com isso, o nosso servidor DHCP já está instalado.

Configurações do DHCP

Vamos comentar cada item que deve ser posto no arquivo */etc/dhcp.conf*. O primeiro item que deve ser posto é referente ao tempo. Veja abaixo:

-  **default-lease-time 86400:** Libera um endereço por 86400 segundos (24 horas), caso o cliente não defina um tempo específico de utilização de endereço.
-  **max-lease-time 7200:** Tempo máximo em segundos para liberação do endereço IP. Caso esse tempo seja ultrapassado, o servidor DHCP irá fornecer um novo endereço, para o cliente DHCP.

Agora, falaremos dos dados da rede, teremos que passar informações da rede tipo IP, máscara, roteador, DNS, etc. Para fins didáticos, usaremos novamente a rede 192.168.200.0.

```
subnet 192.168.200.0 netmask 255.255.255.0
{range 192.168.200.2 192.168.200.100;
range 192.168.200.150 192.168.200.200 }
```

O que foi feito acima? Atribuímos que nossa rede será 192.168.200.0 e nossa máscara 255.255.255.0. O range é a faixa de IP que será atribuída a clientes, seja Windows, Linux, FreeBSD, etc.

```
option domain-name-server 200.200.200.201 ,
200.200.200.202;
option domain-name dns.com.br
```

Nesta outra opção, informamos que será nosso servidor de Nomes - DNS, com seus respectivos endereços IPs.

```
option routers 192.168.200.1
```

Este item informa qual o IP do nosso Gateway. Veja que ele está fora do range que será atribuído às outras máquinas, dando início em 192.168.200.2. É muito importante lembrar quais tem IP fixo e não colocar dentro do range.

```
option broadcast-address 192.168.1.255 . ( Define o
endereço de broadcast ).
```



Esta seria a configuração básica do servidor DHCP, para que ele funcione sem problemas.

Vamos supor que você queira atribuir o mesmo IP para um determinado cliente, pois nas regras de firewall você quer dar acesso a ele apenas em determinados sites da Internet.

Como fazer, já que no firewall você apenas tratou o IP?

Podemos, então, atribuir um IP determinando apenas para uma máquina em particular, usando seu MAC Address ou endereço ethernet.

Para configurá-lo, digite apenas um *ifconfig* e veja um número parecido com algo deste tipo:

```
00:E0:7D:A0:7E:78
```

Como você pode ver, são um conjunto de pares, no total de seis conjuntos. No Windows, através do programa *winipcfg*, é possível verificar o endereço *ethernet*.

Para configurar o DHCP para que sempre atribua determinado IP para um MAC Address, use o esquema abaixo:

```
host conec {hardware ethernet 00:E0:7D:A0:7E:78;
fixed-address 192.168.200.2;
}
```

Com isso, a máquina com o endereçamento de placa 00:E0:7D:A0:7E:78 sempre terá o IP 192.168.200.2. Fácil, não?

Configurando os clientes DHCP Linux e Windows

Agora, como configurar o cliente para que pegue as configurações DHCP?

No **Linux** é muito fácil. Através do *Linuxconf/Rede/Nome da máquina e dispositivo IP da rede*. Na opção *Modo de configuração*, marque a opção DHCP ou vá em */etc/sysconfig/network-scripts/ifcfg-ethx*, onde *x* refere-se a sua placa de rede, altere o item :

```
BOOTPROTO="dhcp"
```

Já no Windows, acesse *Painel de Controle/Rede/TCP/IP/Propriedades*.

Marque a opção pegar IP automaticamente.



Propriedades do TCP

Para verificar se as máquinas que estão com IP alugados no DHCP, edite o arquivo `/etc/dhcpd.leases`.

Outra forma de configurar o servidor DHCP é através do Linuxconf. Abra o Linuxconf e vá a guia *Configurações*, depois expanda o item *Rede/Serviços de Inicialização*, clique em *Boot DHCP/BOOP*.



LinuxConf : Padrões DHCP



Esta primeira guia refere-se aos padrões da máquina e do servidor, onde serão preenchidas informações, como nome da máquina, servidores DNS, NIS, etc.

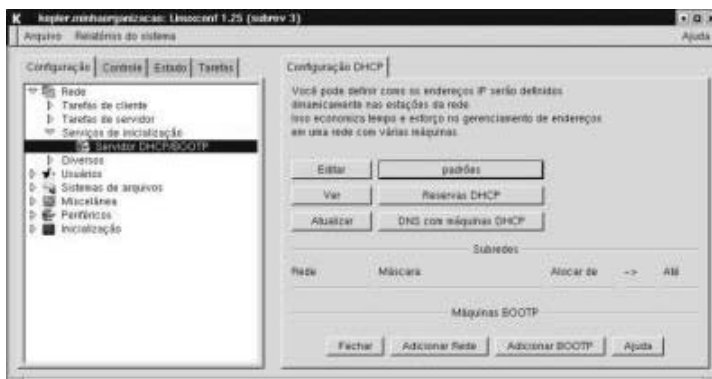
Feito isso uma vez, o Linuxconf não abrirá mais estas informações automaticamente. Para acessar tais informações novamente, teremos que ir por outro caminho.

A próxima vez que acessar, ou logo que terminar as configurações padrão do DHCP, será mostrada uma nova tela, onde faremos configurações de rede, range.

Para adicionar uma configuração de uma rede 192.168.300.0 com seus ranges de IPs, por exemplo, basta clicar no botão *Adicionar Rede* (veja a figura 9.3).

Serão solicitadas informações, como endereço da rede, máscara, faixa de IP, servidor de nomes, etc. Ou seja, todas as configurações necessárias para que você configure o DHCP para uma rede. Pode-se adicionar mais de uma rede. Você pode distribuir o IP da rede 192.168.200 e 192.168.300, por exemplo. É só adicionar mais uma rede.

Para verificar as máquinas conectadas, clique no botão *Ver*.



Padrões Normais do DHCP

Se existir mais de uma placa de rede no servidor, supondo que ele seja o Gateway da rede também, podemos ter problemas. O servidor não será inicializado pelo fato de haver duas placas de rede e o mesmo não saber qual utilizar para entregar informações da rede.

Dessa forma, se faz necessário iniciar o serviço com alguma modificação. A primeira coisa a ser feita é retirar as configurações do setup para que ele inicie automaticamente.



Temos que iniciá-lo informando qual placa distribuirá as configurações da rede para os clientes.

Supondo que temos duas placas, *eth0* para rede externa com IP válido, ou até um *Nat* para outra rede, e uma placa *eth1* de endereço da rede interna que forneceremos configurações através do servidor DHCP.

Inicie o serviço manualmente da seguinte forma:

```
/usr/sbin/dhcpd eth1
```

Para parar o servidor, faz-se normalmente, como qualquer outro serviço:

```
service dhcpd stop
```

Ou:

```
cds ( cd /etc/rc.d/init.d )  
./dhcpd stop
```

Para não termos que iniciar o servidor DHCP todas as vezes que dermos um boot na máquina, podemos colocar nos arquivos de inicialização, como o */etc/rc.d/rc.local*, adicionando a linha do comando que executamos manualmente.

Com este servidor, não teremos mais o trabalho que configurar estações com informações de rede, pois as mesmas podem ser obtidas pelo DHCP, poupando o trabalho e até o tempo que seria utilizado para configurar a rede. Não existe um limite mínimo para usar o DHCP, particularmente, já usei três micros e um servidor. Poupa-me o trabalho de configurar as três máquinas cliente existentes. Logicamente, ele se torna necessário à medida que a rede cresce.

Introdução

Neste capítulo, falaremos sobre o servidor de páginas de Internet, o Apache. Para ser possível visualizar um documento, deve existir um servidor que distribua as solicitações de documentos na Internet.

Através do HTTP, que faz parte do pacote TCP/IP, é possível fazer tal distribuição. O processo, resumidamente, é iniciado através do browser (Internet Explorer, Netscape, Opera, etc), que solicita a página HTML, HTM ou outras terminações do servidor. O servidor verifica se a página existe, confirma a versão do HTTP, se compatível, e envia a página ao solicitante.

Existem vários servidores. Entre os mais famosos, o Apache, com algo em torno de 55% do mercado. Em segundo, encontra-se o IIS da Microsoft.

O Apache, dessa forma, é o servidor WEB mais utilizado no mundo, possuindo versões para várias plataformas, tais como **Linux**, Windows 9x /NT BSD , UNIX, Solares , SunOS. Além de ser um software muito bom, ele tem um diferencial entre os outros: é “free”.

Instalando o Servidor Web

A primeira coisa a fazer é o download dos pacotes do servidor web. Normalmente, quando fazemos a instalação do sistema operacional **Linux**, podemos escolher os pacotes que serão instalados e, normalmente, o Apache é instalado junto ao sistema operacional.

Para verificar, digite no Shell:

```
httpd -v
Server version: Apache/1.3.19 (Unix) (Conectiva/Linux)
Server built:   Jun 29 2001 00:18:04
```



Vemos a versão do Apache instalada.

```
httpd -V
Server version: Apache/1.3.19 (Unix) (Conectiva/Linux)
Server built: Jun 29 2001 00:18:04
Server's Module Magic Number: 19990320:10
Server compiled with....
-D EAPI
-D HAVE_MMAP
-D HAVE_SHMGET
-D USE_SHMGET_SCOREBOARD
-D USE_MMAP_FILES
-D USE_SYSVSEM_SERIALIZED_ACCEPT -D
SINGLE_LISTEN_UNSERIALIZED_ACCEPT
-D HTTPD_ROOT="/etc/httpd"
-D SUEXEC_BIN="/usr/sbin/suexec"
-D DEFAULT_PIDLOG="/var/run/httpd.pid"
-D DEFAULT_SCOREBOARD="/var/run/httpd.scoreboard"
-D DEFAULT_LOCKFILE="/var/run/httpd.lock"
-D DEFAULT_XFERLOG="/var/log/httpd/access_log"
-D DEFAULT_ERRORLOG="/var/log/httpd/error_log"
-D TYPES_CONFIG_FILE="conf/mime.types"
-D SERVER_CONFIG_FILE="conf/httpd.conf"
-D ACCESS_CONFIG_FILE="conf/access.conf"
-D RESOURCE_CONFIG_FILE="conf/srm.conf"
httpd -t
Syntax OK
```

No primeiro item *httpd -v*, vemos a versão do Apache, já o *httpd -V*, mostra os parâmetros e a construção do daemon *httpd*.

O último *httpd -t*, testa os arquivos de configuração, verificando a integridade dos arquivos.

Caso ocorra algum erro, ele reportará o arquivo e a linha que estão com problemas.

Mas nem todos instalam o servidor WEB na primeira instalação, por isso deve-se fazê-lo posteriormente.

Você pode fazer um download do arquivo na página do Apache <http://www.apache.org>, no site da sua distribuição **Linux** <ftp://ftp.conectiva.com.br/pub/conectiva/> e, caso possua o CD de instalação do **Linux**, no CD 1 encontra-se o pacote *rpm* do Apache em */conectiva/RPMS*.



Se você pegar os binários do tipo *apache .x.x.x.tar.gz* para serem compilados, execute o seguinte processo:

```
tar -xzf apache.x.x.x.tar.gz
```

Será descompactado em um diretório com o nome do arquivo.

```
cd apache.x.x.x.tar.gz
./configure
./make
make install
```

No caso do *rpm*, esta tarefa é mais fácil:

```
rpm -ivh apache.x.x.x.i386.rpm
```

É interessante instalar os pacotes *apache*, *apache devel* e *apache doc*, executando:

```
rpm -ivh apache-*
```

A versão do **Conectiva Linux** vem configurada com as opções mais utilizadas (não sendo necessário reconfigurar). Posteriormente, falaremos de algumas configurações, como autenticação e SSL.

A página padrão é parecida com esta abaixo. Para vê-la, precisamos iniciar o nosso daemon?

```
cds (ou)
cd /etc/rc.d/init.d
./http status
```

Mostrará o status do servidor, se está executando ou parado.

```
./httpd start
```

Iniciando HTTPd:

```
[ OK ]
```

Com isso, iniciamos nosso serviço HTTP.

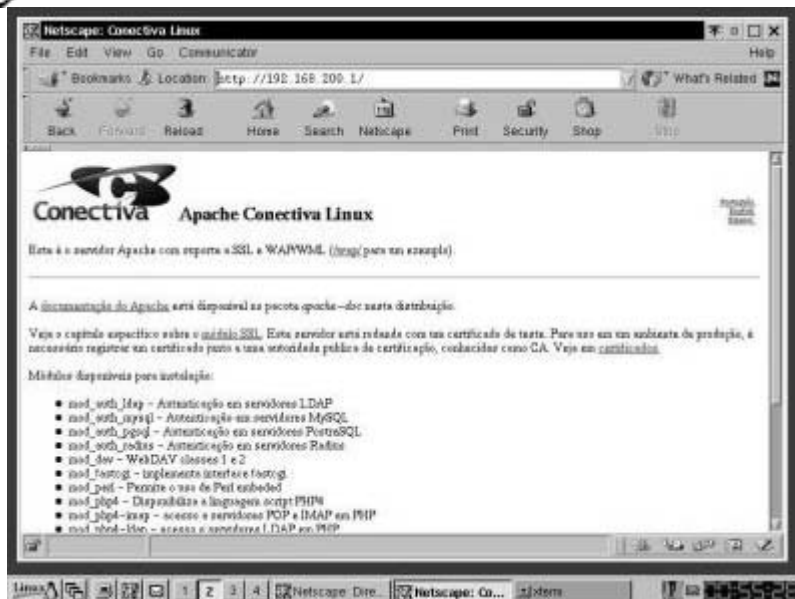
Digite em qualquer browser o seguinte:

<http://localhost>

Ou:

<http://192.168.200.1>

Se este for o seu IP.






Página web padrão do Apache

Configurações




Existem vários arquivos de configurações do Apache.

No diretório `/home/httpd`, existe, a princípio, três diretórios.

-  `cgi-bin` (Onde ficarão as rotinas CGI)
-  `html` (Onde ficarão as páginas)
-  `icons` (Onde ficarão os ícones)

Isso pode ser alterado caso você queira.

Os arquivos de configuração do Apache ficam em `/etc/httpd/conf`. Existem alguns arquivos importantes neste diretório, inclusive o que você usaria para mudar o diretório padrão das páginas do Apache, no `httpd.conf`, na linha referente a *Directory*.

-  **httpd.conf**: Controla o daemon.
-  **srm.conf**: Contém diretivas que controlam as especificações de documento que o servidor fornece aos clientes.
-  **access.conf**: Contém instruções para controlar o acesso aos documentos.



Comentando o Arquivo `httpd.conf`

Aqui, falaremos um pouco sobre o arquivo *httpd.access*, já que ele é o mais importante dos três e, inclusive, pode substituir os outros dois.

As opções, neste arquivo, já estão comentadas, mas mesmo assim, aqui vai uma descrição das mesmas. As principais são:

Parâmetro `inetd` ou `standalone`

```
# ServerType is either inetd, or standalone. Inetd mode
is only supported
# on
# Unix platforms.
#
ServerType standalone
```

Diz ao sistema se o *httpd* vai ser rodado via script próprio (`standalone`) ou a partir do arquivo *inetd.conf* (em *inetd*, o *httpd* fica ocioso, enquanto o *inetd* fica monitorando as requisições. Quando houver alguma, ele avisa e o serviço começa a funcionar.

Diretório Padrão

```
#
# ServerRoot: The top of the directory tree under which
the server's
# configuration, error, and log files are kept.
#
# NOTE! If you intend to place this on an NFS (or
otherwise network)
# mounted filesystem then please read the LockFile
documentation
# (available at http://www.apache.org/docs/mod/
core.html#lockfile);
# you will save yourself a lot of trouble.
#
# Do NOT add a slash at the end of the directory path.
#
ServerRoot /etc/httpd
```

Esta cuida do caminho do diretório onde irão ficar os arquivos de configuração. Pode ser mudado, se necessário.

Timeout - Tempo de Espera

```
#
# Timeout: The number of seconds before receives and
sends time out.
#
Timeout 300
```



Tempo máximo (em segundos) que o servidor esperará, mantendo uma conexão aberta com o cliente. Se o limite for excedido, ele terá de criar uma nova conexão com o mesmo.

KeepAlive On

```
#
# KeepAlive: Whether or not to allow persistent
# connections (more than
# one request per connection). Set to "Off" to
# deactivate.
#
KeepAlive On
```

Diretamente ligado com a opção acima, define se o processo de manter a conexão com o cliente está ativo ou não.

MaxKeepAliveRequests

```
#
# MaxKeepAliveRequests: The maximum number of requests
# to allow
# during a persistent connection. Set to 0 to allow an
# unlimited amount.
# We recommend you leave this number high, for maximum
# performance.
#
MaxKeepAliveRequests 100
```

Número máximo de conexões mantidas, sem necessidade de renovação. Quanto mais alto o número, melhor a performance (com o hardware adequado).

```
#
# KeepAliveTimeout: Number of seconds to wait for the
# next request from the
# same client on the same connection.
#
```

Máximo (de segundos) a espera de nova requisição.

StartServers 10

Número de servers iniciais, ou seja, logo no início do processo, o *httpd* poderia responder a dez conexões simultâneas ao mesmo site.

```
#
# Number of servers to start initially – should be a
# reasonable ballpark
# figure.
#
```



MaxClients 150

Número máximo de conexões simultâneas por clientes ao site. Se for ultrapassada, mostrará a infame mensagem “http server busy”.

```
#
# Limit on total number of servers running, i.e., limit
# on the number
# of clients who can simultaneously connect – if this
# limit is ever
# reached, clients will be LOCKED OUT, so it should NOT
# BE SET TOO LOW.
# It is intended mainly as a brake to keep a runaway
# server from taking
# the system with it as it spirals down...
#
```

Respondendo por mais de um endereço IP

```
#
# Listen: Allows you to bind Apache to specific IP
# addresses and/or
# ports, in addition to the default. See also the
# <VirtualHost>
# directive.
#
#Listen 3000
#Listen 12.34.56.78:80
```

Permite ao __principal__ *httpd server* responder em mais de um IP (descomentando o 12.34.56.78:80, por exemplo, habilitaria ao server HTTP “escutar” em um IP além de seu IP normal (o da própria máquina).

Entrada BindAddress - Domínio Virtual

```
#
# BindAddress: You can support virtual hosts with this
# option. This
# directive
# is used to tell the server which IP address to listen
# to. It can either
# contain “*”, an IP address, or a fully qualified
# Internet domain name.
# See also the <VirtualHost> and Listen directives.
#
BindAddress 192.168.255.108:80
```

Por default, a linha BindAddress vem comentada, pois como esta apresentada acima, habilita o acesso a um domínio virtual (em nosso



caso, o IP 192.168.255.108:80 (o :80 estaria indicando a porta 80) que será explicado mais além. Para cada virtual host, é necessária uma entrada “BindAddress e um número ip”.

```
#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which
was built as a DSO
# you
# have to place corresponding 'LoadModule' lines at this
location so the
# directives contained in it are actually available
_before_ they are used.
# Please read the file README.DSO in the Apache 1.3
distribution for more
# details about the DSO mechanism and run 'httpd -l' for
the list of already
# built-in (statically linked and thus always available)
modules in your
# httpd
# binary.
#
```

Liberando Módulos

```
# Note: The order is which modules are loaded is
important. Don't change
# the order below without expert advice.
#
# Example:
# LoadModule foo_module modules/mod_foo.so
#
#LoadModule mmap_static_module modules/
mod_mmap_static.so
LoadModule env_module          modules/mod_env.so
(seguido de uma lista de LoadModule e mais além,
AddModule)
```

Descomentando quaisquer das linhas que comecem com *LoadModule* ou *AddModule*, valida o carregamento de módulos feito na inicialização do *httpd*. Estes funcionam como opções, por exemplo, habilitar ou não o suporte a arquivos *cgi* no server, etc.

Porta Padrão

```
#
# Port: The port to which the standalone server listens.
For
```



ports < 1023, you will need httpd to be run as root initially.

#

Port 80

Responde por default na porta 80. Neste campo, você poderá modificá-la se quiser.

Notificação de Erros

#

ServerAdmin: Your address, where problems with the server should be

e-mailed. This address appears on some server-generated pages, such

as error documents.

#

ServerAdmin root@localhost

O endereço de e-mail para onde será mandado algo se o servidor acusar erro ou anormalidades.

Nome do Servidor

#

ServerName allows you to set a host name which is sent back to clients for

your server if it's different than the one the program would get (i.e.,

use

"www" instead of the host's real name).

#

Note: You cannot just invent host names and hope they work. The name you

define here must be a valid DNS name for your host. If you don't

understand

this, ask your network administrator.

If your host doesn't have a registered DNS name, enter its IP address

here.

You will have to access it by its address (e.g., http://123.45.67.89/)

anyway, and this will make redirections work in a sensible way.

6*/3

#

ServerName vader.suptel



Determina o nome do server principal. Importante: o nome tem que obrigatoriamente constar em DNS (um IP associado a um nome), pois apenas inventando um, não irá adiantar. O modo de chamá-lo seria `http://nome`, mas se o mesmo não estiver em nenhum DNS, coloque o IP (seria `http://numero-ip` para chamá-lo, então, neste caso).

Diretório Padrão e Opções

```
#
# DocumentRoot: The directory out of which you will
# serve your
# documents. By default, all requests are taken from
# this directory, but
# symbolic links and aliases may be used to point to
# other locations.
#
DocumentRoot "/html"
```

Determina o caminho onde estarão os arquivos HTML do servidor principal. Importante: o diretório deve estar com permissão 755 (chmod 755, sendo leitura, escrita e execução para o dono; leitura e execução para grupo e outros que não sejam do grupo nem donos (others)).

```
#
# This should be changed to whatever you set
# DocumentRoot to.
#
<Directory "/html">
#
# This may also be "None", "All", or any combination of
# "Indexes",
# "Includes", "FollowSymLinks", "ExecCGI", or
# "MultiViews".
#
# Note that "MultiViews" must be named *explicitly* –
# "Options All"
# doesn't give it to you.
#
Options Indexes FollowSymLinks Includes
#
# This controls which options the .htaccess files in
# directories can
# override. Can also be "All", or any combination of
# "Options", "FileInfo",
# "AuthConfig", and "Limit"
```




```
#
AllowOverride None
#
# Controls who can get stuff from this server.
#
Order allow,deny
Allow from all
</Directory>
```

Este conjunto de campos determina as opções dos diretórios onde contém documentos HTML a serem acessados irão ter. A primeira “#This should..” deve conter o mesmo diretório que o “DocumentRoot” tem (o */html*).

Se, por exemplo, você tiver apenas um servidor virtual além do principal que responda no diretório */vh* (veremos como fazer essa associação mais além), você terá de ter as entradas *<Directory “/vh”>* e todas as abaixo desta, nem que sejam apenas copiadas, abaixo do término da última.

```
(# Controls who can get stuff from this server.).
#
# UserDir: The name of the directory which is appended
onto a user's home
# directory if a ~user request is received.
#
UserDir public_html
```

Esta opção é bem útil. Cuida de qual diretório o usuário terá de fazer se quiser ter uma página em seu home. No caso, como está configurado, ele precisará criar um diretório *public_html* (o nome pode ser alterado no campo acima) e colocar algo em HTML, podendo ser acessado com:

```
http://nome.da.maquina/~nome-do-usuário.
#
# Control access to UserDir directories. The following
is an example
# for a site where these directories are restricted to
read-only.
#
<Directory /home/*/public_html>
    AllowOverride FileInfo AuthConfig Limit
    Options MultiViews Indexes SymLinksIfOwnerMatch
    <Limit GET POST OPTIONS PROPFIND>
        Order allow,deny
    Allow from all
```



```
</Limit>
<Limit PUT DELETE PATCH PROPPATCH MKCOL COPY MOVE
LOCK UNLOCK>
Order deny,allow
    Deny from all
</Limit>
</Directory>
```

Esta opção coordena os direitos de acesso ao diretório *public_html* dos usuários e vem, por padrão, comentada. No caso, você deve descomentá-la e modificá-la de acordo com o diretório home de seus usuários (por exemplo, o campo *<Directory /home/*/public_html>* diz que, no diretório */home*, todos que existem dentro dele e que tenham *public_html* vão ser passíveis de acesso sob as regras configuradas abaixo desta linha.

```
#
# DirectoryIndex: Name of the file or files to use as a
# pre-written HTML
# directory index.  Separate multiple entries with
# spaces.
#
DirectoryIndex index.html index.htm index.cgi
```

Esta é bastante importante também, pois determina quais nomes de arquivos serão válidos para se realizar a abertura dos mesmos em um browser HTTP. No caso da configuração acima, o server aceitará arquivos de nome *index.html*, *index.htm* e *index.cgi*, como arquivos iniciais de uma home page.

Opção Canonical

```
#
# UseCanonicalName: (new for 1.3) With this setting
# turned on, whenever
# Apache needs to construct a self-referencing URL (a
# URL that refers back
# to the server the response is coming from) it will use
# ServerName and
# Port to form a "canonical" name. With this setting
# off, Apache will
# use the hostname:port that the client supplied, when
# possible. This
# also affects SERVER_NAME and SERVER_PORT in CGI
# scripts.
#
UseCanonicalName On
```



Se ligada, uma página que por exemplo se chame *http://www.teste.com/teste/* e seja acessada como *http://www.teste.com/teste* (sem o “/” no final) seja válida. Se desligada, ele não irá achar.

Log Level

```
#
# LogLevel: Control the number of messages logged to the
# error_log.
# Possible values include: debug, info, notice, warn,
# error, crit,
# alert, emerg.
#
LogLevel warn
```

Determina em que nível o *httpd* irá rodar. A recomendada é a *warn*, pois não causa acúmulo de atividades no Apache e é uma das mais usadas.

Virtual Host

```
#
# If you want to use name-based virtual hosts you need
# to define at
# least one IP address (and port number) for them.
#
NameVirtualHost 192.168.200.2:80
```

Neste, configuramos o IP e porta que o servidor virtual terá. A definição deste é que você não precisa ter vários computadores rodando HTTP servers neles, com apenas um você pode ter *www.teste.com* e *www.teste1.com*, cada um abrindo uma página diferente (em diferentes diretórios da CPU) e cada um possuindo um IP (mas ambos apontarão para a mesma cpu, isso se chama *IP ALIAS*).

```
#
# VirtualHost example:
# Almost any Apache directive may go into a VirtualHost
# container.
#
#<VirtualHost ip.address.of.host.some_domain.com>
#ServerAdmin webmaster@host.some_domain.com
# DocumentRoot /www/docs/host.some_domain.com
#     ServerName host.some_domain.com
#     ErrorLog logs/host.some_domain.com-error_log
#     CustomLog logs/host.some_domain.com-access_log
# common
#</VirtualHost>
```



```
<VirtualHost teste:80>
ServerAdmin root@localhots
DocumentRoot /vh #Diretorio do Virtual host
ServerName teste.dominiolinux
ErrorLog logs/host.some_domain.com-error_log
CustomLog logs/host.some_domain.com-access_log common
</VirtualHost>
```

Ela cuida do servidor virtual e de suas configurações, sendo que o que está comentado (com um # na frente) dá um exemplo do que deverá ser feito (as linhas descomentadas não estão escritas por padrão, estas apresentadas foram digitadas com base no exemplo). Explicarei cada uma delas:

```
<VirtualHost teste:80>
```

Valida o nome “teste” ao servidor virtual e fala em que porta ele irá atender (80)

```
ServerAdmin root@localhots
```

Se o sistema detectar algo de anômalo, um e-mail será enviado a root@localhost.

```
DocumentRoot /vh
```

Designa a pasta onde os arquivos HTML do servidor virtual serão colocados (Lembrando novamente que este diretório deverá ter permissões 775 em seu caminho inteiro).

```
ServerName teste.dominiolinux
Nome e domínio do servidor virtual.
ErrorLog logs/host.some_domain.com-error_log
```

O relatório de erros do servidor virtual vai ser escrito em *logs/host.some_domain.com-error_log*.

```
CustomLog logs/host.some_domain.com-access_log common
```

Log de acessos vai ser escrito em *logs/host.some_domain.com-access_log common*

```
</VirtualHost>
```

Indica o final da configuração do virtual host.

```
[root@localhost /root]# cds
```

E após:

```
[root@localhost /init.d]# ./httpd stop
[root@localhost /init.d]# ./httpd start
```



Virtual Host - Configurando IP Alias

Depois, é necessário informar ao sistema que o mesmo precisa responder num outro IP (192.168.200.2, como definido nas configurações do virtual host) além do IP verdadeiro (pois um virtual host não é nada mais do que fazer um computador responder em outro IP (e outro nome, se assim especificado no DNS), direcionando o pedido HTTP para este ip “falso” e associando a pasta de html referida ao mesmo).

Vamos utilizar então o Linuxconf para adicionar este IP “falso” (técnica chamada de IP ALIAS, anteriormente mencionada).

Entre como root no Linuxconf:

```
[root@localhost]# linuxconf
```

Vá em:

*Ambiente de Rede/Apelidos de IP para máquinas virtuais/
eth0.*

Configure o IP virtual (no caso do nosso, seria 192.168.200.2) e sua máscara.

Depois, aceite e saia do Linuxconf, ativando as mudanças.

Confirme se o novo IP está realmente on-line usando o comando ping:

```
[root@localhost]# ping 192.168.200.2
```

Coloque algum documento HTML no diretório */html* (que, de acordo com a configuração feita, é a pasta do server principal, podendo ter subpastas dentro desta) e em */vh* (configurada para o virtualhost “teste” e que também pode ter subpastas), todas com as devidas permissões 755 previamente mencionadas.

E, para acessar eles, digite em um browser:

`http://192.168.200.1` (para o principal já que o nome é o do servidor principal)

Ou:

`http://teste` (que é o nome do virtual host)

O primeiro tem que estar respondendo no documento HTML válido na pasta */html* e o segundo na */vh*. Para abrir uma página no home do usuário, após o usuário ter criado a pasta *public_html* em seu home e ter dado permissões 755 para mesma, digite:

`http://nome-da-maquina/~nome-do-usuário`
















Deve responder no HTML do diretório:

`/home/nome-do-usuário/public_html.`

Comandos HTTPd

Temos, ainda, as opções de linha de comando para o *daemon httpd*.

-  **-D name:** Define um nome para usar nas Diretivas *<IfDefine name>*.
-  **-d directory:** Especifica o valor da variável *ServerRoot*.
-  **-f file:** Executa comandos no arquivo identificado na inicialização */.../conf/httpd.conf*.
-  **-C “directiva” :** Processa a diretiva antes de ler a arquivos de configurações.
-  **-c “directive”:** Processa a diretiva após ler os arquivos de configuração.
-  **-v:** Mostra a versão do Apache.
-  **-V:** Mostra configurações do compilação.
-  **-h:** Mostra uma lista de opções para utilizar com o *httpd*.
-  **-l:** Fornece uma lista com os módulos compilados no servidor.
-  **-L:** Mostra uma lista de diretivas.
-  **-S:** Mostra as configurações analisadas a partir do arquivo *config*.
-  **-t:** Executa um *check* nos arquivos de configuração. Verifica entrada no *DocumentRoot*.
-  **-T:** Executa um check nos arquivos de configuração. Não verifica entrada no *DocumentRoot*.

Existem também os arquivos de log, servem para verificar acessos, processos, transferências, etc.

Como a maioria dos *daemons*, ele grava um arquivo *pid*, com o número do processo que está sendo executando no *daemon httpd*, ele arquivo ficam em */var/run/httpd.pid*.

Os erros do servidor Apache são registrados em log no arquivo chamado *erro_log*, que fica em */var/log/httpd*.



As transferências de arquivos são também registradas em log, ficam gravadas no arquivo *access.log*, no mesmo diretório que o anterior.

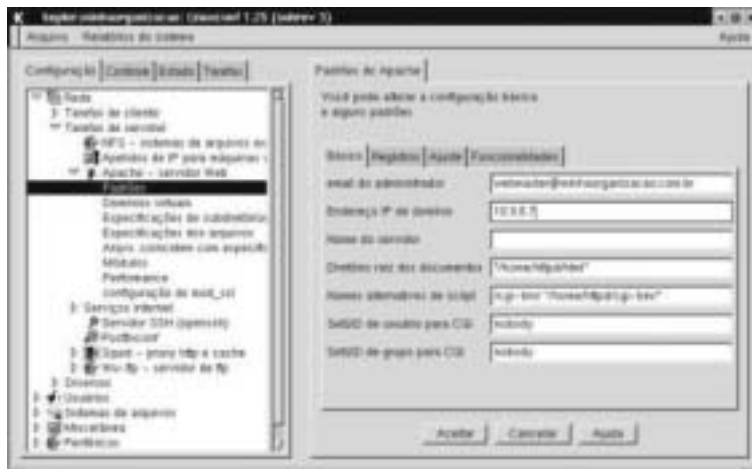
Configurando o Apache pelo Linuxconf

Podemos, ainda, configurar o Apache através do Linuxconf.

Primeiramente, verificamos se o módulo referente ao Apache no Linuxconf está ativado: *Arquivos de controle e sistemas/Configurar os módulos do Configurador Linux*.




Marque o módulo do Apache, saia e entre novamente no Linuxconf.

No Linuxconf, vá em: *Ambiente de rede/Apache - servidor www/Padrões*.








Padrões do Apache







Os campos mais importantes serão explicados abaixo:

-  **Endereço e-mail do administrador:** Para onde o e-mail será enviado automaticamente se houver algum problema.
-  **Domain IP address:** Endereço IP do domínio (do servidor principal), no caso, 192.168.200.1 conforme nosso exemplo.
-  **Nome do servidor:** Um nome válido em DNS do servidor principal, usado se tivermos um servidor DNS na rede.




-  **Diretório raiz dos documentos:** Onde serão lidos os *htmls* do servidor principal.
-  **Registros de transferências, Registros de agentes, Registros de referência:** Logs do sistema, cujo caminho, opcionalmente, pode ser especificado aqui.
-  **Escutando na porta:** Porta onde o servidor principal vai responder (padrão é 80).
-  **Tempo máximo de espera da conexão:** Mesmo que “Timeout” mencionado durante as configurações do *httpd.conf*.
-  **Funcionalidades:** Neste, encontram-se algumas opções que estão explicadas e poderão ser marcadas de acordo com a necessidade.

Aba Funcionalidades

-  **[] Inclusões relativas ao servidor:** Se habilitado, os arquivos com a extensão *.shtml* são processados para expandir tags de inclusões relativas ao servidor, antes dos resultados serem retornados ao cliente.
-  **[] IncludesNOEXEC:** Também habilita inclusões relativas ao servidor, mas a tag *SSI #exec* e *#include:ing CGI-scripts* são desabilitadas.
-  **[] Pode executar programas CGI:** Permite execução de scripts CGI. Normalmente habilitada apenas no sub-diretório *spec* para o diretório CGI.
-  **[] índices:** Quando acessada uma URL terminando em um diretório, o arquivo *index.html* neste diretório é retornado. Se este arquivo não existir e esta opção estiver habilitada, uma lista de diretórios é retornada. Caso esta opção esteja desabilitada, uma mensagem “404 Not Found” é então retornada.
-  **[] Pode seguir links simbólicos:** Caso um diretório ou arquivo acessado seja um link simbólico, o acesso será apenas permitido se esta característica estiver habilitada.
-  **[] Seguir links simb. de mesmo dono:** Como na descrição acima, porém, o dono do link simbólico e seu arquivo alvo devem ser os mesmos. Isto permite que usuários possuam links simbólicos para seus próprios









arquivos, mas não para os arquivos mais sensíveis do sistema, por exemplo */etc/passwd*.

-  [] **Visões múltiplas:** Permitem que o navegador do cliente e o servidor web negociem em qual formato e linguagem os dados devem ser retornados. Uma maneira de fazer o servidor web capaz disso, é armazenar várias imagens e formatos de dados em arquivos com certas extensões e, então, habilitar multivisões.

Saia e acesse o seguinte item:

Domínios Virtuais/Adicionar

-  **Nome da máquina virtual:** Nome da máquina virtual em que o servidor irá rodar.
-  **Endereço e-mail do administrador:** Para onde o e-mail será enviado se ocorrer algo errado.
-  **Domain IP address:** O IP do Servidor Virtual.
-  **Diretório raiz dos documentos:** Pasta onde os documentos HTML estarão.
-  **Nomes alternativos de servidor:** Seria um apelido para este servidor virtual.
-  **Funcionalidades:** Opções relativas a este servidor virtual.



Agora, vamos configurar item de diretórios em *Especificações de subdiretórios*:

Neste item, encontraremos onde os documentos do servidor principal ficarão (inclui ícones, HTMLs e *cgis*). Depois de *Sair* ou *Aceitar* estas opções, entre em *Modules*.

Neste campo, você poderá adicionar ou retirar módulos relativos ao Apache (aumentando as suas funções ou restringindo-as).

Voltando ao menu principal do Apache, selecione *Performance*.

Em:

-  **Inicia servidores:** Número de servidores que iniciam logo no boot do HTTPd.
-  **Núm máx de clientes por servidores, Núm máx de req. por processo filho, Núm mín de serv. aguardando conexão, Núm máx de serv. aguardando conexão:** Pode-se regular o número de servidores, clientes e processos filhos.



Keeps Alive: Nesta seção, pode-se ligar ou desligar o suporte deste recurso, o máximo de pedidos e após quanto tempo o mesmo expira.



Tempo máximo de espera da conexão: “timeout”.

Configurando Domínios Virtuais

Com esta configuração, já é possível disponibilizar páginas em seu servidor Apache principal. Já existe, por padrão, um arquivo *index.html* no diretório */home/httpd/html* e para testar se sua configuração está correta, saia do Linuxconf ativando as mudanças. Caso tenha sido alterado o diretório raiz dos documentos, coloque algum documento HTML neste diretório (lembrando que este deve ter a permissão de acesso 755, ou seja, toda a árvore deve possuir tal permissão).

Explicaremos melhor o que é um Domínio Virtual neste momento, pois fizemos a configuração, mas não sabemos o que significa.

O Domínio Virtual, como o próprio nome diz, é um domínio (referente à web) diferente do original, mas rodando na mesma máquina, ou seja, teremos dois endereços na web, vamos supor que você esteja numa intranet e seu servidor seja 192.168.200.1 *dominiolinux.com.br*, mas também quer o domínio *intranet.dominiolinux.com*.

Existem dois casos utilizando domínio virtual, uma que é configurar os domínios virtuais no mesmo endereço IP e a outra, utilizando endereços IPs diferentes.

O arquivo onde as informações devem ser adicionadas é o:

/etc/httpd/conf/httpd.conf

Neste arquivo, procure pela seção # *VirtualHost example*.

Configurando um domínio utilizando o mesmo IP:

```
NameVirtualHost 192.168.200.1:80
<VirtualHost 192.168.200.1:80>
ServerAdmin root@dominiolinux.com.br
ServerName dominiolinux.com.br
DocumentRoot /html
ErrorLog logs/error_log
CustomLog logs/access_log common
</VirtualHost>
```

Para conferir a configuração, execute:



```
[root@localhost]# httpd -S
VirtualHost configuration: 192.168.200.1:80 is a
NameVirtualHost
default server dominiolinux.com.br (/etc/httpd/conf/
httpd.conf:1008)
192.168.200.1:80 intranet.dominiolinux.com (/etc/httpd/
conf/httpd.conf:1192)
```

Configurando três domínios utilizando três IPs diferentes:

```
<VirtualHost 192.168.200.2:80>
ServerAdmin root@dominiolinux.com.br
ServerName dominiolinux.com.br
DocumentRoot /html
ErrorLog logs/error_log
CustomLog logs/access_log common
</VirtualHost>
```

E, novamente, digite no Shell *HTTPd -S* para verificar se está tudo OK.

Logicamente, isso só vai funcionar se você tiver configurado entradas no servidor DNS. Como não abordamos aqui a montagem de um servidor DNS, temos outra forma de burlar esse problema.

Primeiramente, no seu arquivo */etc/hosts*, coloque as seguintes linhas:

```
127.0.0.1 localhost.localdomain localhost
192.168.200.1 Dominiolinux.com.br DominioLinux
192.168.200.2 intranet.dominiolinux.com Intranet
```

No máquina cliente, se for **Linux** faça como o arquivo acima, caso seja Windows, temos duas possibilidades: o arquivo *C:/Windows/hosts* ou *C:/Windows/lmhosts*, no Windows 2000 esses arquivos *hosts* e *lmhosts* ficam em *WINNT/System32/etc/drivers*.

Adicione as seguintes linhas:

```
192.168.200.1 dominiolinux.sytes.net
192.168.200.2 intranet.dominiolinux.com
```

Se você utilizar o *lmhosts*, vá ao prompt do MS-DOS e digite:

```
nbtstat -R
```

Com isso, veja a possibilidade de você poder ter várias páginas para vários endereços, como *contabilidade.suarede*, *marketing.suarede*, etc.

Aliado a um servidor DNS, ficará ainda mais fácil pelo fato de não ser necessário criar os arquivos *lmhosts* e *hosts*.



Autenticação de Usuário no Apache

Podemos utilizar, juntamente com o recurso de domínios virtuais, o recurso de autenticação de usuário, por exemplo, você tem uma página com o pagamento dos funcionários, que deve estar disponível somente para os usuários do RH.

Criamos um domínio virtual para o RH ou apenas um diretório dentro de `/home/httpd/html/RH`.

Se for feito, no segundo caso, somente o diretório, quando digitar o site deve-se colocar algo do tipo:

`http://dominio/RH`

Enfim, sendo Domínio Virtual ou diretório, deve-se criar um arquivo chamado `.htaccess` no diretório onde se deseja proibir o acesso.

```
vi /home/httpd/html/RH/.htaccess
```

Esse arquivo, deve possuir o seguinte conteúdo:

```
AuthType Basic
AuthUserFile /home/httpd/html/RH/httpd.users
AuthName "Diretorio do RH"
require valid-user
```

Ainda no diretório onde ficará restrito pelo Apache (somente para localização mais rápida, pois na realidade é feito apontamento do arquivo no conteúdo de `.htaccess`) crie um usuário e senha com o comando:

```
[root@localhost root]# htpasswd -c .httpd.users
nomedousuário
```

O parâmetro `-c` é usado somente no primeiro cadastro para a criação do arquivo `.htpasswd`.

Feito isso, certifique-se que o item `AllowOverride` está como no exemplo abaixo, localizando no arquivo `httpd.access`:

```
AllowOverride AuthConfig
```

Para completar, reinicie o Apache com os comandos:

```
[root@localhost root]# cds; ./httpd restart;
```

Dessa forma, sempre que o usuário tentar acessar a página do RH será solicitado nome de usuário e senha, pode-se criar quantos usuários forem necessários.



Para descadastrar os usuários, acesse o arquivo `httpd.user` e retire a linha com o nome de usuário que deseja retirar a permissão.

Criptografia no Apache

A criptografia é muito usada no mundo corporativo. Também é conhecida como SSL.

Porém, quando o servidor Apache está utilizando suporte à SSL, então, primeiramente, deve-se fazer o registro do servidor SSL em empresas devidamente credenciadas.

No site www.apache.org, existem mais informações a respeito.

Os pacotes necessários para instalação são:

- 📁 SSLeay (<ftp://ftp.pca.dfn.de/pub/tools/net/ssleay/SSLeay-0.9.0b.tar.gz>);
- 📁 OpenSSL (<http://www.openssl.org/source/openssl-0.9.6.tar.gz>);
- 📁 mod_ssl (http://www.modssl.org/source/mod_ssl-2.8.1-1.3.19.tar.gz);

O padrão de instalação segue como outros arquivos binários:

- 📁 Instalação SSLeay, necessário para geração de *public* e *private keys*;

```
# tar -xvzf SSLeay-0.9.0b.tar.gz;
# cd SSLeay-0.9.0b.
```

Edite o arquivo *Configure*, alterando a primeira linha de */usr/local/bin/perl* para */usr/bin/perl*.

```
# ./Configure linux-elf
# make
# make install
```

- 📁 Instalação OpenSSL, responsável pela criptografia dos dados:

```
# tar -xvzf openssl-0.9.6.tar.gz
# cd openssl-0.9.6
# ./config
# make
# make test
# make install
```



Instalação `mod_ssl`, módulo para o Apache se relacionar com o OpenSSL:

```
# tar -xvzf mod_ssl-2.8.1-1.3.19.tar.gz
# cd mod_ssl-2.7.1-1.3.14
# ./configure --with-apache=../apache_1.3.19
```

No caso, a versão 1.3.19.

Habilitando PHP no Apache

Outro item muito usado nos servidores é o suporte ao PHP.

Para habilitar o módulo PHP, edite o arquivo `/etc/httpd/conf/httpd.conf` e descomente as seguintes linhas (retire o símbolo “#”):

```
LoadModule php4_module          modules/libphp4.so
AddModule mod_php4.c
  AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
DirectoryIndex index.php index.html index.wml
```

Para funcionar de forma correta, você deve ter o pacote PHP4 e `mod_php4` instalados.

Reinicie o serviço HTTPd. Se não ocorrer erro, é quase certo que está tudo OK.

Primeiramente, para testar a configuração do Apache, execute um browser e acesse o endereço da máquina local.

Se esta página não for mostrada, verifique o arquivo `/etc/httpd/logs/error_log` que exhibe os erros do Apache e tente configurá-lo novamente.

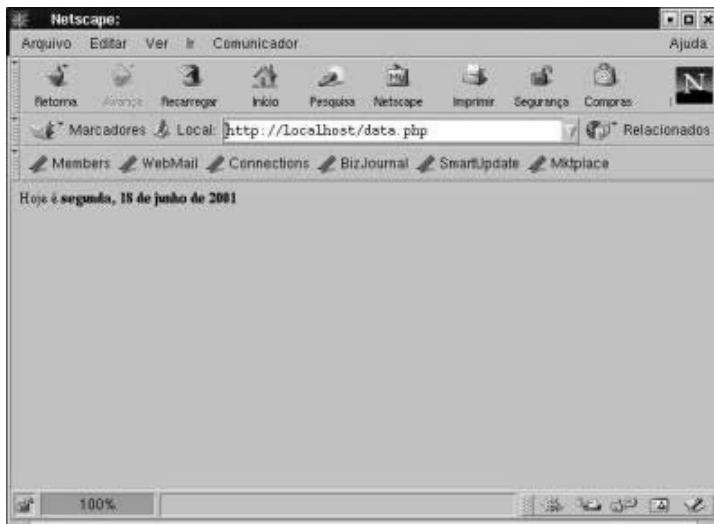
Para testar a configuração do PHP, primeiramente acesse o diretório de arquivos Apache em `/home/httpd/html`. Em seguida, utilizando o seu editor de textos, crie um arquivo chamado `data.php`, colocando o seguinte conteúdo:

```
<html>
<? setlocale ("LC_TIME", "pt_BR"); ?>
<body>
Hoje é <b><? print(strftime ("%A, %d de %B de %Y"));
?><b>
</body>
</html>
```

Use o Netscape® para visualizar a página.

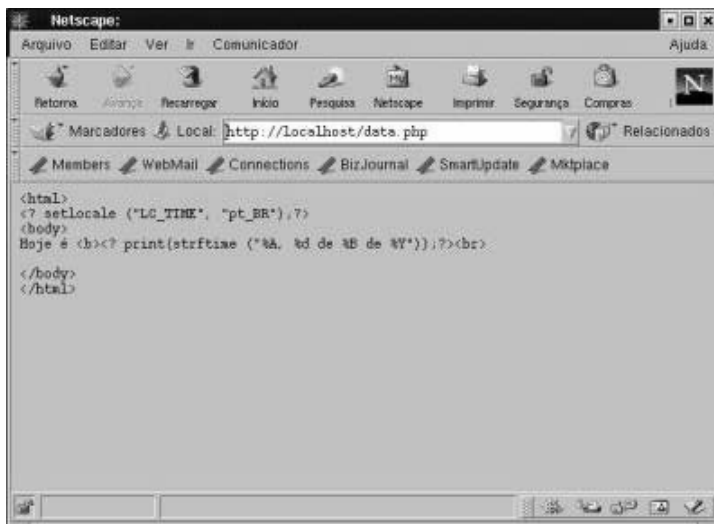


Se o *mod_php* estiver funcionando, você verá a página semelhante a da figura 10.3.



Teste PHP

Se a configuração estiver incorreta, você provavelmente irá ver a figura 10.4:



Falha no PHP

Se este for o caso, revise os passos da instalação para verificar o que está incorreto. Além disso, verifique o arquivo */etc/httpd/error_log*, que deverá conter informações sobre o erro.



11

Servidor FTP

Introdução

O significado de FTP é *File Transfer Protocol*. Como nós sabemos, ele faz parte do TCP/IP. O FTP é um dos protocolos mais utilizados na Internet. Por quê? Ele está disponível para quase todos os sistemas operacionais, tais como **Linux**, **Unix**, **Windows**, etc.

E além do cliente poder usar comandos como *wget* e FTP no **Linux**, existem interfaces gráficas para clientes Windows e **Linux**. E além de tudo isso, pode usar o próprio browser como Netscape , Opera, Internet Explorer, entre outros, para iniciar um download de qualquer FTP.

O FTP é utilizado para transferência de arquivos, seja transferindo do servidor para o cliente, como do cliente para o servidor. Possui recursos de comandos que veremos logo adiante e a possibilidade de gerenciar diretórios, como criar e excluir diretórios, mover arquivos de um para outro, listar, renomear, etc.

Pode-se utilizar o método de FTP anônimo e com uso de usuários. No nosso caso, instalaremos das duas formas.

O **Conectiva Linux** utiliza o servidor FTP chamado WU-FTP por padrão. Existem outros, mas veremos apenas este.

Comandos FTP
















Será mostrado, aqui, os comandos do FTP já conectado e antes de conectar.

Abaixo, vemos uma listagem com dos comandos que podem ser usados no cliente sob o prompt ou Shell:

















-v: A opção *Verbose* força o FTP a mostrar todas as saídas do servidor remoto, bem como reportar os dados estatísticos de transferência.
















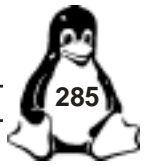
-  **-n:** Impede o FTP de tentar estabelecer um “auto-login” numa conexão inicial.S.
-  **-i:** Desliga os prompts interativos durante múltiplas transferências de arquivos.
-  **-d:** Ativa depuração.
-  **-g:** Desabilita o uso de coringas (*, ?, [a-z], etc.) em nomes de arquivos.
-  **! [comando [parâmetros]]:** Abre uma sessão interativa na máquina local. Se existirem parâmetros, o primeiro é tomado como um comando a ser executado diretamente com os parâmetros restantes como opções desse comando.
-  **\$ nome-da-macro [parâmetros]:** Executa a macro específica. Parâmetros são passados para a macro sem o uso de coringas.
-  **account [senha]:** Fornece uma senha suplementar exigida pelo sistema remoto para acessar recursos após uma entrada ter sido feita com sucesso. Se nenhum parâmetro for incluído, o sistema solicitará uma senha para o usuário.
-  **append arquivo-local [arquivo-remoto]:** Adiciona um arquivo local a um arquivo na máquina remota.
-  **ascii:** Ajusta o tipo de transferência de arquivo para ASCII na rede. Este é o tipo padrão.
-  **bell:** Faz com que um sinal sonoro seja soado após cada comando de transferência de arquivo completado.
-  **binary:** Ativa e desativa o uso de transferência de arquivos para suportar binários.
-  **bye:** Termina a sessão FTP com o servidor remoto e sai do FTP.
-  **Case:** Quando case está habilitado, os nomes de arquivos do computador remoto onde todas as letras sejam maiúsculas serão escritos com letras minúsculas no diretório local. O padrão é que o case esteja desabilitado.
-  **cd diretório-remoto:** Muda o diretório de trabalho na máquina remota para diretório-remoto.
-  **cdup:** Muda o diretório de trabalho na máquina remota para o diretório imediatamente superior ao atual.




















-  **chmod modo nome-do-arquivo:** Muda as permissões do arquivo *nome-do-arquivo* no sistema remoto.
-  **Close:** Termina a sessão FTP com o servidor remoto e retorna ao interpretador de comandos.
-  **cr:** Habilita/desabilita a retirada do retorno de carro (CR).* durante a recepção de arquivos ASCII.
-  **delete arquivo-remoto:** Apaga o arquivo *arquivo-remoto* na máquina remota.
-  **debug [valor-de-depuração]:** Habilita/desabilita o modo de depuração.
-  **dir [diretório-remoto] [arquivo-local]:** Mostra uma listagem do conteúdo do diretório no diretório-remoto, e, opcionalmente, coloca a saída no arquivo-local.
-  **disconnect:** Um sinônimo para *close*.
-  **form formato:** Ajusta a forma de transferência de arquivo para formato. O formato padrão é “file”.
-  **get arquivo-remoto [arquivo-local]:** Transfere o arquivo-remoto e o armazena na máquina local. Se o nome do arquivo local não for especificado, ele terá o mesmo nome que na máquina remota).
-  **glob:** Habilita/desabilita a expansão de nomes de arquivos para os comandos *mdelete*, *mget* e *mput*. Se a expansão for desabilitada com o comando *glob*, os nomes de arquivos são interpretados literalmente e não são expandidos.
-  **hash:** Habilita/desabilita um sinal de marcação (“#”) para cada bloco de dados transferido. O tamanho de um bloco de dados é de 1024 bytes.
-  **help [comando]:** Mostra uma mensagem informativa sobre o significado do comando. Se nenhum parâmetro for especificado, o FTP mostrará uma lista de todos os comandos válidos.
-  **idle [segundos]:** Ajusta o contador de inatividade na máquina remota para segundos. Se *segundos* for omitido, o valor atual é mostrado.
-  **lcd [diretório]:** Muda o diretório de trabalho na máquina local. Se nenhum diretório for especificado, o *homedir* do usuário será usado.
















-  **ls [diretório-remoto] [arquivo-local]:** Mostra uma listagem do conteúdo de um diretório da máquina remota.
-  **macdef nome-da-macro:** Define uma macro.
-  **mdelete [arquivos-remotos]:** Apaga os arquivos-remotos na máquina remota.
-  **mdir arquivos-remotos arquivo-local:** Funciona como o *dir*, exceto que múltiplos arquivos podem ser especificados.
-  **mget arquivos-remotos:** Expande os arquivos-remotos na máquina remota e faz um *get* para cada nome de arquivo obtido. Cria um diretório na máquina remota.
-  **mls arquivos-remotos arquivo-local:** Funciona como o *nlist*, exceto que múltiplos nomes de arquivos podem ser especificados e que arquivo-local precisa ser especificado.
-  **mode [nome-do-modo]:** Ajusta o modo de transferência de arquivos para nome-do-modo. O modo padrão é o modo “stream”.
-  **modtime nome-do-arquivo:** Mostra a hora da última modificação do arquivo na máquina remota.
-  **mput arquivos-locais:** Carrega os arquivos especificados. Se coringas forem fornecidos, os nomes são expandidos.
-  **newer nome-do-arquivo [arquivo-local]:** Recebe o arquivo apenas se a hora de modificação do arquivo remoto for mais recente que a do arquivo que se encontra atualmente no sistema.
-  **nlist [diretório-remoto] [arquivo-local]:** Mostra uma lista dos arquivos de um diretório da máquina remota. Se o diretório-remoto não for especificado, o diretório atual será usado.
-  **nmap [padrão-de-entrada padrão-de-saída]:** Ativa ou desativa o mecanismo de mapeamento de nomes de arquivos.
-  **ntrans [caracteres-de-entrada [caracteres-de-saída]]:** Ativa ou desativa o mecanismo de tradução de caracteres em nomes de arquivos. Se nenhum parâmetro for especificado, o mecanismo de tradução de caracteres em nomes de arquivos será desativado.



-  **open máquina [porta]:** Estabelece uma conexão para o servidor FTP da máquina especificada. Um número de porta opcional pode ser especificado, fazendo com que o FTP tente contatar um servidor FTP nesta porta.
-  **prompt:** Habilita/desabilita o prompt interativo.
-  **proxy comando-do-ftp:** Executa um comando do FTP numa conexão de controle secundária.
-  **put [arquivo-local] [arquivo remoto]:** Carrega um arquivo da máquina local para a máquina remota.
-  **pwd:** Exibe o diretório atual da máquina remota.
-  **quit:** Sinônimo de *bye*.
-  **quote [arg] [arg1]:** Envia uma lista de argumentos para o servidor.
-  **recv arquivo remoto [arquivo local]:** Sinônimo de *get*.
-  **reget arquivo-remoto [arquivo-local]:** *Reget* age semelhante ao *get*, exceto pelo fato de comparar o arquivo local com o remoto. Se remoto for maior, é provável que houve uma interrupção e o *reget* iniciará a transferência de onde havia sido interrompida.
-  **remotestatus [nome-do-arquivo]:** Sem nenhum parâmetro, mostra a situação da máquina remota. Se *nome-do-arquivo* for especificado, mostra a situação do nome-do-arquivo na máquina remota.
-  **rename [de] [para]:** Renomeia o arquivo *de* na máquina remota para o arquivo *para*.
-  **reset:** Limpa a fila de respostas.
-  **restart marcação:** Reinicia o próximo *get* ou *put* na marcação indicada.
-  **rmdir nome-do-diretório:** Deleta um diretório na máquina remota.
-  **runique:** Habilita/desabilita a armazenagem de arquivos no sistema local com nomes de arquivos únicos.
-  **send arquivo-local [arquivo-remoto]:** Um sinônimo para *put*.
-  **sendport:** Habilita/desabilita o uso do comando *PORT*.



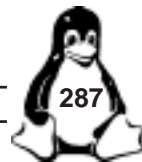
-  **site arg1 arg2...**: Os parâmetros especificados são enviados para o servidor FTP remoto como um comando *SITE*.
-  **size nome-do-arquivo**: Retorna o tamanho do nome-do-arquivo na máquina remota.
-  **status**: Mostra a situação atual do FTP.
-  **struct [nome-da-estrutura]**: Ajusta a estrutura de transferência do arquivo para *nome-da-estrutura*. Por padrão, a estrutura “stream” é usada.
-  **sunique**: Habilita/desabilita a armazenagem de arquivos na máquina remota sob nomes de arquivos únicos.
-  **system**: Mostra o tipo do sistema operacional rodando na máquina remota.
-  **tenex**: Ajusta o tipo de transferência de arquivo para aquele necessário para falar com máquinas TENEX.
-  **trace**: Habilita/desabilita o rastreamento de pacotes.
-  **type [nome-do-tipo]**: Ajusta o tipo da transferência de arquivo para *nome-do-tipo*. Se nenhum tipo for especificado, o valor atual é mostrado. O valor padrão é o ASCII para redes.
-  **umask [nova-máscara]**: Ajusta a máscara padrão para o usuário na máquina remota para *nova-máscara*. Se o parâmetro *nova-máscara* for omitido, o valor atual é mostrado.
-  **user nome-do-usuário [senha] [senha-para-conta]**: Identifica o usuário para o servidor FTP remoto.
-  **verbose**: Habilita/desabilita o modo de detalhamento. No modo de respostas detalhadas, todas as resposta vindas do servidor FTP são mostradas para o usuário.
-  **? [comando]**: Sinônimo de help.

Utilizando o FTP no Cliente

São muitos os comandos, com certeza. Mas não é necessário decorar todos. A maioria dos usuários utiliza os principais comandos para trabalhar com o FTP, serão mostrados a seguir.

O primeiro passo é conectar no host:

```
[root@Stato /root]# ftp dominiolinux.sytes.net
```



```
Connected to dominiolinux.sytes.net.  
220 Stato.Dominiolinux FTP server (Version wu-2.6.1(1)  
Sun Sep 24 18:36:35 BRT 2000) ready.  
Name (dominiolinux.sytes.net:root): stato  
331 Password required for stato.  
Password:  
230 User stato logged in.  
Remote system type is UNIX.  
Using binary mode to transfer files.  
ftp>
```

Veja que foi digitado, o nome do site para o FTP conectar, posteriormente, foi solicitado nome de usuário e senha. Em um servidor que aceita conexão anônima, o nome de usuário e senha seriam:

```
username : anonymous  
password : email(stato@dominiolinux.com.br)
```

O e-mail necessariamente não precisa existir pelo fato de não ser verificado (pelo menos no momento), mas tem que ser de formato de e-mail, tipo nome@dominio.

Depois de digitado o nome de usuário e a senha, o FTP fica aberto esperando os comandos.

Um dos comandos mais utilizados é o *get*. Veja o exemplo abaixo:

```
get arquivo.txt
```

Com isso, ele salvará o *arquivo.txt* do servidor na máquina local.

Para verificar qual diretório está no servidores, utilize:

```
pwd
```

Para verificar qual o diretório local:

```
lcd
```

Podemos pegar vários arquivos ao mesmo tempo:

```
mget *.txt
```

Outros comandos são de envio, supondo que você tenha um diretório seu no servidores FTP, onde você guarde seus arquivos por falta de espaço no seu HD.

```
put arquivo.txt
```

Ou:

```
mput *.txt (Para vários arquivos)
```




Certamente, conhecendo esses comandos você já poderá gerenciar seu download de qualquer servidor FTP, mas agora vamos mais adiante: montar nosso próprio servidor.



Instalando Servidor FTP

O primeiro item que temos que discutir é se o nosso servidor FTP terá ou não acesso anônimo. No nosso exemplo ele terá. Caso não queira liberar acesso anônimo, não instale o pacote *anonftp*.

Para instalar nosso servidor, utilizaremos três pacotes:

-  **Wu-ftp**
-  **anonftp**
-  **linuxconf-wuftp**

Após instalar os pacotes acima, certifique-se primeiramente de que a linha abaixo esteja presente no arquivo */etc/inetd.conf* e que ela não esteja comentada:

```
ftp stream tcp nowait root /usr/sbin/tcpd in.ftpd -l -a
```

Feito isso, precisamos parar e reiniciar o serviço inet:

```
service inet stop  
service inet start
```

Com isso, nosso servidor já está em pleno funcionamento, mas precisamos configurar alguns itens.

Configurando Servidor FTP

Execute o Linuxconf e vá até o menu *Configuração/Rede/Tarefas de servidor/Wu-ftp/servidor de FTP*.








Configurações Básicas do Server FTP



Acima, temos a figura que consta com configurações básicas do servidor FTP. As configurações são as seguintes:


Na aba *Diversos* devem ser configuradas as seguintes opções:

-  **E-mail do administrador:** Endereço de correio eletrônico do administrador do sistema.
-  **Grupo de convidados:** Aqui, você pode informar o grupo do **Linux** ao qual o usuário *anonymous* pertencerá. Por padrão, o usuário *anonymous* pertence ao grupo *nobody* se este campo não for preenchido.
-  **Arquivo de banner:** O conteúdo do arquivo de *banner* que será mostrado aos usuários no momento em que estes acessarem com sucesso o seu servidor. O conteúdo do arquivo será mostrado ao usuário antes da conexão em si. Quando terminado, ele gravará um arquivo chamado *.message* no diretório padrão. Caso queira criá-lo manualmente, não haverá problema algum.
-  **Mensagem de encerramento:** Se o arquivo informado existir, sempre que um usuário tentar acessar o servidor, seu conteúdo será mostrado e o servidor irá fechar a conexão.
-  **Permitir acesso anônimo:** Pode permitir ou proibir os acessos anônimos.

Temos outras configurações que devem ser feitas na guia *Controle*. Na realidade, existem três telas de configuração de controle de acesso:

- 1. Usuários reais:** Controle de acesso dos usuários reais do seu sistema. Esses são os usuários que têm contas em sua rede.
- 2. Usuários convidados:** Controle de acesso a usuários convidados.
- 3. Anônimos:** Controle de acesso de usuários que se conectam anonimamente.

As três telas são absolutamente idênticas. As opções disponíveis são:

-  **Pode requisitar arquivos comprimidos:** Define se o usuário pode requisitar que seus arquivos sejam compactados.



Pode requisitar arquivos tar: Especifica se o usuário tem permissão de solicitar o arquivamento de arquivos transmitidos com o tar.



Pode usar chmod para arquivos: Especifica se o usuário pode modificar as permissões de arquivos localizados no servidor FTP.



Pode excluir arquivos: Define se o usuário tem permissão de apagar arquivos localizados no servidor FTP.



Pode atualizar arquivos: Define se o usuário pode sobrescrever arquivos no servidor.



Pode renomear arquivos: Especifica se o usuário pode modificar o nome dos arquivos localizados no servidor.



Registrar transferências recebidas: Especifica se o servidor deve manter um registro de arquivos recebidos.



Registrar transferências expedidas: Especifica se o servidor deve manter um registro de arquivos enviados.

E na última guia, *Tempos de Espera*, deixamos como está, pois já estão preenchidos os padrões para o funcionamento do servidor.

Com essas configurações, o servidor já está pronto, mas normalmente os administradores não deixam ninguém gravar nada no servidor, mas se por algum motivo específico for necessário gravar, usa-se uma solução diferente de deixar todos os usuários gravar em seus homes.

A Conectiva recomenda que você não permita o acesso de escrita em seu servidor. Caso não tenha escolha, você deve ter alguém responsável por monitorar os arquivos gravados em seu servidor.

A solução seria criar um local e um grupo específico que possa gravar no servidor.

Crie um diretório chamado *incoming* dentro do diretório FTP:

```
# cd /home/ftp
# mkdir incoming
```

Crie um usuário e grupo específicos para:

```
# chown usuario incoming
# chgrp grupo incoming
# chmod 3773 incoming
```



Devemos, por último, editar o arquivo */etc/ftpaccess* para permitir a escrita ao diretório */incoming*.

```
upload HOME DIR GRAVA USUARIO GRUPO PERMS DIRS
```

Veja a linha que você terá que adicionar no arquivo:

```
upload /home/ftp /incoming yes usuario grupo nobody 0400  
dir
```

Em PEMS, depois de gravados, os arquivos terão as permissões trocadas para estas permissões. Em nosso exemplo, definimos que os arquivos gravados neste diretório passariam a ter permissão 0400, ou seja, apenas para leitura do dono (no caso, *usuário.grupo*).

Para testar a configuração, use um Shell e digite:

```
ftp localhost
```

Faça logon como usuário normal, anonymous e nos grupos criados para gravar.

Com o nosso servidor no ar, se algo ocorrer errado, verifica as permissões dos usuários reais pelo Linuxconf se a linha do arquivo *ftpaccess* foi escrita corretamente.



12

Servidor NFS






Introdução

Neste capítulo, falaremos sobre o servidor NFS (Network File System), que é um protocolo muito popular, presente na maioria dos sistemas operacionais, usado para compartilhar informações na rede.

Utilização

Como dito anteriormente, o NFS é utilizado para compartilhar discos ou HD em uma rede. Com isso, muitos usuários podem compartilhar áreas de disco em um servidor que para eles é transparente, parecendo ser um HD da própria máquina.

Alguns dos principais recursos do NFS estão descritos abaixo:

-  O NFS pode ser usado tanto em rede pequenas como numa corporativa;
-  Utilizando este recurso, automaticamente torna-se transparente para o usuário;
-  É fornecido pelo cache acesso rápido às informações em disco local;
-  O NFS é implementado de forma que a administração fique centralizada, facilitando e reduzindo a administração, por exemplo, para backup das pastas de usuários.
-  O NFS fornece suporte para sistemas de cliente sem disco e sem dados. Esta é uma solução muito interessante, onde teremos um terminal “burro”, que se conecta ao servidor, loga e o NFS disponibiliza um diretório para o usuário. Esta solução economiza muito hardware.



Compartilhando um diretório com o NFS, o mesmo parece ser parte do sistema de arquivo local de uma máquina cliente ao invés de um sistema remoto, tornando isso transparente para este usuário. Ele pode ser instalado em redes corporativas, mas não tem restrição quanto a redes pequenas.

Suponha que você tenha uma máquina no seu quarto e outra no escritório, podemos montar um servidor NFS no escritório, já que é o micro mais usado, e quando estivermos no quarto, continuaríamos a ter acesso a esse micro no escritório.

O NFS é um protocolo, mas pode ser usado com o TCP/IP sem problema algum. Aliás, ele foi projetado para isso, pois o TCP/IP está presente em todas as redes praticamente.

Ainda temos a possibilidade de ser, ao mesmo tempo, um servidor e um cliente NFS, obtendo recursos de outro servidor simultaneamente. O protocolo NFS fornece a capacidade dos clientes manipularem arquivos e diretórios como se fosse localmente, como *rename*, *mkdir*, *write* e *read*.

Instalando o NFS

A instalação do servidor é tão simples como os outros aplicativos. Localize os pacotes abaixo em seu CD no Conectiva ou faça download na Internet dos mesmos:

```
nfs-server*
nfs-utils
rpm -ivh nfs-*

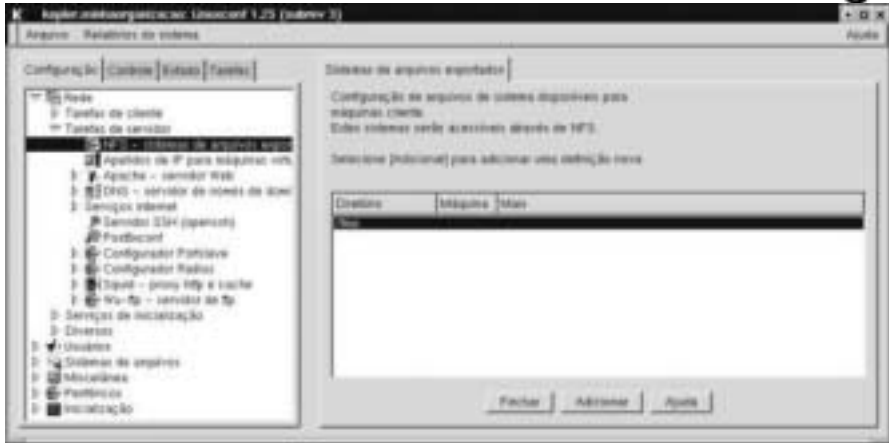
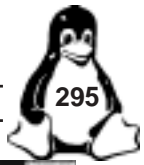
nfs-server #####
nfs-utils   #####
```

Configurando o Servidor NFS

Pronto. Já temos nosso servidor instalado. Agora, precisamos configurá-lo de modo que ele possa exportar os diretórios que serão compartilhados.

Podemos configurá-lo através do Linuxconf em *Ambiente de Rede/Tarefas de Servidor/NFS*.

Veja a figura abaixo:



Configuração básica NFS no Linuxconf

Na tela *Sistema de Arquivos Exportados*, mostrará os diretórios que estão sendo exportados.

Para editá-lo, dê um clique duplo sobre o mesmo.

A exportação de arquivos também pode ser feita manualmente, veremos adiante como funciona, mas primeiramente vamos ver como funciona a criação de um novo diretório.

Nesta mesma tela no Linuxconf onde está o arquivo compartilhado, clique em *Adicionar*, abrir-se-á uma guia chamada *Um Sistema de Arquivos Exportados*. Nestes campos, serão inseridos informações sobre o compartilhamento. Na pior das hipóteses, você fará isso em dez minutos. Pois é, muito fácil exportar tais diretórios. O princípio básico é que a rede esteja configurada corretamente.

Abaixo, segue a descrição dos campos para criação de um novo diretório a ser exportado.



Caminho para exportar: Aqui, você define o diretório a ser exportado. Este será o diretório que será montando pelo cliente.



Comentário: Apenas um comentário ilustrativo. Pode ser usado para informar alguma observação sobre o diretório.



Nome do cliente: Neste campo, você pode definir quais máquinas cliente (separadas por vírgula) poderão acessar este diretório. Se nenhum cliente for especificado, qualquer máquina poderá conectar-se. Você pode, ainda, utilizar coringas para definir as máquinas clientes. Por



exemplo, se você quer dar acesso a todas as máquinas de seu domínio, poderá especificar algo como:



***.minharede.** Observe que este e os próximos campos se repetem, permitindo que você defina opções para grupos diferentes de clientes.



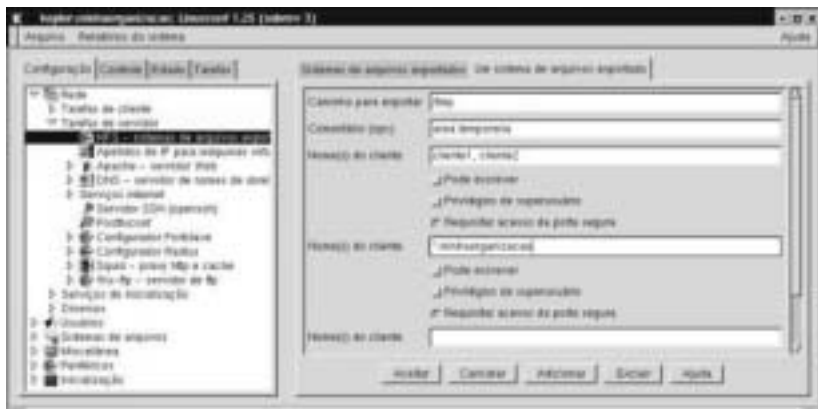
Pode escrever: Indica se o diretório será exportado apenas para leitura ou se os usuários poderão gravar nele.



Privilégios de superusuário: Normalmente, o superusuário acessa diretórios remotos com privilégios de nobody. Você pode especificar esta opção para que o superusuário continue tendo esse acesso quando entrar no diretório.



Requisitar acesso da porta segura: Você pode determinar que apenas conexões seguras (com número baixo de porta de origem) possam montar o diretório.



Configurações de Diretórios Exportados

Agora vamos falar sobre a configuração manual do *arquivo /etc/exports*. Quando fizemos todo esse processo de exportação, na realidade alteramos apenas o arquivo */etc/exports*.

Vamos fazer um teste criando um diretório chamado teste na raiz:

```
cd /
mkdir teste
```

Agora, no Linuxconf iremos compartilhar este diretório:

1. No caminho, coloque */teste*.
2. No primeiro cliente, coloque 192.168.200.2 ou o nome da máquina caso haja um DNS na rede ou até mesmo o nome da máquina no arquivo hosts do servidor.











3. No segundo cliente, coloque 192.168.200.3.
4. Marque a opção pode escrever.
5. Saia do Linuxconf.

Agora, dê uma olhada no arquivo `/etc/exports`.

```
/teste 192.168.200.3(rw) 192.168.200.2(ro)
```

Vejam que ele colocou o diretório que será exportado e quais suas permissões para 192.168.200.2 (somente leitura) e 192.168.200.3 (leitura e escrita).

Abaixo, temos uma lista das principais permissões para uso do arquivo `exports`:

-  **root_squash:** Esta opção nega ao superusuário, nos hosts especificados, qualquer direito de acesso.
-  **no_root_squash:** Esta opção, ao contrário da primeira, permite direitos ao superusuário, ou seja, desativa a `root_squash`. Utilizada muito para usuários sem disco.
-  **squash_uids and squash_gids:** Essa opção especifica uma lista de *uids* (ou *gids*) que estão sujeitos a mapeamento anônimo.
-  **insecure:** Esta opção, ao contrário de quando marcado acesso da porta segura, não verifica se a origem tem da requisição parte de uma porta de número baixo.
-  **all_squash:** Esta opção é utilizada para mapear todos *uids* e *gids* para o usuário anônimo.
-  **map_daemon:** Essa opção ativa o mapeamento dinâmico de *uids* e *gids*. Cada *uid* e *gid* será traduzido para seus equivalente no servidor.
-  **map_static:** Essa opção ativa o mapeamento estático de *uid* e *gid*.
-  **map_nis:** Essa opção ativa o mapeamento de *uid* e *gid* baseado em NIS (Network Information Service).

Depois disso, é só iniciar o serviço NFS.

```
cds
./nfs start
iniciando quotas (NFS)           [ OK ]
Iniciando mountd (NFS)          [ OK ]
Iniciando statd (NFS)           [ OK ]
Iniciando nfsd (NFS)            [ OK ]
Iniciando os serviços NFS:      [ OK ]
Configurando o Cliente NFS
```



Para as configurações do cliente, tudo se torna mais fácil ainda, pois, na realidade, não é necessário nenhum software adicional a ser instado para que possamos usar o compartilhamento.

Preparando o Kernel

O que deve ser visto se ocorrer algum erro, é se o Kernel tem suporte a NFS, e para configurar, deverá compilar o Kernel.

Vá ao diretório `/usr/src/linux`.

Se estiver no modo gráfico, digite:

```
make xconfig
```

Caso esteja em modo texto, digite:

```
make menuconfig
```

Na janela que será aberta, procure pelo item *filesystem*, e dentro deste, *Network File System*. O item *NFS filesystem suport* deve estar marcado em *y* (yes) ou *m* (modulo). Caso não esteja, compile o Kernel e dê uma olhada no capítulo de dicas que mostra como fazer isto.



Configurações do Kernel

Utilizando o Linuxconf

Mas se o Kernel já tem suporte ao NFS, que é a maioria dos casos, então iremos montar um diretório como outro qualquer.

Crie um diretório onde estará montado o diretório exportado:

```
mkdir /mnt/dirNFS
```



Na máquina cliente, digite:

```
mount 192.168.200.1:/teste /mnt/dirNFS
```

Agora, é só usá-lo logicamente, dependendo das permissões.

Obviamente, existem formas de configurar no modo gráfico através do Linuxconf, do KDE, etc.

No Linuxconf, selecione *Sistemas de arquivos/Montar volume NFS*. Insira o nome do servidor no campo *servidor*, o diretório exportado no campo *volume* e o diretório onde esse disco deve ser montado na máquina cliente em *ponto de montagem*. É possível, também, configurar uma série de opções de montagem nas abas *Opções* e *Opções NFS*.

Através da aba *Opções*, é possível configurar o diretório de montagem para somente para leitura, ativar cotas de usuário e grupo ou impedir que programas sejam executados a partir dele, entre outras opções. Se você quiser montar esse diretório manualmente através do Linuxconf ative a opção *Não montar na inicialização* (boot), e utilize os botões *Montar* e *Desmontar* para efetuar essas operações.



Montando Volumes NFS

Para tornar mais prático ainda, configure o arquivo `/etc/fstab` para montar automaticamente no inicialização do sistema.












```
192.168.200.1:/teste      /mnt/dirNFS      nfs
defaults                  1 1
```

Com esta linha acima, sempre que a máquina reiniciar, será montado automaticamente em `/mnt/dirNFS` o diretório exportado da máquina `192.168.200.1 /teste` do tipo NFS.






Parâmetros para Montagem de Sistema


Abaixo, temos uma lista das opções mais comuns do mount?

-  **-V:** Imprime a versão.
-  **-h:** Imprime a mensagem de ajuda.
-  **-v:** Mensagens detalhadas.
-  **-a:** Monta todos os sistemas de arquivos (ou aqueles com os tipos mencionados) descritos em *fstab*.
-  **-F** (usado em conjunto com **-a**): Gera uma nova instância do comando *mount* para cada dispositivo. Assim sendo, a montagem em diferentes dispositivos ou diferentes servidores NFS ocorrerá em paralelo. A grande vantagem será a velocidade, porém, os tempos de espera do NFS correrão em paralelo. Uma desvantagem nesta montagem será a sua ordem indefinida. Ou seja, não se pode usar esta opção caso se deseje montar tanto */usr* como */usr/spool*.
-  **-f:** Faz com que tudo seja executado, exceto a montagem efetiva em si, como se fosse um teste. Apesar de não ser tão óbvia, esta opção permite que falsas montagens sejam realizadas e é útil quando, em conjunto com **-v**, pois permite determinar o que o comando *mount* está tentando fazer.
-  **-l:** Adiciona os rótulos *ext2* na saída da montagem.
-  **-n:** Montagem sem gravação de */etc/mtab*. Isso é necessário, por exemplo, quando o sistema de arquivos */etc* está somente com permissões de leitura.
-  **-s:** Tolerar o uso de opções “sujas” de montagem para sistemas de arquivos ao invés de falhar. Esta opção irá ignorar as opções não suportadas pelo tipo do sistema de arquivos. Nem todos os sistemas de arquivos suportam esta opção, a qual é disponibilizada para suportar a montagem automática do **Linux**, baseada em *autofs* (*automounter*).
-  **-r:** Monta o sistema de arquivos somente com permissões de leitura. Um sinônimo é **-o ro**.
-  **-w:** Monta o sistema de arquivos com permissões de leitura e gravação. Este é o padrão e um sinônimo de **-o rw**.














-  **-L nome:** Monta a partição que tem o nome especificado.
-  **-U uid:** Monta a partição que tem o *uid* especificado.
-  **-t vsftype:** O argumento seguinte a *-t*, é usado para indicar o tipo do sistema de arquivos. Os tipos atualmente suportados são listados em *linux/fs/filesystems.c*:

adfs, affs, autofs, coda, coherent, devpts, efs, ext, ext2, hfs, hpfs, iso9660, minix, msdos, ncpfs, nfs, ntfs, proc, qnx4, romfs, smbfs, sysv, udf, ufs, umsdos, vfat, xenix, xiafs.

-  **-o:** As opções são especificadas com um sinalizador *-o*, seguido por uma lista de opções separadas por vírgula. Algumas dessas opções são úteis somente quando aparecem no arquivo */etc/fstab*.

As opções a seguir aplicam-se a qualquer sistema de arquivos que esteja sendo montado:

-  **async:** Todas as operações de E/S no sistema de arquivos devem ser realizadas assincronamente.
-  **atime:** Atualiza a data de acesso ao *inode* do sistema de arquivos para cada acesso que seja realizado. Esta é a opção padrão.
-  **auto:** Pode ser montado com a opção *-a*.
-  **defaults:** Usa as opções padrão: *rw, suid, dev, exec, auto, nouser* e *async*.
-  **dev:** Interpreta dispositivos especiais de blocos ou caracteres no sistema de arquivos.
-  **exec:** Permite a execução de binários.
-  **Noatime:** Não atualiza a data de acesso no *inode* deste sistema de arquivos.
-  **noauto:** O arquivo somente pode ser montado explicitamente (ou seja, a opção *-a* não montará o sistema de arquivos).
-  **nodev:** Dispositivos especiais de blocos ou caracteres não devem ser interpretados no sistema de arquivos.
-  **Noexec:** Não permite a execução de qualquer binário no sistema de arquivos montado.
-  **nosuid:** Não permite o uso dos bits de configuração de identificação de usuário ou de grupo.



nouser: Proíbe que um usuário comum (qualquer um que não seja o superusuário) monte o sistema de arquivos. Este é o padrão.



Remount: Tenta remontar um sistema de arquivos já montado. Isso é comumente usado para mudar indicadores de montagem para sistemas de arquivos, especialmente para tornar sistemas de arquivos montados como para somente leitura e de leitura e escrita.



ro: Monta o sistema de arquivos somente para leitura.



rw: Monta o sistema de arquivos com permissão de leitura e gravação.



suid: Permite o uso dos bits de configuração de identificação do usuário e do grupo.



sync: Todas as operações de E/S do sistema de arquivos devem ser realizadas de modo síncrono.



user: Permite que um usuário normal possa montar o sistema de arquivos. Esta opção tem implicações com as opções *noexec*, *nosuid*, e *nodev* (a menos que seja sobreposta pelas opções subseqüentes, como as da linha de opções: *user*, *exec*, *dev*, *suid*).

Com esta última explanação do *mount*, terminamos o capítulo sobre NFS. Neste capítulo, foi possível explorar o NFS, tal como montar e configurar um servidor para este fim.

Utilizamos um cliente para fazer a conexão para o servidor, e fizemos por último a montagem automática do diretório exportado.

Nosso próximo capítulo será referente ao servidor Samba, para compartilhamento de diretório e recursos, tais como impressora, CD-ROM (que entra como diretório entre o **Linux** e Windows). Ainda abordaremos aspectos do Samba como servidor de Login e wins; veremos muito pouco de *netlogon*, usado por máquinas Microsoft para mapear diretórios como drivers locais, atualizar hora, entre outros serviços.

Servidor Samba: Windows e Linux Juntos






Introdução

O Samba é um conjunto de aplicativos rodando sobre a plataforma **Linux**, que utiliza um protocolo chamado SMB (Server Message Block) nativo do Windows. Ele é utilizado em redes Windows para compartilhar recursos, tais como impressoras, discos.

Com o crescimento do **Linux**, tornou-se necessário redes onde as duas plataformas se comunicassem de modo transparente. Por isso, foi criado o Samba, que é a implementação livre do protocolo SMB, que permite que as estações **Linux** e Windows trabalhem em rede utilizando o protocolo NetBIOS.

Mas, não é só a sobrevivência de duas plataformas distintas na mesma rede entendendo-se. O SMB foi portado para outras plataformas tais como IBM - OS/2, variantes do Unix, Macintosh, Amiga OS, Novell Netware, etc.

Outras vantagens abaixo:

-  Compartilhamento de arquivos tanto do **Linux** no Windows, como do Windows no **Linux**, tratando com permissões de usuários e grupos.
-  Compartilhamento de impressoras. Da mesma forma que o anterior, você pode compartilhar uma impressora que está no **Linux** com um cliente Windows ou o inverso, compartilhando uma impressora que está no Windows com um cliente **Linux**.
-  Autenticação de usuários e implementação de *netlogon*.
-  Fornecimento de conversão de nome, em servidor Win.
-  Administração via WEB



Instalando o Samba

Você pode usar os pacotes que vem no CD de instalação do Conectiva ou fazer download da versão mais nova no próprio site do samba www.samba.org.

Para verificar a existência do mesmo, digite:

```
rpm -qa | grep samba
samba-doc-2.2.1a-1U70_1cl
linuxconf-samba-1.25r3-27U70_1cl
samba-clients-2.2.1a-1U70_1cl
samba-swat-2.2.1a-1U70_1cl
```

Caso não esteja instalado, use o comando `rpm -ivh` para instalar os pacotes acima listados. Automaticamente, ele cria um daemon chamado SMB em `/etc/rc.d/init.d`. Para iniciar, digite:

```
./smb start (No diretório /etc/rc.d/init.d)
```

Ou:

```
service smb start (Em qualquer diretório)
```

Mas creio que primeiro queremos implementá-lo, não é? Vamos começar do mais básico, compartilhando um diretório de nossa máquina, deixando público. O arquivo principal do Samba fica em `etc`:

```
vi /etc/smb.conf
```

Podemos configurá-lo, também, através do Linuxconf. Para iniciar a configuração, inicie o Linuxconf e acesse a opção Configuração/Rede/Tarefas de servidor/Samba -servidor de arquivos/Padrões. Veja a figura a seguir:



Padrões do Samba



A aba *Configuração base* deve ser preenchida com o grupo de trabalho das estações Windows. No nosso caso, vamos chamar o grupo de trabalho ou domínio de *Dominiolinux*.

Configure na aba *Senhas* se o Samba deve utilizar as senhas criptografadas (recurso adotado como padrão pelo Windows 98 e superiores).

O Windows normalmente utiliza o método de segurança por recurso, você deve alterar o comportamento padrão do Samba, que é o de fazer a autenticação por usuário. Para fazer isso, apenas altere a opção *Modo de Autenticação* para *Compartilhar*. É possível utilizar um servidor Windows para fazer a autenticação, inserindo o seu nome NetBIOS no campo servidor de senha.

Além da autenticação do usuário, é possível restringir ou permitir o acesso ao servidor Samba a partir de determinadas máquinas, listando-as nos campos *Permite máquinas* e *Proíbe máquinas* na aba acesso. Você poderá especificar as máquinas utilizando seus nomes, seus números IP ou o endereço da rede mais uma máscara.

O Samba pode exportar qualquer diretório escolhido pelo administrador, além do mais ele provê um meio bastante simples de exportar os diretórios home dos usuários do sistema. Veremos a seguir, como exportar os diretórios home dos usuários e, em seguida, como exportar um diretório qualquer do sistema.

Para exportar os diretórios *home*, acesse a opção *Configuração -/ Rede/Tarefas de servidor/Samba - servidor de arquivos/Opções padrão do diretório home de usuários*; insira uma descrição desse serviço no campo *comentário/descrição* e selecione a opção *Este serviço está ativo para permitir que o Samba exporte os diretórios home*. Você poderá selecionar a opção navegável para fazer com que esses diretórios sejam visíveis pelas máquinas Windows.

Selecione na aba *Acesso à opção Gravável* para permitir que os usuários possam escrever arquivos em seus diretórios, caso contrário, o Samba exportará o diretório com permissão de leitura apenas.

A opção *Acesso público* permite que os diretórios sejam acessados sem senha, mas com as permissões da conta convidado.

Na aba *Usuários*, você poderá definir quais usuários poderão utilizar esse serviço (deixe o campo em branco para permitir que todos os usuários possam utilizar o serviço). Adicionalmente, é possível configurar algumas permissões do serviço baseadas no usuário que o acessa, como por exemplo, os usuários com direito de escrita.



Para exportar um diretório qualquer, utilize a opção *Configuração/Rede/Tarefas de servidor/Samba - servidor de arquivos/ Compartilhamento de disco* e clique no botão *Adicionar*. Na tela que surgirá, preencha o nome que se deseja dar a esse diretório compartilhado, bem como um comentário sobre ele, utilizando os campos *Nome do compartilhamento* e *Comentário/descrição*.

Para compartilhar as impressoras do sistema, acesse a opção *Configuração/Rede/Tarefas de servidor/Samba -servidor de arquivos/Opções padrão para impressoras*. Insira no campo *Comentário/descrição* uma breve descrição do serviço.

Senhas Descriptografadas no Windows 95

O padrão do Windows 95 é não criptografar as senhas. Assim, é provável que esta configuração não seja necessária para você. Porém, versões mais recentes do Windows 95 (como o Windows 95 OSR2) mudaram seu comportamento. Se você tiver problemas com senhas, siga o procedimento abaixo:

A configuração do Windows 95 pode ser feita de duas maneiras. Uma delas seria a utilização do arquivo *Win95_PlainPassword.reg*, que acompanha o Samba. Este arquivo pode ser localizado no diretório *docs/* da documentação on-line do Samba:

```
# cd /usr/share/doc/samba*/docs
```

Você pode copiá-lo para as estações Windows® 95. Depois, basta abrir o arquivo (dando dois cliques sobre ele) para que seja instalado.

Outra maneira de habilitar senhas é editar o registro através do utilitário *regedit* do Windows® para alterar ou incluir a chave:

```
[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\VNETSUP]
"EnablePlainTextPassword"=dword:00000001
```

Com esta opção, as senha que estarão trafegando serão criptografadas.

Senhas Descriptografadas no Windows 98

Como padrão, o Windows criptografa todas as senhas que trafegam pela rede. Para poder utilizar estações, o Windows com sua rede, siga o procedimento descrito nesta seção.



A configuração de senhas não criptografadas no Windows® 98 pode ser feita de duas maneiras. Uma delas seria a utilização do arquivo *Win98_PlainPassword.reg*, que acompanha o Samba.

Este arquivo pode ser localizado no diretório *docs/* da documentação on-line do Samba:

```
# cd /usr/share/doc/samba*/docs
```

Você pode copiá-lo para as estações Windows® 98. Depois, basta abrir o arquivo (dar dois cliques sobre ele) para que seja instalado.

Outra maneira de habilitar senhas é editar o registro através do utilitário *regedit* do Windows® para alterar ou incluir a chave:

```
[HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\VxD\NETSUP]
```

```
"EnablePlainTextPassword"=dword:00000001
```

Senhas Descriptografadas no Windows NT

O Windows NT criptografa senhas em algumas versões e não criptografa em outras. Constatou-se que, após o terceiro pacote de consertos (Service Pack 3 ou SP3), ele passou a criptografar todas as senhas trafegadas pela rede.

Assim, recomenda-se seguir os procedimentos descritos nesta seção para assegurar que ele funcione como cliente de uma rede Samba.

Além disso, recomendamos que você tenha, pelo menos, o Windows NT SP3 instalado.

A configuração de senhas não criptografadas no Windows NT pode ser feita de duas maneiras. Uma delas seria a utilização do arquivo *NT4_PlainPassword.reg*, que acompanha o Samba.

Este arquivo pode ser localizado no diretório *docs/* da documentação on-line do Samba:

```
# cd /usr/share/doc/samba*/docs
```

Você pode copiá-lo para as estações Windows NT. Depois, basta abrir o arquivo (dando dois cliques sobre ele) para que seja instalado.

Outra maneira de habilitar senhas é editando o registro através do utilitário *regedit*, do Windows, para alterar ou incluir a chave:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Rdr\Parameters]
```

```
"EnablePlainTextPassword"=dword:00000001
```



Senhas Descriptografadas no Windows 2000

O Windows 2000 criptografa todas as senhas que trafegam pela rede; para poder utilizar estações Windows 2000 com sua rede, siga o procedimento descrito nesta seção.

A configuração de senhas não criptografadas no Windows 2000 também pode ser feita de duas maneiras. Uma delas é utilizar o arquivo *Win2000_PlainPassword.reg*, que acompanha o Samba. Este arquivo pode ser localizado no diretório *docs/* da documentação on-line do Samba:

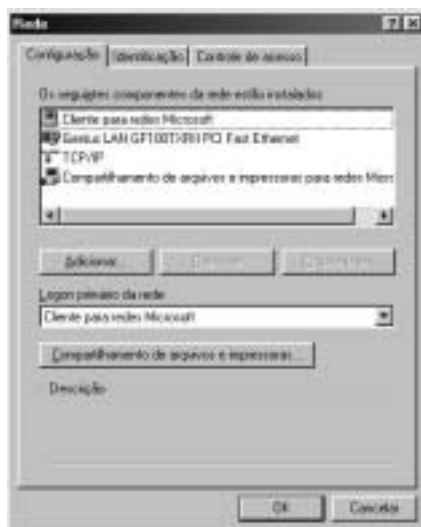
```
# cd /usr/share/doc/samba*/docs
```

Você pode copiá-lo para as estações Windows® 2000. Depois, basta executar o arquivo dando dois cliques para que seja instalado.

Outra maneira de habilitar senhas, é editar o registro através do utilitário *regedit* do Windows para alterar ou incluir a chave:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\  
LanmanWorkStation\Parameters]  
"EnablePlainTextPassword"=dword:00000001
```

A primeira parte está pronta, só precisamos testar para verificarmos se todas as configurações estão OK. Neste momento, estaremos configurando o cliente Windows. Para começarmos a configurar o cliente Windows, vá em *Painel de Controle/Redes*. Deverá aparecer as propriedades da rede como a imagem abaixo:



Propriedades da Rede Windows



Clique na pasta *Identificação*, na tela de configuração de redes e digite o nome de sua máquina e o nome do grupo de trabalho. Em nosso exemplo, a máquina *DEZAO* fará parte do grupo de trabalho *Dominiolinux*.

Veja a próxima figura para conferir suas configurações.



Configurações do Grupo de Trabalho

Clique na pasta *Controle de Acesso* e verifique se a tela está configurada como na figura seguinte:



Controle de acesso no Windows



É interessante colocar o endereçamento IP do servidor da propriedade do TCP/IP na guia configuração do DNS.

Boot a máquina e o procedimento está pronto.

Mas é melhor testarmos através do nosso servidor, existe um comando *testparm*, ele verifica se existe erros em nosso arquivo *smb.conf*.

Vamos nos aprofundar mais um pouco neste arquivo. Primeiramente, vamos criar um diretório público onde todos terão acesso:

```
mkdir /publico
```

Agora, editemos o arquivo */etc/smb.conf*.

Existem duas formas básicas que permitem compartilhar arquivos do **Linux** para rede Windows.

A primeira delas faz com que qualquer pessoa tenha acesso ao diretório compartilhado, sem que seja necessário entrar com uma senha. Já a outra, exige que o usuário entre com um nome de usuário com um password.

Para entender melhor como o compartilhamento funciona, crie dois diretórios. Vamos usar dois diretórios, o público e o comum. Para fazer isso, use o comando:

```
mkdir
```

Neste caso, é altamente recomendável que você guarde uma cópia do arquivo *smb.conf*. Isto irá garantir que você possa recuperá-lo se, por acaso, alterar alguma configuração por engano. Use o seguinte comando para fazer a cópia:

```
cp smb.conf smb.conf.cópia
```

Vejamos o arquivo propriamente dito:

```
#Global parameters
[global]
#nome do seu grupo de trabalho
workgroup = Dominiolinux
server string = servidor Samba
security = SHARE
log file = /var/log/samba/log.%m
max log size = 50
printcap name = /etc/printcap
dns proxy = No
guest account = guest
hosts allow = 192.168.200.192.168.254.127.
```




```
[Printers]
comment = Impressora do Linux
security = serve
path = /var/spool/lpd/lp
browseable = no
printable = yes
public = yes
writable = no
```

```
[homes]
comment = Home Directories
read only = No
browseable = No
writable = yes
```

```
[public]
comment = Public Stuff
path = public
write list = @staff
read only = No
guest ok = yes
```


```
[comum]
comment = Pasta comum
path = /comum
read only = No
guest ok = yes
write list = @staff
```


Vejamos, agora, o que significa cada parâmetro do arquivo:


 **Seção [global]:** Define as configurações globais do Samba.

A relação abaixo apresenta a explicação dos parâmetros do exemplo:

 **comment:** Comentário para este Host na Rede.

 **Workgroup:** Especifica o Domínio ou Workgroup a que o host pertence na rede.

 **security:** Por padrão, o Samba utiliza a segurança a nível de usuário (security = user) com as opções:

 **security = share:** Senhas de acesso serão solicitadas por cada recurso compartilhado e não por usuário, ou seja, cada diretório ou impressora poderá ter uma senha



única conhecida pelos usuários autorizados. Esta opção é geralmente usada para estações de trabalho **Linux**, onde em diversos casos simplifica o acesso a dados locais quando necessário.



security = user: As permissões são dadas de acordo com o login do usuário ou através dos grupos (@grupo).



security = server: O Samba tentará validar a senha do usuário, enviando os dados para outro servidor SMB, como outro servidor Samba ou um servidor Windows. Deve-se incluir o parâmetro *.password server = x.x.x.x* na seção *[global]* do *smb.conf*.



security = domain: Usado se o host for adicionado a um Domínio Windows através do comando *smbpasswd*. Neste caso, as informações de usuário e senha serão enviadas para o PDC da rede, exatamente como o servidor NT faria. Note que é necessário que a conta do usuário exista tanto no **Linux** quanto no servidor primário (mais adiante isso será explicado de forma detalhada).



os level: Este parâmetro não é obrigatório se você não possui um servidor **Linux** ou Windows na rede, mas deve ser usado caso tenha um ou mais. A variável é um número de 1 a 255, onde 65 é a mesma variável utilizada pelo servidor Windows. Especifique um número maior que este (como 100, por exemplo) para garantir que o servidor Samba seja eleito na escolha de validação do login das estações.



announce as: Permite especificar o tipo de servidor NetBIOS (nmbd) que será divulgado na rede. As opções aceitas pelo Samba: “NT Server”, “NT Workstation”, “Win95” ou “WfW”.



domain logons: Usado para validar o login na rede, apenas para estações Windows.










logon script: Indica qual arquivo de logon script será executado para os usuários. A variável *%u* corresponde ao usuário na rede. Deve, também, ser criado um compartilhamento de nome *[netlogon]*, apontando para o diretório dos scripts.



logon path: Indica o caminho do perfil remoto do usuário. A variável *%L* corresponde ao nome do servidor



NetBIOS (que pode ser o próprio Samba). O *logon path* é útil quando os usuários costumam efetuar logon em mais de um host na rede, pois seu perfil é trazido com o logon. No caso do exemplo, o diretório “Profiles” deve conter os scripts (em formato Microsoft, usando NET USE, etc) e os scripts devem ser criados com o notepad do Windows, por exemplo, a fim de conservar o formato do arquivo.

-  **domain master:** Indica se o host será o Domain Master Browser da rede inteira (WAN).
-  **local master:** Indica se o host será o Master Browser da rede local.
-  **preferred master:** Este parâmetro força a eleição do Samba como Master Browser para o workgroup. É recomendável utilizar este parâmetro em conjunto com o “domain master = yes” para garantir a eleição. Mas tome cuidado: se você possui uma rede com servidores Windows e Samba e já possui um servidor como Domain Master, não use esta opção e deixe o parâmetro “os level = 65” para haver equilíbrio.
-  **guest account:** O Samba trabalha melhor em redes Microsoft com a existência de uma conta *guest* (visitante em inglês). Por padrão, a conta usada é nobody (a mesma utilizada pelo Apache).
-  **wins server:** Indica qual o servidor de Wins da rede. Se o próprio host for o servidor de Wins, não utilize este parâmetro, pois haverá um loop e o sistema travará!
-  **wins support:** Permite ao Samba ser o servidor de Wins na rede. Isto significa que o Samba terá uma tabela com o ambiente completo da rede, garantindo que as estações tenham acesso a estas informações e ganho em velocidade para encontrar e acessar os compartilhamentos e impressoras. O Wins Server deve ser especificado na configuração de rede (TCP/IP) das estações, indicando o endereço IP do servidor.
-  **keep alive:** Como máquinas rodando Windows tendem a travar com o passar do tempo, este parâmetro é usado para verificar o estado da conexão, evitando tráfego desnecessário na rede. Também pode ser usado para estações **Linux**.



debug level: Parâmetro usado para dar flexibilidade à configuração do sistema. Permite ao Samba trabalhar corretamente com algumas situações de erro, por exemplo.



winpopup command: Especifica qual comando será executado quando o servidor receber mensagens Winpopup. Aqui, muitas opções podem ser usadas de acordo com a preferência do Administrador. Se sua rede utiliza mensagens deste tipo, é interessante definir um comando para o parâmetro, evitando, assim, possíveis mensagens de erro para quem enviou a mensagem ao servidor.



log file: Indica o arquivo de log do Samba. A variável %u corresponde ao nome de logon do usuário. O samba, por padrão, gera arquivos de log em `/var/log/samba` que indicam, por exemplo, os horários de logon dos usuários, quem acessou determinado arquivo, etc. Esteja atento para estas informações para consultas quando necessário.



null passwords: Indica se será ou não possível que usuários tenham senha nula de logon (logon sem senha).



unix password sync: Se este parâmetro for ativado (= yes), então clientes SMB (como estações Windows) poderão trocar sua senha de login.



socket options: Este parâmetro permite configurações extras para o protocolo, possibilitando uma melhor performance do servidor em lidar com os pacotes na rede.



printing: Indica qual o sistema de impressão padrão utilizado pelo **Linux**.



printcap name: Indica o arquivo para busca das definições das impressoras.









load printers: Disponibiliza as impressoras para a rede.









hosts allow: Indica quais máquinas tem acesso ao servidor Samba. Pode-se utilizar o endereço IP ou o nome da máquina. Para garantir acesso a toda uma rede, por exemplo, escreva: “hosts allow = 192.168.1”. Este parâmetro pode ser usado na seção Global, mas, preferencialmente, nas demais seções.



-  **hosts deny:** Como em “hosts allow”, mas para restringir o acesso ao servidor Samba.
-  **Seção [homes]:** Define os parâmetros para as pastas pessoais dos usuários na rede (*home dir*):
-  **comment:** Comentário para este compartilhamento.
-  **public:** Também conhecido como “guest ok”, permite ou não acesso de outros usuários.
-  **browseable:** Define se o compartilhamento será ou não visível para o Ambiente de Rede.
-  **writable:** Indica se o usuário poderá ou não escrever em sua pasta pessoal (*home dir*).

Demais Seções [shares]

Correspondem aos compartilhamentos presentes na rede. Os parâmetros abaixo são apenas alguns dos possíveis que podem ser utilizados:

-  **comment:** Comentário para o compartilhamento.
-  **path:** Caminho do diretório compartilhado.
-  **valid users:** Este parâmetro é usado para destacar quem terá acesso ao compartilhamento na rede. É importante destacar que estações Win95/98/ME têm diferenças entre si, que em muitas situações representam um problema para acesso e segurança. Acontece algumas vezes de você definir o “write list” e o “read list” corretamente, mas mesmo assim, usuários do “read list” conseguem escrever no compartilhamento (!). Para resolver este problema, inclua o “valid users”, indicando os usuários que têm acesso e, em seguida, inclua o “write list” e o “read list” conforme sua necessidade.
-  **writable:** Indica se será ou não possível criar ou excluir arquivos ou diretórios do compartilhamento.
-  **public/guest ok:** Indica se será ou não permitido o acesso de outros usuários.
-  **browseable:** Define se o compartilhamento será ou não visível para o Ambiente de Rede do Windows (apresentado na rede).



write list: Define os usuários e/ou grupos com acesso de escrita no compartilhamento. Para mais de um usuário, separe os nomes por vírgula (*user1, user2*, etc.) e para grupos, utilize *@* antes do nome do grupo.



read list: Como em *.write list*., mas define quem terá permissão de apenas leitura.



force create mode: Diz ao Samba para forçar o tipo de permissão dos arquivos criados (o mesmo que usar o *chmod*). Esta permissão tem menor prioridade que os parâmetros *write list* e *read list*.



force directory mode: O mesmo que *force create mode*, mas para os diretórios criados no compartilhamento.



admin users: Indica quais são os usuários com permissão completa para o compartilhamento (permissão de root).



copy: Permite copiar os parâmetros de outra seção, como um template, por exemplo. Para alterar parâmetros, basta informá-los na seção atual.



hosts allow: Indica quais máquinas podem acessar o compartilhamento. Pode-se utilizar o endereço IP ou o nome da máquina. Para garantir acesso a toda uma rede classe C, por exemplo, escreva: “hosts allow = 192.168.1”.



hosts deny: Como em “hosts allow”, mas para restringir o acesso ao compartilhamento.



max connections: Permite especificar o número máximo de conexões simultâneas ao compartilhamento.

Depois de explicados os itens, vamos analisar o *smb.conf* de exemplo que foi dado anteriormente:



[global]: Em global, veremos as configurações globais do servidor.



workgroup = Dominiolinux: Este é o nome do nosso grupo de trabalho, posteriormente, será o nosso domínio para login.









server string = servidor Samba: Em *server string*, colocamos o que será o nome de nosso servidor.



security = SHARE: Como visto anteriormente, este é um tipo de configuração de segurança para compartilhamento.





-  **log file = /var/log/samba/log.%m:** Onde será gravado nosso log.
-  **max log size = 50:** Tamanho máximo do nosso log.
-  **printcap name = /etc/printcap:** Carrega a configuração das impressoras que estão instaladas no servidor Samba.
-  **dns proxy = No:** Diz ao Samba se é ou não para tentar resolver nomes NetBIOS através do *nslookup* do DNS.
-  **guest account = guest:** Transforma todos os usuários que de algum modo foram indicados como visitantes em usuário *guest*.
-  **hosts allow = 192.168.200.192.168.254.127:** Quais hosts estão permitidos para acessar o servidor.





Esta é a primeira parte referente ao global, agora veremos as outras linhas:

-  **[Printers]:** O nome deste compartilhamento, alias e este nome que aparecerá compartilhado para o usuário.

Para evitar de ficarmos repetindo as mesmas configurações, não me atarei a detalhes das que já foram vistas anteriormente.

-  **comment = Impressora do Linux**
-  **security = serve:** Segurança remota por usuário. O Samba pega o nome do usuário e a senha, autenticando junto a outro servidor, que poderá ser outro **Linux** rodando Samba ou um Windows NT. Apesar da autenticação ser remota, ainda é necessário criar os usuários Unix localmente em determinados casos.

path = /var/spool/lpd/lp

-  **browseable = no:** Define se o compartilhamento será ou não visível para o Ambiente de Rede do Windows.
-  **printable = yes:** Permitirá a gravação através de operações de geração de arquivos temporários de impressão.
-  **public = yes:** Também conhecido como “guest ok”, permite ou não acesso de outros usuários sem senha.
-  **writable = no:** Permite que não se escreva no diretório.

[homes]

comment = Home Directories



```
read only = No - Somente leitura ou não
browseable = No
writable = yes
[public]
comment = Public Stuff
path = public
```



write list = @staff: Permissão de gravação somente para quem pertencer ao grupo suporte *staff*.

```
read only = No
guest ok = yes
[comum]
comment = Pasta comum
path = /comum
read only = No
guest ok = yes
write list = @staff
```

Na sessão global, temos um padrão muito utilizado no Samba. A segurança é feita por compartilhamentos. Não é dos melhores, mas funciona. Temos uma conta *guest* que pode estar acessando nosso servidor.

Em printers, temos outro tipo de autenticação que, na realidade, envia para outro servidor validar, este outro servidor pode ser um NT ou o próprio Samba em outra máquina. Um detalhe é que ela não será visível no Ambiente de Rede do Windows.

O compartilhamento *home* é referente ao home do usuário. Veremos logo após este item como criar usuários no Samba.

Já no *public*, temos outro item interessante, que é o caso do *write list*, onde só quem é do grupo poderá escrever neste diretório.

Sempre que usarmos o item **security = user**, o usuário terá que fornecer nome de usuário e senha.

Vamos fazer uma configuração que faça com que o cliente Windows largue em uma Rede NT, no caso, o nosso próprio servidor, e em seguida mapeia algumas unidades. A primeira coisa a fazermos é criar os usuários e os grupos. Criaremos dois grupos e dois usuários:

```
adduser -g adm andre
adduser -g users stato
```

Para adicionar usuários já existentes no grupo, vá ao arquivo */etc/group* para verificar qual os números ID dos grupos:

```
adm:x:505:
users:x:506:
```



Vamos supor que tenhamos um usuário chamado *joao* que deverá fazer parte do grupo *adm*. Vejamos o arquivo */etc/passwd*:

```
andre:x:508:505::/home/carlos:/bin/bash
stato:x:507:506::/home/junior:/bin/bash
joao:x:502:502::/home/thais:/bin/bash
```

O que faremos é adicionar mais uma linha:

```
joao:x:502:505::/home/joao:/bin/bash
```

Lembre-se de apenas adicionar, pois caso contrário trocará pelo grupo que o usuário já possui.

Agora, temos três usuários, *andre* e *joao*, que fazem parte do grupo *adm*, e *joao* que faz parte do *users*.

Precisamos vincular os usuários do **Linux** com os do Samba, para isso existe o comando *smbadduser*.

Faça da seguinte forma:

```
smbadduser joao:joao
smbadduser stato:stato
smbadduser andre:andre
```

Vamos, agora, criar alguns diretórios:

```
mkdir /adm
mkdir /usuarios
mkdir /home/netlogon
chmod -R 777 /adm
chmod -R 777 /usuarios
```

Precisamos alterar o arquivo de configuração do Samba, o *smb.conf*.

Seção do arquivo que seta as configurações que serão utilizadas por todos:

```
# os usuários
[global]

workgroup = Dominiolinux      #grupo de trabalho
netbios name = Linux          #nome da máquina que
aparecerá nas máquinas Windows

server string = Samba Server printcap      # carrega as
características das impressoras disponíveis

name = /etc/printcap

load printers = yes          # carregar impressoras
printing = lprng             #sistema de impressão
```



```
log file = /var/log/samba/log.%m      #logs de acesso
max log size = 50                     #tamanho máximo do logdebug
level = 2                             #nível do log a ser feito
security = user                       #tipo de acesso
password level = 8                    # tamanho máximo para a
                                     senha
username level = 8                    # tamanho máximo para o
                                     usuário
encrypt passwords = yes               #utilização de senhas
criptografadas (win95B/win98)
smb passwd file = /etc/smbpasswd
socket options = TCP_NODELAY SO_RCVBUF=8192
SO_SNDBUF=8192
local master = yes                   # Servidor master
os level = 34                         # Define quem é o servidor
master da rede inteira
domain master = yes                  # Indica se ele será o
servidor master de domínio da rede inteira
preferred master = yes               # Força a eleição do
servidor samba como master
domain logons = yes                  #Efetua o logon de máquinas
win95/win98
logon script = %U.bat                #carrega os compartilhamento
de cada usuário
dns proxy = no
# Diretório particular de cada usuário
[homes]
comment = Home Directories
browseable = no
writable = yes
# Local onde ficarão os scripts dos usuários
[netlogon]
comment = Network Logon Service
path = /home/netlogon
guest ok = yes
writable = no
share modes = no
browseable = no
```




```
#compartilhamento de impressoras
[printers]
comment = All Printers
path = /var/spool/samba
browseable = no
# Set public = yes to allow user 'guest account' to
print
guest ok = no
writable = no
printable = yes
# Exemplos de compartilhamento
#Neste compartilhamento todos possuem acesso de cópia e
escrita
[publico]
path = /publico          #caminho do diretório na
    máquina linux
public = yes             #Todos podem visualizar este
diretório
only guest = yes
writable = yes           #permite que todos possam gravar
neste diretório
printable = no
# Só poderá utilizar este compartilhamento que tiver ele
em seu script
[Administracao]
path = /adm
only guest = yes
writable = yes
printable = no
write list = @adm
# Só poderá gravar neste compartilhamento quem
pertencer ao grupo adm os outros usuários só poderão ver
e copiar os arquivos.
[usuarios]
path = /usuarios
only guest = yes
writable = yes
```



```
printable = no
write list = @users #permissão de gravação somente para
quem pertencer ao grupo users
# Faz o compartilhamento do cd-rom para que os usuários
possam acessar
[cdrom]
path = /mnt/cdrom
public = yes
only guest = yes
writable = no
printable = no
```

Agora, precisamos do script que fará os compartilhamentos nas máquinas Windows.

Abaixo, segue um exemplo de script que faz o mapeamento dos compartilhamentos dos diretórios criados no **Linux** transformado em unidades de rede no Windows e altera o horário da máquina Windows, deixando com o mesmo horário da máquina Samba. Abaixo, um script criado no Windows (Notepad) por questões de formatação. Este script pode ser usado pelos usuários *andre* e *joao*.

```
rem Logon script padrão para a rede.
net time \\ /set /yes
@echo off
if %OS%.==Windows_NT. goto WinNT
:Win95
net use X: /HOME
net use Y: \\Llinux\adm
net use Z: \\Linux\cdrom
goto end
:WinNT
net use X: \\Linux\adm /persistent:no
net use Y: \\Linux\adm /persistent:no
net use Z: /HOME /persistent:no
:end
```

Salve-o como *nomeusuario.bat*, por exemplo, *andre.bat* e *joao.bat*. Coloque no diretório do servidor Samba em */home/netlogon*.

Não confunda, o arquivo foi feito em uma máquina Windows, mas deverá ser gravado no servidor **Linux**.

Para o *stato*, podemos criar o seguinte script:

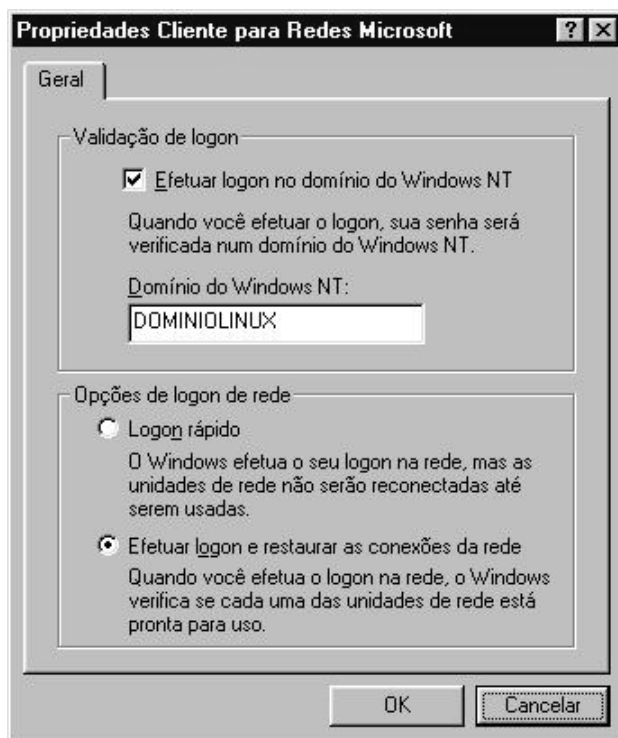
```
rem Logon script padrão para a rede.
net time \\ /set /yes
@echo off
```



```
if %OS%.==Windows_NT. goto WinNT
:Win95
net use X: /HOME
net use Y: \\Llinux\usuarios
net use Z: \\Linux\cdrom
goto end
:WinNT
net use X: \\Linux\usuarios /persistent:no
net use Y: \\Linux\cdrom /persistent:no
net use Z: /HOME /persistent:no
:end
```

Este último deverá ser salvo como *stato.bat*, também deverá estar em */home/netlogon*.

Já no Windows, precisamos configurar alguns itens ainda: Vá em *Painel de Controle/Confederações/Redes* e selecione *Clientes para rede Microsoft*. Em seguida, clique em propriedades, veja a figura abaixo:



Configurações de Domínio

Marque o item *Efetuar logon* no domínio do Windows NT. E no domínio Windows NT, coloque o nome do Workgroup ou Domínio Configurado no Servidor Samba, no arquivo *smb.conf*.



Agora, é só reiniciar a máquina Windows. Será solicitado nome de usuário, domínio e senha. No *nome de usuário* coloque *stato*, *andre* ou *joao*, que são os usuários criados. O domínio será conforme nosso exemplo Dominiolinux e a senha será aquela que foi informada quando criado o usuário com o comando *smbadduser*.

Logicamente, o servidor Samba deve estar funcionando. Verifique:

```
service smb status
```

Após logar o script, como o *nome.bat*, a hora será atualizada conforme o servidor e serão mapeados os compartilhamentos que foram colocados no script.










Acredito que esta é uma das melhores maneiras de compartilhar diretórios para o Windows usando o Samba, pois assim, se for informado nome de usuário ou senha, nada será compartilhado, já que o servidor está trabalhando no modo *user*, e não *share*.

O que quer dizer? Que para qualquer diretório do Samba, o usuário deve existir. Caso não esteja logado não terá acesso ao servidor.









Se você notar, em nosso script existe uma variável nas linhas:

```
log file = /var/log/samba/log.%m  
logon script = %u.bat
```

Estas são variáveis que o Samba permite. A primeira *%m* será substituída pelo nome NetBIOS da máquina cliente e *%U* substitui pelo nome de usuário da sessão. Abaixo, temos uma lista mais completa das variáveis permitidas pelo Samba:

-  **%S:** Nome do Serviço (compartilhamento) atual.
-  **%P:** Diretório raiz root.
-  **%u:** Nome do usuário.
-  **%g:** Nome do grupo.
-  **%H:** Nome do diretório pessoal do usuário (home dir).
-  **%G:** Nome primário do grupo.
-  **%v:** Versão do Samba que está sendo executada.
-  **%h:** Nome do host Internet em que o Samba está sendo executado.
-  **%m:** Nome da máquina cliente fornecido pelo NetBIOS.



-  **%L:** Nome do servidor NetBIOS, permitindo que a configuração desejada seja alterada de acordo com o cliente que vai acessar o sistema.
-  **%N:** Nome do servidor do diretório inicial do NIS.
-  **%p:** Caminho do diretório inicial do serviço NIS.
-  **%M:** Nome Internet da máquina cliente.
-  **%d:** ID do processo atual do servidor.
-  **%a:** Sistema operacional da máquina remota, onde os reconhecidos são WfW, Win95, Win2000.
-  **%I:** O endereço IP da máquina cliente.
-  **%T:** Data e horário atuais.

Falamos bastante do compartilhamento Windows para **Linux**. O que não falamos foi do **Linux** para Windows e do **Linux** para o **Linux**, que serão nossos próximos pontos.

Em um primeiro passo, vamos falar do **Linux** para o Windows. Suponhamos que você queira montar um diretório que se encontra em uma máquina Windows e, ainda, acessar o diretório C:, que foi compartilhado com o nome C.

O processo deve ser feito através da linha de comando. A máquina Windows chama-se *estacao1* e você montará tal partição em */mnt/WinC*:

```
smbmount //estacao1/c /mnt/WinC
```

Existem dois tipos de compartilhamento do Windows do tipo *fat*, como Windows 95, 98 e ME.

No compartilhamento por recurso, é dada uma senha por diretório. O compartilhamento por usuário, se torna mais interessante, pois ele trará os usuários cadastrados no Samba.

No primeiro caso, após digitar o comando *smbmount*, será solicitada uma senha para o recurso compartilhado.

No segundo caso, através da linha de comando, deve-se informar o nome de usuário e senha. Veja o exemplo abaixo com o mesmo compartilhamento acima:

```
smbmount //estacao1/c /mnt/WinC -o  
username=carlos,password=senha
```

Com isso, o drive C será montado no diretório */mnt/WinC*.



Pode haver algum tipo de problema quanto à montagem quando não houver na rede um servidor DNS, pois o Samba pode não resolver o IP da estação. Neste caso, deve-se adicionar no arquivo */etc/hosts* o endereço conforme abaixo:

```
192.168.200.3  estacao.dominiolinux    estacao
```

Com isso, já será possível montar sem nenhum problema.

Observe que em alguns casos (você está montando um volume de um servidor Windows NT, por exemplo) será necessário que você também informe um usuário e uma senha para poder utilizar o comando *smbmount*. Isso funcionará se o usuário estiver cadastrado na máquina *estacao1* ou se o compartilhamento estiver usado os usuários do domínio no servidor Samba e não Local.

Agora, no nosso segundo caso, **Linux** para **Linux**, existe um servidor Samba compartilhado em alguns diretórios.

O programa usado para se utilizar dos recursos do Samba chama-se *smbclient*, similar ao FTP, ou seja, ele abre um prompt onde serão utilizados comandos, como o FTP. Em contrapartida, ele não possui recursos para montagem em diretório local.

Vamos para exemplo:

```
Servidor: Server
Pasta: Diretório
Cliente Linux: estacao1
smbclient //Server/Diretorio
INFO: Debug class all level = 2    (pid 1928 from pid
1928)
added interface ip=192.168.200.1 bcast=192.168.200.255
nmask=255.255.255.0
Password:
```

















Veja que foi solicitado uma senha, este tipo de conexão também só funcionará se o tipo de segurança do servidor Samba for do tipo Share, ou compartilhamento. No nosso segundo caso, no qual montamos um servidor como um Domínio NT, é necessário informar também o nome de usuário.

```
smbclient //Server/Diretorio -U userINFO: Debug class
all level = 2    (pid 1933 from pid 1933)
added interface ip=192.168.200.1 bcast=192.168.200.255
nmask=255.255.255.0
Password:
Domain=[Server] OS=[Unix] Server=[Samba 2.2.1a]
smb: \>
```
















Veja que, no exemplo, temos um prompt onde serão digitados comando como *ls*, *cd*, *etc*.





Abaixo, temos uma lista dos comandos possíveis:

-  **? [comando]:** Sozinho, traz uma lista dos comandos disponíveis; junto a um comando, traz uma breve descrição do comando.
-  **! [comando do shell]:** Se especificado, um Shell é executado localmente com o comando especificado do Shell.
-  **cd [nome do diretório]:** Altera para o diretório especificado.
-  **del <mask>:** Uma solicitação é feita para o servidor excluir a partir do diretório de trabalho atual todos os arquivos que correspondem com o mask.
-  **dir <mask>:** Utilizado para mostrar um lista de arquivos que corresponde com mask no diretório de trabalho.
-  **exit:** Termina a conexão com o servidor e fecha.
-  **get <arquivo remoto> [arquivo local]:** Copia o arquivo remoto do servidor para a máquina local.
-  **help [comando]:** Igual ao “?”.
-  **lcd [diretório]:** O diretório atual da máquina local é alterado para o diretório específico.
-  **lowercase:** Alterna as letras minúsculas do nomes de arquivos. Utilizado com o *get* e *mget*.
-  **ls <mask>:** Refere-se ao comando *dir* nesta tabela.
-  **mask <mask>:** Permite ao usuário configurar uma máscara a ser utilizada durante a operação recursiva dos comandos *mget* e *mput*.
-  **md <diretório>:** Veja o comando *mkdir*.
-  **mget <mask>:** Copia arquivos que correspondem à mask do servidor para máquina local.
-  **mkdir <diretório>:** Cria um novo diretório no servidor.
-  **mput <mask>:** Copia arquivos que correspondem às masks do servidor para máquina local.



















-  **print <arquivo>:** Imprime o arquivo especificado da máquina local para um serviço de impressão no servidor.
-  **printmode <graphics or text>:** Configura o modo de impressão para binários ou texto.
-  **prompt:** Alterna a solicitação aos nomes de arquivo para os comandos mget e mput.
-  **put <Arquivo Local> [Arquivo Remoto]:** Copia o arquivo local do cliente para o Servidor com o nome arquivo remoto
-  **queue:** Exibe lista de impressão.
-  **quit:** Igual ao comando exit.
-  **rd <diretório>:** Igual ao rmdir.
-  **recurse:** Alterna a solicitação de recursão do diretório para os comandos mget e mput.
-  **rm <mask>:** Remove do servidor arquivos que correspondem com mask
-  **rmdir <diretório>:** Remove o diretório especificado do Servidor
-  **tar <c | x>[IXbgNa]:** Realiza operação com o tar.
-  **blocksize <blocksize>**
-  **setmode <filename> <perm=[+ | \-]rsha>:** Similar ao *attrib* do DOS, serve para configurações de permissões de arquivos.





Este são os comandos que você pode usar dentro do prompt do Samba, mas existem os comandos que podem ser usados com o próprio *smbclient*, como o *-U*, para informar o nome do usuário ao servidor, por exemplo. Abaixo, a relação dos parâmetros que podem ser usados com o comando:

-  **servicename:** O nome do serviço que você quer utilizar no servidor. Por exemplo *//Server/Diretorio*.
-  **password:** A senha requerida para acessar o serviço no servidor.
-  **-s smb.conf:** Especifica o local do arquivo *smb.conf*.
-  **-O opções de soquete:** Utilizado para configurar as opções de TCP para o cliente.



-  **-R ordem_de_conversão_de_nome:** Permite que seja informado qual a forma de resolver nome será utilizada pelo servidor. As opções são *lmhosts*, *hosts*, *wins*, *bcast*.
-  **-M nomeNetBios:** Esta opção permite enviar mensagens, usando o protocolo WinPopup para outros computadores.
-  **-i escopo:** Especifica o escopo de NetBIOS que o programa *smbcliente* utilizará para comunicar-se ao gerar nomes NetBIOS.
-  **-N:** Suprime o prompt de senha do cliente para o usuário.
-  **-n nome NetBIOS:** Permite anular o nome de host.
-  **-d nível de depuração:** Quanto mais alto for o nível, mais detalhados serão os dados registrados no log. O nível vai de 0 a 10. O padrão é 0.
-  **-p porta:** O número da porta de TCP que será utilizado ao fazer conexões com o servidor. O padrão é 139.
-  **-l nome_do_arquivo_de_log:** Especifica o nome do arquivo onde os dados operacionais serão conectados.
-  **-h:** Imprime uma mensagem de help.
-  **-I endereçamento_IP:** O endereço de IP do servidor ao qual conectar-se.
-  **-E:** Faz com que o *smbclient* escreva mensagens para o fluxo padrão de erro ao invés do fluxo de saída padrão.
-  **-U username:** Especifica o nome de usuário para ser utilizado na conexão com o servidor.
-  **-L:** Permite ver os serviços disponíveis, ou seja, os diretórios e impressoras compartilhados. Muito útil.
-  **-A arquivo:** Esta opção permite especificar um arquivo onde estará contido o nome de usuário e senha que será usado na conexão.
-  **-t código_de_terminal:** Especifica a maneira como interpretar nomes de arquivos que vêm do servidor.
-  **-b tamanho_do_buffer:** Altera o tamanho do buffer de *transmit* e *send*. O padrão é 65250 bytes.



-  **-W workgroup:** Define o grupo de trabalho ao qual você irá se conectar, anulando o grupo de trabalho padrão especificado no arquivo *smb.conf*.
-  **-T opções_de_Tar:** O *smbclient* pode ser usado para criar arquivos do tipo *tar* no compartilhamento.
-  **-D direito_inicial:** Muda para o diretório atual antes de iniciar um processo, pode-se usar junto com o *tar* para que seja compactado no diretório inicial.
-  **-c string_de_comando:** Uma lista de comandos do tipo *csv*, ou seja, separados por ponto-e-vírgula que devem ser executados.

Está é a lista dos comandos que podem ser usados com o *smbclient*.

Mas quase esqueço de falar sobre o SWAT, Samba Web Administration Tool.

O SWAT é uma interface web para administrar os compartilhamentos, configurações do servidor Samba, alterando diretamente o arquivo *smb.conf*. Já havíamos instalado-o no começo do capítulo, agora, vamos fazer algumas configurações para que o mesmo funcione.

Verifique o arquivo */etc/service*.

Neste arquivo, deve existir uma entrada como a descrita abaixo:

```
swat  901/TCP
```

Caso não exista, crie-a na mão.

Outro arquivo muito importante é o */etc/inetd.conf*, onde também deve existir uma entrada do tipo:

```
swat  stream  tcp  nowait.400  root    /usr/sbin/swat
swat
```

Provavelmente, esta linha já existirá, mas estará comentada com o símbolo *#* na frente, retire-o.

Feito estes dois passos, devemos reiniciar do daemon do *inetd*:

```
cd  /etc/rc.d/init.d
./inetd stop
./inetd start
```

O SWAT está praticamente pronto para uso, ele trabalha com qualquer browser direcionado para porta 901.



Abra qualquer browser e digite:

<http://localhost:901>

Será solicitado um nome de usuário e senha. Normalmente, quem tem permissão para administrar é o próprio administrador do sistema, ou seja, o root.

Neste caso, coloque nome de usuário e senha para ter acesso ao Administrador do Samba via web.

Veja abaixo a figura do browser quando solicita nome de usuário e senha para acesso a porta 901.



Solicitação de senha para acesso ao SWAT

O que aconteceu? Ele disse que você não tem permissão de acesso, *alm*. Acho que deixamos alguma configuração de lado.

Abra o arquivo */etc/smb.conf*, verifique uma linha que contenha as palavras *hosts allow* nas Configurações Globais.

Achou? Certo de que está configurada? Há, está faltando a sua rede, não é isso? Então, somente adicione sua rede e retire as que não interessam.

```
host allow = 192.168
```

Neste ponto, é necessário reiniciar o servidor Samba:

Tente novamente acessar o *swat*. Funcionou desta vez?

Logo que abrimos o SWAT temos uma mensagem de boas-vindas e links para ajuda, que são listados abaixo:

Samba Documentation

Daemons

 *smbd*: O daemon *smbd* do Samba.



 *nmbd*: O servidor de nomes NetBIOS.

 *winbindd*: O daemon *winbind*.


Configuration Files

 *smb.conf*: Quando solicitado, ele traz as páginas do manual, o Main.

 *lmhosts*: O arquivo hosts.

 *smbpasswd*: O arquivo de senha do Samba.


Administrative Utilities

 *smbcontrol*: Controla o envio de mensagens para os daemons do Samba.

 *smbpasswd*: Gerencia senha no Samba.


 *SWAT*: Ferramenta de configuração web.

 *make_smbcodepage*: Criação de codepage.

 *make_unicodemap*: Criação de arquivo de mapeamento unicode.


 *smbrun*: Utilitário interno do *smbd*.


Client Tools

 *rpcclient*: Utilitário de comando de linha MS-RPC.


 *smbtar*: Ferramenta de backup do SMB.

 *smbclient*: Samba cliente de comando de linha.


 *smbmnt*: Utilitário de ajuda para montagem de diretório SMB em hosts **Linux**.

 *smbmount*: Ferramenta usada para montar arquivos SMB dentro do **Linux**.


 *smbspool*: Cliente de impressão SMB e comando de linha.

 *smbumount*: Usado para desmontar diretórios SMB dentro do **Linux**.


 *smbstatus*: Monitora os daemons do SMB.

 *testparm*: Valida o arquivo de configuração.

 *testprns*: Teste a configuração da impressora.

 *nmblookup*: Ferramenta para resolver nome NetBIOS.



 *Books* e *Howto* são documentações disponibilizadas pelo Samba:











Interface do SWAT

Logo na primeira página, vemos algumas opções: *Home*, *Globals*, *Share*, *Printers*, *Status*, *View* e *Password*. O *Home* simplesmente é essa primeira página que estamos vendo, a página inicial do SWAT.

Em *Globals*, como você já deve ter desconfiado, trará informações de Configurações Globais (opção setada no arquivo `smb.conf`), como *Workgroup*, *security*, *host allow*, *logs*. Na realidade, são as opções globais descritas no item *[Global]* no arquivo `/etc/smb.conf`. Estas configurações podem ser alteradas diretamente no SWAT.

As Configurações Globais, no SWAT, estão divididas conforme abaixo:




-  Base Options
-  Security Options
-  Base Options
-  Logging Options
-  Tuning Options
-  Printing Options
-  Browse Options
-  WINS Options



Quase que auto-explicativa. Ao lado de cada uma das opções, existe um link de ajuda que, ao ser clicado, levará-lo direto à parte que contém informações dentro do Man (Arquivos de ajuda, manuais).

O segundo item do SWAT são os *Shares*. Clicando nele, a página que contém informações sobre os diretórios compartilhados será aberta, bem como suas permissões.

De início, serão mostradas apenas três itens:

-  **Choose Share:** Ver as opções de compartilhamento do diretório escolhido.
-  **Delete Share:** Deletar o compartilhamento.
-  **Create Share:** Criar novo compartilhamento.

Do lado de *Choose Share*, existe um espécie de ListBox. Clicando nele, serão listados todos os compartilhamentos. Você deve escolher um para depois optar entre *Choose* e *Delete Share*.

Do mesmo o modo, o *Printers* é usado para administrar as configurações de compartilhamento de impressoras. Trabalha da mesma forma que *shares*, tendo as opções *choose*, *delete* e *create printers*, que têm a mesma função no *share*. Escolhendo uma *share* ou impressora, e em seguida clicando em *Choose*, serão mostradas as configurações, em particular do *share* ou *printer*, onde poderão ser alteradas as permissões de compartilhamento e todas as outras configurações que já vimos anteriormente.

A quinta opção, *Status*, mostra o status do servidor, ou seja, se ele está ativo ou não. Mas além disso, é possível parar ou iniciar os serviços *smbd*, *nmbd*. Informa qual versão do Samba está sendo utilizada.

O que acho mais interessante são as informações quanto às conexões existentes no servidor. Em *Active Connection*, serão listados todos os PIDs (processos) da conexão cliente. Serão informados o PID, o nome do cliente, o endereçamento IP e, ainda, a opção de matar o processo, ou seja, desconectar o usuário.

Em *Active Share*, serão mostrados todos os compartilhamentos que estão sendo usados no momento, informados os seus nomes, o usuário que está utilizando, o grupo que o usuário faz parte para poder acessar tal compartilhamento, o nome da máquina que o cliente está utilizando e a data de acesso.

E, por último, temos *Open Files*, que como o nome diz, traz informações de arquivos que estão sendo utilizados pelo cliente, além



de informações do PID, sobre o tipo de compartilhamento, se o cliente pode ler e escrever, o nome do arquivo, a data e a hora que o mesmo foi acessado.

Abaixo, temos uma figura que mostra os compartilhamentos ativos em uma máquina exemplo. Veja que o usuário *thais* está compartilhando três diretórios, que seu grupo é o suporte e que o mesmo está acessando seu diretório *home* e usando o arquivo *rede2.bmp*.

Com essa ferramenta, é possível verificar furos dentro das permissões. Por exemplo, alguém que não deveria acessar certo compartilhamento, está listado em *Active Connections*, utilizando um arquivo que ele deveria ver. Podemos, através disso, rever nosso esquema de permissão.

The screenshot shows the SWAT web interface. The 'Active Connections' section contains a table with one entry for user 'thais' from IP 192.168.200.3. The 'Active Shares' section shows three shares: 'reports', 'stats', and 'data', all owned by 'thais'. The 'Open Files' section shows a file named 'rede2.bmp' being accessed by 'thais' with 'DENY_WRITE' permissions.

PID	Client	IP address	Date	Kill
1267	thais	192.168.200.3	Thu Mar 23 01:22:44 2000	[X]

Share	User	Group	PID	Client	Date
reports	thais	reports	1267	thais	Thu Mar 23 01:22:49 2000
stats	thais	reports	1267	thais	Thu Mar 23 01:22:49 2000

PID	Sharing	R/W	Oplock	File	Date
1267	DENY_WRITE	READONLY	EXCLUSIVE+ATCH	/home/thais/rede2.bmp	Thu Mar 23 01:22:27 2000

Conexões Ativas listadas no SWAT

Temos o item *View* logo após ao *Status*.

O item *View* apenas mostra o conteúdo do arquivo *smb.conf*, ele tem duas opções: *Normal View* e *Full View*.

Em *Normal*, serão trazidas as opções mais utilizadas no item *Global*; já no modo *full*, ele trará todas as opções possíveis dentro do arquivo *smb.conf*, as que estão sendo usadas ou não.

Devo lembrar que só é possível visualizar, não alterar.

Na última página, *Password*, podemos alterar, adicionar e remover usuários do Samba.



O Samba realmente tem assunto para um livro todo, assim como já existe e pode ser consultado livremente.

No site http://samba.he.net/using_samba/ o livro *Using Samba*, da editora O'Reilly.

No próprio site do samba (www.samba.org) pode-se também obter muitas informações.

Com o que foi disponível neste livro, já é possível usar o Samba compartilhado, com permissões que chegam a nível de usuário, um servidor emulando Windows NT.

Também foi possível conectar através de cliente **Linux**, máquinas Windows e o próprio servidor *Linux*.

Com o Samba, é possível fazê-lo PDC, servidor Wins, entre outros.

Começamos implantando somente uma pasta para uso da rede e, quando vemos, estamos explorando as permissões para cada diretório em conjunto com os scripts de *netlogon*.

Com certeza, o Samba foi uma idealização muito feliz e muito bem vinda no mundo OpenSource.

Introdução

Neste capítulo, falaremos sobre administração do servidor, manutenção e criação de usuários, enfim, tudo que seria possível fazer na máquina, só que de forma remota.

Para isso, abordaremos o Telnet, SSH que trabalha como um Shell local. Mais adiante, veremos a administração via web através do Linuxconf Web e o Webmim, que é software distribuído à parte da Conectiva.

Telnet

Para quem não conhece o Telnet, é como o FTP, ele abre um Shell remoto e ao invés da manipulação de arquivos, como é caso do FTP, o Shell permite executar comandos, tais como *adduser*, *ipchains*, *iptables*, *start* e parar qualquer serviço.

Suponha que você queira abrir uma porta no seu servidor, mas você está em outra cidade, o que fazer?

Através do Telnet, é possível acessar a máquina remota e adicionar uma regra ao firewall.

Para instalamos o servidor Telnet, são necessários dois pacotes:

 `telnet-server`

 `telnet`

```
[root@localhost]# rpm -q telnet-*
```

Caso não estejam instalados, para instalá-los, coloque o CD 1 do **Conectiva Linux** no drive de CD-ROM e monte-o:

```
[root@localhost]# mount /dev/cdrom /mnt/cdrom
```



Vá até o diretório das RPMs:

```
[root@localhost]# cd /mnt/cdrom/conectiva/RPMS
```

Execute o comando de instalação:

```
[root@localhost]# rpm -ivh telnet-*
```

Após terminarmos a instalação, precisamos habilitar a porta para o uso.

Vá ao arquivo */etc/inetd.conf*, verifique se existe a seguinte linha:

```
telnet stream tcp nowait root /usr/sbin/tcpd in.telnetd
```

Caso elas estejam comentadas, apenas remova o “#”.

Agora para ativar as mudanças:

```
[root@localhost]# cds
[root@localhost]# ./inet stop
[root@localhost]# ./inet start
```

Verifique se no *ntsysv* o *inet* está selecionado.

```
[root@localhost]# ntsysv
```

Para testar a configuração, digite:

```
[root@localhost]# telnet localhost
```

Deverá aparecer uma tela solicitando nome de usuário e senha, como abaixo:

```
xterm
[root@stato init.d]# telnet localhost
Trying 127.0.0.1...
Connected to localhost.localdomain.
Escape character is '^]'.
Conectiva Linux 7.0
Kernel 2.2.19-15cl

login: stato
Password:
Last login: Thu Mar 23 01:16:38 from localhost.localdomain
[stato@stato stato]$
```



Deve-se liberar a porta 23 no *ipchains* ou *iptables* conforme o Kernel.

Outra observação importante é que, por padrão, o usuário root não pode fazer login por dois motivos. O primeiro é que um o root é um superusuário e o segundo é porque o telnet não é criptografado e suscetível a ataques hacker, com isso, sua senha trafega sem criptografia.

Mas é possível habilitar o usuário root para login no Telnet. Abaixo, está a descrição do que deve ser feito para habilitá-lo.

Deve lembrar que é altamente recomendável não habilitar. Caso haja necessidade de utilizar o usuário root, faça isso através do comando *su* e mude de usuários:

```
[stato@Stato init.d]$ su root
Password:
[root@Stato init.d]#
```

Edite o arquivo */etc/securetty* e insira as seguintes linhas ao final deste:

```
[root@localhost]# mcedit /etc/securetty
```

Insira algumas linhas, conforme exemplo:

```
0
1
2
3
4
5
6
```

Para sistemas baseados em Kernel 2.0.x, substitua 0, 1, ... por

```
ttyp0
ttyp1
ttyp2
ttyp3
ttyp4
ttyp5
ttyp6
```

Essas linhas representam em qual console será disponibilizado o acesso através do Telnet para o root. Caso não deseje ter conexões simultâneas com o usuário root através do Telnet, insira apenas uma linha.

No Conectiva 5, o processo também é similar. Ao invés de colocar apenas os números, como no primeiro caso, acrescente as linhas como no exemplo abaixo (o princípio é o mesmo):



```
pts/0
pts/1
pts/2
pts/3
pts/4
pts/5
pts/6
```

Neste caso, serão liberados até sete acessos como superusuário (root).

Salve o arquivo e reinicialize o *inet*:

```
[root@localhost]# cds
[root@localhost]# ./inet stop
[root@localhost]# ./inet start
```

Com esta configuração, já estaremos habilitados a usar o Telnet, inclusive com o usuário root logando diretamente.

SSH

O SSH é um programa que permite a execução de comandos em uma máquina remota, utilizando para isso um canal de comunicação encriptado. Ele pode ser utilizado como uma alternativa segura a comandos tradicionais do UNIX, como o *telnet*, *rlogin*, *rsh*, *rcp* e *rdist*.

A utilização de métodos de encriptação na comunicação entre duas máquinas torna o SSH uma ferramenta bastante útil na administração de máquinas, uma vez que permite ao administrador verificar e configurar uma máquina remotamente de forma segura, podendo até mesmo executar aplicações como o Linuxconf na máquina remota.




Do ponto de vista prático, ao se utilizar o SSH, é como se o administrador ou usuário estivesse efetivamente sentado em frente ao computador remoto, podendo rodar programas e utilizar recursos deste computador.

Além da característica de utilizar pacotes de comunicação criptografados, que impedem a utilização de programas *farejadores* [1] para capturar logins e senhas, o SSH apresenta como vantagem uma forma de autenticação mais avançada, podendo utilizar chaves assimétricas para usuários e máquinas, além da capacidade de criar *túneis* criptográficos, podendo, assim, executar até mesmo aplicações gráficas remotamente.



É possível utilizar interfaces gráficas remotamente, ou seja, uma vez habilitadas as configurações do XFree para que o mesmo seja utilizado remotamente, é possível através do SSH rodar aplicações gráficas.

Através do *synaptic* os pacotes:

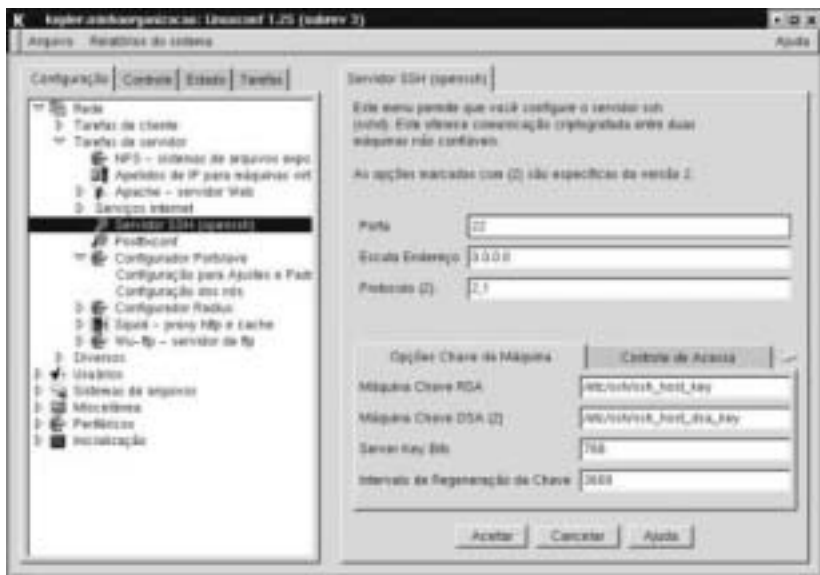
-  `openssh-server`
-  `linuxconf-opensshd`
-  `openssh-client`

Ou abra um terminal e utilize os comandos abaixo:

```
# apt-get install openssh-server
# apt-get install linuxconf-opensshd
# apt-get install openssh-client
```

Ou através dos próprios pacotes *rpm* contidos no CD do Conectiva Linux.

A configuração do servidor SSH pode ser feita utilizando-se a opção *Configuração/Rede/Tarefas de servidor/Servidor SSH (openssh)* do Linuxconf. Ao acessar essa opção, uma tela como a da próxima figura será mostrada:



Configuração do SSH

A configuração padrão do servidor SSH é bastante completa e segura. Veremos a seguir apenas as opções mais importantes.



O campo *Porta* contém o número da porta que o servidor SSH utilizará. A alteração do valor padrão exigirá alterações também no cliente SSH, portanto, sugerimos que seja mantido esse valor.

O campo *Escuta endereço* define quais endereços de rede do servidor deverão ser utilizados para receber conexões SSH. O valor padrão define que todos os endereços serão utilizados.

A versão do protocolo a ser utilizada é definida no campo *Protocolo*. O valor padrão especifica que deverão ser utilizadas as versões 2 e 1 (nesta ordem, de preferência).

Veremos a seguir as principais opções de cada guia apresentadas:



Opções Chave da Máquina: Nesta guia, são configuradas as opções relativas às chaves de autenticação do servidor. O pacote de instalação do *Openssh* gera automaticamente essas chaves e as coloca nos locais corretos, logo não há necessidade de se alterar nada nesta guia.







Controle de Acesso: Aqui, podem ser configuradas restrições ao acesso via SSH. O campo *Permite Grupos* restringe o acesso apenas a usuários de determinados grupos e o *Permite Usuários* apenas a usuários de determinados grupos. Insira os valores nos campos, separando-os com espaços em branco. Valores em branco nesses campos significam que todos os usuários e grupos são permitidos e ao se inserir um valor neles, apenas o usuário ou grupo listado será permitido. Caracteres coringas são permitidos nesses campos. Os campos *Nega Grupos* e *Nega Usuários* podem ser utilizados para restringir o acesso dos usuários ou grupos listados nestes campos. E, por fim, o campo *Permite Registro de Root* especifica se o usuário root pode se logar ou não utilizando o SSH.



Opções de Segurança: Nesta guia, são definidas as opções de autenticação que o servidor SSH usará. A opção *Autenticação de Senha* configura o servidor para utilizar a autenticação por senhas em texto plano através do tunelamento encriptado, ou seja, o usuário poderá utilizar a sua senha do sistema. Se você quiser forçar os seus usuários a utilizarem apenas a autenticação com *passphrases* e chaves criptográficas assimétricas, configure esse campo com o valor "NO". Os campos



Autenticação RSA e Autenticação DSA configuram se o servidor deve utilizar esse tipo de chave assimétrica para autenticar os usuários. O campo *Registra Grace Time* configura quanto tempo o servidor SSH deve esperar pela autenticação do usuário antes de desconectar a conexão.

-  **Opções Gerais:** Aqui, se encontram algumas opções relativas ao comportamento do servidor após o usuário estabelecer a conexão, se o mesmo deve ou não verificar se há mensagens novas para o usuário ou imprimir a mensagem do dia.
-  **Opções do X11:** Por padrão, o servidor SSH não permite o tunelamento de aplicações gráficas via conexão SSH e para habilitá-las você deverá selecionar o valor “yes” no campo *Envio do X11*. Se você deseja utilizar aplicações gráficas remotamente através do SSH, habilite essa opção.
-  **Suporte do skey:** Especifica se a autenticação *skey* é permitida.
-  **Suporte a Kerberos:** Nesta guia, se encontram opções de configuração relativas à autenticação via Kerberos. Utilize essas opções apenas se você possuir um servidor de autenticação com suporte a Kerberos.

Finalize a configuração do servidor SSH ativando a inicialização automática do serviço SSHD e acessando a opção *Controle/Painel de controle/Controle de atividade dos serviços do Linuxconf*.

Agora, vamos fazer a configuração do cliente. O cliente SSH não necessita de nenhuma configuração para funcionar, no entanto, se você pretende utilizar autenticação com chaves assimétricas, como a chave RSA, será necessário gerar essas chaves. Para gerar essas chaves, cada usuário deverá utilizar o comando *ssh-keygen*. Veja como proceder:

Abra um terminal e digite o comando:

```
$ ssh-keygen
```

O comando irá gerar a sua chave dentro do diretório *.ssh* dentro de sua área home, e utilizará por padrão o arquivo *identity*. Ao aparecer a mensagem abaixo, pressione *Enter* para confirmar ou insira um outro nome de arquivo.

```
Generating public/private rsa1 key pair.  
Enter file in which to save the key (/home/usuario/.ssh/  
identity):
```



O comando solicitará, então, que você entre com uma *passphrase* (uma frase-senha). É possível utilizar uma frase-senha vazia pressionando *Enter*, logo a autenticação será feita utilizando-se apenas as chaves. Insira a sua frase-senha ao aparecer a mensagem:

```
Enter passphrase (empty for no passphrase):
```

Insira-a novamente para confirmar ao aparecer a mensagem:

```
Enter same passphrase again:
```

O *ssh-keygen* criará as chaves e mostrará uma mensagem como esta:

```
Your identification has been saved in /home/usuario/.ssh/identity.
Your public key has been saved in /home/usuario/.ssh/identity.pub.
The key fingerprint is:
69:85:b7:a9:74:d9:92:ea:5c:e3:92:cb:47:b2:70:03  \
usuario@meuservidor.minhaorganizacao
```

Para funcionar corretamente, é necessário que o diretório *.ssh/* possua as permissões corretas.

Ele deve ter permissão de leitura, escrita e execução apenas para o dono. Você poderá ajustar as permissões deste diretório, utilizando o seguinte comando:

```
$ chmod 700 ~/.ssh
```

Entre no diretório *.ssh/* e adicione a sua chave pública ao arquivo de chaves autorizadas utilizando o seguinte comando:

```
$ cat identity.pub >> authorized_keys
```

Ajuste a permissão do arquivo contendo as chaves com o comando:

```
$ chmod 600 authorized_keys
```

As configurações necessárias na máquina que será o alvo da conexão SSH estão prontas. É necessário apenas que você copie o arquivo *identity* para a máquina de onde você pretende se conectar, pois ele é a sua chave privada. Crie nessa máquina o diretório *.ssh* e coloque a chave privada lá ou simplesmente especifique o arquivo que contém a chave privada na linha de comando ao se conectar.

Para testar a configuração do servidor SSH, tente se conectar a ele utilizando o cliente SSH. Você poderá fazer isso com o seguinte comando:

```
$ ssh usuario@meuservidor
```













Estamos prontos para utilizar. Com isso, já temos duas ferramentas que pode nos ser útil na administração remota. Neste momento, abordaremos mais duas ferramentas que serão usadas via web: o Linuxconf Web e o Webmim.

Linuxconf




O Linuxconf é um aplicativo avançado de administração para um sistema **Linux**. Ele centraliza tarefas como configuração do sistema e monitoração dos serviços existentes na máquina. Na verdade, o Linuxconf é um gerenciador de módulos, cada qual responsável por executar uma tarefa específica.

Este aplicativo possui quatro abas: *Configuração*, *Controle*, *Estado* e *Tarefas*. A primeira trata de configurações relacionadas basicamente a uma rede ou servidor. A aba *Controle* trata basicamente de padrões e do gerenciamento do próprio Linuxconf; a aba *Estado* permite a visualização de logs e informações gerais do sistema; já a aba *Tarefas* possui uma série de diálogos que podem ser úteis em algumas configurações mais complexas.




Na aba *Configuração*, são fornecidas opções para a configuração de uma rede de modo geral.. Através dela, é possível configurar uma máquina na rede, um servidor, um firewall, buscar informações sobre a rede, etc. Vejamos com detalhes cada uma das abas:

-  **Tarefas de Cliente:** Aqui é possível configurar uma máquina cliente que será incluída na rede. Informações sobre o IP, se fixo, Dns e rede estão localizadas nesta aba.
-  **Tarefas de Servidor:** São fornecidas opções para a configuração de um servidor para os mais diversos objetivos. É possível configurar:
 -  Servidor de Arquivos de Rede (NFS);
 -  Servidor Web (Apache);
 -  Servidor de nomes de domínio (DNS);
 -  Servidor de e-mail (Sendmail, Postfix);
 -  Servidor SSH (Openssh);
 -  Servidor de Alta Disponibilidade;
 -  Servidor de IRC;





-  Servidor de Serviços Internet (por exemplo, Telnet e FTP);
-  Servidor de Autenticação de usuários (Radius e Portslave);
-  Servidor Proxy (Squid);

Esta aba fornece praticamente todas as opções para que se construa qualquer tipo de servidor que se deseje. É possível configurar uma rede TCP/IP a partir do zero.



-  **Serviços de Inicialização:** Essas configurações determinam redes com particularidades. Pode ser montada uma rede com boot remoto. Através destas opções, pode ser também montado um servidor que utiliza o protocolo RARP (que determina o endereço IP a partir de um endereço físico da rede) e um servidor DHCP/BOOTP, para a configuração de IPs dinamicamente.
-  **Firewall:** Opções para a configuração de um firewall, muito importante para a segurança de uma rede.
-  **Diversos:** Opção utilizada para configurações adicionais à rede, como um gerador de gráficos para múltiplas rotas (MRTG) e a configuração de acesso ao Linuxconf via rede, além da inclusão de informações sobre outras máquinas ou outras redes.

A aba *Controle* define o comportamento do Linuxconf em relação à máquina e em relação a outros aplicativos. Ela também fornece opções de ativação, ou seja, controla atividades de serviços e configurações feitas na aba *Configuração*.

Esta opção é subdividida nas seguintes seções:

-  **Painel de Controle:** Controla execução de tarefas, como inicialização de serviços e montagem de sistema de arquivos. Não permite tais configurações, apenas ativa/desativa as mesmas.
-  **Gerenciamento do Linuxconf:** Permite o gerenciamento do Linuxconf em relação a propriedades, permissões, arquivos de configuração e módulos. Esta opção personaliza o Linuxconf.



-  **Data e Horário:** Configura a data e o horário de uma máquina cliente.
-  **Gerenciamento de pacotes RPM:** Permite a instalação ou atualização de pacotes RPM individualmente ou por grupo. Pode-se, também, fazer pesquisas e ver o estado dos pacotes.

Por fim, a aba *Estado* permite a visualização do estado do sistema e do hardware, bem como seus detalhes, além da descrição de registros referentes ao sistema e a inicialização de aplicativos.





Demos uma visão geral do Linuxconf e agora iremos habilitá-lo para uso via web.

O primeiro passo para habilitar o acesso ao Linuxconf através da rede é ir até o menu *Configuração/Ambiente de Rede/Diversos/Acesso ao Configurador Linux via rede* e marque a opção *Ativa Acesso via Rede*.

Além desta opção, seria útil selecionar *Acesso de registro no arquivo/var/log/htmlaccess.log* para que os acessos fiquem registrados neste arquivo.

Logo abaixo destas opções, você encontra alguns campos para definir quais máquinas podem acessar o Linuxconf via rede. Se estes não forem preenchidos, o Linuxconf aceitará apenas conexões da rede local da primeira placa de rede detectada pelo sistema, o que é considerado um funcionamento razoavelmente seguro.

Pode-se, ainda, habilitar outros usuários ou até a rede externa para tenha acesso ao Linuxconf.

-  *um nome de máquina;*
-  *um endereço IP;*
-  *um par de endereços IP e máscara de rede;*
-  *um nome de dispositivo (*eth0*, *eth1*, etc).*

Como esta interface que o Linuxconf roda através do *inetd*, é necessário verificar se ele está habilitado.

Este procedimento pode ser efetuado através da caixa de diálogo *Controle/Painel de Controle/Controle de atividade dos serviços*.

Nesta caixa de diálogo, procure pelo serviço Linuxconf, o qual deverá estar marcado como *Inativo*, como ilustrado na figura.



Habilitando o inet para o linuxconf

Também é possível editar manualmente o arquivo `/etc/inetd.conf`, retirando o comentário da última linha correspondente ao Linuxconf. Caso ela não exista, adicione-a.

```
# linuxconf stream tcp wait root /bin/linuxconf
linuxconf -http
```

Verifique no arquivo `/etc/services` se a linha referente ao Linuxconf está comentada, se estiver descomente, se não existir, inclua:

```
linuxconf    98/tcp                # Linuxconf web
interface
```

Depois ative essa mudança:

```
[root@localhost]# cds
[root@localhost]# ./inet stop
[root@localhost]# ./inet start
```

Pronto. Agora basta apontar seu navegador para `http://<sua_máquina>:98`.

Nota: Em algumas versões de navegadores do Internet Explorer não funciona.



Linuxconf WEB

Tudo que pode ser feito no Linuxconf também poderá ser feito via web. Na figura acima, podemos ver que existe *Rede*, *Usuários*, *Periféricos*, *Controle* (*Painel de Controle*), enfim, tudo que é preciso para administrar seu servidor **Linux**.

Mas temos também um software que pode ser utilizado para administração via web: o Webmin. O primeiro passo é fazer download do pacote *webmin-0.950.tar.gz* no site www.webmin.com, ele tem algo em torno de 4,9 Mb. Feito download, vá até o diretório onde está o arquivo *tar* e execute o seguinte comando:

```
[root@localhost~]# cp webmin-0.950.tar.gz /usr/local
[root@localhost~]# cd /usr/local
[root@localhost local]# gunzip webmin-0.950.tar.gz
[root@localhost local]# tar xf webmin-0.950.tar
[root@localhost local]# cd webmin-0.950
```

Com isso, será criado o diretório *webmin-0.950*, acesse-o e execute o script *setup.sh*.

```
[root@localhost local]# cd webmin-0.950
[root@localhost local]# ./setup.sh
```

Serão feitas muitas perguntas, como qual o local de instalação, distribuição e versão do **Linux**. Na maioria, é só clicar *Enter*, pois já tem um padrão preestabelecido.



Após definida a configuração, será informado que o *webmim* terminou a instalação e pode ser usado na porta 10000, que é a porta padrão para ele. Isso pode ser alterado através do próprio Webmim. Para acessá-lo, digite no browser:

<http://localhost:10000>

Caso você tenha criado um usuário diferente do root para administrar o Webmim, será necessário logar com ele, pois o Webmim solicitará nome de usuário e senha. Logo que o Webmim for acessado, será apresentado com a imagem igual a da figura.



Tela inicial do Webmin

Temos seis itens logo acima que são:













-  WebMin
-  System
-  Servers
-  Hardware
-  Cluster
-  Others

A primeira tela é referente às opções do Webmim, é a mesma da figura.



Temos as seguintes opções:

Configuração do Webmim

-  *Controle de Acesso do IP:* Dá permissão a todos os endereços.
-  *Porta e Endereço:* Porta 10000 (por padrão), podendo ser modificada se for desejado.
-  *Ligação no endereço IP:* Localhost, seus endereçamentos IP de várias placas *eth0* , *eth1*, nome no domínio, etc.
-  *Criação de historial:* Referente ao log,
-  *Servidores Proxy:* Só no caso de acessar um servidor proxy, User Interface, são configurações de cores.
-  *Módulos do Webmin:* São usados para adicionar mais módulos depois de instalado. Um exemplo seria o *ipchains* e o *iptables*: deve-se fazer download do módulo no próprio site do Webmim e em seguida instalá-lo. Depois de instalado pode-se configurar por este item.
-  *Sistema Operativo:* Referente ao sistema operacional. Desde o **Linux**, varia distribuições - Solaris, FreeBSD. Caso esteja configurado incorretamente, altere-o nesta guia.
-  *Linguagem:* A linguagem padrão do Webmim está em Português (*Brazilian*).
-  *Index Page Options:* Opção de indexação de página, colunas, etc.
-  *Upgrade Webmin:* É possível fazer upgrade do arquivo local ou diretamente do site, onde mesmo verifica qual a última versão é instala.
-  *Authentication:* Opções de autenticação, onde é possível até configurar segurança para evitar ataques de força bruta.
-  *Reassign Modules:* Este item é bastante interessante, pois lhe dá a flexibilidade de colocar os módulos onde quiser. Por exemplo: o servidor DHCP está dentro da guia Servers, mas poderia ser mudado sem nenhum problemas para Network ou Hardware, o que não faria muito sentido.



Edit Categories: Neste item, é possível criar novas categorias, que poderão ser usadas para organizar o módulos em Reassign Modules. As categorias são as seguintes:

webmin	system	syslet
servers	cluster	info
net	hardware	other



Webmin Themes: O Webmin dá suporte a temas. Na instalação, vem com KDE, Caldera, MSC.Linux Theme.



Trusted Referers: Trata de permissões de outros domínios e sites.



Encriptação SSL: Referente às configurações SSL.



Certificate Authority: Usado junto com Encriptação SSL para configuração dos Certificados.



Usermin Cofiguration: Trata das configurações de usuários do Webmin.



Utilizadores Webmin: Mostra os usuários cadastrados para utilizar o Webmin, bem como quais módulos os mesmos têm acesso. Podendo ser alterado nesta mesma guia.



Webmin Actions Log: Utilizado para pesquisa no log através do nome do usuário, data, módulos, etc.



Index de Servidores Webmin: Utilizado para procurar outros servidores que possuem Webmin instalado.

Esta é a primeira guia. Logo em seguida, temos grupo *System*, que como o próprio nome diz, são tarefas de sistema, como backup para iniciar daemon, Sistema de Arquivo, montagem de sistemas de arquivos (NFS), criação e alteração de usuários, entre outras coisas.

Abaixo, segue a lista das tarefas possíveis. Não achei necessário comentar as tarefas, pois o próprio nome já diz o que pode ser feito:



Change Passwords;



Configuração de Inicialização SysV;
















Exportações de NFS;



Filesystem Backup;



-  Historiais do Sistema;
-  Iniciar e Encerrar;
-  MON Service Monitor;
-  NIS Client and Server;
-  PAM Authentication;
-  Pacotes de Software;
-  Processos em Curso;
-  Páginas do Manual;
-  Quotas de Disco;
-  Scheduled Commands;
-  Sistema de Ficheiros de Disco e Rede;
-  Tarefas Agendadas (Cron);
-  Utilizadores e Grupos.





















Vemos que é possível administrar o sistema de uma forma bem completa. Temos Quotas para usuário, se o sistema permitir, Scheduled Commands e até agendamento via Cron. Abaixo, veja a figura, que mostra todos os módulos da guia *System*.



Módulos de System



A próxima guia será a *Serves*. No meu ponto de vista, uma das mais interessantes, pois é possível configurar servidores de e-mail, entre eles: Sendmail, Postfix, Servidores Samba, Servidores DNS, Banco de Dados, etc. Abaixo, a lista dos servidores:

-  Administrador de Listas Majordomo;
-  Configuração do Sendmail;
-  Fetchmail Mail Retrieval;
-  Jabber IM Server;
-  Partilha de Ficheiros de Windows com Samba;
-  Postfix Configuration;
-  PostgreSQL Database Server;
-  ProFTP Server;
-  Qmail Configuration;
-  SSH Server;
-  SSL Tunnels;
-  Servidor Proxy Squid;
-  Servidor WU-FTP;
-  Servidor Web Apache;
-  Servidor de DHCP;
-  Servidor de DNS BIND;
-  Servidor de DNS BIND 4;
-  Servidor de base de dados MySQL;
-  Serviços de Protocolos de Internet;
-  Usernames e Passwords de Acesso de Utilizadores PPP.







Veja abaixo a figura do item *Servidores*:



Módulos de Servers

Só para ilustrar, vamos verificar a configuração do Servidor Proxy Squid.

Clique no item *Squid Proxy Server*, que será aberta uma nova tela com as seguintes opções:

-  **Ports and Networking:** Configura a Porta de Servidor por padrão 3128.
-  **Other Caches:** No caso de haver outro cache (por exemplo, um domínio em uma porta do tipo 8080).
-  **Memory Usage:** Configurações do uso de memória (por padrão, somente 8 Mb - RAM).
-  **Logging:** Referente a log de arquivos.
-  **Cache Options:** Opções de configuração do *Squid*, como local do cache, tamanho máximo, etc.
-  **Helper Programs:** Outras configurações do *Squid*, como DNS, FTP, ping cache.



Access Control: Certamente, uma das mais importantes dentro do Squid. Nesta guia, é possível fazermos nossas ACLs ou nossa lista de controle de acesso. Como não somos super conhecedores de todas as ACLs possíveis, este, além de configurar, é uma ótima forma de aprendermos. Para se ter uma idéia, ele configura 23 tipos de ACLs, incluindo as conhecidas : URL Port, URL Regexp, URL Protocol, Data and Time.



Administrative Options: Algumas configurações de administração do *Squid*.



Proxy Authentication: Configurações de autenticação, inclusive inclusão de novos usuários na lista de acessos permitidos.



Miscellaneous Options: Outras configurações, como Teste de DNS, Mensagem de erro, Configurações de Estatística.



Cache Manager Statistics: Como o nome diz, são estatísticas de uso do *Squid*, bastante completo, mas complexo. Na lista, constam mais de trinta itens que podem ser verificados.



Clear and Rebuild Cache: Com este item, será apagado o cache do *Squid* e reiniciado.











Configurações do Squid Proxy Server






A próxima guia é a *Hardware*, que trata das configurações deste. Temos configurações de impressora, do lilo, grub, rede, partições, criação de haid.

Os itens estão listados abaixo:

-  Administração de Impressoras;
-  CD Burner;
-  Configuração de Arranque do Linux (LILO);
-  Configuração de Rede;
-  GRUB Boot Loader;
-  Partições em Discos Locais;
-  RAID do Linux;
-  Tempo do Sistema.


Em seguida, veremos a guia *Cluster*. Este *Cluster* é de Alta Disponibilidade, ou seja, temos duas máquinas com as mesmas configurações (por exemplo, Apache), mas somente uma ativa. Se por algum motivo a máquina ativa parar de funcionar, automaticamente, o *Cluster* faz com que a segunda máquina seja ativada.

Dentro de *Cluster* temos três opções:

-  Cluster Software Packages;
-  Cluster Users and Groups;
-  Heartbeat Monitor.

Na última opção padrão, temos *Outher*, também muito interessante.

Existem seis módulos neste item:

-  **Administrador de Ficheiros:** Trabalha semelhante a um gerenciador de arquivos. Neste item, é possível administrar, ler, escrever, enviar e fazer downloads de arquivos da máquina local para o servidor, deletar, criar, enfim, gerenciar arquivos como se estivesse localmente.



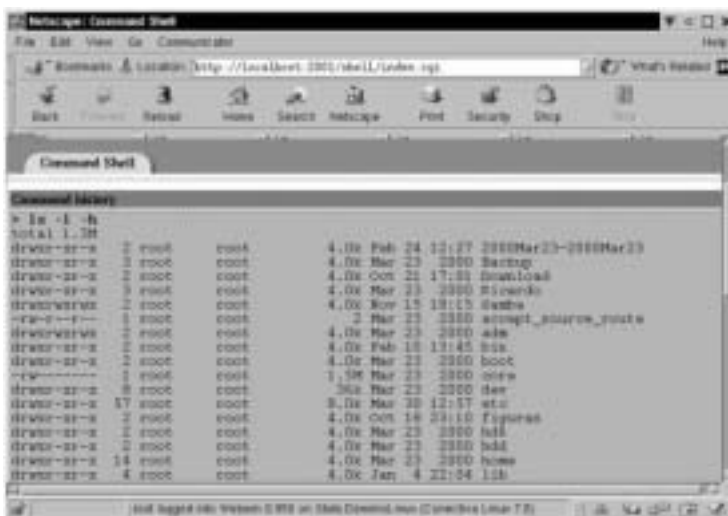
Administrador de Ficheiros



Comandos Personalizados: Neste item, é possível criar comandos ou cascatas de comandos, usando inclusive variáveis. Depois de criado, se for necessário, passe o parâmetro e clique em executar.






Command Shell: Também muito interessante, pode-se executar qualquer comando, este será executado e o resultado, se houver um, será trazido para a interface do Webmin. Veja abaixo a figura depois de executado o comando `ls -l -H`:



Command Shell



-  **Login via SSH/Telnet:** Faz uma conexão telnet com o servidor.
-  **Perl Modules:** Instala novos módulos perl.
-  **System and Server Status:** Mostra o status dos servidores que a máquina Webmin possui. Ainda é possível agendar monitoramento e configurar para que de fato ocorra, sendo enviado e-mail para determinada configuração.

Alguns módulos não vêm junto ao padrão. Muitos módulos são desenvolvidos pelos próprios usuários do Webmin e, posteriormente, disponibilizados no site do Webmin. Dessa forma, você poderá, ainda, colocar quantos módulos quiser. Como disse no exemplo, pode-se colocar *ipchains* ou *iptables*, interfaces de uso do MySQL, etc.

Certamente, este é um ótimo Administrador Remoto, mas infelizmente, às vezes não temos uma interface gráfica ou até mesmo um browser para podermos utilizá-lo, sendo assim, torna-se necessário Administração via telnet ou SSH. Mas sempre que possível, será escolhido o Webmin para administração.

O próximo capítulo terá um apanhado de dicas retirados de sites, revistas, livros, etc. Coloquei apenas alguns, aqueles que julguei de importância.

Estarei disponibilizando uma enquete sobre o livro no site www.dominiolinux.com.br, caso o leitor tiver interesse em fazer algum tipo de crítica.



15

Dicas

Introdução

Este último capítulo tem com objetivo ajudar o leitor em certas tarefas que são executadas diariamente, ou até mesmo abordar algo que não se enquadrou dentro dos assuntos estudados no livro.

Um Mini-HOWTO de Atualização do Kernel

Este texto foi escrito por João Alexandre Voss de Oliveira (joalex@pro.via-rs.com.br), a princípio, como uma mensagem na lista **Linux-BR** e colocado no site **Linux in Brazil**.

O foco de João e os exemplos que ele usa são todos para um upgrade do Kernel 2.2.13 para o 2.2.14. Mas todos os procedimentos expostos são genéricos o suficiente para qualquer upgrade da série 2.2 e, eventualmente, das séries posteriores.

Introdução

Peço desculpas pela pretensão, mas como tenho observado várias dúvidas de companheiros quanto à atualização do Kernel de suas distros, resolvi sair do “ghost” e enviar este “mini-HOWTO”, indicando os passos para a atualização do Kernel.

IMHO, a única necessidade “real” de recompilar o Kernel para o usuário novato é a de adequá-lo, assim que instalado, a seu tipo de processador, para não sub-utilizar a máquina. Para isso, é interessante uma primeira compilação.

Claro que a adequação do Kernel ao tipo de hardware do usuário também é um bom motivo, mas o usuário comum pode esperar um pouco até dominar um o **Linux**.



Entretanto, há aqueles que não se contentam com uma simples compilação e adoram perder horas de sono e convívio familiar somente pelo prazer de “fuçar” na máquina. Felizmente (?) eu me enquadro nessa categoria, pois senti aquele “comichão” quando vi lançado o Kernel 2.2.14 - meu Slackware 7.0 não podia ficar para trás :) .

Apesar de eu não acreditar muito em “receitas”, seguem os passos que eu executei para a atualização do Kernel 2.2.13 para o 2.2.14, no Slackware.

A Receita...

Os passos a seguir dizem respeito à compilação do Kernel com a construção e instalação de novos módulos adequados às necessidades do usuário. Você deve estar logado como root (passo “2” em diante):

1. Baixar o *linux-2.2.14.tar.gz* do www.kernel.org ou algum mirror.
2. Copiar o *linux-2.2.14.tar.gz* para */usr/src* `cp linux-2.2.14.tar.gz /usr/src`.
3. Em */usr/src*, excluir o link simbólico “linux”: `rm linux`.
4. Descompactar o fonte do Kernel copiado em */usr/src* - será criado um diretório chamado “linux”: `tar -zxvf linux-2.2.14.tar.gz`.
5. Renomear o diretório “linux” recém descompactado: `mv /usr/src/linux /usr/src/linux-2.2.14`.
6. Criar um novo link simbólico “linux”, desta vez apontando para o novo diretório *linux-2.2.14*: `ln -s /usr/src/linux-2.2.14 linux`.

Neste ponto, você deve ter em */usr/src* os dois diretórios e um *symlink*:

```
# pwd
# /usr/src
# ls
# linux@ linux-2.2.13 linux-2.2.14 rpm sendmail
```

Essa é a listagem em */usr/src* em meu sistema.

O link simbólico deve apontar para o diretório *linux-2.2.14*. É nele que o trabalho a seguir deve ser feito.



Não delete o diretório *linux-2.2.13* ainda, você pode precisar dele mais tarde se as coisas derem erradas.

Os próximos passos devem ser executados em */usr/src/linux*;

```
# cd /usr/src/linux
# make menuconfig
```

O comando acima iniciará a configuração em modo gráfico do Kernel.

Opcionalmente, estando em algum ambiente X (KDE, Windowmaker, Blackbox, etc.), o comando poderá ser:

```
# make xconfig
```

Escolha a opção “Store configuration to file” e salve a configuração atual em “/” com um nome bem claro, tipo “config.original” ou algo parecido.

Mais tarde, se você precisar restaurar a configuração original, basta rodar o *make menuconfig* e carregar os parâmetros a partir deste arquivo.

Em seguida, escolha a opção “Quit without save” e saia.

Continuando, digite (em */usr/src/linux*):

```
# make mrproper
```

Isto apagará a configuração anterior do Kernel no sistema.

Em seguida, novamente:

```
# make menuconfig (ou make xconfig, se for o caso)
```

Escolha as opções que quiser, adapte o Kernel ao seu hardware. Salve a configuração.

Novamente no prompt:

```
# make dep
# make clean
# make zImage
```

Os comandos acima criam as dependências necessárias, limpam arquivos temporários e de instalação e criam a imagem compactada do novo Kernel, em */usr/src/linux/arch/i386/boot*.

No meu caso, como a imagem do Kernel a ser gerado é muito grande, tenho de usar “make bzImage” ao invés de “make zImage”. Significa, literalmente, “make a BIG zImage”.



Após algum tempo (que varia de acordo com a máquina), a imagem é gerada e gravada no diretório acima referido.

Vá agora para o diretório */lib/modules*. O comando *ls* dá a listagem do conteúdo:

```
# cd /lib/modules
# ls
# 2.2.13
```

O único diretório listado, 2.2.13, é onde encontram-se os módulos originais da primeira instalação. Caso você esteja apenas compilando um Kernel existente, renomeie para 2.2.13.old, por exemplo. Caso seja uma atualização, deixe intacto, pois a seguir será criado um novo diretório.

Volte para */usr/src/linux*:

```
# cd /usr/src/linux
```

Digite os comandos:

```
# make modules
# make modules_install
```

Os comandos acima criarão e instalarão os novos módulos em */lib/modules*. No caso da atualização desta receita, será criado um diretório “2.2.14” em */lib/modules*.

O trabalho de criação dos módulos pode demorar um pouco, dependendo do tipo de máquina (processador, RAM disponível, etc.).

Após o prompt ser novamente liberado (!), mova a imagem compactada do Kernel de */usr/src/linux/arch/i386/boot* para o diretório raiz (/):

```
# mv /usr/src/linux/arch/i386/boot/bzImage /
```

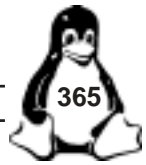
Vá para o diretório raiz e renomeie o novo Kernel recém compilado para um nome facilmente identificável:

```
# cd /
# mv bzImage vmlinuz-2.2.14
```

O novo Kernel, agora, terá o nome acima, *vmlinuz-2.2.14*.

Observe que será necessário indicar, ao LILO, qual a imagem compactada que deverá ser inicializada no boot da máquina. Vá para */etc* e edite o arquivo *lilo.conf*:

```
# cd /etc
# pico lilo.conf
```



Meu */etc/lilo.conf* se parece com isto:

```
# LILO configuration file
# generated by 'liloconfig'
#
# Start LILO global section
boot = /dev/hda
message = /boot/boot_message.txt
prompt
timeout = 20
vga = normal
image = /vmlinuz
root = /dev/hda1
label = Linux
read-only
# Linux bootable partition config ends
```

Veja que apenas uma imagem pode ser inicializada. Para adicionar a nova imagem, na seção:

```
image = /vmlinuz
root = /dev/hda1
label = Linux
read-only
```

Adicione mais uma entrada:

```
image = /vmlinuz-2.2.14
root = /dev/hda1
label = New
read-only
```

O arquivo *lilo.conf* deve ficar assim:

```
# LILO configuration file
# generated by 'liloconfig'
#
# Start LILO global section
boot = /dev/hda
message = /boot/boot_message.txt
prompt
timeout = 20
vga = normal
image = /vmlinuz
root = /dev/hda1
label = Linux
read-only
image = /vmlinuz-2.2.14
root = /dev/hda1
```



```
label = New
read-only
# Linux bootable partition config ends
```

Salve o arquivo *lilo.conf*. No prompt, para salvar as alterações, digite:

```
# lilo
```

Você poderá ver:

```
Added
Linux *
New
```

Indica que a imagem *vmlinuz-2.2.14* poderá ser inicializada no boot da máquina, digitando “new” no prompt do *lilo*. Posteriormente, você poderá eliminar a imagem antiga (*vmlinuz*) em “/” e deletar o diretório */lib/modules/2.2.13*. Mas espere alguns dias, observando se não ocorrem erros em módulos e se todas as funcionalidades estão operantes (ppp, etc).

Efetue um reboot na máquina e a nova inicialização deverá apresentar:

```
Welcome to linux 2.2.14
```

Indica que tudo correu como devia e o novo Kernel está instalado. Boa diversão!

Se as Coisas Derem Errado...

Observe as mensagens de inicialização, às vezes alguns módulos “teimam” em não ser instalados corretamente...

Em caso de *Kernel Panic*, inicialize a partir de um disco de inicialização (Você não tem um? O que está esperando para fazer um?) e efetue as alterações necessárias. É possível voltar ao antigo Kernel 2.2.13 simplesmente desfazendo as alterações. Lembre-se que o comando “linux” digitado no prompt do *lilo* inicializará a imagem antiga do Kernel 2.2.13, e o comando “new” a nova imagem, *vmlinuz-2.2.14*.

Conclusão

Não deixe de consultar a documentação referente ao Kernel, em especial o “kernel HOWTO” (em *linuxdoc.org*) e o guia para o Kernel 2.2.XX.



Como dito, esta é uma receita simples, que visa orientar os iniciantes a trilharem o caminho das pedras... não tem a pretensão de ser nada mais além disso.

Espero ter sido útil, estamos a disposição para eventuais esclarecimentos (e mesmo correções por parte dos “mais velhos” da lista).

Adicionando Memória RAM

Em alguns Kernels mais antigos e em suas distribuições, às vezes, quando temos mais 64 MB, o **Linux** não consegue verificar. Para colocá-la manualmente, edite o arquivo `/etc/lilo.conf`:

```
vi /etc/lilo.conf
```

Adicione a seguinte linha ao arquivo:

```
append="mem=128M"
```

Salve o arquivo e execute o comando:

```
lilo
```

Logicamente, se você tiver mais de 128Mb, coloque o valor correspondente subtraindo a quantidade de memória se ela for compartilhada.

Como Fazer para o CDR-RW IDE Funcionar no Linux

```
### CL70 e anteriores ###
```

Para fazer funcionar seu drive de CD-R/CD-RW, você precisa, primeiramente, estar certo de que a instalação na máquina esteja correta e que funcione como leitora normal, somente para leitura. Se isto está OK, então passaremos à configuração dele como gravador propriamente dito.

Antes de mais nada, certifique-se de entrar no sistema como root e verifique se o pacote `mkisofs-1.8-4cl` está instalado. Para verificar, dê o seguinte comando:

```
[root@localhost]# rpm -qa |grep -i mkisofs
```

Se não retornar nada, é por que não está instalado, então, você precisa instalá-lo. Para isso, coloque seu CD 1 no drive de CD-ROM, monte-o e instale o pacote. Faça-o da seguinte maneira:



```
[root@localhost]# mount /mnt/cdrom
[root@localhost]# cd /mnt/cdrom/conectiva/RPMS
[root@localhost /RPMS]# rpm -ivh mkisofs-1.8-*
```

Espere aparecerem as cerquilhas (####), após isso, o pacote estará instalado e poderá ser usado pelo script de gravação (já serão mencionados mais adiante).

Agora desmonte seu CD-ROM:

```
[root@localhost /RPMS]# cd /
[root@localhost]# umount /mnt/cdrom
```

Agora, verifique se o pacote *cdrecord-1.8* está instalado. Para isso, faça:

```
[root@localhost]# rpm -qa | grep -i cdrecord
```

Caso não apareça nada, é necessário instalar o pacote que se encontra no CD 2. Coloque este no drive de CD-ROM, monte-o e instale-o da seguinte maneira:

```
[root@localhost]# [root@localhost]# mount /mnt/cdrom
[root@localhost]# cd /mnt/cdrom/conectiva/RPMS
[root@localhost /RPMS]# rpm -ivh cdrecord-1.8-*
```

Espere as cerquilhas (####) aparecerem, após isso, o pacote estará instalado e poderá ser usado pelo script de gravação (que serão mencionados mais adiante).

Agora que os pacotes necessários para o processo foram instalados, vamos dar uma olhada nos módulos que estão levantados na sua máquina, para isso, dê o seguinte comando:

```
[root@localhost]# lsmod
```

Deve dar como resultado algo do tipo:

Module	Size	Used by
nls_cp437	3920	1 (autoclean)
ide-cd	24304	1 (autoclean)
isofs	17888	1 (autoclean)
soundcore	2608	0 (autoclean) (unused)
nfsd	162336	8 (autoclean)
nfs	46272	4 (autoclean)
lockd	43760	1 (autoclean) [nfsd nfs]
sunrpc	58672	1 (autoclean) [nfsd nfs]
lockd]		
8139too	11744	1 (autoclean)
agpgart	18640	0 (unused)



O módulo que nos interessa é o *ide-cd* e precisamos removê-lo, para isso, faça:

```
[root@localhost]# rmmmod ide-cd
```

Após ter feito isso, verifique se ele já não está dando o mesmo comando, *lsmod*, como descrito acima, pois este não deve mais aparecer na lista. Agora, precisamos levantar os módulos para emular drives SCSI, faça:

```
[root@localhost ]# modprobe sg; modprobe ide-scsi
```

Após isso, edite o arquivo */etc/rc.d/rc.local* e copie esta linha de comando no fim do arquivo:

```
[root@localhost]# mcedit /etc/rc.d/rc.local
```

Coloque no final de tudo:

```
modprobe sg; modprobe ide-scsi
```

Após isso, dê *F2* para salvar e *F10* para sair.

Agora, precisamos testar para verificar se a gravadora foi testada. Siga o procedimento abaixo:

```
[root@localhost ]# cdrecord -scanbus
```

Se tudo ocorrer bem, irá parecer algo do tipo:

```
Cdrecord 1.8 (i686-pc-linux-gnu) Copyright (C) 1995-2000
Jörg Schilling
Using libscg version 'schily-0.1'
scsibus0:
0,0,0      0) 'HP          ' 'CD-Writer+ 9300 ' '2.0g'
Removable CD-ROM
cdrecord: Warning: controller returns wrong size for CD
capabilities page.
0,1,0      1) 'CREATIVE' 'CD4820E CS990211' '1.03'
Removable CD-ROM
0,2,0      2) *
0,3,0      3) *
0,4,0      4) *
0,5,0      5) *
0,6,0      6) *
0,7,0      7) *
```

Feito isso, a gravadora e os outros drivers de CD-ROM da máquina deixarão de responder por um device IDE (hdXn) e passarão a responder por um device SCSI (srn).



O X representa se o dispositivo está como *primary master/slave* ou *secondary master/slave* (**hda**, **hdb**, **hdc** ou **hdd**) o **n** representa o número da partição no dispositivo.

Para ter certeza que esses drivers estão respondendo como dispositivos SCSI, digite:

```
[root@localhost]# dmesg |grep -i scsi
```

Irá aparecer o seguinte:

```
scsi0 : SCSI host adapter emulation for IDE ATAPI
devices
scsi : 1 host.
Detected scsi CD-ROM sr0 at scsi0, channel 0, id 0, lun
0
sr0: scsi3-mmc drive: 16x/48x cd/rw xa/form2 cdda tray
<- Dispositivo
SCSI (sr0)
```

Agora que sabemos que o dispositivo pelo qual o *CDR/CDRW* irá responder é o *sr0*, vamos tentar montar um CD:

```
[root@localhost]# mount /dev/sr0 /mnt/cdrom
```

Caso volte para tela uma mensagem de erro do tipo:

```
mount: the kernel does not recognize /dev/sr0 as a block
device (maybe  nsmod driver'?)
```

É porque não existe, na máquina, um dispositivo *sr0*. Para criá-lo, digite:

```
[root@localhost]# mknod /dev/srn b 11 m
```

Caso tenha mais de um CD-ROM, mesmo que não seja gravador, você também precisa criar um device para que ele assim faça o seguinte: troque o “n” pelo número do device, ou seja, se for o primeiro, coloque *sr0*; se for o segundo, *sr1*, e assim por diante. O mesmo para o “m” da sentença acima. O exemplo abaixo mostrará como criar dois devices no caso de se ter dois drives de CD-ROM, um gravador e outro não; mas não faz diferença se ambas forem gravadoras. Ex:

```
[root@localhost]# mknod /dev/sr0 b 11 0
[root@localhost]# mknod /dev/sr1 b 11 1
```

Agora, para facilitar a montagem dos dispositivos no seu sistema, é interessante, mas não necessário, criar um link para ele, como mostra o exemplo:

```
[root@localhost]# ln -sf /dev/sr0 /dev/cdrw
[root@localhost]# ln -sf /dev/sr1 /dev/cdrom
```



Perceba que eu estou assumindo como sendo seu gravador de CD o *sr0*. Caso seja o *sr1* ou qualquer outro número, mude conforme necessitar. Vamos criar o diretório que será usado como ponto de montagem padrão para o *CDR/CDRW*:

```
[root@localhost]# mkdir /mnt/cdrw
```

Agora, vamos editar o *fstab* para poder ter facilidade de montagem no sistema. Para isso, faça:

```
[root@localhost]# mcedit /etc/fstab
```

/dev/hda1	/	ext2	defaults
1 1			
/dev/hda2	swap	swap	defaults
0 0			
/dev/fd0	/mnt/floppy	auto	user,noauto
0 0			
/dev/cdrom	/mnt/cdrom	iso9660	
user,noauto,ro	0 0 <		
/dev/cdrw	/mnt/cdrw	iso9660	
user,noauto,ro	0 0 <		
none	/proc	proc	defaults
0 0			
none	/dev/pts	devpts	
gid=5,mode=620	0 0		

Deixe as linhas marcadas com ‘<’ conforme o exemplo.

Pronto. Agora você tem configurado seu gravador de CD e o seu drive de CD no **Conectiva Linux**. A seguir, mostraremos como você pode gravar seus CDs ou arquivos usando o **Linux**. O primeiro passo é baixar os scripts que se encontram no endereço:

```
http://www.conectiva.com.br/~suporte-cl/programas/cdrwtool.tgz
```

Ainda como root, copie este para o “/” (raiz) do seu disco. Imaginemos que você o baixou da rede e está no seu home:

```
[root@localhost]# mv ~/cdrwtool.tgz /
```

Agora, vá até o “/”:

```
[root@localhost]# cd /
```

Descompacte o script da seguinte maneira:

```
[root@localhost]# tar xzvf cdrwtool.tgz
```

Após tê-lo descomprimido, poderá começar a fazer uso dos mesmos.



Neste arquivo, vão dois scripts: um para criar imagens *iso* dos arquivos ou CDs que desejar gravar e outro que gera o CD a partir destas imagens *iso*. Esta é a maneira como nós recomendamos que você faça suas gravações. Caso você conheça outra, sinta-se à vontade para usá-la.

Neste arquivo, estão dois scripts: o *make_iso*, que serve para criar imagens, e o *run_cdrw*, para gravar o CD propriamente. A seguir, passarei a explicar brevemente como funciona cada um deles (ambos são executados como root).

Make_iso

Funciona criando uma imagem a partir de um diretório, sendo que, recursivamente, agrupa todos os arquivos e subdiretórios que possam existir no diretório indicado. Por exemplo, vou criar a imagem de um CD que está no CD-ROM. Primeiramente, coloque o CD que quer copiar no drive de CD-ROM e monte ele da seguinte maneira:

```
[root@localhost]# mount /mnt/cdrom
```

Vá ao diretório que você deseja que a imagem fique. Imaginemos que criei o diretório */CDRW-ISO* para armazenar as imagens:

```
[root@localhost]# mkdir /CDRW-ISO
[root@localhost]# cd /CDRW_ISO
```

Considere que as imagens apenas criam um arquivo com todo o conteúdo do disco ou diretório dado, assim, este não comprime e o tamanho do arquivo de imagem é de aproximadamente o tamanho do total do CD ou diretório. Caso o CD esteja cheio, são aproximadamente 650MB que precisam estar disponíveis no disco. Se não tiver esta capacidade livre no disco, o procedimento não vai funcionar. Para verificar o espaço livre em disco, entre com o comando:

```
[root@localhost]# df
```

Por exemplo:

Filesystem	1k-blocks	Used	Available	Use%
Mounted on				
/dev/hda1	2015936	1611844	301680	84%
/				

Por exemplo, no disco acima, tenho apenas cerca de 301 MB livres (Available) e 84% usado, assim, não poderei fazer uma imagem maior que esta capacidade. Caso você tenha este problema, libere



espaço no seu disco e comece o procedimento depois de liberado o espaço necessário.

Agora, estando no diretório */CDRW-ISO*, vamos executar o script, passando os parâmetros certos, da seguinte maneira:

```
[root@localhost /CDRW-ISO]# make_iso <dir_base>
<nome_da_imagem> <nome_CD>
```

No nosso exemplo, digamos que quero copiar meu CD 1 do **Conectiva Linux 6.0** (é totalmente possível e legal fazer isto):

```
[root@localhost]# make_iso /mnt/cdrom/ linux.img linux
```

Isto quer dizer que vou criar um arquivo chamado *linux.img* no diretório */CDRW-ISO* a partir do diretório */mnt/cdrom/*, que será meu CD-ROM montado com o CD 1 do **Conectiva Linux 6.0** e, quando eu mandar gravar o CD, seu label ficará com o nome “linux”. Agora, é só aguardar e ele criará a imagem. Quando o prompt retornar, é porque esta é a saída padrão. Algo como:

```
[root@snake /tmp]# make_iso /tmp/ tmpimage.img exemplo
mkisofs: Option -a is obsolete. All files are included
by default.
Total extents actually written = 2065
Total translation table size: 12007
Total rockridge attributes bytes: 27770
Total directory bytes: 81920
Path table size(bytes): 686
Max brk space used 37864
2065 extents written (4 Mb)
[root@snake /tmp]#
```

Claro que este é um exemplo, mas serve como base. Criei uma imagem de 4MB no meu */tmp* que se chama *tmpimagem* e se eu gravar um CD com esta imagem, o CD chamar-se-á *exemplo*. Pronto. Agora com a imagem em mãos podemos gerar nosso disco, usando outro script, o *run_cdrw*, que passarei a explicar mais abaixo.

Run_cdrw

Este script é simples e auto-explicativo, é só ter o cuidado de desmontar o *CDR/CDRW* e depois ir no diretório onde a imagem criada está. Siga o exemplo:

```
[root@localhost]# cd /CDRW-ISO
[root@localhost /CDRW-ISO]# umount /mnt/cdrw
[root@localhost /CDRW-ISO]# run_cdrw
```



Sairá uma lista e logo uma mensagem para você escolher qual CD deve escolher. Escolha o número do *CDR/CDRW*, caso tenha mais de um CD na máquina; caso só tenha um, escolha a única opção, coloque que está após a mensagem “Selecione um Gravador de CD”.

Caso escolha o número errado ou este não for um *CDR/CDRW*, o próprio programa voltará, solicitando uma outra opção, pois aquela é incorreta.

Logo será solicitado o nome da imagem da qual se quer gerar o disco, então, coloque o seu nome: *linux.img*.

Logo solicitará a velocidade com a qual você deseja gravar. Isto vai depender da sua gravadora e da mídia que você estiver usando. No meu caso, minha gravadora suporta gravar em 8x e a mídia também. Colocarei, então, para gravar em 8x:

Velocidade de gravação [0/2/4/8/10]: 8

Após isso, o comando será mostrado, verificado e uma mensagem de configuração aparecerá. Deve colocar *yes* ou *no*, tal como está entre colchetes []:

Verificando comando:

```
cdrecord -v speed=8 dev= linux.img
```

Você tem certeza que deseja fazer isto ? [No/yes] yes

O CD começará a ser gerado. É interessante não mexer na máquina enquanto grava, mesmo que os equipamentos mais avançados tenham buffer e o sistema também, pode ser que algum dos aplicativos venha a prejudicar a gravação e você perca uma mídia, o que não é muito agradável.

Editando o PATH

Para vermos os diretórios que estão no PATH, digite:

```
echo $PATH
```

Para adicionar um novo diretório ao PATH, faça o seguinte:

```
PATH=$PATH:/novo_diretório
```

Este comando é válido somente durante a sessão para colocar o path permanentemente no seu PATH particular. Edite o arquivo *.bash_profile* localizado no seu diretório pessoal e acrescente o diretório na linha PATH.



Já para torná-lo global, ou seja, disponibilizar para todos os usuários, coloque estas configurações na linha PATH do arquivo */etc/profile*.

Fazendo o BackSpace Funcionar

Após a instalação do Gnome ou até mesmo de uma distribuição como a RedHat, às vezes a tecla BackSpace não funciona ou deixa de funcionar (ocorrem com teclados us-acentuado ou abnt).

Existe um conjunto de programas que pode ser muito útil, como *showkey*, que mostra qual o código da tecla digitada; *dumpkey*, que mostra o mapa do teclado corrente; *loadkey*, que carrega o teclado especificado.

Uma outra forma é criando um arquivo qualquer chamado *BackS*, por exemplo, e colocar o seguinte dentro dele:

```
keycode 22 = BackSpace
```

Salve o arquivo e digite em qualquer Shell:

```
xmodmap /diretorio_do_arquivo/BackS
```

Ou ainda:

```
xmodmap -e "Keycode 0x16 = BackSpace"
```

Como descobrir em qual pacote determinado arquivo está? No seguinte endereço:

```
http://www.conectiva.com.br/~suporte-cl/programas
```

Existe o programa *procura_rpm.sh* que verifica de qual pacote *rpm* que o arquivo faz parte. Baixe o arquivo para um diretório de sua preferência e transforme-o em executável com o seguinte comando:

```
[root@localhost]# chmod +x /tmp/procura_rpm.sh
```

Substitua */tmp* pelo diretório no qual o arquivo se encontra.

Caso o CD-ROM ainda não esteja montado, monte-o:

```
[root@localhost]# mount /mnt/cdrom
```

Acesse o diretório das RPMs:

```
[root@localhost]# cd /mnt/cdrom/conectiva/RPMS
```

Execute o comando de verificação, por exemplo, para descobrir qual o pacote que o arquivo *libc.so.5* está:



```
[root@localhost]# /tmp/procura_rpm.sh libc.so.5
```

Novamente, lembre-se de alterar o diretório */tmp* pelo diretório em que o programa se encontra.

Informações sobre o Sistema

Para obter informações sobre o sistema, acesse o diretório */proc*.

Para ver informações sobre:

Item	Veja
Cpu	cat cpuinfo
Memória	cat meminfo
Partições	partition
Módulos	modules
PCI	pci
Diretórios	mount

Linux com Windows 2000

Segue abaixo uma receitinha para rodar Windows 2000 (NTFS) e **Linux** na mesma máquina.

1. Primeiramente, instale o Windows.
2. Depois, instale o **Linux**. Quando for instalar o *lilo*, não instale na MBR, instale-o na mesma partição onde está a distribuição.
3. Não esqueça de criar o disco de boot quando estiver instalando o **Linux** e for solicitado.
4. Dê boot na máquina com o disquete de boot criado na instalação do **Linux**.
5. Após iniciar o **Linux**, pegue um novo disquete: vamos copiar o superbloc onde está *lilo*.
6. Digite no prompt o seguinte comando:

```
dd if=/dev/hda2 of=/mnt/floppy/bootsect.lnx bs=512  
count=1
```

7. Substitua o */dev/hda2* pela partição onde se encontra o seu **Linux**. Será criando um *bootsect.lnx* no floppy.



8. Dê outro boot e entre no Windows. Edite o arquivo *c:\boot.ini*.
9. Onde estiver *c: \ "MS-DOS"* e coloque *c:\bootsect.lnx = "Start Linux"*.
10. Copie o arquivo do disquete *bootsect.lnx*, para o *C:*.
11. Reinicie o computador e escolha a opção *Start Linux*.

Renomear Nomes de Arquivos de Maiúsculas para Minúsculas

Para renomear todos os arquivos de um diretório de maiúscula para minúscula:

```
for file in * ;do mv $file `echo $file > tr [:upper:] [:lower:]` 2>/dev/null; done;
```

Reiniciar Downloads

Para downloads grandes e que possam reiniciar, use o *wget*:

```
wget ftp.dominio.com.br/arquivo.zip
```

Caso queira reiniciar um download começado, digite:

```
wget -c ftp.dominio.com.br/arquivo.zip
```

Existe uma interface gráfica para o mesmo, chamada *kaitoo*.

Ativando o Numlock na Inicialização

Insira o script abaixo no arquivo *rc.local*.

```
INITTY=/dev/tty[1-8]
for tty in $INITTY; do
    settleds -D +num < $tty
done
```

Dicas sobre Pacotes RPM

Para verificar se um pacote está instalado, digite o seguinte comando:

```
rpm -qa | grep -i <pacote>
```



Para listar os arquivos contidos em um pacote RPM:

```
rpm -ql <nome_do_pacote>
```

Para listar os arquivos de um pacote ainda não instalado:

```
rpm -qlp <nome_do_arquivo.rpm>
```

Para obter informações de um pacote instalado em seu computador:

```
rpm -qi <nome_do_pacote>
```

Para obter informações de um pacote ainda não instalado em seu computador:

```
rpm -qip <nome_do_pacote.rpm>
```

Fontes TrueType no Linux

Instale o pacote *Freetype*. Ele pode ser encontrado no endereço <http://www.freetype.org> ou no próprio CD-ROM da Distribuição.

Crie o diretório onde as fontes serão instaladas.

```
mkdir /usr/X11R6/lib/X11/fonts/TrueType
```

Copie as fontes *truetype* do Windows para este diretório que foi criado. Se você possui Linux e Windows, monte a partição e faça como no exemplo. Caso contrário, terá que compartilhar na rede através de um servidor Samba (procure no capítulo 13 para maiores detalhes sobre o Servidor Samba).

```
cp /Windows/fontes_do_Windows/*.ttf /usr/X11R6/lib/X11/fonts/TrueType
```

Em */Windows/fontes_do_Windows*, altere pelo caminho real montado ou pela disponibilização dos pacotes via rede.

Crie uma lista de fontes. Esta lista será usada como referência por todos os programas do XFree do **Linux**:

```
cd /usr/X11R6/lib/X11/fonts/TrueType
ttmkfdir -o fonts.scale
mkfontdir
```

Adicione o caminho para o local onde estão as fontes:

```
chkfontpaht -add /usr/X11R6/lib/X11/fonts/TrueType
```

Reinicie a máquina e pronto... as alterações já estão feitas.



Login em Modo Texto ou Gráfico

Com uma simples configuração, pode-se alterar a forma como o **Linux** inicia, se em modo gráfico ou modo texto.

Esta configuração é feita através do arquivo */etc/inittab*. Procure por uma linha:

```
id:5:initdefault
```

Neste caso acima, a máquina iniciará em modo gráfico. Para fazê-la iniciar em modo texto, mude o 5 para 3:

```
id:3:initdefault
```

Disco de Boot

Para criar um disco de boot no **Linux**, insira um disquete formatado no drive e digite:

```
lilo -b /dev/fd0
```

Com este comando, será gravado o seu lilo atual no disquete. Com esse disquete, é possível iniciar a máquina utilizando-o.

