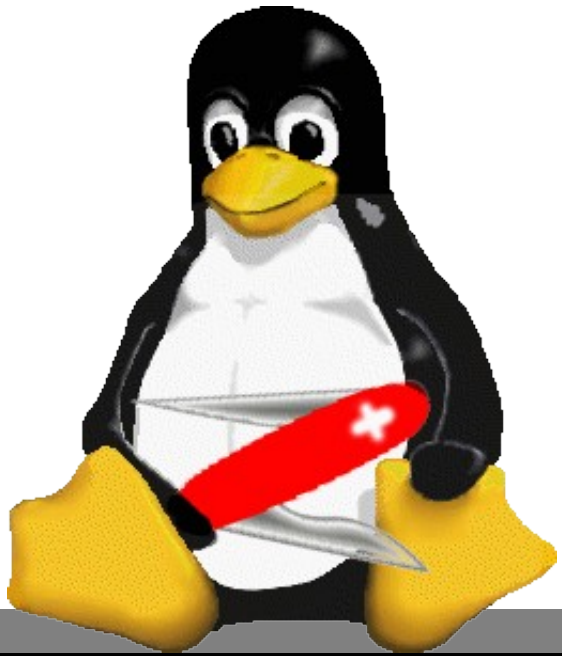


Linux
Básico, Gerência, Segurança e Monitoramento de Redes

Odilson Tadeu Valle Fevereiro de 2009

Prefácio

Esta apostila pretende atender as demandas curriculares de algumas disciplinas dos cursos na área de telecomunicações da Unidade São José do Centro Federal de Educação Tecnológica de Santa Catarina.
Nela são abordados dezenas de tópicos desde a parte introdutória até a parte mais avançada na administração de serviços de rede usando o Linux. A relação de tópicos abordados é extensa, mas não pretendemos nos aprofundar muito em cada um deles, para isto existem literaturas específicas, muitas vezes abordando um único tópico.
Gostaríamos de salientar também que toda a parte de instalação e configuração está escrita em cima dos procedimentos necessários para a distribuição Mandriva 2008, portanto, dependendo da distribuição adotada, podem ser necessários ajustes.
Alguns tópicos foram escritos de próprio punho e em outros os textos foram coletados da Internet e adaptados, procuramos ser fiéis nas referências, mas peço desculpas antecipadas caso tenha omitido alguma.

É isto!
Odilson Tadeu Valle
Fevereiro de 2009

Sumário

Linux Básico

Gerência de Redes

Segurança e Monitoramento de Redes

Linux Básico

1 Sistema Operacional

O Sistema Operacional é um programa especial que gerencia todos os recursos da máquina, tais como memória, teclado, vídeo (monitor), mouse, entre outros. É através do Sistema Operacional que executamos outros programas, gravamos ou lemos informações em disquetes, visualizamos textos em vídeo ou impressora, etc. Sem o Sistema Operacional não conseguiríamos realizar estas tarefas. Ou seja, simplesmente não poderíamos utilizar o computador.
Existem inúmeros Sistemas Operacionais, tais como: MS-DOS, UNIX, OS/2, VM/CMS, QNN, etc. Cada um deles possuem características próprias e são executados em máquinas diferentes. Assim, não podemos executar um programa em Sistemas Operacionais distintos, a não ser que o fabricante do programa nos garanta esta portabilidade.
É de responsabilidade do Sistema Operacional:

- Carregar e executar programas.
- Controlar dispositivos de entrada e saída (teclado, monitor, mouse, etc).
- Gerenciar arquivos e diretórios.
- Gerenciar a memória RAM

Todo e qualquer programa executado em um computador utiliza a memória RAM. Da mesma forma, o Sistema Operacional deve ser carregado, ou seja, copiado do disco rígido ou disco flexível para a memória RAM. Denominamos este processo de BOOT. Toda vez que ligamos o computador, é feita uma série de testes para verificar o funcionamento dos periféricos e se tudo estiver perfeito, o Sistema Operacional pode ser carregado.
Os Sistemas Operacionais ainda podem ser classificados quanto ao número de pessoas que podem utilizar os recursos ao mesmo tempo e quanto ao número de programas que podem ser executados em uma mesma máquina.

1. **Monousuário**: permitem apenas um usuário.
2. **Multiusuário**: permitem vários usuários.
3. **Monotarefa**: apenas um programa pode ser executado de cada vez.
4. **Multitarefa**: vários programas podem ser executados ao mesmo tempo.

Em geral Sistemas Operacionais que são multiusuários são também multitarefa, como o UNIX, QNN e atuais versões do Windows, onde podemos ter vários usuários em terminais distintos executando, cada um, uma série de programas diferentes ao mesmo tempo.
Além disto, Sistemas Operacionais podem ser classificados quanto ao tipo de comunicação com o usuário, podendo ser:

1. **Interface por linha de comando**: quando o usuário tem que digitar o comando por extenso na tela do computador. A comunicação, em geral

é feita em modo texto. Preferencialmente utilizada por especialistas.
2. **Interface gráfica para usuários (GUI)** : quando os comandos são executados em um ambiente gráfico com o uso do mouse. Voltada principalmente para o usuário final.

2 Linux

2.1 Histórico

O Sistema Operacional UNIX foi desenvolvido nos laboratórios da AT&T para uso próprio, baseado em um antigo projeto que deveria ser o primeiro Sistema Operacional multiusuário e multitarefa, o MULTICS. Porém, este projeto estava muito além da capacidade dos equipamentos para a época. Desta forma o projeto foi arquivado, mas alguns de seus idealizadores (Ken Thompson, Dennis Ritchie e Rudd Canadaday) resolveram escrever uma versão simplificada e monousuário para um computador com menores recursos. O resultado impressionou, mesmo sendo utilizada uma máquina limitada.
Assim, o código foi reescrito para outros computadores melhores, apresentando excelentes resultados. Por coincidência ou não, estes computadores para os quais o Sistema Operacional foi reescrito eram utilizados por quase todas as Universidades que se interessaram por este Sistema Operacional muito superior aos que vinham sendo utilizados nos laboratórios de computação.
A partir de então, a AT&T licenciou seu mais novo projeto para as Universidades, mostrando uma enorme visão e capacidade inovadora, pois além do Sistema Operacional, foi cedido o código do mesmo para as Universidades, que não mediram esforços em depurar o programa e incluir novas características.
Foi dentro das Universidades que o UNIX cresceu e adquiriu muitas das características que o tornam poderoso, dando origem a diversas versões além da original proveniente dos laboratórios da AT&T. Esta característica tornou o UNIX um sistema poderoso na medida em que foi concebido não apenas por uma equipe de projetistas, mas sim por toda uma comunidade de pessoas interessadas em extrair o melhor das máquinas. A princípio, o código do UNIX foi escrito em linguagem assembler ou de máquina que é altamente dependente do hardware ou parte física do computador. Para que o código fosse reescrito, era necessário muito esforço e tempo.

Entretanto, um dos criadores do Sistema Operacional UNIX resolveu utilizar uma nova linguagem para escrever o UNIX, era a linguagem C que oferecia o poder da linguagem de máquina com a facilidade das linguagens estruturadas de alto nível.

A grande vantagem de se utilizar a linguagem C ao invés da linguagem de máquina própria do computador é a de que a primeira é altamente portável, isto é, um programa escrito em C para um determinado computador poderá ser executado quase sem nenhuma modificação em

outro tipo de máquina completamente diferente. Enquanto que se fosse feito um programa em linguagem de máquina para um determinado computador o programa seria executado somente neste tipo de computador e não nos demais, para isto, seria preciso reescrever todo o programa.
O UNIX foi projetado para ser executado em computadores de grande capacidade, ou seja, mini e supercomputadores, pois somente estas máquinas podiam oferecer suporte aos recursos necessários para o ambiente gerado pelo Sistema Operacional.
Crescendo longe do alcance dos usuários de microcomputadores, o UNIX atingiu uma estabilidade e estrutura jamais alcançada por um Sistema Operacional. Mas nestes quase quarenta anos de existência do UNIX, os microcomputadores evoluíram a ponto de fornecer o mínimo de condições para que este poderoso Sistema Operacional pudesse ser implementado para os micros IBM -PC e compatíveis.
Diversas versões do UNIX foram escritas e licenciadas para venda com nomes semelhantes (XENIX, UNISYS, AIX, etc) porém com as mesmas características essenciais, sendo que atualmente existem inúmeras versões comerciais e outras tantas versões livres que foram desenvolvidas em Universidades ou por *hackers* através da rede Internet.
Apesar de ter sido desenvolvido para lidar com dispositivos de caracteres, UNIX foi pioneiro na área de gráficos em estações de trabalhos. As primeiras interfaces gráficas para usuários (GUI) foram projetadas e utilizadas em Sistemas operacionais UNIX, desenvolvidas pelo MIT (*Massachussets Institute of Technology*). Trata-se do *X Window System*.
Como se pode notar, UNIX é um sistema de inúmeras possibilidades. Praticamente todos os recursos que os sistemas operacionais mais atuais utilizam já haviam sido executados em UNIX há muito. Todas as áreas da computação puderam ser desenvolvidas com o UNIX.
As tendências atuais levam a uma tentativa de padronizar o Sistema Operacional UNIX combinando as melhores características das diversas versões do mesmo. Prova disto é a criação do POSIX, um padrão de Sistema Operacional desenvolvido pela IEEE (*Institute of Eletrical and Eletronic Engineers*). Além da OSF (*Open System Fundation*) que reúne as principais líderes do mercado de equipamentos para definir o padrão de GUI (interfaces gráficas) para UNIX.
A versão que será abordada durante este curso é o LINUX, um clone do Sistema Operacional UNIX para microcomputadores IBM -PC 386 e compatíveis. O LINUX foi desenvolvido inicialmente por Linus Torvalds na Universidade de Helsinski na Finlândia.
O LINUX possui a vantagem de ser um software livre e ser compatível com o padrão POSIX. Além de unir em um único Sistema Operacional as vantagens das diferentes versões de UNIX comerciais disponíveis. Desta forma, LINUX torna -se a melhor opção para que usuários de microcomputadores possam usufruir da capacidade do UNIX.
Apesar de não poder rodar aplicativos para MS- DOS, o LINUX pode rodar todos os softwares de desenvolvidos para UNIX, além de estarem disponíveis softwares que permitem a emulação do MS-DOS e do WINDOWS.
O LINUX pode ser útil em empresas que desejam possuir estações de

trabalho com poder razoavelmente comparável às estações existentes como SUNs e outras usando PCs, com fiel semelhança no seu uso.
O LINUX pode conviver pacificamente com outros sistemas operacionais no PC. Existe uma infinidade de formas de instalá-lo: em uma partição DOS já existente, pode ainda ser instalado em um HD exclusivamente dedicado a ele.
Para conviver com outros sistemas operacionais, existem algumas maneiras de carregar o sistema operacional, o Lilo (Linux Loader) que pode funcionar como um *BOOT manager* no qual se escolhe qual partição ou drive irá dar a partida, o loadlin que é um utilitário DOS para carregar o LINUX a partir do DOS, ou por meio de um disco de boot.
O LINUX pode ser obtido de diversas formas diferentes, existem diversos livros à venda, os quais incluem CDs com distribuições do LINUX. Outra forma de obtê-lo inteiramente grátis e via ftp pela INTERNET.
Existe hoje, um movimento no sentido de tornar o LINUX um sistema popular, dado que superioridade técnica ele já possui.
Existem algumas outras versões de UNIX para PCs, tais como Xenix, SCO Unix, FreeBSD e NetBSD, as últimas duas também livres, no entanto além de mais popular, o LINUX possui uma série de características a mais, não encontradas em outras versões, mesmo comerciais, de UNIX.

2.2 Uma visão geral do LINUX

Um Sistema Operacional deve gerenciar os recursos da máquina da melhor maneira possível de forma a poder oferecer aos usuários o máximo do computador. Dentre as principais funções do sistema Operacional, podemos destacar:

- Criar e manipular uma estrutura de arquivos e diretórios.
- Controlar o acesso à memória e outros dispositivos controlados pelo microprocessador, como monitor, teclado, impressora, etc.
- Gerenciar a execução de programas, trazendo-os da memória para o microprocessador.

A primeira vista, parece que o LINUX nada possui de diferente de qualquer outro Sistema Operacional, mas nenhum é tão bom em unir e integrar o que há de melhor em um computador de forma harmoniosa e eficiente devido a sua própria origem em meio a toda uma comunidade de pessoas interessadas em obter o máximo e o melhor em desempenho. Cabe ressaltar também que o Linux possui todas as características que fazem do UNIX um excelente sistema operacional, entre elas : **Portabilidade, Multiusuário e Multitarefa, Estrutura hierárquica de arquivos, Ferramentas e Utilitários, Comunicação com outros sistemas.**
Daremos uma rápida olhada em algumas das principais características e vantagens que fazem o LINUX único :
Multitarefa. Linux, como as outras versões do UNIX é um sistema multitarefa, possibilitando a execução de múltiplas aplicações de diferentes usuários no mesmo sistema ao mesmo tempo.
O X Window System é, de fato, um padrão na indústria de sistemas gráficos para máquinas UNIX. Uma versão completa do X Window System, conhecida

como *Xfree86* está disponível para Linux.
TCP/IP (Transmission Control Protocol / Internet Protocol), este é um conjunto de protocolos que liga milhões de universidades e empresas numa rede mundial conhecida como **Internet.** Com uma conexão Ethernet o Linux permite que seja feita uma conexão da Internet a uma rede local.
Memória Virtual. O Linux pode usar parte do seu HD como memória virtual, "aumentando" assim a capacidade da memória RAM.
Compatibilidade com o IEEE POSIX. Linux foi desenvolvido com a portabilidade de software em mente.

2.3 A estrutura do LINUX

2.3.1 Kernel/Shell

Kernel é o núcleo do Sistema Operacional LINUX, que permanece residente na memória. Através dele que o usuário possui o acesso aos recursos oferecidos pelo hardware (o computador em si). Todo o gerenciamento de memória, dispositivos, processos, entre outros é coordenado pelo kernel. Basicamente está dividido em duas partes:

- *Gerenciamento de dispositivos*: supervisiona a transmissão de dados entre a memória principal e os dispositivos periféricos. Desta forma, o kernel abrange todos os drivers controladores de dispositivos que podem ser ligados a um computador
- *Gerenciamento de processos*: aloca recursos, escalona processos e atende a solicitação de serviços dos processos

Shell é o interpretador de comandos do LINUX. É ele quem fornece uma interface para que o usuário possa dizer ao Sistema Operacional o que deve ser feito. O shell traduz o comando digitado em chamadas de sistema que são executadas em linguagem de máquina pelo kernel. Além disto, fornece um ambiente programável através de *scripts*.
Existem inúmeros shells cada um com ligeiras diferenças entre si. Muitas vezes é possível utilizar vários shells diferentes em um mesmo micro rodando LINUX, isto porque ele é multitarefa e multiusuário, de modo que cada usuário poderia utilizar o shell que lhe agradar mais. Entre os mais utilizados estão o Bourne Shell, o C shell e o Korn Shell.

2.3.2 Utilitários

Existem centenas de utilitários (comandos) para a realização de tarefas especializadas ou rotineiras, entre elas manipulação e formatação de textos, cálculos matemáticos, gerenciamento e manutenção de arquivos e diretórios, administração de sistemas, manutenção de segurança, controle da saída para impressora, desenvolvimento de programas e filtragem de dados.
Cada um destes utilitários é digitado na linha de comando do LINUX que será interpretado pelo Shell do sistema. Este por sua vez se encarregará de realizar diversas chamadas ao Kernel para a execução do comando.
Como as interfaces gráficas são muito recentes, o LINUX teve toda a sua

potencialidade explorada em termos de ambiente de desenvolvimentos. Isto equivale a dizer que o que se pode fazer com um software de Formatação de Textos do tipo aponte e clique, pode ser realizada através do antigo modo de linha de comando no LINUX.
Mas isto não impede que as facilidades do ambiente de janelas seja explorado, pelo contrário. Os mais profissionais programas aplicativos rodam sobre o Sistema Operacional LINUX, entre eles o Gerenciador de Banco de Dados ORACLE, INGRES e FoxBASE+, formatadores de textos Postscript etc.

3 Processos

Quando um programa ou utilitário é executado, passa a se chamar processo. Cada processo iniciado possui um estado indicando sua condição (em execução, parado, interrompido, etc) e a prioridade. Sendo que os processos do sistema possuem prioridades sobre os do usuário.
Com base nas informações sobre os processos em andamento, a CPU precisa escalonar os processos para dedicar a cada um, um determinado tempo dando a impressão de que vários processos estão sendo executados ao mesmo tempo.
Para vermos uma “fotografia” dos processos rodando na máquina podemos usar o comando **ps aux**, ax mostra todos os processos e u informa a mais os usuários donos dos processos.
Podemos usar também o **top**. Neste caso haverá atualização periódica da tela, fazendo uma amostra *on-line* processos ativos. Mostra ainda outras informações da máquina, como uso de memória, tempo de atividade, uso de cpu etc. Para navegar entre janelas usa-se as teclas <>.
Para matarmos um processo em execução usamos o comando **kill** seguido do número do processo (PID). A principal flag é o -9, que mata o processo sem salvar dados da memória, se existirem. Podemos usar também o **killall** seguido do nome do processo (comando). Neste caso mata-se todos os processos com mesmo nome.

4 Comandos básicos

4.1 Introdução

Certos comandos são *interativos* e outros *não-interativos*. Comandos interativos são aqueles que após serem executados, exigem que algumas perguntas sejam respondidas para que possam prosseguir. Comandos não interativos simplesmente executam os comandos sem nada perguntar e retornam à linha de comando do LINUX.
Exemplos de comandos não interativos:

```
ls    exibe lista do conteúdo do diretório corrente

date  exibe a data e hora do sistema

cal <ano>    exibe calendário do ano especificado
```

```
who   exibe lista de todos os usuários ativos no sistema

clear limpa a tela
```

Exemplos de comandos interativos:

```
passwdmodifica a senha

ftp   permite transferência de arquivos
```

4.2 Ciclo de Execução do Comando

O *shell* analisa a linha do comando separando seus vários componentes com o uso de espaços em branco. Este procedimento é conhecido como *parsing* (análise), e é composto dos seguintes passos:

1. O *shell* examina se há algum caractere especial a ser interpretado na linha de comando;
2. Supondo que os caracteres até o primeiro branco se referem a um comando, o *shell* procura um arquivo executável (programa) com o mesmo nome;
3. Se o *shell* localiza o programa, ele verifica se o usuário que fez o pedido tem permissão de acesso para usar o comando;
4. O *shell* continua a examinar o resto da linha de comando para ver a formatação;
5. Finalmente, ela informa ao *kernel* para executar o programa, passando todas as opções e argumentos válidos para o programa;
6. Enquanto o *kernel* copia o arquivo executável do disco para a memória e executa-o, o *shell* permanece inativo até que o programa tenha encerrado. O programa em execução na memória é chamado de *processo*;
7. Quando o processo termina de ser executado, o controle retorna ao *shell* que exibe novamente o *prompt* para avisar que está pronto para o próximo comando;

4.3 Login

Por ser um Sistema Operacional que suporta vários usuários (multiusuário), antes de tudo, é preciso se identificar. O LINUX então se encarregará de permitir ou não seu acesso verificando sua senha, se estiver correta libera o diretório de entrada e executa arquivos de inicialização locais e o interpretador apropriado. Após este processo, você estará apto a executar os comandos do LINUX.
Quando o terminal estiver ligado, provavelmente será apresentado um sinal de prontidão do sistema da seguinte forma:

```
Login:
```

Isto significa que o sistema está esperando para que o usuário se identifique com o nome de usuário que lhe foi concedido pelo Administrador de Sistema junto com uma senha de acesso. Após digitar o nome de usuário,

pressione ENTER. Será apresentada um novo sinal de prontidão:

```
Password:
```

Este sinal pede que seja digitada a sua senha. Note que a medida que forem digitados os caracteres, eles não aparecerão no vídeo por medidas de segurança.
Se algo deu errado (foi novamente apresentado o sinal de login), tente novamente, certificando-se de ter digitado o nome de usuário e a senha exatamente como recebeu do Administrador pois o LINUX diferencia as letras maiúsculas das minúsculas. Isto quer dizer que para o LINUX **A** (letra "a" maiúscula) é diferente de **a** (letra "a" minúscula). Esta é uma dica que serve não apenas para iniciar a sessão, mas também para todos os comandos LINUX.
Tendo o usuário se identificado com o nome da conta e a senha (se esta existir, pois existem contas criadas especialmente para uso sem senha), o LINUX checa em um arquivo de configuração pelo nome da conta e a senha correspondente devidamente encriptada. Estando ambas registradas e corretas, o Sistema Operacional permite o acesso ao usuário executando o shell indicado também neste arquivo.

O shell providencia uma interface de comunicação entre o kernel e o usuário. Esta interface consiste de uma *linha de comando* (ou *prompt*) na qual deve ser digitado o comando por extenso seguido por seus parâmetros (se tiver). Em uma linha de comando podemos ter mais de um comando em seqüência para serem executados.

Em geral esta linha de comando é formada por um símbolo que pode ser de porcentagem (%) ou cifrão ($) para usuários comuns e grade (#) para usuários com privilégio de raiz, dependendo do tipo de shell usado.

Os parâmetros que aparecem após o nome do comando podem ser nomes de arquivos e/ou caminhos de diretórios. Eles devem sempre ser digitados com um espaço entre si e depois do comando. Muitas vezes alguns símbolos que aparecem na linha de comando não são parâmetros, mas sim comandos para o shell determinando a seqüência em que o(s) comando(s) devem ser executados.

O LINUX aceita e executa um comando quando, ao terminarmos de digitarmos o comando, pressionarmos a tecla ENTER, RETURN ou ↵ (varia de computador para computador).

Caso seja encontrado algum erro na digitação do comando antes que a tecla ENTER seja pressionada, podemos corrigí-lo utilizando as teclas de direção e para posicionarmos o cursor na posição em que o erro foi cometido. Cursor é o símbolo gráfico que aparece logo após a linha de comando e que se movimenta a medida em que caractere são digitados e aparecem na tela.

4.4 Logout

Este comando permite sairmos de nossa seção shell, ou seja, desconectarmos

nosso usuário do sistema.

4.5 Reboot

Este comando é equivalente à “init 6” e com ele podemos reiniciar nosso sistema, sem desligamento do hardware.

4.6 Halt

Este comando é equivalente à “shutdown -r now” e permite desligar o sistema, caso nossa máquina tenha fonte ATX.

4.7 Man

Páginas de manual ou **man pages** são pequenos arquivos de ajuda que podem ser invocados pelo comando **man** a partir de linha de comando de sistemas baseados em Unix e Linux.
A forma de invocar a ajuda é:

```
$ man [<seção>] <nome-da-página>
```

5 Estrutura de Arquivos e Diretórios

Existem 4 tipos básicos de arquivos em LINUX :

- Arquivo diretório;
- Arquivo convencional;
- Arquivo de dispositivo;
- Arquivo simbólico ou de ligação;

Um **arquivo diretório** nada mais é do que um tipo de arquivo contendo informações sobre arquivos que conceitualmente (e não fisicamente) estão contidos nele. Isso significa que o conteúdo de seus arquivos não está armazenados dentro do diretório. Assim sendo, não há limite para o tamanho de um diretório. Teoricamente você poderia colocar no seu diretório tantos arquivos quanto quisesse, até o ponto de estourar a capacidade do seu disco.
Os dados contidos no arquivo diretório são apenas o nome de cada arquivo e seu ponteiro para uma tabela de informações de controle de todos os arquivos do sistema. Esta tabela contém informações administrativas do arquivo, como dados de segurança, tipo, tamanho, datas de acesso e dados que indicam onde ele está gravado no disco.
Quando você vai usar um arquivo, o sistema operacional consulta o diretório para verificar se existe no disco um com o nome que você especificou. Em caso afirmativo, o sistema obtém, da tabela as informações necessárias para poder manipulá- lo. Caso contrário, o sistema envia uma mensagem informando que não foi possível encontrar o arquivo.
Um diretório pode conter outros diretórios, aos quais chamamos subdiretórios. Um subdiretório pode conter outros arquivos e subdiretórios, que também podem conter arquivos e subdiretórios e assim por diante. Este é um relacionamento pai/filho entre um diretório e seus

arquivos e diretórios subordinados. Cada diretório pai guarda informações sobre os arquivos e diretórios que estão a um nível abaixo dele-seus filhos.
Um **arquivo convencional** é um conjunto de caracteres presentes em algum meio de armazenamento, como por exemplo um disco. Ele pode conter texto para uma carta, código de programa ou qualquer informação armazenada para um futuro uso.
Um **arquivo de dispositivo**, como um diretório, não contém dados. Ele é basicamente um ponteiro para um dispositivo periférico, como por exemplo uma unidade de disco, um terminal ou uma impressora. Os arquivos especiais associados aos dispositivos periféricos estão localizados no diretório /dev.
Um **arquivo simbólico** é um arquivo convencional que aponta para outro arquivo em qualquer lugar do sistema de arquivos LINUX.

5.1 Diretórios

Todos os arquivos fazem parte de algum diretório. Assim, eles são mantidos organizadamente. Se todos os arquivos do sistema fossem armazenados em um mesmo lugar, o LINUX levaria muito tempo para verificar todos os arquivos até encontrar aquele que está procurando. Os diretórios são um meio de oferecer endereços dos arquivos, de maneira que o LINUX possa acessá-los rápida e facilmente.
Ao entrar pela primeira vez em sua conta, você já está em um subdiretório do sistema LINUX, chamado seu diretório de entrada (*home directory*). A menos que você crie alguns subdiretórios em sua conta, todos os seus arquivos serão armazenados em seu diretório de entrada. Teoricamente, você pode fazer isso, mas a manutenção de seus arquivos será mais eficiente se você criar seu próprio sistema se subdiretórios. Assim ficará mais fácil manter o controle de seus arquivos porque eles estarão agrupados em diretórios por assunto ou por tipo. O LINUX também realiza buscas de maneiras mais eficiente em diretórios pequenos que nos grandes.

5.1.1 Diretório de Entrada

Seu diretório de entrada é aquele em que você é colocado quando abre uma sessão em um sistema LINUX. Esse diretório tem o mesmo nome que seu nome de login. Você pode pensar em sua conta como uma versão em miniatura do sistema de arquivos do LINUX. No alto de seu sistema pessoal de arquivos, em vez do diretório raiz, está seu diretório de entrada. Abaixo dele estarão os subdiretórios que você criar, que podem, por sua vez, se ramificar em subdiretórios e/ou arquivos.
Os diretórios de entrada dos usuários são iguais a qualquer outro diretório de um diagrama de sistema de arquivos. Entretanto, sendo o diretório principal de sua conta, seu diretório de entrada tem um status especial. Sempre que você entra no sistema, o LINUX define uma variável chamada HOME que identifica o seu diretório de entrada. O LINUX usa o valor da variável HOME como ponto de referência para determinar quais arquivos e diretórios do sistema de arquivos você pode acessar e também para orientar-se para onde

levá-lo quando você deseja mudar de diretório corrente.

5.1.2 Diretórios Corrente

O diretório corrente, ou de trabalho (*working directory*), é o diretório em que você está em um determinado momento. Por exemplo, quando você entra no sistema, o diretório corrente é sempre seu diretório de entrada. Se você passar para um de seus subdiretórios, este passará a ser o diretório corrente.
Durante toda a sessão, o LINUX mantém o controle de seu diretório corrente. Todos os comandos são executados sobre o diretório corrente, a menos que você especifique outro. Por exemplo, qualquer arquivo ou subdiretório que você criar será em princípio criado no diretório corrente. Sempre que você digitar ls, verá uma lista dos arquivos e diretórios do diretório corrente.
Todos os diretórios do LINUX contém um arquivo chamado **.** (ponto), que é um arquivo especial que representa o diretório corrente (um sinônimo). Sempre que você quiser se referir ao diretório corrente, pode fazê-lo usando um ponto (.). Outro arquivo especial, chamado .. (dois pontos) representa o diretório pai do diretório corrente (o diretório ao qual o diretório corrente pertence). Quando precisar se referir ao diretório pai do diretório corrente, você pode usar dois pontos (..) em vez do nome do diretório.
Quando você digita um comando que opera sobre um arquivo ou diretório, precisa especificar o nome do arquivo ou do diretório desejado. O caminho, de um arquivo ou diretório é a lista de todos os diretórios que formam a ligação entre ele e o diretório raiz.
Você só pode identificar individualmente cada arquivo e diretório por seu nome e caminho, porque seu nome pode ser idêntico ao de outro arquivo em outro local do sistema. Por exemplo, suponha que haja duas contas de usuário, chamadas luciene e alfredo, cada uma contendo um subdiretório chamado vendas. O LINUX pode diferenciar esses dois subdiretórios por seus caminhos. Um deles seria /.../luciene/vendas e o outro seria /.../alfredo/vendas, onde as reticências representam os diretórios intermediários.
Embora você possa se referir a um arquivo ou diretório dentro de seu diretório de entrada usando apenas seu nome, o LINUX sempre interpretará o nome do arquivo ou diretório como seu nome e caminho inteiro, porque ele mantém o controle de seu diretório corrente e pode preencher a parte do nome de caminho que falta.
Além do caminho absoluto, você também pode usar o caminho relativo, de um arquivo ou diretório. O **caminho relativo** não começa com o diretório raiz, mas com o diretório mais próximo do diretório cujo caminho está sendo definido. Para especificar um caminho relativo para seu diretório de entrada, você pode começar o caminho com
$HOME ou com um ~ (til), que é um sinônimo para $HOME. Por exemplo, se seu diretório de entrada é marco, a variável HOME terá o valor /.../marco, onde as reticências representam os diretórios entre o diretório raiz (/) e o diretório marco. Sempre que você digitar $HOME ou ~ como parte de um nome de caminho, o LINUX o interpretará como o nome de caminho

completo de seu diretório de entrada.
Para especificar um caminho a partir do diretório corrente, você pode iniciar o caminho com um . (que representa o diretório corrente), ou com o nome do primeiro subdiretório naquele caminho. O ponto é opcional neste caso porque se o nome de caminho não começar com uma /, o LINUX considera que você quer que ele comece com o diretório corrente.
Se você já tiver mudado de diretório algumas vezes, talvez não esteja seguro de qual é o diretório corrente no momento. Para descobrir, use o comando ls e poderá se lembrar do nome do diretório pela lista dos arquivos que ele contém. Entretanto, uma maneira mais simples de saber qual o diretório corrente é digitar pwd, que será apresentado mais adiante. O comando pwd imprime o caminho completo do diretório corrente.

5.2 Substituição do Nome do Arquivo

Três caracteres especiais permitem a referência a grupos de arquivos ou diretórios em uma linha de comando. Estes caracteres são chamados Meta caracteres ou Coringas.

5.2.1 Asterisco

O * substitui qualquer conjunto de caracteres.

5.2.2 Ponto de interrogação

O caractere ? substitui qualquer caractere.

5.2.3 Colchetes

O símbolo [] contém uma lista de caracteres. Um dos caracteres dentro do colchetes será substituído. Um hífen separando os caracteres que estão entre colchetes indica um intervalo. Um ! dentro do colchetes indica o sentido da procura invertido.
Esses caracteres especiais poupam tempo de digitação. O mais importante é que eles podem ser usados para fazer referência a arquivos cujo nome não se conhece exatamente.
Exemplos :
1- Liste todos os arquivos com extensão .new :

```
$ ls *.new

File.new arquivo.new
```

2- Liste todos os arquivos cujo nome termine com um numero entre 1 e 5 :

```
$ ls *[1-5]

file1

arquivo3

dir5
```

3- Liste todos os arquivos cujo nome tem três caracteres e começam com f :

```
$ ls f??

fig fin
```

5.3 Marcação do Caractere Especial

Para usar literalmente um caractere especial sem que o Shell interprete seu significado ele deve ser marcado. O shell trata um caractere especial marcado como um caractere normal.

5.3.1 Aspas

Quando se coloca um caractere especial entre aspas " " , o Shell ignora todos os caracteres especiais exceto o cifrão ($), o acento grave (') e a barra invertida (\);

5.3.2 Apóstrofe

O apóstrofe é mais restritivo. Todos os caracteres especiais entre apóstrofes são ignorados ;

5.3.3 Barra invertida

Geralmente, a barra invertida faz o mesmo que colocar um caractere entre apóstrofes. Quando uma barra invertida é usada, ela deve preceder cada caractere a ser marcado;

6 Manipulando Arquivos e Diretórios

6.1 Introdução

A função essencial de um computador é armazenar informações (arquivos) e catalogá-los de forma adequada em diretórios, fornecendo, se possível, algum esquema de segurança de modo que pessoas não autorizadas não tenham acesso a arquivos importantes.
Neste capítulo você aprenderá como manipular arquivos e diretórios no LINUX. Saber copiar, mover, exibir o conteúdo de um arquivo, e localizar um arquivo são algumas das atividades que veremos neste capítulo.
Os comandos aqui apresentados não são a totalidade dos comandos disponíveis, mas certamente são suficientes para que você consiga executar funções típicas e usuais de um programador ou de um usuário de aplicativos em ambiente LINUX.

6.2 Identificando o Diretório Corrente

pwd

O comando **pwd** (*print working directory*) não possui nenhuma opção ou

argumento. Este comando mostra o nome do diretório corrente ou de diretório de trabalho (*working directory*). Você pode utilizá-lo para se situar no sistema de arquivos. Por exemplo, é sempre útil verificar o diretório corrente antes de criar ou remover arquivos e diretórios. Do mesmo modo, o **pwd** é útil para confirmar o diretório corrente após várias trocas de diretórios.

6.3 Criando diretórios

mkdir

Nos diretórios podemos agrupar informações afins, isto é, arquivos que possuem alguma inter-relação. O nome do diretório deve ser significativo e permitir um acesso e uma localização rápida dos arquivos armazenados no seu sistema de arquivos.
O comando **mkdir** (*make directory*) é utilizado para criar diretórios. Os nomes dos diretórios a serem criados são passados como argumentos para o comando. Estes nomes podem ter até 255 caracteres, e devem ser únicos, isto é, não pode haver dois diretórios com mesmo nome dentro de um mesmo subdiretório, nem mesmo um arquivo e um diretório iguais em um mesmo subdiretório.
Opções:
-m modo Especifica o modo de permissão de acesso para o diretório que está sendo criado;
-p Cria os diretórios pai citados no nome do diretório que está sendo criado;
Exemplos :
1-Crie um diretório chamado teste com o seguinte modo de permissão 711

```
$mkdir -m 711 teste

$ls-l

total 1

drwx--x-- x 2 guest users 1024 May 15 21:27 teste/
```

2-Crie um diretório chamado curso com um diretório filho chamado aula1

```
$mkdir-p curso/aula1

$ls-l

total 2

drwx--x-- x 2 guest users 1024 May 15 21:27 teste/

drwxr- xr- x 3 guest users 1024 May 15 21:33 curso/

$ls-l curso total 1

drwxr- xr- x 2 guest users 1024 May 15 21:33 aula1/
```

6.4 Listando diretórios

ls [AaCFpdlmRrstucx] [nomes]
Normalmente o conteúdo de um diretório é listado em ordem alfabética, um item por linha. As diversas opções do comando ls permitem adaptar o formato da listagem.
Se nada for especificado em nomes todos os itens do diretório corrente são listados. Entretanto em nomes é possível determinar máscaras (filtros) para selecionar padrões de nomes de itens a serem listados.
Principais opções:

- **-a** **All.** Lista todos os itens , inclusive os que começam com pontos;
- **-l** **Long.** Lista o conteúdo de um item que é diretório;
- **-R** **Recursive.** Lista todos os diretórios encontrados e seus subdiretórios;
- **-t** **Time.** Ordena os itens por hora/data de modificação;

Exemplo:

```
$ ls -la
```

6.5 Mudando de diretório

cd <nome-do-diretório>
O comando **cd** (*change directory*) é utilizado para mudar o diretório de trabalho corrente. Não há opções para este comando. O nome do novo diretório de trabalho é indicado em nome-do-diretório. Se você não especificar um diretório, **cd** fará com que o seu diretório de entrada (*home directory*) se torne o seu diretório corrente. Se nome-do-diretório for um subdiretório do seu diretório corrente, basta informar o nome dele. Caso contrário você pode informar o nome relativo ou absoluto do diretório para o qual você quer mudar.

6.6 Criando arquivos vazios

touch <arquivo>
Cria um arquivo vazio de nome arquivo.
Exemplo:

1- Crie um arquivo vazio de nome teste.file

```
$ touch teste.file
```

6.7 Inserindo texto em arquivos

echo “texto a ser inserido” >> <arquivo>
Insere o “texto a ser inserido” ao final do arquivo.
Exemplo:

```
$ echo “teste texto” >> teste.file
```

6.8 Conteúdo de um arquivo

cat [svte] <arquivos>
O comando **cat** mostra o conteúdo de arquivos (ou da entrada padrão), apresentado-o na tela (de fato, na saída padrão). É possível utilizar o cat para criar, exibir e juntar arquivos. Quando utiliza-se o **cat** para concatenar arquivos, os arquivos da origem permanecem intactos.
Opções:
Exemplo:
1- Mostre o conteúdo do arquivo teste.file :

```
$cat teste.file curso

teste texto
```

6.9 Copiando arquivos

cp <arquivo-origem> <arquivo-destino>
O comando **cp** (*copy*) copia, isto é, cria uma cópia de um arquivo com outro nome ou em outro diretório sem afetar o arquivo original. Você pode usar esse comando para criar cópias de segurança de arquivos importantes ou para copiar arquivos que queira modificar. Se há algum arquivo que você quer ter em mais de um diretório, pode usar o comando **cp** para copiá-lo para outros diretórios.
Na linha de comando, arquivo-origem é o nome do arquivo que você quer copiar e arquivo-destino é o nome que você quer dar à cópia. Lembre-se: se você fizer uma cópia de um arquivo no mesmo diretório, ela não poderá ter o mesmo nome de arquivo-origem. Com este comando você pode acidentalmente perder arquivos se já existir um arquivo com o nome arquivo-destino, neste caso o comando **cp** escreve o novo arquivo por cima do antigo.
Exemplo:

1- Copie o arquivo file.teste para teste:
$ cp file.teste file.teste.2

6.10 Movendo/Renomeando arquivos

mv <arquivo-origem> <arquivo-destino>
O comando **mv** (move) funciona com arquivos da mesma maneira como funciona com diretórios. Pode-se usar **mv** para renomear um arquivo ou para movê-lo para outro diretório, dependendo dos argumentos que você utilizar. Na linha de comando, arquivo-origem é o nome do arquivo cujo nome você deseja mudar, e arquivo-destino o novo nome para este arquivo. Se arquivo-destino já existir, **mv** primeiro remove o arquivo já existente e depois renomeia arquivo-origem com o novo nome. Para evitar este problema, você tem duas opções:

- Examinar o conteúdo do diretório antes de renomear um arquivo, para verificar se o novo nome que você deseja atribuir já existe;
- Usar a opção **-i** (*interactive*) que permite uma confirmação da remoção

de um arquivo entes de o comando **mv** removê-lo.
Se arquivo-destino for o nome de um diretório presente no diretório corrente, então o comando **mv** entende que arquivo-origem deve ser movido para o diretório arquivo-destino, e não que este deve ser eliminado e substituído por arquivo-origem.
Se você mover um arquivo para um novo diretório, o arquivo terá o mesmo nome de arquivo-origem, a menos que você especifique o novo nome também, dando o nome do caminho (relativo ou absoluto) antes do nome do arquivo.
Exemplo:
1- Mova o arquivo file.teste para o diretório teste interativamente :

```
$ mv-i file.teste.2 teste
```

6.11 Como ligar arquivos

ln [-opções] fonte destino
Uma ligação é uma entrada em um diretório que aponta para um arquivo. O Sistema operacional cria a primeira ligação a um arquivo quando este é criado. O comando **ln** é geralmente usado para criar múltiplas referências ao arquivo em outros diretórios. Uma ligação não cria uma cópia de um arquivo, ela é simplesmente outra indicação para os mesmos dados. Quaisquer alterações em um arquivo são independentes do nome usado para se referir ao arquivo, ou seja, alterando o arquivo ou a ligação o efeito é o mesmo.
As ligações não podem ser feitas entre sistemas de arquivos, a menos que a opção **-s** seja usada. Esta opção cria uma ligação simbólica que é um arquivo que contém o nome do caminho do arquivo ao qual ele está ligado.
Opções:
-s Permite a construção de um arquivo de ligação simbólica para ligar um arquivo em um outro sistema de arquivos. Um arquivo de ligação simbólica contém o nome absoluto do arquivo no outro sistema de arquivos;
Exemplo:
1- Crie uma ligação simbólica para o arquivo teste chamado slink :

```
$ ln -s file.teste slink

$ ls -l
```

6.11.1 Notas:

- Quando você liga um arquivo a outro, não está criando outro arquivo, mas simplesmente dando ao arquivo antigo outro endereço. As mudanças feitas no arquivo ou em uma de suas ligações afetam tanto o arquivo como todas as suas ligações.
- As permissões são as mesmas para todas as ligações de um arquivo. Alterar as permissões de uma das ligações implica em alterar as permissões de todas as ligações automaticamente.
- As ligações criadas com **ln** podem ser removidas com **rm**. Isto não

significa que o arquivo original será removido.

6.12 Como remover arquivos

rm [-opções] arquivo(s)

O comando **rm** remove o arquivo e/ou as ligações. Quando a última ligação é removida, o arquivo não pode mais ser acessado e o sistema libera o espaço ocupado pelo arquivo para outro uso. Se o arquivo for de ligação simbólica, a ligação do arquivo é removida.
Para remover um arquivo é exigida a permissão de gravação do diretório pai do arquivo. Entretanto não é exigido o acesso de leitura ou gravação ao arquivo. Os caracteres especiais podem ser usados para se referir a vários arquivos sem indicar cada nome separadamente.
Opções :
-f Força a remoção de arquivos com proteção de gravação.
-r Remove recursivamente o diretório citado e seus subdiretórios.

6.13 Localizando arquivos

find diretórios [expressão]

O comando **find** procura recursivamente por arquivos em diretórios do sistema de arquivos.
O argumento diretórios especifica em quais diretórios a busca deve ocorrer. A busca recursiva faz com que a busca ocorra não apenas nos diretórios especificados, mas em todos os subdiretórios dos diretórios especificados, nos subdiretórios dos subdiretórios deles, etc. O argumento expressão consiste em um ou mais argumentos, que podem ser um critério de busca ou uma ação que o find deve tomar, ou ainda ambos os casos. Se vários argumentos forem especificados, eles devem ser separados por espaço em branco.
O comando **find** também possui um grande número de opções que podem ser utilizados na busca por arquivos em um sistema de arquivos. Neste curso vamos abordar apenas os mais usuais, suficientes para compreender o funcionamento do comando.
Expressão:
-iname arquivo Seleciona os arquivos com nomes que correspondam a arquivo, ignorando (**i**) maiúsculas e minúsculas, sendo que arquivo pode ser um nome de arquivo, ou um padrão de nomes de arquivos (especificado com o uso de *), mas deve ser precedido de uma barra invertida;
-user nome Seleciona arquivos que pertencem ao usuário nome;
-exec cmd '{}' \; Executa o comando cmd nos arquivos selecionados pelo comando find; Um par de chaves representa cada nome de arquivo que está sendo avaliado; Um ponto e vírgula marcado encerra a ação;
! Inverte o sentido do argumento que o sucede. Por exemplo, "!-iname arquivo" seleciona um arquivo cujo nome não corresponde a arquivo;
-o Permite uma seleção disjuntiva de arquivos, especificada por dois argumentos distintos. Isto é, quando usamos dois argumentos para especificar a busca, o arquivo é selecionado se satisfizer ambos os critérios

de busca. Com -o podemos selecionar arquivos que satisfazem um ou outro dos argumentos especificado para a busca. Por exemplo, "-iname arquivo-o-user nome" selecionará arquivos que possuem nomes correspondentes a arquivo ou cujo usuário seja correspondente a nome;

Exemplos :

1-Encontre o arquivo teste.file a partir do seu diretório base, imprima os caminhos

```
$ find ~ -iname teste.file

./curso/aula1/teste.file

./teste.file
```

2-Encontre todos os seus arquivos a partir do diretório curso :

```
$ find ~/curso -user aluno

/home/guest/curso/aula1

...
```

3-Encontre e apague todos os seus arquivos teste.file :

```
$ find ~ -iname teste.file -exec rm {} \;
```

6.13.1 Notas:

Como o comando **find** verifica todos os arquivos em um diretório especificado em todos os subdiretórios contidos nele, o comando pode tornar-se demorado. Veja mais adiante como executá- lo em *background*.

6.14 Procurando nos arquivos

grep [-opções] 'sequência de caracteres' arquivo(s)

O comando **grep** procura uma seqüência de caracteres em um ou mais arquivos, linha por linha. A ação especificada pelas opções é executada em cada linha que contém a seqüência de caracteres procurada.

Se mais de um arquivo for indicado como argumento do comando, o **grep** antecede cada linha de saída que contém a seqüência de caracteres com o nome do arquivo e dois pontos. O nome do arquivo é mostrado para cada ocorrência da seqüência de caracteres em um determinado arquivo.

Opções :

-i Ignora maiúsculas ou minúsculas;

-n Mostra o número de linhas com o *output* das linhas que contêm a sequência de caracteres;

-v Análise contrária. Mostra todas as linhas que **não** contém a seqüência de caracteres;

Exemplos:

1- Procure a seqüência 'root' no arquivo /etc/passwd :

```
$ grep root /etc/passwd
```

2-Procure a seqüência 'root' em todos os arquivos do diretório /etc, independente de maiúsculas ou minúsculas :

```
$ grep -i root /etc/*
```

6.15 More/less

Paginadores para visualização em tela. Quando usamos o cat para ver o conteúdo de um arquivo, e se o mesmo for muito extenso, teremos dificuldade em ver o início do mesmo. Com o more/less é possível "navegar" pelo conteúdo dos arquivos ou da da saída padrão.

Exemplo:

```
$ more /etc/passwd
```

6.16 Head e Tail

Mostram na tela as 10 primeiras ou 10 últimas linhas de um arquivos, respectivamente. Muito úteis para análise de log's.

6.16.1 Opções

-n num muda a quantidade de linhas a serem apresentadas. Por exemplo -n 30 mostra 30 linhas do arquivo.

-f atualiza continuamente o conteúdo do arquivo, ou seja, se o arquivo estiver sendo modificado as alterações aparecerão na tela.

Exemplos:

```
$ tail -n 20 /etc/passwd

$ head -n 10 /etc/passwd
```

6.17 Gzip e Gunzip

Compacta e descompacta arquivos, respectivamente. Um único por vez, mudando a extensão do mesmo.

Exemplos:

```
$ gzip teste.file.2

$ ls -l

$ gunzip teste.file.2.gz

$ ls -l
```

6.18 Tar[1]

TAR ou **tar** (abreviatura de **T**ape **AR**chive), é um formato de arquivamento de arquivos. Apesar do nome "tar" ser derivado de "*tape archive*", o seu uso não se restringe a fitas magnéticas. Ele se tornou largamente usado para armazenar vários arquivos em um único, preservando informações como datas e permissões. Normalmente é produzido pelo comando "tar". É suportado pelo programa Winrar.
tar também é o nome de um programa de arquivamento desenvolvido para armazenar (*backup*) e extrair arquivos de um arquivo tar (que contém os demais) conhecido como tarfile ou tarball. O primeiro argumento para tar deve ser uma das seguintes opções: Acdrtux, seguido por uma das seguintes funções adicionais. Os argumento finais do tar são os nomes dos arquivos ou diretórios nos quais eles podem ser arquivados. O uso de um nome de diretório, implica sempre que os subdiretórios sob ele, serão incluídos no arquivo.

6.18.1 Usando o TAR

O que o GZIP não consegue fazer, o TAR (Tape ARchives) faz. Ele é um aplicativo capaz de armazenar vários arquivos em um só. Porém, não é capaz de compactar os arquivos armazenados. Como é possível notar, o TAR serve de complemento para o GZIP e vice-versa. Por isso, foi criado um parâmetro no TAR para que ambos os programas possam trabalhar juntos. Assim, o TAR "junta" os arquivos em um só. Este arquivo, por sua vez, é então compactado pela GZIP.
O TAR também consegue gravar a propriedade e as permissões dos arquivos. Ainda, consegue manter a estrutura de diretórios original (se houve compactação com diretórios), assim como as ligações diretas e simbólicas.
A sintaxe do TAR é:

tar [parâmetros] [-f arquivo] [arquivos...].

Abaixo, segue a lista de parâmetros.
Parâmetros principais:

-c cria um novo arquivo tar;
-p mantém as permissões originais do(s) arquivo(s);
-r acrescenta arquivos a um arquivo tar;
-t exibe o conteúdo de um arquivo tar;
-v exibe detalhes da operação;
-x extrai arquivos de um arquivo tar;
-z comprime o arquivo tar resultante com o gzip;
-f especifica o arquivo tar a ser usado;

A seguir mostramos exemplos de utilização do TAR. Em alguns parâmetros o uso de '-' (hífen) não é necessário. Desta vez, os comandos não serão explicados. Execute-os e descubra o que cada um faz. Repare na combinação de parâmetros e tente entendê-la. Assim, você saberá exatamente o que está fazendo. Bom aprendizado!

```
tar -cvf arq.tar arq1 arq2

tar -czvf arq.tgz *
```

1 http://pt.wikipedia.org/wiki/TAR

```
tar -rf arq.tar arq*

tar -tzf arq.tar

tar -xv -f arq.tar
```

7 Permissão de Acesso à Diretórios e Arquivos

Há uma maneira de restringir o acesso aos arquivos e diretórios para que somente determinados usuários possam acessá-los. A cada arquivo e diretório é associado um conjunto de permissões. Essas permissões determinam quais usuários podem ler, e escrever (alterar) um arquivo e, no caso de ser um arquivo executável, quais usuários podem executá-lo. Se um usuário tem permissão de execução para um diretório, significa que ele pode realizar buscas dentro daquele diretório, e não executá-lo como se fosse um programa.
Quando um usuário cria um arquivo ou um diretório, o LINUX determina que ele é o proprietário (*owner*) daquele arquivo ou diretório. O esquema de permissões do LINUX permite que o proprietário determine quem tem acesso e em que modalidade eles poderão acessar os arquivos e diretórios que ele criou. O super-usuário (root), entretanto, tem acesso a qualquer arquivo ou diretório do sistema de arquivos.

7.1 Permissões de acesso:

O conjunto de permissões é dividido em três classes: proprietário, grupo e usuários. Um grupo pode conter pessoas do mesmo departamento ou quem está trabalhando junto em um projeto. Os usuários que pertencem ao mesmo grupo recebem o mesmo número do grupo (também chamado de Group Id ou GID). Este número é armazenado no arquivo /etc/passwd junto com outras informações de identificação sobre cada usuário. O arquivo /etc/group contém informações de controle sobre todos os grupos do sistema. Assim, pode -se dar permissões de acesso diferentes para cada uma destas três classes.
Quando você executa ls -l em um diretório qualquer, os arquivos são exibidos de maneira semelhante a seguinte:

```
total 403196
drwxr-xr-x 4 odilson admin 4096 Abr 2 14:48 BrOffice_2.1_Intalacao_Windows/
-rw-r--r-- 1 luizp admin 113811828 Out 31 21:28 broffice.org.2.0.4.rpm.tar.bz2
-rw-r--r-- 1 root root 117324614 Dez 27 14:47 broffice.org.2.1.0.rpm.tar.bz2
-rw-r--r-- 1 luizp admin 90390186 Out 31 22:04 BrOo_2.0.4_Win32Intel_install_pt-BR.exe
-rw-r--r-- 1 root root 91327615 Jan 5 21:27 BrOo_2.1.0_070105_Win32Intel_install_pt-BR.exe
```

As colunas que aparecem na listagem são:

1. Esquema de permissões;
2. Número de ligações do arquivo ou diretório;
3. Nome do usuário dono do arquivo ou diretório;
4. Nome do grupo dono do arquivo ou diretório;
5. Tamanho do arquivo, em bytes;
6. Mês da criação do arquivo;

7. Dia da criação do arquivo;
8. Hora da criação do arquivo;
9. Nome do arquivo;

O esquema de permissões está dividido em 10 colunas, que indicam se o arquivo é um diretório ou não (coluna 1), e o modo de acesso permitido para o proprietário (colunas 2, 3 e 4), para o grupo (colunas 5, 6 e 7) e para os demais usuários (colunas 8, 9 e 10).
Existem três modos distintos de permissão de acesso: leitura (*read*), escrita (*write*) e execução (*execute*). A cada classe de usuários você pode atribuir um conjunto diferente de permissões de acesso. Por exemplo, atribuir permissão de acesso irrestrito (de leitura, escrita e execução) para você mesmo, apenas de leitura para seus colegas, que estão no mesmo grupo que você, e nenhum acesso aos demais usuários. A permissão de execução somente se aplica a arquivos que podem ser executados, obviamente, como programas já compilados ou script shell. Os valores válidos para cada uma das colunas são os seguintes:

- 1 d se o arquivo for um diretório; -se for um arquivo comum;
- 2,5,8 r se existe permissão de leitura; -caso contrário;
- 3,6,9 w se existe permissão de alteração; -caso contrário;
- 4,7,10 x se existe permissão de execução; -caso contrário;

A permissão de acesso a um diretório tem outras considerações. As permissões de um diretório podem afetar a disposição final das permissões de um arquivo. Por exemplo, se o diretório dá permissão de gravação a todos os usuários, os arquivos dentro do diretório podem ser removidos, mesmo que esses arquivos não tenham permissão de leitura, gravação ou execução para o usuário. Quando a permissão de execução é definida para um diretório, ela permite que se pesquise ou liste o conteúdo do diretório.

7.2 Verificando as permissões de acesso

O comando ls -l mostra os atributos dos arquivos e dos diretórios. Normalmente as permissões padrão para os diretórios (rwxrwxrwx) permitem o acesso de leitura, gravação e execução para todos os usuários (proprietário, membros do grupo e outros). Para os arquivos as permissões padrão (rw-rw-rw-) permitem acesso de leitura e gravação para o proprietário, membros do grupo e todos os demais usuários. As permissões padrão podem ser modificadas com o uso do comando umask que será apresentado mais adiante.

7.3 Alterando a permissão de acesso

chmod modo-de-*permissão arquivo*
O modo-de-permissão na linha de comando é representado em um dos dois formatos: octal (absoluto) ou simbólico. O formato octal usa valores numéricos para representar as permissões.

7.3.1 Formato octal do modo de permissões

Há oito valores numéricos possíveis (0 -7) que representam o modo de permissão para cada tipo de usuário. Estes valores são obtidos pela soma do tipo de permissão desejada, segundo a tabela abaixo:

permissão	r	w	x
valor	4	2	1

Exemplo : Usando o formato octal, mude o modo de permissão do arquivo teste.file para que o proprietário tenha acesso total e todos os outros usuários (grupo e outros) tenham apenas permissão de leitura e execução :

```
$ chmod 755 teste.file      ==> 7=rwx (4+2+1); 5=r-x (4+1)

$ ls-l teste.file

-rwxr-xr-x 1 aluno aluno 1475 May 20 11:02 teste.file
```

7.3.2 Formato simbólico do modo de permissões

O formato simbólico usa letras e símbolos para indicar o modo de permissão. Ele é composto de três elementos :

Tipo de usuário

- **u** Usuário (Proprietário)
- **g** Grupo
- **o** Outros
- **a** Todos

Ação

A ação significa como serão alteradas as permissões.

- **+** Acrescenta permissão(ões)
- - Remove permissão(ões)
- = Atribui a permissão explicitamente

Os operadores + e -acrescentam e removem as permissões relativas ao modo de permissão corrente. O operador = reinicializa todas as permissões explicitamente (exatamente como indicado)

Tipo de permissão

- **r** Leitura
- **w** Gravação
- **x** Execução

A combinação desses três elementos formam o modo de permissão no formato simbólico.

Exemplos :

1- Tire a permissão de execução, sobre o arquivo teste, do grupo e dos outros usuários :

```
$ chmod go-x teste

$ ls-l teste
```

```
-rwxrw-rw- 2 guest user s 512 May 20 14:04 teste
```

2- Mude as permissões do arquivo prog2 para que todos os usuários possam ler e executá- lo:

```
$ chmod a=rx prog2

$ ls-l

-r-xr-xr- x 1 guest users 1986 May 20 08:26 prog2
```

7.4 Mudando as permissões padrão

umask [permissão]

Modifica os modos padrão de permissão para os novos arquivos que você criar. No comando, número é um número octal de três dígitos, como visto no comando chmod. Entretanto aqui você especifica de maneira inversa, isto é, em chmod se você utilizar número igual a 777, você estará concedendo autorização de leitura+escrita+execução para você mesmo, para o grupo e para todos os demais usuários. Com o comando umask se você especificar número igual a 777, você estará negando acesso a todas as classes em qualquer modo. De fato, a permissão que será concedida é dada pela diferença entre a permissão padrão original, que é 777 para diretórios e 666 para arquivos, e a permissão especificada em umask. Por Exemplo :

Diretórios:

- Permissão padrão 777 (rwxrwxrwx)
- Valor de umask 023
- Novas permissões 754 (rwxr -xr--)

Arquivos:

- Permissão padrão 666 (rw-rw- rw-)
- Valor de umask 022
- Novas permissões 644 (rw-r --r --)

Sem especificar um número umask mostrará o valor corrente da máscara de permissões. Os arquivos e diretórios criados antes do uso do comando permanecem com as permissões inalteradas.

Exemplo :

1- Mostrar o valor atual da máscara de permissões :

```
$ umask

0022
```

2- Mudar o valor da máscara para que os novos aquivos tenham a seguinte permissão : proprietário com acesso a leitura e escrita, grupo com acesso a leitura e execução e outro somente para leitura :

```
$ umask 012
```

7.5 "group-id" de um arquivo

chgrp grupo arquivo
O comando **chgrp** muda a identificação do grupo de um arquivo. Pode ser utilizado para conceder permissão de leitura e escrita para outro grupo que não o seu, sem ter que conceder as mesmas permissões para todos os demais usuários. Você só poderá mudar o grupo do arquivo que você mesmo criou. Além de você somente o super-usuário poderá fazer isso.
Exemplo : Mude o grupo do arquivo memo1 para users2 :

```
$ ls -l memo1

-rw- r --r-- 1 guest users 984 May 12 11:02 memo1

$ chgrp users2 memo1

$ ls -l memo1

-rw-r--r-- 1 guest users2 984 May 12 11:02 memo1
```

7.6 "owner" de um arquivo

chown usuário arquivo
Usado para mudar a identificação de proprietário associada a um arquivo. Você só poderá aplicar este comando aos arquivos que você mesmo criou. Além de você somente o super-usuário poderá fazê-lo. Observe que uma vez que você tenha alterado a identificação de proprietário que está associada a um arquivo, você não é mais o proprietário, e não poderá mais fazer a alteração inversa.
Exemplo : Mude a propriedade do arquivo prog1 para guest2 :

```
$ ls -l prog1

-rw-r-xr-- 1 guest users 1765 May 17 13:34 prog1

$ chown guest2 prog1

$ ls-l prog1

-rw-r-xr-- 1 guest2 users 1765 May 17 13:34 prog1
```

8 Redirecionamentos

8.1 Entrada e Saída dos comandos

Quase todos os comandos do LINUX usam uma entrada e produzem uma saída. A entrada para um comando são os dados sobre os quais o comando irá operar. Esses dados podem vir de um arquivo especificado pelo usuário, de um arquivo de sistema do LINUX, do terminal (do teclado)

ou da saída de outro comando. A saída de um comando é o resultado da operação que ele realiza sobre a entrada. A saída do comando pode ser impressa na tela do terminal, enviada a um arquivo, ou servir de entrada para outro comando.
Neste capítulo você vai aprender a manipular estas entradas e saídas, para poder criar e ler arquivos durante o uso de alguns comandos, e também aprenderá a encadear comandos, fazendo com que um comando utilize como entrada a saída de outro.

8.2 Entrada e Saída Padrão

Alguns comandos têm apenas uma fonte possível para a entrada, por exemplo o comando date sempre utiliza o sistema interno de relógio para indicar a data e hora. Outros comandos exigem que você especifique uma entrada. Se não especificar uma fonte de entrada juntamente com esses comandos, o LINUX considera que ela virá do teclado, isto é, ele esperará que você digite a entrada. Por isso o teclado é chamado de entrada padrão. As informações do teclado são utilizadas no processamento, e para sua facilidade o LINUX também ecoa (apresenta na tela) o que você digitar. Desta forma, você pode certificar-se de ter digitado os comando corretamente.
Normalmente, quase todos os comandos enviam suas saídas para a tela do terminal, que é chamada de saída padrão. Como com as entradas, você também pode redirecionar as saídas dos comandos para outro destino que não é a saída padrão, por exemplo para arquivos ou para a entrada de outros comandos.
Alguns comandos, como rm, mv e mkdir não produzem nenhuma saída. Entretanto esses comandos e muitos outros podem apresentar mensagens de erro na tela se não obtiverem sucesso no seu processamento. Isto ocorre porque a tela do terminal também é a saída de erros padrão, isto é, o local para onde são enviadas as mensagens de erro. As mensagens de erro dos comandos não devem ser confundidas com as saídas dos comandos.
O shell do LINUX redireciona a fonte e o destino da entrada, de modo que o comando não percebe se a entrada padrão está direcionada para o teclado do terminal ou para um arquivo. Da mesma forma, o comando não percebe se a saída padrão está direcionada para a tela do terminal, para um arquivo ou para a entrada de outro comando.

8.3 Redirecionamento de E/S

Há três métodos básicos para redirecionar a entrada ou saída de um comando. Uma delas é simplesmente fornecer como argumento para o comando o nome do arquivo que deve ser usado como entrada ou saída para o comando. Este método funciona com alguns comandos, como por exemplo cat, pg e outros. Já comandos como o pwd não podem receber um arquivo como argumento. Mesmo com os filtros (classe de comandos a qual pertencem o cat e o pg) nem sempre é possível especificar a saída.
Outro método para redirecionar a entrada ou saída é utilizar os símbolos

de redirecionamento. Como muitos comandos podem receber arquivos de entrada sob a forma de argumentos, os símbolos de redirecionamento são mais utilizados para direcionar a saída dos comandos.
Um terceiro método de redirecionar entradas e saídas é usando pipes, que enviam a saída de um comando para outro, ou seja, a saída de um comando serve como entrada para outro comando.

8.3.1 Símbolos de redirecionamento

Os caracteres especiais utilizados na linha do comando para fazer O shell redirecionar a entrada, saída ou erro do programa estão listados e descritos a seguir. O shell interpreta esses caracteres antes do comando ser executado.

8.3.2 Redirecionamento de entrada

comando < arquivo
O símbolo < (menor que) faz com que a entrada padrão seja direcionada a um arquivo. Em muitos casos, especificar < funciona exatamente como especificar o nome do arquivo como argumento do comando. Por exemplo:

```
$ cat Arquivo.teste

$ cat < Arquivo.teste
```

produzirão exatamente o mesmo efeito.

8.3.3 Redirecionamento de saída

comando > arquivo ou **comando >> arquivo**
Os símbolos > e >> redirecionam a saída de um comando para um arquivo. O símbolo > escreve a saída do comando dentro do arquivo que você indicar, quer esse arquivo exista ou não, sendo que o conteúdo do arquivo já existente será substituído. O símbolo >>, ao contrário, anexa ao arquivo a saída do comando indicado em vez de substituir os dados que ele já continha. No C Shell é necessário que o arquivo já exista, para que o símbolo possa ser utilizado, caso contrário ocorrerá um erro.
Exemplos :
1- Guarde no arquivo data.de.hoje a saída do comando date :

```
$ date > data.de.hoje

$ cat data.de.hoje
```

2- Acrescente ao arquivo data.de.hoje a saída do comando who :

```
$ who >> data.de.hoje

$ cat data.de.hoje
```

8.3.4 Pipes

Os símbolos de redirecionamento permitem realizar mais de uma operação em um mesmo arquivo. Somente com esses símbolos você já tem condições de realizar tudo oque quiser sobre um arquivo. Suponha, entretanto, que você queira fazer um conjunto de operações diferentes em um mesmo arquivo. Cada operação implicaria a criação de um novo arquivo, sendo que o único propósito desses arquivos seria servir como entrada para outro comando. Entretanto, tudo o que importa é o resultado final. Para situações como essas o LINUX possui outra maneira de redirecionar entradas e saídas: os *pipes*.

comando 1 | comando 2

Este símbolo, "|", pode ser usado para enviar a saída de um comando para a entrada de outro. Você pode usar vários *pipes* em uma linha de comando, de maneira que é possível combinar tantos comandos quantos necessários, bastando intercalá-los por símbolos de *pipe*. Uma seqüência de comandos encadeados desta maneira é chamada de *pipeline*.

Existem algumas regras básicas para compor um *pipeline* em uma linha de comandos LINUX. Essencialmente essas regras são o endosso da intuição de um usuário um pouco mais experiente, que facilmente percebe que em um *pipeline* não pode haver "vazamentos" nem "entupimentos" do *pipe*, isto é, não pode haver no meio do *pipeline* um comando que não produza saídas (como é o caso do mkdir ou rm), ou um comando que não aceite entradas (como é o caso do date e pwd). O primeiro comando do *pipeline* deve ser um produtor de saída, obviamente.

Exemplos :

1- Conte o número de arquivos que começam com a sub-string 'arq' no diretório corrente :

```
$ ls | grep arq | wc -l

3
```

2- Conte o número de usuários que estão presentes no sistema neste momento :

```
$ who | wc -l

2
```

8.3.5 Redirecionamentos múltiplos

tee *[iau] arquivo*

O comando tee "divide" a saída de um comando e redireciona-a para dois destinos: para um arquivo especificado e para a saída padrão. O comando tee em geral é utilizado como um pedaço de um *pipeline*. Se não estiver em um *pipeline*, o comando tee se comporta de maneira semelhante ao comando cat: recebendo linhas na entrada e ecoando-as na saída.

Opções:

-a Faz a saída ser anexada aos arquivos especificados, em vez de

substituir seus conteúdos;
-i Ignora o sinal de interrupção;
Exemplos :
1- Conte o número de arquivos que começam com a sub-string ‘arq’ no diretório corrente e guarde os arquivos encontrados no arquivo nomes :

```
$ ls | grep arq | tee nomes | wc -l

3

$ cat nomes arq

arq2

arquivo
```

2- Conte o número de ocorrências da cadeia “Linux” no arquivo arq2, guarde as ocorrências no arquivo resp :

```
$ cat < arq2 | grep Linux | tee resp | wc -l

2

$ cat resp
```

8.3.6 Redirecionamento de erro padrão

A mensagem de erro gerada por um comando é normalmente direcionada pelo shell para a saída de erro padrão, que é a mesma da saída padrão. A saída de erro padrão também pode ser redirecionada para um arquivo, utilizando o símbolo >. Uma vez que este símbolo também é utilizado para redirecionar a saída padrão, é necessário fazer uma distinção mais detalhada para evitar ambigüidade.
Os descritores de arquivos a seguir especificam a entrada padrão, saída padrão e saída de erro padrão:

- 0 Entrada padrão;
- 1 Saída padrão;
- 2 Saída de erro padrão;

O descritor do arquivo deve ser colocado imediatamente antes dos caracteres de redirecionamento. Por exemplo, 1> indica a saída padrão, enquanto 2> indica a saída de erro padrão. Assim, o comando mkdir temp 2> errfile faz o shell direcionar qualquer mensagem de erro para o arquivo errfile. As indicações da entrada padrão (0>) e saída padrão (1>) são necessárias apenas para evitar ambigüidade.
Exemplo :

```
$ find / -name xinetd.conf > find.res 2> find.erro

$ cat find.resp

$ cat find.erro
```

9 Editor vi

O editor vi é bastante simples e muito utilizado por ser encontrado em todas as distribuições Linux. Poderíamos optar por um editor um pouco mais avançado, mas com o inconveniente de encontrarmos uma distribuição/instalação onde não dispuséssemos deste editor. Isto é válido principalmente para o chamado “Linux embarcado” onde não dispomos de memória para outros editores.

O editor vi não objetiva formatar textos: negritos, indentações, justificação, etc.

Na prática o vi é muito usado para editar textos que não necessitam de formatação em nenhum momento, como por exemplo códigos fonte de programas em alguma linguagem de programação, e que não carreguem o texto com caracteres especiais.

Neste capítulo vamos aprender alguns comandos do vi, suficientes para que você entenda o funcionamento do editor e consiga editar arquivos simples.

9.1 Os três modos de operação do VI

O editor vi tem três modos de operação distintos, que são: **modo insert** ,

modo escape (também chamado de modo comando), **modo last line**;

O **modo insert** é usado para a digitação do texto. Neste modo o vi funciona como uma máquina de escrever, com a diferença de que você pode retroceder sobre o texto já digitado para corrigir eventuais erros. Cada caracter que for digitado aparecerá na tela exatamente como foi digitado.

No **modo escape** os caracteres comuns (letras, números e sinais de pontuação) têm um significado especial e quase todos os caracteres funcionam como comandos; portanto, existem muitos comandos. Alguns comandos servem para passar para o modo insert, outros para movimentar o cursor sobre as linhas do texto, alterar trechos do texto, buscar palavras, etc.

No modo **last line** você digita os comandos em uma linha especial que aparece no final da tela quando se digita : (dois pontos) no modo escape. Parte dos comandos do modo escape possuem similares no modo last line, como por exemplo os comandos de edição, que veremos mais adiante. Os comandos no modo last line devem ser seguidos por ENTER, contrariamente ao que acontece no modo escape.

9.2 O Buffer de edição

Quando você edita um arquivo com o vi, na verdade você não está alterando o arquivo em si. As alterações feitas são aplicadas em um buffer (uma área na memória, que passa a conter o arquivo sendo editado). Quando você

quiser que as alterações fiquem permanentemente aplicadas ao arquivo, é necessário copiar o conteúdo do buffer para o disco, usando o comando write (w) no modo last line. Portanto, se o comando write não for executado antes de deixar o vi, as alterações contidas no buffer não serão aplicadas ao arquivo que está no disco.

9.3 Criação e edição de arquivos

Nesta etapa vamos dar enfoque somente aos comandos e usos principais, objetivando atender as demandas de um administrador de rede.

Para criarmos um arquivo simplesmente digitamos vi seguido do nome do arquivo. Por exemplo:

```
vi primeiro.arquivo
```

Após isto será aberto o editor com conteúdo vazio, no modo comando. Para podermos editar qualquer coisa devemos entrar no modo inserção, para isto basta teclarmos <i> (aparecerá -- INSERT -- na base da janela). Em seguida digitamos o texto propriamente dito, usando o teclado normalmente.

Para salvarmos o texto devemos teclar <Esc> (modo comando), <:> (modo last line> e <w> (write). Assim teremos o nosso texto salvo.

Agora vamos a alguns comandos úteis:

- Para **copiarmos algumas linhas** do texto colocamos o cursor no início do texto a ser copiado e, no modo de comando, teclamos <n>+<y>+<y>, onde n é número de linhas que desejamos copiar. Por exemplo se digitarmos <5>+<y>+<y> copiaremos 5 linhas para o buffer.
- Para **excluirmos algumas linhas** do texto colocamos o cursor no início do texto a ser excuído e, no modo de comando, teclamos <n>+<d>+<d>, onde n é número de linhas que desejamos copiar. Por exemplo se digitarmos <3>+<y>+<y> apagaremos 3 linhas do texto mas que serão armazenadas no buffer.
- Para **colarmos o conteúdo do buffer** para alguma parte do texto colocamos o cursor no ponto onde pretendemos inserir o texto e, no modo de comando, teclamos <p> (paste) para inserirmos o texto abaixo da linha do cursor e <P> para inserir o texto acima da linha do cursor.
- Para **encontrarmos alguma palavra**, no modo de comando, teclamos </> <palavra> <Enter>. O vi mostrará a primeira ocorrência da mesma. Para ir para a próxima ocorrência teclamos <n> (next).
- Para **substituirmos uma palavra por outra**, no modo de comando, teclamos <:s/><palavra></><outra><Enter>. Por exemplo se quisermos substituir velha por nova: <:s/><velha></><nova><Enter>, assim teremos a troca da primeira ocorrência de velha por nova.
- Para **substituirmos todas as ocorrências** acrescentamos <%> entre <:> e <s> do caso anterior. Exemplo: <:

%s/><velha></><nova><Enter>

- Para **inserirmos o conteúdo de um texto externo em nosso texto**, no modo de comando, teclamos <:r>+<caminho/arquivo>+<Enter>.
- Para salvarmos com outro nome, no modo de comando, teclamos <:w>+<novo.nome>+<Enter>.

10 KDE

O KDE é um ambiente *desktop* gráfico poderoso para estações com Linux/UNIX. O KDE combina a facilidade de uso, funções atuais, projeto gráfico proeminente com a tecnologia do sistema operacional UNIX/Linux.

O KDE, como a grande maioria dos ambientes, gráficos é intuitivo e fácil de utilizar. É recomendado (obrigatório) para uso em estações clientes mas não para o caso de servidores de rede, principalmente por ser “pesado” e mais suscetível a problemas.
Após o login o usuário terá uma janela semelhante a da Ilustração 1.

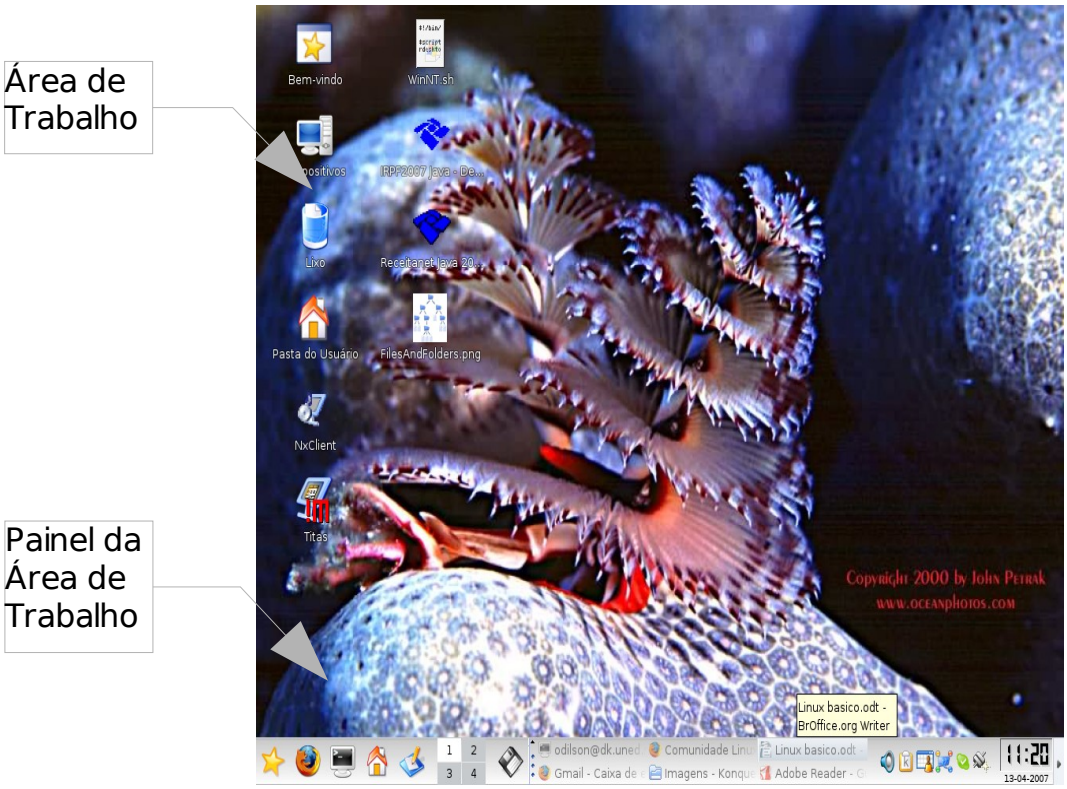


Ilustração 1: Aparência do KDE

Como na maioria dos ambientes gráficos temos a **Área de Trabalho**, onde se localizam os ícones de acesso rápido, e o **Painel da Área de Trabalho**, que permanece sempre visível e também tem a função de acesso rápido.
Ao clicarmos em ☆ será aberto um menu como mostrado na Ilustração 2.

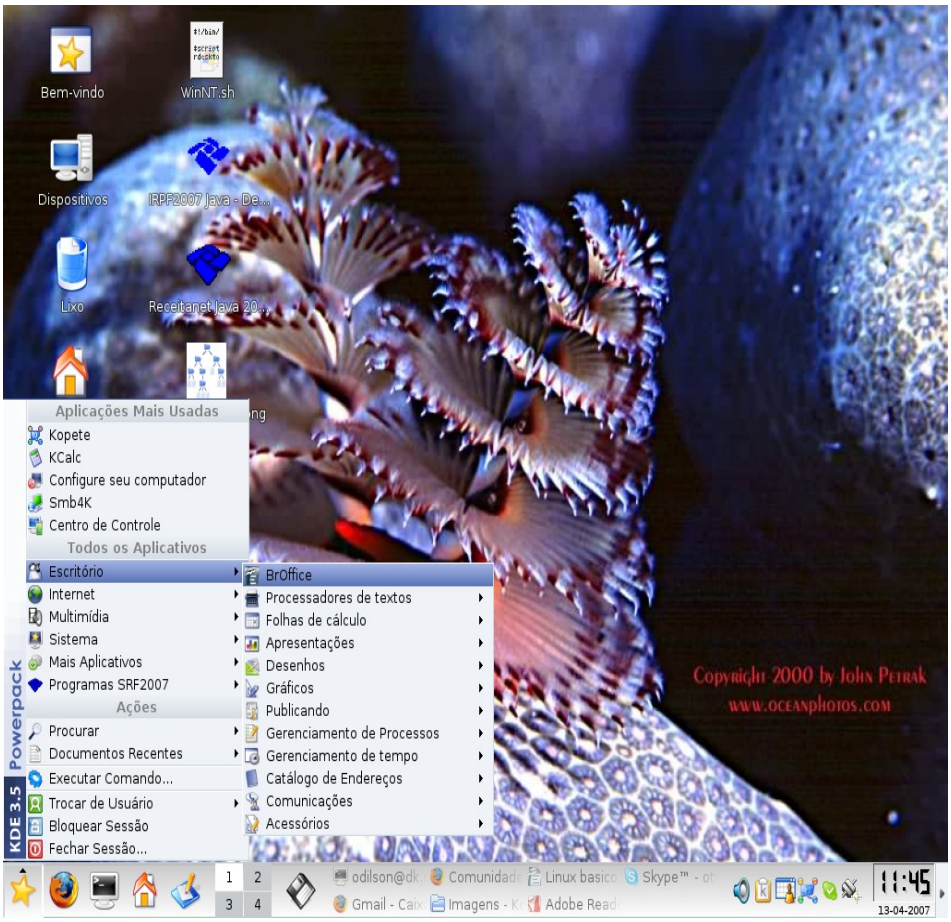


Ilustração 2: Menus do KDE

Todos os menus e painéis são configuráveis e personalizáveis, no sentido de facilitar a vida do usuário. Em geral os menus são sensíveis ao contexto e podem ser personalizados diretamente com auxílio do mouse.

10.1 Alguns aplicativos do KDE

Na instalação padrão, o Mandriva já insere uma série de aplicativos gráficos, prontos para o uso, que provêem o suporte a maioria das necessidades de uso no dia-a-dia do usuário. Se houver necessidade de mais algum aplicativo basta instalar conforme o roteiro do capítulo 11.
A título de exemplo vamos citar/conhecer alguns.
Dia – Editor de diagramas. ☆ - Escritório – Gráficos. Ilustração 3.

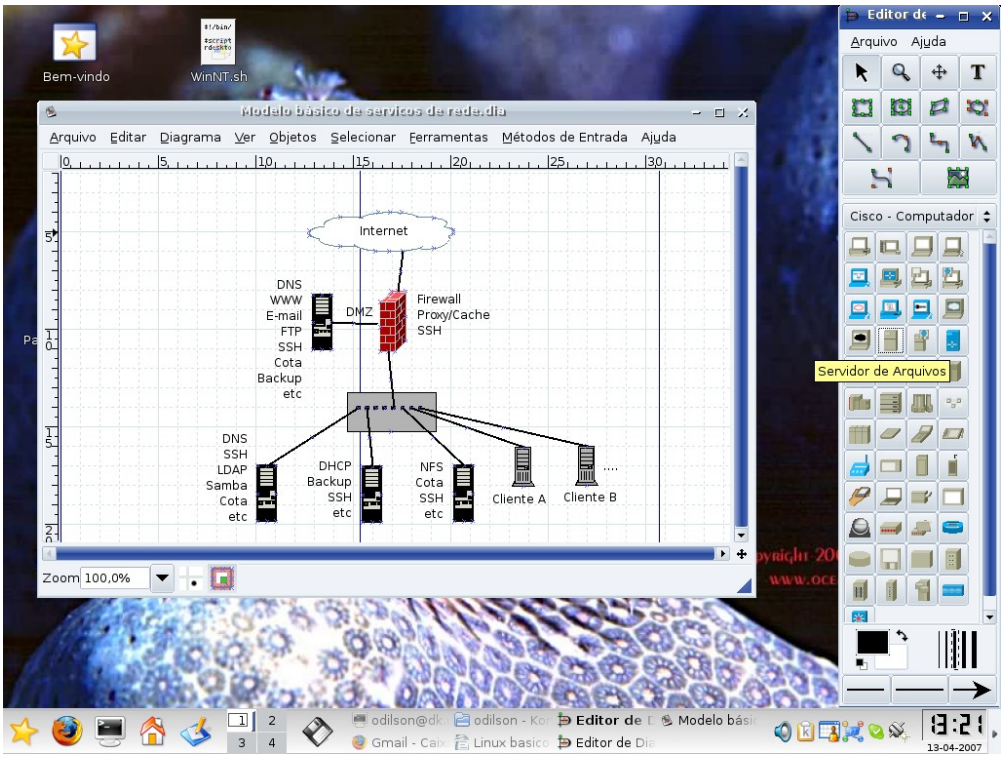


Ilustração 3: Editor de diagramas - DIA

Scribus – Editoração Gráfica (~Corel Draw). ☆ - Escritório – Publicando – Scribus. Ilustração 4.

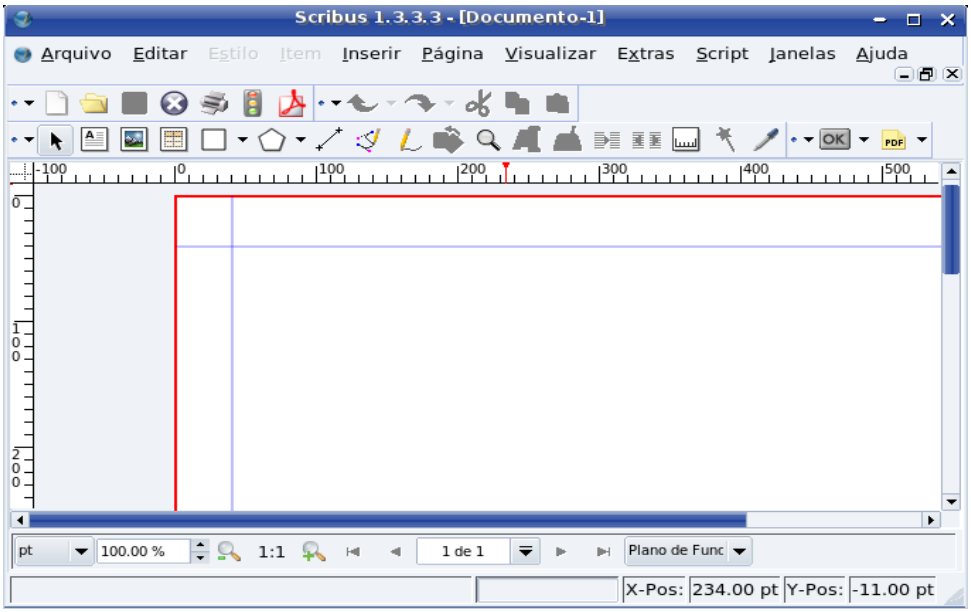


Ilustração 4: Editoração Gráfica - Scribus.

K3B – Gravador de CD's e DVD's. ☆ - Sistema – Arquivar – Gravador de CD. Ilustração 5.

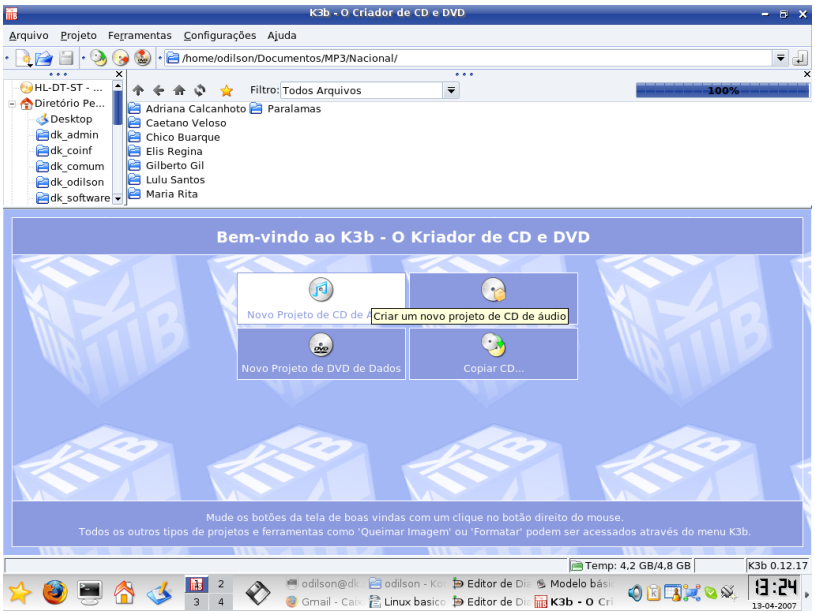


Ilustração 5: K3B - Gravador de CD's e DVD's

Centro de Controle – Configurações de software e hardware da máquina. ☆ - Sistema – Configuração. Ilustração 6.

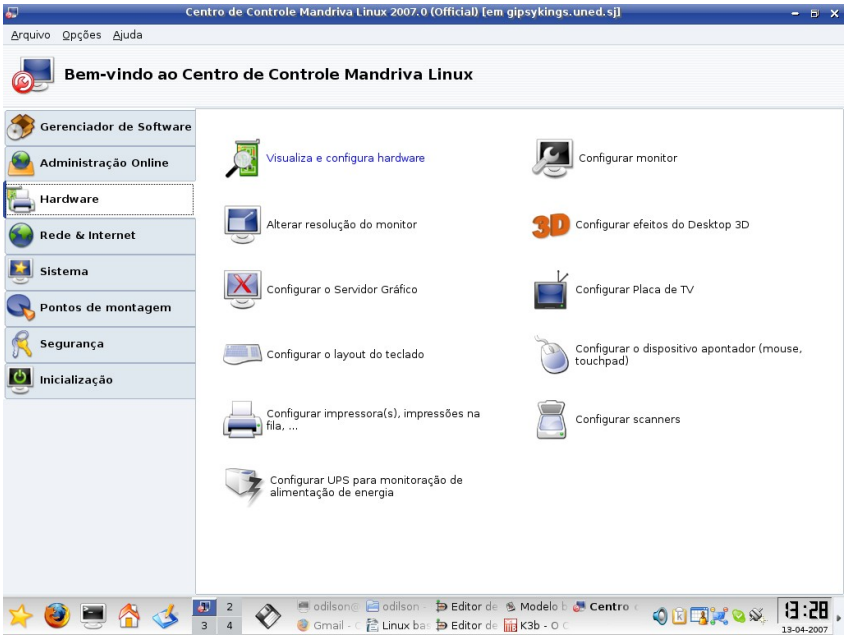


Ilustração 6: Centro de Controle

Gerência de Redes

11 Gerência de Redes

Como princípio o gerenciamento de redes consiste em prover serviços de rede aos usuários da maneira transparente e fácil aos mesmos.
Normalmente o usuário procura, além dos sistemas próprios da empresa, os seguintes serviços: correio eletrônico, navegação na internet, servidor para hospedar as páginas de seus projetos e um lugar seguro para guardar seus dados e documentos.
Por outro lado, do ponto de vista do administrador, para prover estes serviços de maneira segura são necessários uma série de outros serviços que, a princípio, não interessa ao usuário e em muitos casos “atrapalha” o mesmo.

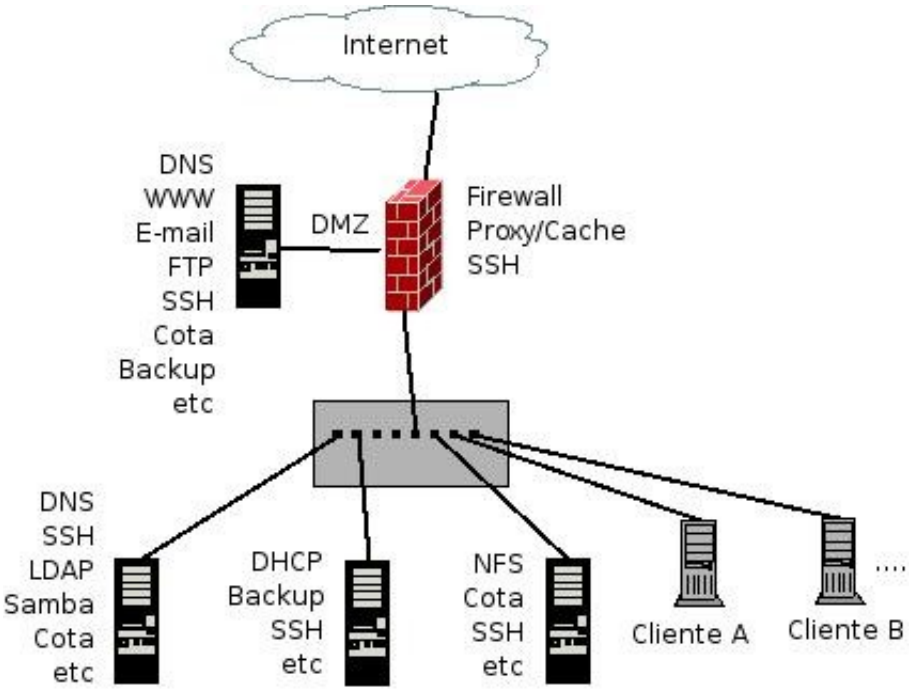


Ilustração 7: Serviços de rede

Na Ilustração 7 vemos um diagrama com os serviços básicos de rede que um administrador de sistemas deve ofertar. Devemos observar que os serviços podem ou não estar agrupados na mesma máquina.
Nos capítulos abaixo vamos detalhar uma série de serviços que podemos fornecer aos usuários ou que são necessários à boa administração do sistema. Nosso objetivo é prover uma instalação básica de todos os serviços para o conhecimento básico teórico/prático dos mesmos. Em caso de necessidades de configurações “avançadas” nos serviços, devemos consultar as referências bibliográficas.

12 Inittab[2]

Antes que qualquer script de inicialização tenha sido executado, o arquivo /etc/inittab é lido. Cada linha neste arquivo possui o seguinte formato:

```
ID:runstate:ação:processo
```

Cada um destes campos indicam:

```
ID:         identificador da entrada

runstate:   nível de operação na qual esta entrada é usada

ação:       indica como o processo é executado. Por exemplo, o valor wait
            indica que o processo deve ser executado e aguardar pelo seu
            encerramento.

processo:   indica o comando ou processo a ser executado.
```

A linha

```
s3:3:wait:/sbin/rc3
```

indica que o script /sbin/rc3 é executado quando o sistema se encontra no nível de operação de número 3 e que o processamento deve ser encerrado antes que qualquer ação adicional seja tomada.
Uma das principais atribuições deste arquivo é a definição do nível de inicialização do sistema (*run level*), que podem ser:

0 - halt (não o deixe como padrão)
1 – modo monousuário
2 – Modo multiusuário, sem NFS (basicamente sem rede)
3 – Modo multiusuário completo (com rede)
4 – Não usado (pode ser usado para definir um modo próprio)
5 – X11 (ambiente gráfico)
6 - reboot (não o deixe como padrão)

Por exemplo:

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

indica que está máquina inicializa no modo gráfico.
Outra atribuição deste arquivo é habilitar ou não o reboot pela associação das teclas <Ctrl>+<Alt>+<Delete>, com uma linha do tipo:

```
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

Se comentarmos esta linha o reboot pelo teclado será desabilitado.

13 Instalação de aplicativos com RPM[3]

RPM, a simplificação de *Red Hat Package Manager* é um sistema de gerenciamento de pacotes para Linux. RPM instala, atualiza, desinstala e verifica softwares. RPM é o formato base da Linux Standard Base. Originalmente desenvolvido pela Red Hat Linux. RPM é agora usado por muitas distribuições Linux e também é portado para outros sistemas operacionais como NetWare da

2 Texto obtido de http://www.dicas-l.com.br/dicas-l/19980517.php
3 Texto obtido de http://pt.wikipedia.org/wiki/RPM

Novell e AIX da IBM.

13.1 Base de Dados RPM

Na base do gerenciador de pacotes está o banco de dados rpm. Ele consiste de uma lista duplamente ligada que contém todas as informações de todos os rpm instalados. O banco da dados lista todos os arquivos que são criados ou modificados quando um usuário instala um programa e facilita a remoção destes mesmos arquivos. Se o banco de dados é corrompido, as ligações duplas garantem que eles possa ser reconstruído sem nenhum problema. Nos computadores com o sistema operacional RedHat e derivados instalado, este banco da dados se encontra em /var/lib/rpm.

13.2 Rótulo dos Pacotes

Todo pacote RPM tem um rótulo de pacote (*package label*), que contém as seguintes informações:

- o nome do software
- a versão do software (a versão tirada da fonte original do pacote)
- a edição do pacote (o número de vezes que o pacote foi refeito utilizando a mesma versão do software)
- a arquitetura sob a qual o pacote foi feito (i386, i686, athlon, ppc, noarch[4] etc.)

os arquivos RPM têm normalmente o seguinte formato:

<nome>-<versão>-<release>.<arquitetura>.rpm

Um exemplo:

nano-0.98-2.i386.rpm

Note que o rótulo do pacote está contido no arquivo e não precisa necessariamente ser o mesmo que o nome do arquivo.
O código-fonte também pode ser distribuído em pacotes RPM. O rótulo de tais pacotes não contém a parte destinada para a arquitetura e em seu local inserem "src". Exemplo:

libgnomeuimm2.0-2.0.0-3mdk.src.rpm

13.3 Vantagens e desvantagens do formato

As vantagens de utilizar os pacotes RPM em com relação a outros métodos de adquirir e instalar software são:

- Um método uniforme para o usuário instalar programas.
- Maior simplicidade para desinstalar os programas.
- Popularidade: muitos pacotes disponíveis.
- Instalação não-interativa: facilita uma instalação automática.
- Código-fonte original incluído (.tar.gz, .tar.bz2): fácil de verificar.
- Verificação criptográfica com o GPG e o md5.

As desvantagens incluem:

- Comumente tem mudanças no formato de pacote incompatíveis com versões anteriores.

4 Independente de arquitetura

- Documentação incompleta e desatualizada.
- Pouca aprendizagem sobre os pacotes.

13.4 Acessórios relacionados

O RPM é comumente usado por outros acessórios para manipular dependências, como o *Yellow dog Updater Modified* yum ou o (versão compatível com RPM) *Advanced Packaging Tool* (apt).
Alguns gerenciadores de pacotes são

- dpkg usado com o *Advanced Packaging Tool* (apt) no Debian Linux.
- portage usado no Gentoo Linux.
- urpmi usado no Mandriva.

13.5 Instalação/desinstalação de aplicativos com URPMI

No caso do Mandriva sempre a opção mais fácil é usar o urpmi, já que o mesmo "tenta" adivinhar o que estamos querendo instalar e instala todas as dependências, se for o caso. Por exemplo, se desejarmos instalar o digikam (software para manipulação e gerenciamento de fotos), mas não lembramos exatamente o nome e digitamos

```
urpmi digi
```

O sistema retornará algo assim:

```
nenhum nome de pacote digi
Os seguintes pacotes contém digi:
acroread-plugins-digitalsignature
digicamerge
digikam
digikamimageplugins
digitemp
libdigidoc2
libdigidoc2-devel
libdigikam0
libdigikam0-devel
rmedigicontrol
vdr-plugin-digicam
x11-driver-input-digitaledge
```

então digitamos:

```
urpmi digikam
```

Para desinstalar (extrair) basta digitarmos:

```
urpme pacote
```

13.6 Mídias do URPMI

Mídia é o local onde temos pacotes rpm para o Mandriva. O cd-rom, o dvd, diretório nfs, ftp, http todos são mídias. Geralmente a maioria chama de

repositório devido ao costume de se trabalhar com o Debian e Conectiva com a ferramenta APT. Estas são facilmente gerenciadas com alguns comandos:

urpmi.addmedia	Adiciona mídias à base de dados
urpmi.removemedia	Remove mídias da base de dados

As mídias podem ser sites da internet, permitindo assim o administrador manter o sistema sempre atualizado. Existe um excelente site para cadastro de mídias que é o *Easy URPMI* http://easyurpmi.zarb.org/. Neste site configuramos as nossas mídias de acordo com nossa versão e necessidades, basta seguir o roteiro do site.

14 Sistema de arquivos[5]

Sistema de arquivos é a forma de organização de dados nos discos de armazenamento. Sabendo do sistema de arquivos de um determinado disco, o Sistema Operacional pode decodificar os dados armazenados e lê-los ou gravá-los.

Fazendo analogias, tal organização assemelha-se a uma biblioteca escolar. O bibliotecário organiza os livros conforme o seu gosto, cuja busca, convenientemente, procura deixar mais fácil, sem ocupar muitas prateleiras e assegurando a integridade deste. Ainda, certamente, organiza os livros segundo suas características (assunto, censura, etc). Depois de organizados, ou durante a organização, o bibliotecário cria uma lista com todos os livros da biblioteca, com seus assuntos, localizações e códigos respectivos.

O Sistema Operacional seria o bibliotecário da "biblioteca de dados" do computador: o disco de armazenamento. Exatamente igual à organização de uma biblioteca, o Sistema Operacional guarda os dados nos espaços vazios do disco, rotulando-os com um FCB (*File Control Block*, Bloco de Controle de Arquivo) e ainda criando uma lista com a posição deste dado, chamada de MFT (*Master File Table*, Tabela de Arquivos Mestre).

Sabendo a posição do arquivo a ser aberto/gravado, o Sistema Operacional solicita a leitura desta, decodifica/codifica e realiza a abertura/gravação do dado.

Um sistema de arquivos é, assim, uma forma de criar uma estrutura lógica de acesso a dados numa partição. Sendo assim, também é importante referir que nunca poderá ter 2 ou mais tipos de sistemas de arquivos (formatos) numa mesma partição.

O MBR (*Master Boot Record*) é um arquivo de dados interligado com a BIOS (*Basic Input Output System*) cuja importância é o reconhecimento do sistema de arquivos, como também na inicialização de sistema operacionais.

5 Texto obtido de http://pt.wikipedia.org/wiki/Sistema_de_ficheiros

Particionar um dispositivo é dividi-lo de forma que cada uma das suas partes, denominadas *partições*, possa receber um tipo de sistema de arquivo e esteja preparada para receber as informações.

Sistema de arquivos e partições são normalmente confundidos, quando na verdade são conceitos totalmente diferentes. As partições são áreas de armazenamento, criadas durante o processo de particionamento, sendo que cada partição funciona como se fosse um disco rígido (ou dispositivo utilizado). Para se utilizar uma partição, entretanto, deve-se criar um sistema de arquivos, ou seja, um sistema que organize e controle os arquivos e diretórios desta partição. Uma partição pode ter apenas um sistema de arquivo, já um disco com várias partições pode ter vários sistemas de arquivos.

Quando um disco rígido é formatado com um sistema de arquivos no Linux, o mesmo é dividido em 4 partes, Ilustração 8.

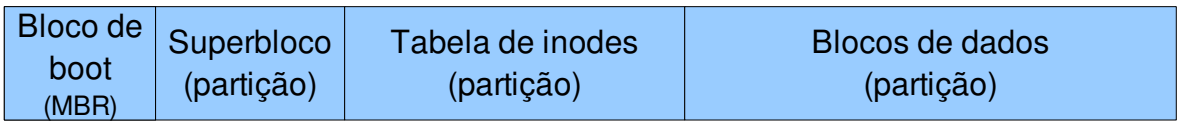


Ilustração 8: Estrutura de um sistema de arquivos genérico no Linux

O bloco de boot contém o boot do sistema operacional.

O superbloco contém informações sobre o sistema de arquivos, como número de inodes, inodes livres, número de blocos, blocos livres, etc.

A tabela de inodes contém informações sobre cada arquivo (diretório é um tipo especial de arquivo). Cada inode tem 64 bits e contém as seguintes informações:

- UID e GID (identificação do usuário e grupo dono).
- Tipo de arquivo. Arquivo comum, diretório, link, dispositivo etc., ou 0 (zero) se o inode não estiver em uso.
- Permissões.
- Data e hora de criação, acesso e última modificação.
- Número de links para o mesmo.
- Tamanho.
- Localização dos blocos onde está armazenado seu conteúdo.

O bloco de dados contém os dados propriamente ditos dos arquivos.

O Linux tem suporta à dezenas de sistemas de arquivos, sendo que os principais são:

- **ext**: sistema de arquivos estendido (*extended filesystem*). É o sistema de arquivos mais utilizado no Linux. Existem ramificações (ext2 e ext3), sendo o ext3 o mais amplamente utilizado pela comunidade Linux atualmente. Ele fornece padrões para arquivos regulares, diretórios, arquivos de dispositivos, *links* simbólicos e suporte a transações

(*journalling*), entre outras características avançadas.

- **vfat**: este é o sistema de arquivos (volume FAT) dos sistemas Windows®9x e Windows NT®.
- **ntfs**: este é o sistema de arquivos dos sistemas Windows2000®, Windows XP® e NT®, entre outros. O Linux só o suporta em modo de leitura.
- **nfs**: sistema de arquivos de rede, utilizado para acessar diretórios de máquinas remotas, que permite o compartilhamento de dados na rede.
- **reiserfs:** sistema de arquivos com suporte a características como, por exemplo, melhor performance para diretórios muito grandes e suporte a transações (*journalling*). Não suporta cota para usuários e grupos.
- **Xfs:** Projetado especialmente para trabalhar com arquivos grandes (de até 9 mil "petabytes") e diretórios com vários arquivos. Oferece suporte a quotas para usuários e grupos. Suporta transações (*journalling*).
- **iso9660:** sistema de arquivos do CD-ROM.

14.1 Particionando e formatando discos

O Linux trata os discos, diferentemente do Windows, por nomes hda, hdb, hdc e hdd para disco IDE; sda, sdb etc para discos SATA e SCSI.

As partições são numeradas dentro de cada disco, do tipo hda1, hda2 etc. Sendo que o Linux normalmente cria uma partição primária, uma estendida e as demais lógicas dentro desta estendida. Sendo assim ou nomes das partições seriam hda1, hda5, hda6 etc.

Para particionarmos discos podemos usar as ferramentas fdisk, cfdisk ou, no caso específico do Mandriva, o diskdrake.

Exemplo:

Muito cuidado para não "detonar" a máquina.

Vamos criar uma partição no espaço livre em disco da nossa máquina. Para isso executamos a seguinte seqüência de comandos:

1. diskdrake, abrirá uma janela conforme Ilustração 9.

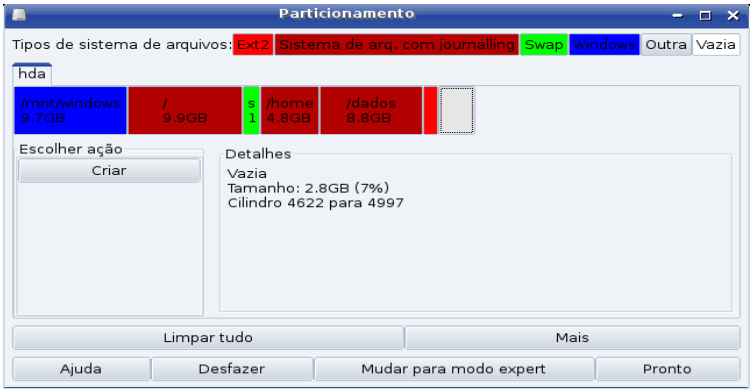


Ilustração 9: Janela do DiskDrake

2. Clicar no espaço vazio (cinza).
3. Clicar em Criar, aparecerá uma janela que deverá ser completada conforme Ilustração 10.

Ilustração 10: Especificação da nova partição

4. Clicar em OK.
5. Clicar em Formatar. **Tenha certeza que a partição selecionada é a nova**.
6. Clicar em Montar.
7. Clicar em Pronto. Aparecerá uma janela, Ilustração 11, que requer a confirmação se deseja-se ou não atualizar o arquivo /etc/fstab, ver próximo item, Montando partições. Clique Sim.

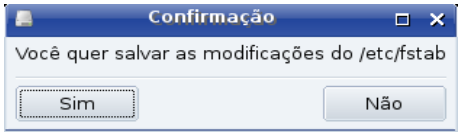


Ilustração 11: Confirmação para gravação no fstab

Tudo pronto!

Caso tivéssemos optado pelo cfdisk deveríamos permitir a “leitura” da nova partição pelo sistema operacional reiniciando a máquina.

Em seguida deveríamos formatar a partição com o comando:

```
mkfs.ext3 /dev/hdaX
```

Onde **X** é o número da partição que acabamos de criar. **Muito cuidado**, pois se informarmado o número errado “detona-se” uma partição indevida. Se não lembrar do número use o cfdisk para ter certeza.

14.2 Montando partições[6]

Uma vez formatado devemos montar a nova partição para que ela se torne acessível a nós. O primeiro passo é criar um diretório onde será montado a nova partição e em seguida a montagem. Por exemplo:

mkdir /dados

mount /dev/hda**X** /dados

Assim teremos a nova partição disponível para uso mas, deste modo, isto valerá somente até reiniciarmos a máquina. Se desejamos usar sempre tal partição devemos informar ao sistema sobre isto.

O arquivo /etc/fstab (*File System Table*) é o responsável pelas montagens das partições desejadas. É através desse arquivo que são montadas as partições para que o Gnu/Linux inicie corretamente, pois sem ele não teria como iniciar o sistema. O fstab também serve para montar outras partições com outros sistemas de arquivos, facilitando para você não precisar montar aquela sua partição windows toda vez que ligar o computador, e através dele também são montados os dispositivos (cd-rom, dvd, floppy...). Este documento se baseia no manual do fstab (man fstab) e do manual do mount (man mount).
Então vamos lá.
Abaixo temos um exemplo de um arquivo /etc/fstab:

Device	**Moint point**	**File Sistem**	**Options**
/dev/hda4	/	ext3	relatime 1 1
/dev/hda5	/var	ext3	relatime 1 2
/dev/hda6	swap	swap	relatime 0 0
/dev/hdb	/mnt/cdrom	iso9660	relatime,noauto,user 0 0
/dev/fd0	/mnt/floppy	auto	relatime, noauto, user 0 0
/dev/hda2	/mnt/win	vfat	noexec,uid=100,gid=100 0 0

O modelo do arquivo fstab é assim:
[dispositivo] [ponto de montagem] [sistema de arquivos] [opções] [opção para o dump] [opção para o fsck]
dispositivo:
Nesse campo é colocado o dispositivo a ser montado ou um sistema de arquivos remoto. Para montagens NFS deve ser colocado <máquina>:<dir>, exemplo: dark.dark.net:/home/minhapasta.

ponto de montagem:
Ele identifica em qual pasta será montada a partição, para partições swap esse campo deve ser especificado como 'swap'. No nosso exemplo na primeira linha coloquei a pasta raiz do Gnu/linux, já na segunda linha identifiquei como a pasta /var que seria montado no sistema.
sistema de arquivos:
Nesse campo você descreve qual o sistema de arquivo. Consulte /proc/filesystems para saber quais sistemas de arquivos são suportados pelo seu kernel.

6 Texto obtido de http://www.linuxbsd.com.br/phpLinuxBSD/modules/artigos_tecnicos/fstab.htm

opções:

Segue abaixo explicação de algumas das opções disponíveis no fstab:

noauto: Essa opção faz com que o dispositivo não montado automaticamente durante o boot, é a opção que deve ser usada para disquetes e cd-roms no fstab, pois senão o Gnu/Linux iria tentar monta-los mesmo que não tivessem discos neles.

user: Essa opção é ótima também para discos removíveis, ela permite que qualquer usuário possa montar esse dispositivo.

noexec: Essa opção é muito útil para quando montamos partições windows, pois ele não gerencia os arquivos com permissões como no Gnu/Linux, com isso os arquivos ficam todos como executáveis, se você clicar em cima de um arquivo mp3, usando o konqueror por exemplo, ele vai tentar "executar" o arquivo é claro que sem funcionar, usando essa opção você faz com que isso não ocorra.

uid: Essa opção também é útil quando se monta partições FAT, pois elas não trabalham com permissões de arquivo, assim todas as partições que forem montadas estarão com o dono dos arquivos seja o root. Assim você com um usuário normal não poderia ter total controle desse diretório, com essa opção você pode mudar o dono do arquivo usando o uid dele que pode ser encontrado no arquivo /etc/passwd:
jean:x:144:200::/home/jean:/bin/false
Nesse exemplo o uid do usuário jean seria 144.

gid: Com essa opção você pode mudar o grupo do diretório, na verdade é a mesma função da opção acima, só que faz isso com o grupo.
jean:x:144:200::/home/jean:/bin/false
Nesse exemplo o gid do usuário jean é 200.

umask: serve para indicar quais serão as permissões dos arquivos, já que os sistemas Fat e derivados não tem sistema de permissões. O padrão é a máscara do processo atual. O valor é dado em formato octal. O padrão geralmente é representado por 022, ou seja, bit 'w'(permissão de escrita) apenas para o dono.

ro: O dispositivo será montado somente para leitura

rw: Monta o sistema de arquivos com permissão de leitura e gravação.

exec: Permite a execução de binários.

suid: Permite o uso dos bits de configuração de identificação do usuário e do grupo.

dev: Interpreta dispositivos especiais de blocos ou caracteres no sistema de arquivos.

relatime: Usa as opções padrão: rw, suid, dev, exec, auto, nouser, e async.

opção para o dump:

Essa opção é usada pelo comando dump para determinar quais sistemas de arquivos precisam ser copiados, caso não tenha sido escrito nada nesse quinto campo o valor dele será considerado zero, e o dump assumirá que esse sistema de arquivos não precisa ser copiado.

opção para o fsck:

Nesse campo você deve colocar a ordem em que os sistemas de arquivos serão verificados durante o boot. A partição raiz (/), sempre como 1, e os outros sistemas de arquivos devem ter esse campo a partir de 2 fazendo

seqüência de acordo com o número de partições que você quiser montar. Sistemas de arquivos em um mesmo dispositivo, serão verificados seqüencialmente, e sistemas de arquivos em dispositivos diferentes, serão verificados ao mesmo tempo para utilizar o paralelismo disponível com o hardware. Caso esse campo não exista ou esteja com o valor 0 o fsck não irá checar essa partição ao inicializar o Gnu/Linux.

14.3 A estrutura de diretórios

O GNU/Linux segue o Filesystem Hierarchy Standard para nomeação de arquivos e diretórios. Este padrão permite aos usuários e programas predizerem a localização de arquivos e diretórios. O diretório de nível raiz é representado simplesmente pela /. No nível raiz, todos os sistemas incluem estes diretórios:

bin	Binários de comandos essenciais
boot	Arquivos estáticos e gerenciador de inicialização
dev	Arquivos de Dispositivos
etc	Configuração do sistema específico da máquina
home	Diretórios de usuários
lib	Bibliotecas essenciais compartilhadas e módulos do kernel
media	Ponto de montagem para montar um sistema de arquivos temporariamente
proc	Diretório virtual de informações do sistema
root	Diretório home do usuário root
sbin	Binários essenciais do sistema do usuário root
tmp	Arquivos temporários
usr	Hierarquia secundária
var	Dados variáveis
opt	Aplicativos adicionais e pacotes de softwares

15 *LVM- Logical Volume Manager*

15.1 Introdução

O LVM tem por objetivo facilitar a administração do espaço em disco e facilitar a ampliação de partições com a inserção de novos discos nos servidores. Como principais características podemos citar:

- O LVM permite o aumento ou diminuição do tamanho de partições sem a reformatação e a reinicialização da máquina.
- Em partições XFS e ReiserFS não é necessário sequer o desmonte da partição para alterar o tamanho da mesma. Ou seja, pode-se alterar o tamanho de uma partição com o sistema *on-line*.

O melhor momento de configuração do LVM é na instalação do servidor. Deve-se reservar uma área mínima para o / (raiz), já que somente o / e o /boot não podem ser montados em partições LVM, e deixar todo o espaço restante do disco como LVM, Ilustração 12.

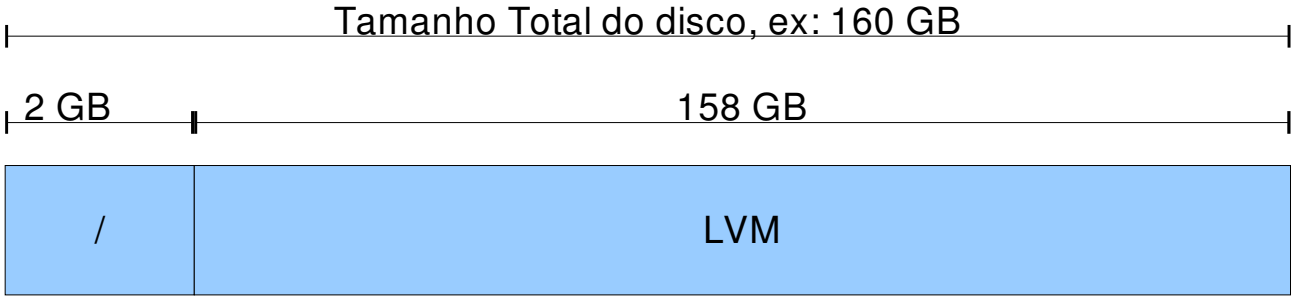


Ilustração 12: Particionamento com LVM

O Gerenciador de Volumes Lógicos consiste em uma camada adicional entre os dispositivos físicos e a interface de E/S no kernel para fornecer uma visão lógica no armazenamento.
Ao contrário do método tradicional de particionamento, a implementação **LVM** cria um grande disco virtual, que pode inclusive ter mais de um dispositivo de armazenamento, e *divide* em *partições virtuais*. A grande vantagem é permitir o redimensionamento das áreas de modo dinâmico, ou seja, com o sistema operacional sendo utilizado.
A grande desvantagem é que por ser um único disco virtual, a recuperação de dados em uma eventual pane no sistema de armazenamento é bastante prejudicada.
Tecnicamente o LVM é montado/composto conforme descrito abaixo, Ilustração 13.

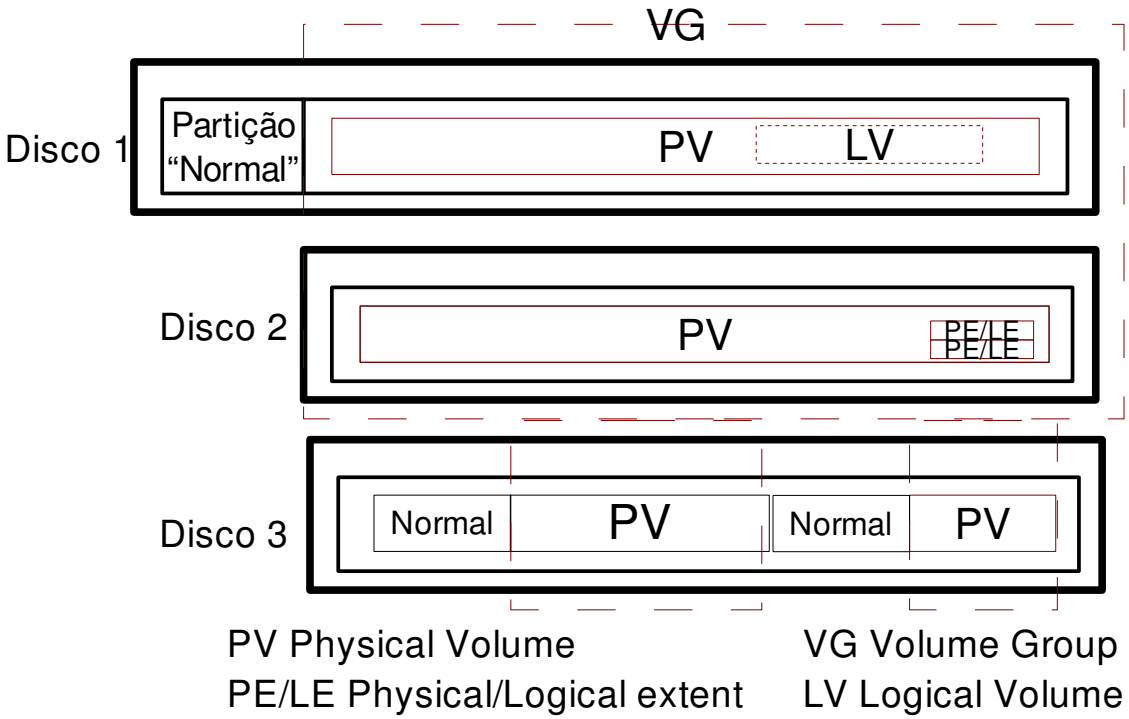


Ilustração 13: Como é fisicamente o LVM

PV (Physical Volume) - Os volumes físicos são as partições de discos alocadas para o LVM. No Linux é necessário criar a partição e alterar o tipo para "Linux LVM", tipo 8e do fdisk, para que ela possa ser utilizada no LVM.

VG (Volume Group) - Um conjunto de PV podem ser necessários para criar

filesystems maiores que a limitação física de um disco rígido. Esses PV são agrupados em um VG.

PE (Physical Extent) - Quando um PV é inserido em um VG o LVM o divide em várias partes de igual tamanho e essas partes são associadas a uma **LE** (Logical Extent), o menor valor de alocação dentro de um VG (do ponto de vista do LVM). No AIX são conhecidos como PP (Physical Partition) e LP (Logical Partition), respectivamente.

LV (Logical Volume) - Esse elemento é uma área de alocação das LE, na qual criamos o filesystem. Ao criarmos um volume lógico, recebemos um device para referenciarmos, ao criar ou manipular, o sistema de arquivos. O nome do device é /dev/NOME_DO_VG/NOME_DO_LV.

VGDA (Volume Group Descriptor Area) - Numa analogia mais grosseira, essa área é uma tabela de alocação do VG. Nela há todos os dados do VG. É dividida em quatro partes básicas: descritor de PV, descritor de VG, descritor de LV e vários descritores de PE e LE. Os backup automáticos da VGDA são guardados em /etc/lvm-conf/.

15.2 Implantando LVM

Se o LVM não foi instalado juntamente com o sistema podemos instalar/configurar o mesmo posteriormente. Como primeira ação devemos instalar os pacotes necessários para gerenciamento e criação de partições LVM. Para isto usamos o comando:

```
urpmi lvm2
```

No exemplo abaixo vamos criar um grupo de volumes de nome vg, numa partição já existente na máquina. Primeiro "inicialize" o LVM com o comando:

```
vgscan
```

Crie o volume físico com o comando:

```
pvcreate /dev/hdaX (X = número da partição)
```

Inclua o(s) volume(s) físico(s) no grupo de volumes com o comando:

```
vgcreate vg /dev/hdaX /dev/hdbX ...
```

Atualize o LVM com o comando:

```
vgscan
```

Ative o volume lógico com o comando

```
vgchange -a y
```

Verifique a criação com o comando

```
vgdisplay -v vg
```

Dentro do grupo de volumes crie, por exemplo, dois volumes lógicos lv1 de 300 MB e lv2 de 500 MB com os comandos:

```
lvcreate -L 300M -n lv1 vg
```

```
lvcreate -L 500M -n lv2 vg
```

Ative os volumes lógicos com os comandos:

```
lvchange -a y /dev/vg/lv1

lvchange -a y /dev/vg/lv2
```

Formate as "partições" lógicas criadas com os comandos abaixo, ou de acordo com o sistema de arquivos usado.

```
mkfs.xfs /dev/vg/lv1

mkfs.xfs /dev/vg/lv2
```

Para o caso de ext3 teríamos:

```
mkfs -t ext3 /dev/vg/lv1 e

mkfs -t ext3 /dev/vg/lv2
```

Crie dois diretórios, onde serão montadas as partições, com o comando:

```
mkdir /dados /backup
```

Monte as partições com os comandos:

```
mount /dev/vg/lv1 /dados

mount /dev/vg/lv2 /backup/
```

Verifique as partições montadas com o comando

```
df
```

As partições já estão disponíveis para uso. Podem ser copiados arquivos e diretórios. Se for necessário pode-se aumentar o tamanho da partições.

15.3 Aumentando ou diminuindo o tamanho de partições LVM

Para aumentar o tamanho de uma partição é preciso primeiro ver seu tamanho e a disponibilidade de espaço no volume group.

Para verificar quais são os volumes lógicos utiliza-se o comando:

```
lvdisplay
```

O espaço disponível pode ser visto com o comando:

```
vgdisplay
```

No exemplo abaixo aumenta-se o volume lógico lv em 2 GB:

```
lvextend -L +2G /dev/vg/lv
```

Podemos reduzir, no formato reiserfs, com o comando:

```
lvreduce -L -2G /dev/vg/lv
```

Pronto a partição já foi aumentada. O único problema é que o sistema de arquivos ainda não sabe disso. Execute o próximo passo de acordo com o seu sistema de arquivos:

- Sistemas de arquivos XFS (pode ser executado com o sistema *on-line*, ou seja, com a partição montada):

```
xfs_growfs /dados
```

- Sistemas de arquivos ReiserFS (pode ser executado com o sistema on-line, ou seja, com a partição montada):

```
resize_reiserfs -f /dev/vg/lv1
```

- Sistemas de arquivos EXT3:

```
resize2fs /dev/vg/lv1
```

16 Gerência de usuários e grupos

16.1 Introdução

Um usuário Linux é uma entidade que possui uma identificação no sistema onde os principais parâmetros são: *login*, senha, e número de identificação. Estas informações permitem ao Linux controlar como o acesso é garantido aos usuários e o que eles podem fazer depois de obter a permissão de acesso.
Um grupo é um conjunto de usuários. Cada grupo também possui identificação única no sistema, um nome e um número. O administradores de sistemas normalmente fazem controle de acesso por meio dos grupos.

16.2 Criação de conta

Para criar um grupo usamos o comando:

```
groupadd nome_do_grupo
```

Para criação de uma conta de usuário usa-se o comando:

```
adduser login
```

Podemos ainda “sofisticar” a criação de contas com algumas *flags*, as principais são:

- -d caminho – Podemos informar qual será o diretório home do usuário.
- -g grupo – Podemos definir o grupo primário a que o usuário pertencerá. Se este não for informado o linux cria um grupo com o mesmo nome de usuário.
- -G grupo1,grupo2 – Define o(s) grupo(s) suplementar(s) ao(s) qual(is) o usuário pertencerá.
- -c comentário – Normalmente o nome completo do usuário.
- -s shell – Esta opção é interessante se desejamos, por exemplo, que um usuário não acesse a máquina com sua conta. Para isto informamos /bin/false.

Em seguida **definimos a senha** com o comando:
`passwd login`

16.3 Parâmetros das Contas

As contas de usuários ficam armazenadas nos seguintes arquivos passwd, group e shadow.
O arquivo **/etc/passwd** define todos os usuários cadastrados no sistema, segundo o molde:

login:x:503:500:comentário:/home/login:/bin/bash

A descrição dos campos são:

o login do usuário.

senha, mais comumente encontrada em shadow.

UID, *User Identification*, número que identifica o usuário.

GID, *Group Identification*, número que identifica o grupo primário.

comentários, pode conter nome de usuário, endereço, etc entre aspas simples e campos separados por vírgulas.

define o diretório home do usuário.

o shell do usuário.

O arquivo **/etc/group** tem uma relação dos grupos do sistema, segundo molde:

nome_do_grupo:senha:GID:lista_de_usuários

As descrições dos campos são:

o nome do grupo

a senha (criptografada) do grupo. Se este campo estiver vazio, nenhuma senha é necessária. (gpasswd)

o identificador numérico do grupo.

nomes de usuário de todos os membros suplementares do grupo, separados por vírgulas.

O arquivo **/etc/shadow** contém as senhas dos usuários, segundo os moldes:

login:1/clCUNfk$9ULCTE27T94Po9qRp5oJi0:13581:0:99999:7:::

As descrições dos campos são:

login
senha criptografada
dias, desde 01/01/1970, que a senha sofreu a última alteração
dentro de quantos dias a senha não pode ser alterada.
dentro de quantos dias a senha deverá ser alterada.
quantos dias antes da expiração da senha o usuário receberá aviso.
quantos dias após a expiração da senha a conta será desabilitada.
dias, desde 01/01/1970, que a conta está desabilitada.
campo reservado.

O arquivo **/etc/login.defs** contém uma série de diretivas e padrões que serão utilizados na criação das próximas contas de usuários. Seu principal conteúdo é:

MAIL_DIR dir # Diretório de e-mail
PASS_MAX_DAYS 99999 #Número de dias até que a senha expire
PASS_MIN_DAYS 0 #Número mínimo de dias entre duas trocas senha
PASS_MIN_LEN 5 #Número mínimo de caracteres para composição da senha
PASS_WARN_AGE 7 #Número de dias para notificação da expiração da senha
UID_MIN 500 #Número mínimo para UID
UID_MAX 60000 #Número máximo para UID
GID_MIN 500 #Número mínimo para GID
GID_MAX 60000 #Número máximo para GID
CREATE_HOME yes #Criar ou não o diretório home

Como o login.defs o arquivo **/etc/default/useradd** contém padrões para criação de contas. Seu principal conteúdo é:

GROUP=100 #GID primário para os usuários criados
HOME=/home #Diretório a partir do qual serão criados os “homes”
INACTIVE=-1 #Quantos dias após a expiração da senha a conta é desativada
EXPIRE=AAAA/MM/DD #Dia da expiração da conta
SHEL=/bin/bash #Shell atribuído ao usuário.
SKEL=/etc/skel #Arquivos e diretórios padrão para os novos usuários.

16.4 Alterando parâmetros das contas

Para modificarmos uma conta já existente podemos usar o comando

```
usermod opçoes login
```

onde as principais opções são:

-c comentário
-d diretório_home
-e data_de_expiração data, na forma YYYY-MM-DD, que a conta será desativada
-g grupo grupo primário
-G grupo grupo(s) suplementar(es)
-l novo_login *login*
-L trava a senha
-s shell

16.5 Removendo Contas

Para remover um grupo usamos o comando:

```
groupdel nome_do_grupo
```

Para remover contas de usuário usamos o comando:

```
userdel login
```

A principal opção é:

-r apaga o diretório home do usuário e todo seu conteúdo.

17 Permissão de Acesso à Diretórios e Arquivos

Há uma maneira de restringir o acesso aos arquivos e diretórios para que somente determinados usuários possam acessá-los. A cada arquivo e diretório é associado um conjunto de permissões. Essas permissões determinam quais usuários podem ler, e escrever (alterar) um arquivo e, no caso de ser um arquivo executável, quais usuários podem executá-lo. Se um usuário tem permissão de execução para um diretório, significa que ele pode realizar buscas dentro daquele diretório, e não executá-lo como se fosse um programa.
Quando um usuário cria um arquivo ou um diretório, o LINUX determina que ele é o proprietário (*owner*) daquele arquivo ou diretório. O esquema de permissões do LINUX permite que o proprietário determine quem tem acesso e em que modalidade eles poderão acessar os arquivos e diretórios que ele criou. O super-usuário (root), entretanto, tem acesso a qualquer arquivo ou diretório do sistema de arquivos.

17.1 Permissões de acesso:

O conjunto de permissões é dividido em três classes: proprietário, grupo e usuários. Um grupo pode conter pessoas do mesmo departamento ou quem está trabalhando junto em um projeto. Os usuários que pertencem ao mesmo grupo recebem o mesmo número do grupo (também chamado de Group Id ou GID). Este número é armazenado no arquivo /etc/passwd junto com outras informações de identificação sobre cada usuário. O arquivo /etc/group contém informações de controle sobre todos os grupos do sistema. Assim, pode -se dar permissões de acesso diferentes para cada uma destas três classes.
Quando você executa ls-l em um diretório qualquer, os arquivos são exibidos de maneira semelhante a seguinte:

```
total 403196
drwxr-xr-x 4 odilson admin 4096 Abr 2 14:48 BrOffice_2.1_Intalacao_Windows/
-rw-r--r-- 1 luizp admin 113811828 Out 31 21:28 broffice.org.2.0.4.rpm.tar.bz2
-rw-r--r-- 1 root root 117324614 Dez 27 14:47 broffice.org.2.1.0.rpm.tar.bz2
-rw-r--r-- 1 luizp admin 90390186 Out 31 22:04 BrOo_2.0.4_Win32Intel_install_pt-BR.exe
-rw-r--r-- 1 root root 91327615 Jan 5 21:27 BrOo_2.1.0_070105_Win32Intel_install_pt-BR.exe
```

As colunas que aparecem na listagem são:

1. Esquema de permissões;
2. Número de ligações do arquivo;
3. Nome do usuário dono do arquivo;
4. Nome do grupo associado ao arquivo;
5. Tamanho do arquivo, em bytes;
6. Mês da criação do arquivo; Dia da criação do arquivo;
7. Hora da criação do arquivo;
8. Nome do arquivo;

O esquema de permissões está dividido em 10 colunas, que indicam se o arquivo é um diretório ou não (coluna 1), e o modo de acesso permitido

para o proprietário (colunas 2, 3 e 4), para o grupo (colunas 5, 6 e 7) e para os demais usuários (colunas 8, 9 e 10).
Existem três modos distintos de permissão de acesso: leitura (***r****ead*), escrita (***w****rite*) e execução (*e****x****ecute*). A cada classe de usuários você pode atribuir um conjunto diferente de permissões de acesso. Por exemplo, atribuir permissão de acesso irrestrito (de leitura, escrita e execução) para você mesmo, apenas de leitura para seus colegas, que estão no mesmo grupo que você, e nenhum acesso aos demais usuários. A permissão de execução somente se aplica a arquivos que podem ser executados, obviamente, como programas já compilados ou script shell. Os valores válidos para cada uma das colunas são os seguintes:

- 1 d se o arquivo for um diretório;-se for um arquivo comum;
- 2,5,8 r se existe permissão de leitura;-caso contrário;
- 3,6,9 w se existe permissão de alteração;-caso contrário;
- 4,7,10 x se existe permissão de execução;-caso contrário;

A permissão de acesso a um diretório tem outras considerações. As permissões de um diretório podem afetar a disposição final das permissões de um arquivo. Por exemplo, se o diretório dá permissão de gravação a todos os usuários, os arquivos dentro do diretório podem ser removidos, mesmo que esses arquivos não tenham permissão de leitura, gravação ou execução para o usuário. Quando a permissão de execução é definida para um diretório, ela permite que se pesquise ou liste o conteúdo do diretório.

17.2 Verificando as permissões de acesso

O comando ls-l mostra os atributos dos arquivos e dos diretórios. Normalmente as permissões padrão para os diretórios (rwxrwxrwx) permitem o acesso de leitura, gravação e execução para todos os usuários (proprietário, membros do grupo e outros). Para os arquivos as permissões padrão (rw-rw-rw-) permitem acesso de leitura e gravação para o proprietário, membros do grupo e todos os demais usuários. As permissões padrão podem ser modificadas com o uso do comando umask que será apresentado mais adiante.

17.3 Alterando a permissão de acesso

chmod modo-de-*permissão arquivo*
O modo-de-permissão na linha de comando é representado em um dos dois formatos: octal (absoluto) ou simbólico. O formato octal usa valores numéricos para representar as permissões.

17.3.1 Formato octal do modo de permissões

Há oito valores numéricos possíveis (0 -7) que representam o modo de permissão para cada tipo de usuário. Estes valores são obtidos pela soma do tipo de permissão desejada, segundo a tabela abaixo:

permissã	r	w	x

o			
valor	4	2	1

Exemplo : Usando o formato octal, mude o modo de permissão do arquivo prog1 para que o proprietário tenha acesso total e todos os outros usuários (grupo e outros) tenham apenas permissão de leitura e execução :

```
$ chmod 755 prog1      7=rwx (4+2+1); 5=r-x (4+1)

$ ls-l prog1

-rwxr-xr-x 1 guest users 1475 May 20 11:02 prog1
```

17.3.2 Formato simbólico do modo de permissões

O formato simbólico usa letras e símbolos para indicar o modo de permissão. Ele é composto de três elementos :

Tipo de usuário

u Usuário (Proprietário)

g Grupo **o** Outros **a** Todos

Ação

A ação significa como serão alteradas as permissões.

+ Acrescenta permissão(ões)

- Remove permissão(ões)

= Atribui a permissão explicitamente

Os operadores + e - acrescentam e removem as permissões relativas ao modo de permissão corrente. O operador = reinicializa todas as permissões explicitamente (exatamente como indicado)

Tipo de permissão

r Leitura

w Gravação

x Execução

A combinação desses três elementos formam o modo de permissão no formato simbólico.

Exemplos :

1- Tire a permissão de execução, sobre o arquivo teste, do grupo e dos outros usuários :

```
$ chmod go-x teste

$ ls-l teste

-rwxrw-rw- 2 guest user s 512 May 20 14:04 teste
```

2- Mude as permissões do arquivo prog2 para que todos os usuários possam ler o executá- lo :

```
$ chmod a=rx prog2

$ ls-l

-r-xr-xr- x 1 guest users 1986 May 20 08:26 prog2
```

17.4 Mudando as permissões padrão

umask [permissão]
Modifica os modos padrão de permissão para os novos arquivos que você criar. No comando, número é um número octal de três dígitos, como visto no comando chmod. Entretanto aqui você especifica de maneira inversa, isto é, em chmod se você utilizar número igual a 777, você estará concedendo autorização de leitura+escrita+execução para você mesmo, para o grupo e para todos os demais usuários. Com o comando umask se você especificar número igual a 777, você estará negando acesso a todas as classes em qualquer modo. De fato, a permissão que será concedida é dada pela diferença entre a permissão padrão original, que é 777 para diretórios e 666 para arquivos, e a permissão especificada em umask. Por Exemplo :
Diretórios: Permissão padrão 777 (rwxrwxrwx) Valor em umask 023
Novas permissões 754 (rwxr -xr--)
Arquivos: Permissão padrão 666 (rw-rw- rw-) Valor em umask 022
Novas permissões 644 (rw-r --r --)
Sem especificar um número umask mostrará o valor corrente da máscara de permissões. Os arquivos e diretórios criados antes do uso do comando permanecem com as permissões inalteradas.
Exemplo :
1- Mostrar o valor atual da máscara de permissões :

```
$ umask

0022
```

2- Mudar o valor da máscara para que os novos aquivos tenham a seguinte permissão : proprietário com acesso a leitura e escrita, grupo com acesso a leitura e execução e outro somente para leitura :

```
$ umask 012
```

17.5 "group-id" de um arquivo

chgrp *grupo arquivo*
O comando **chgrp** muda a identificação do grupo de um arquivo. Pode ser utilizado para conceder permissão de leitura e escrita para outro grupo que não o seu, sem ter que conceder as mesmas permissões para todos os demais usuários. Você só poderá mudar o grupo do arquivo que você mesmo criou. Al ém de você somente o super-usuário poderá fazer isso.
Exemplo : Mude o grupo do arquivo memo1 para users2 :

```
$ ls-l memo1

-rw- r --r-- 1 guest users 984 May 12 11:02 memo1

$ chgrp users2 memo1

$ ls-l memo1
```

```
-rw- r --r-- 1 guest users2 984 May 12 11:02 memo1
```

17.6 "owner" de um arquivo

chown *usuário arquivo*
Usado para mudar a identificação de proprietário associada a um arquivo. Você só poderá aplicar este comando aos arquivos que você mesmo criou. Além de você somente o super-usuário poderá fazê-lo. Observe que uma vez que você tenha alterado a identificação de proprietário que está associada a um arquivo, você não é mais o proprietário, e não poderá mais fazer a alteração inversa.
Exemplo : Mude a propriedade do arquivo prog1 para guest2 :

```
$ ls -l prog1

-rw-r-xr-- 1 guest users 1765 May 17 13:34 prog1

$ chown guest2 prog1

$ ls-l prog1

-rw-r-xr-- 1 guest2 users 1765 May 17 13:34 prog1
```

18 Cotas em disco para usuários e grupos

18.1 Introdução

O sistema de cotas em disco é muito importante pois permite um controle efetivo do espaço em disco a ser utilizado por usuários e grupos. Permite que o administrador controle o sistema de tal modo a não ocorrer travamentos por partições lotadas.
A implementação do sistema de cotas no Linux se dá por partições, ou seja, em todas as partições onde desejamos um efetivo controle devemos configurar o sistema de cotas. As cotas são determinadas por usuário e/ou grupo, sendo que podemos impor limites por espaço ocupado e/ou por número de arquivos e diretórios criados. Por exemplo, se um determinado usuário recebe uma cota de 100 MB, ele poderá ocupar no máximo 100 MB de espaço na partição, seja qual for o diretório da partição. Ao mesmo tempo não terá contabilizado em sua cota algum arquivo ou diretório salvo em outra partição.

18.2 Implementação

Como primeira etapa devemos instalar os pacotes que permitirão o uso do sistema de cotas, com o comando:

```
urpmi quota
```

Em seguida devemos informar ao sistema de arquivos em qual(is) partição(ões) pretendemos implantar o sistema de cotas. Para isto devemos editar o arquivo

/etc/fstab e inserir ao final da quarta coluna, separado por vírgula, a diretiva usrquota e/ou grpquota, dependendo se desejamos quotas para usuários, grupos ou ambas. No exemplo abaixo habilitamos cotas para usuários no /home e para usuários e grupos no /dados.

- Arquivo original
  /dev/hda5 / ext3 relatime 1 1
  /dev/hda7 /home ext3 relatime 1 2
  /dev/hda9 /dados ext3 relatime 1 2
  /dev/hda /mnt/cdrom auto umask=0,users,iocharset=utf8,noauto,ro,exec 0 0
  /dev/hda1 /mnt/win_c ntfs umask=0,nls=utf8,ro 0 0
  /dev/hda8 /mnt/win_d vfat umask=0,iocharset=utf8 0 0
  none /proc proc relatime 0 0
  /dev/hda6 swap swap relatime 0 0
- Arquivo modificado
  /dev/hda5 / ext3 relatime 1 1
  /dev/hda7 /home ext3 relatime,usrquota 1 2
  /dev/hda9 /dados ext3 relatime,usrquota,grpquota 1 2
  /dev/hda /mnt/cdrom auto umask=0,users,iocharset=utf8,noauto,ro,exec 0 0
  /dev/hda1 /mnt/win_c ntfs umask=0,nls=utf8,ro 0 0
  /dev/hda8 /mnt/win_d vfat umask=0,iocharset=utf8 0 0
  none /proc proc relatime 0 0
  /dev/hda6 swap swap relatime 0 0

Após isto devemos desmontar e montar a(s) partição(ões) com parâmetros modificados para que o sistema de arquivos do kernel releia o arquivo fstab e monte segundo os novos parâmetros. Para isto devemos usar a seguinte seqüência de comandos:

```
init 3                  # Chaveamos para o modo texto puro, “matando” o modo
                        gráfico

Logar como root

umount -a               # Desmontamos todas as partições. Exceção ao /. Pode ser
                        necessário aguardar um certo tempo eté ser possível o
                        desmonte das partições onde será implementado o sistema de
                        cotas

mount -a                # Montamos todas as partições do fstab

quotacheck -augv        # Vai inicializar o sistema de cotas
```

Após isto serão criados os arquivos /home/aquota.user, /home/aquota.group, /dados/aquota.user e /dados/aquota.group. Estes arquivos são uma espécie de banco de dados que contém uma relação entre usuários/grupos e o espaço_em_disco/arquivos_e_diretórios usados pelos mesmos.
Em seguida devemos ativar o sistema de cotas com o comando:

```
quotaon -augv
```

Poderíamos desativar, caso desejássemos, o sistema de cotas, sem editar manualmente cada cota de usuário, com o comando:

```
quotaoff -augv
```

Agora podemos restabelecer o modo gráfico com o comando:

```
init 5
```

e estabelecer as cotas por usuários ou grupos com o comando:

```
edquota login
```

que abrirá um editor com algo parecido com:

Disk quotas for user login (uid 534):

Filesystem	blocks	soft	hard	inodes	soft	hard
/dev/hda7	43460	102400	112640	482	0	0
/dev/hda9	57360	62400	72640	432	0	0

Na primeira linha tem a informação do usuário. As demais linhas são divididas em colunas com os significados:

- Filesystem informa qual(is) partição(ões) tem o sistema de cotas habilitada(s).
- blocks informa o espaço em disco já em uso pelo referido usuário.
- soft a cota em disco
- hard o limite máximo a ser atingido pelo usuário. O limite acima de *soft* e até *hard* poderá ser usado por até uma semana, após a qual o usuário não conseguirá salvar mais nenhum arquivo, só conseguirá apagar até que baixe do valor de *soft*.
- inodes informa o número de arquivos e diretórios em nome do usuário.
- soft e hard informa as cotas para número de arquivos e diretórios. Zero significa que não há limites.

Para edição da cota de grupo o processo é exatamente o mesmo sendo que deve-se adicionar a *flag* -g no comando, por exemplo:

```
edquota -g grupo
```

18.3 Estabelecendo cotas para vários usuários e/ou grupos

No caso que tenhamos que implantar cotas para vários usuários e/ou grupos não é viável ficarmos editando as cotas individualmente. Neste caso recomenda-se editar a cota para um determinado usuário padrão e replicar a mesma para os demais com o comando:

```
edquota -p padrão login
```

18.4 Verificando cotas de usuários

Podemos verificar a cota individual de algum usuário com o comando:

```
quota login
```

Para verificar a cota de todos os usuários usamos o comando:

```
repquota -a
```

19 Agendamento de tarefas com Crontab

19.1 Introdução

O "*cron*" é um programa de agendamento de tarefas. Com ele pode-se fazer a programação para execução de qualquer programa numa certa periodicidade ou até mesmo em um exato dia, numa exata hora. Um uso bem comum do cron é o agendamento de tarefas administrativas de manutenção do seu sistema, como por exemplo, análise de segurança do sistema, backup, entre outros. Estas tarefas são programadas para, todo dia, toda semana ou todo mês, serem automaticamente executadas através da crontab e um *script shell* comum. A configuração do cron geralmente é chamada de crontab.
Os sistemas Linux possuem o cron na instalação padrão. A configuração tem duas partes: uma global, e uma por usuário. Na global, que é o root quem controla, o crontab pode ser configurado para executar qualquer tarefa de qualquer lugar, como qualquer usuário. Já na parte por usuário, cada usuário tem seu próprio crontab, sendo restringido apenas ao que o usuário pode fazer (e não tudo, como é o caso do root).

19.2 Uso do Crontab

Primeiramente devemos iniciar o servidor cron:

```
service crond start
```

Para configurar um crontab por usuário, utiliza-se o comando "*crontab*", junto com um parâmetro, dependendo do que se deseja fazer. Abaixo uma relação:

Comando	**Função**
crontab -e	Edita a crontab atual do usuário logado
crontab -l	Exibe o atual conteúdo da crontab do usuário
crontab -r	Remove a crontab do usuário

Se você quiser verificar os arquivos crontab dos usuários, você precisará ser root. O comando crontab coloca os arquivos dos usuários no diretório:

```
/var/spool/cron/usuario
```

Onde "*usuario*" corresponde ao usuário dono do arquivo crontab.
Agora se deseja-se editar o crontab global, este fica no arquivo "*/etc/crontab*", e só pode ser manipulado pelo root.
Vamos estudar o formato da linha do crontab, que é quem vai dizer o que executar e quando. Vamos ver um exemplo:

```
0       4       *       *       *       who
```

A linha é dividida em 6 campos separados por tabs ou espaço:

Campo	**Função**

1o.	Minuto
2o.	Hora
3o.	Dia do mês
4o.	Mês
5o.	Dia da semana
6o.	Programa para execução

Todos estes campos, sem contar com o 6o., são especificados por números. Veja a tabela abaixo para os valores destes campos:

Campo	**Função**
Minuto	0-59
Hora	0-23
Dia do mês	1-31
Mês	1-12
Dia da semana	0-6 (o "0" é domingo, "1" segunda, etc)

Então o que nosso primeiro exemplo estava dizendo? A linha está dizendo: "*Execute o comando 'who' todo dia de todo mês sendo o dia qualquer dia da semana, às 4 horas e 0 minutos.*".
Vamos analisar mais alguns exemplos:

```
1,21,41 *       *       *       *       echo "Meu crontab rodou mesmo!"
```

Aqui está dizendo: "*Executar o comando do sexto campo toda hora, todo dia, nos minutos 1, 21 e 41*".

```
30      4       *       *       1       rm -rf /tmp/*
```

Aqui está dizendo: "*Apagar todo conteúdo do diretório /tmp toda segunda-feira, as 4:30 da manhã.*"

```
45      19      1,15    *       *       /usr/local/bin/backup
```

Aqui está dizendo: "*Executar o comando 'backup' todo dia 1 e 15 às 19:45.*".
E assim pode-se ir montando inúmeros jeitos de agendamento possível. No arquivo do crontab global, o sexto campo pode ser substituído pelo nome do usuário, e um sétimo campo adicionado com o programa para a execução, como mostrado no exemplo a seguir:

```
*/5 * * * * root /usr/bin/mrtg /etc/mrtg/mrtg.cfg
```

Aqui está dizendo: "*Executar o mrtg como usuário root, de 5 em 5 minutos sempre.*"

```
0       19-23/2      *    *       *       /root/script
```

Aqui está dizendo: “Executar o 'script' entre as 19 e 23 de 2 em duas horas.”

20 *Backups* e políticas

20.1 Introdução

Um dos principais quesitos de segurança de redes é a integridade física dos dados e informações armazenadas.

O *backup* é uma ou várias cópias de segurança dos dados, para a recuperação dos dados em caso de acidentes. Objetiva assegurar a integridade contra possíveis quedas do sistema ou problemas com o disco principal. Assegurar a recuperação de arquivos de usuários apagados/corrompidos acidentalmente ou não.

Existem várias formas de se garantir a disponibilidade da informação, a mais importante sem dúvidas é a cópia destes dados em local seguro, ou seja, o *backup* de dados, pois traz flexibilidade à instituição de, a qualquer momento, voltar no tempo com os seus dados ou ao menos deveria isto ser possível.

O conceito de um local seguro por muitas vezes é o maior ponto de variação dentro do assunto *backup* e este merece atenção especial, pois por muitas vezes pensamos que o local seguro possa ser a torre do prédio ao lado, o que nem sempre é verdade.

Existem várias formas de se fazer o *backup* dos dados. Formas simples e baratas para pequenas empresas e usuários domésticos, que possuem poucas informações. Formas mais complexas e caras nas médias e grandes corporações, onde a quantidade de informações é imensa e também precisa de um *backup* desses dados. Isso nos leva a questão de política de *backup* e forma de armazenamento, onde existe então esta variação de custo X segurança. Entre estes pontos é possível se chegar a extremos de confiabilidade o que por muitas vezes é diminuído devido ao custo da solução. A escolha de uma boa política aliada a uma forma de armazenamento suficientemente adequada a situação pode trazer ao administrador um custo compatível com o valor da informação que ele deseja salva-guardar.

20.2 Tipos de backup

O tipo de backup a ser utilizado varia de acordo com cada organização, dependendo da quantidade de informação, e da velocidade que estas informações são atualizadas, cabe ao administrador de rede e/ou gestor de política de segurança analisar e definir a melhor forma. Basicamente existem 3 tipos.

20.2.1 Backups totais

Um *backup* total captura todos os dados, incluindo arquivos de todas as unidades de disco rígido. Cada arquivo é marcado como tendo sido submetido a backup; ou seja, o atributo de arquivamento é desmarcado ou redefinido. Uma fita atualizada de *backup* total pode ser usada para restaurar um servidor completamente em

um determinado momento.
Vantagens

- Cópia total dos dados - Isso significa que você tem uma cópia completa de todos os dados e se for necessária uma recuperação do sistema torna-se mais prático.
- Acesso rápido aos dados de *backup* - Você não precisa pesquisar em várias fitas para localizar o arquivo que deseja restaurar, porque os *backups* totais incluem todos os dados contidos nos discos rígidos em um determinado momento.

Desvantagens

- Dados redundantes - *Backups* totais mantêm dados redundantes, porque os dados alterados e não alterados são copiados para fitas sempre que um *backup* total é executado.
- Tempo - *Backups* totais levam mais tempo para serem executados e podem ser muito demorados.

20.2.2 Backups incrementais

Backup incremental captura todos os dados que foram alterados desde o *backup* total ou incremental mais recente. Você deve usar uma fita de *backup* total (não importa há quanto tempo ela tenha sido criada) e todos os conjuntos de *backups* incrementais subseqüentes para restaurar um servidor. Um *backup* incremental marca todos os arquivos como tendo sido submetidos a *backup*; ou seja, o atributo de arquivamento é desmarcado ou redefinido.
Vantagens

- Uso eficiente do tempo - O processo de *backup* leva menos tempo porque apenas os dados que foram modificados ou criados desde o último *backup* total ou incremental são copiados para a fita.
- Uso eficiente da mídia de *backup* - O *backup* incremental usa menos fita porque apenas os dados que foram modificados ou criados desde o último *backup* total ou incremental são copiados para a fita.

Desvantagens

- Restauração completa complexa - Você pode precisar restaurar os dados de um conjunto incremental de várias fitas para obter uma restauração completa do sistema.
- Restaurações parciais demoradas - Você pode ter que pesquisar em várias fitas para localizar os dados necessários para uma restauração parcial.

20.2.3 Backups diferenciais

Um *backup* diferencial captura os dados que foram alterados desde o último *backup* total. Você precisa de uma fita de *backup* total e da fita diferencial mais recente para executar uma restauração completa do sistema. Ele não marca os

arquivos como tendo sido submetidos a *backup* (ou seja, o atributo de arquivamento não é desmarcado).
Vantagem

- Restauração rápida – A vantagem dos *backups* diferenciais é que eles são mais rápidos do que os *backups* incrementais, porque há menos fitas envolvidas. Uma restauração completa exige no máximo dois conjuntos de fitas — a fita do último *backup* total e a do último *backup* diferencial.

Desvantagens

- *Backups* demorados e maiores - *Backups* diferenciais exigem mais espaço em fita e mais tempo do que *backups* incrementais porque quanto mais tempo tiver se passado desde o *backup* total, mais dados haverá para copiar para a fita diferencial.
- Aumento do tempo de *backup* - A quantidade de dados dos quais é feito *backup* aumenta a cada dia depois de um *backup* total.

20.3 Modos de *backup*

O modo de *backup* determina como o *backup* deve ser executado em relação ao tipo de dados a serem incluídos nele. Há duas maneiras de executar os *backups* de dados.

20.3.1 Backups on-line

São *backups* feitos em servidores que precisam estar 24h por dia disponível aos usuários. Geralmente são banco de dados, servidores de e-mail, etc. Um detalhe bastante importante é que o software de *backup* e a aplicação precisam ter suporte a este tipo de *backup*.

Vantagem

- Servidor sempre disponível podendo ser realizado o *backup* durante o expediente normal de trabalho.

Desvantagem

- O desempenho do servidor é prejudicado durante a realização do *backup*.

20.3.2 Backups offline

São *backups* de dados feito quando ninguém está tentando acessar as informações. Geralmente é agendado para ser realizado à noite.

Vantagem

- Como o servidor estará apenas fazendo o *backup* dos dados é mais rápido que o processo de *backup on-line*.

Desvantagem

- Ninguém poderá acessar os dados durante a execução do *backup*.

20.4 Armazenamento

Unidades de armazenamento costumam ser a parte mais almejada e menos realizada ao se pensar em uma estrutura de *backups*, isto porque a qualidade e tamanho da mesma costuma subir de acordo com seu custo, trazendo muitas vezes a solução para o nível considerado bom e não ótimo. Hoje quando falamos em unidades de armazenamento também precisamos pensar em local para guardarmos estas unidades.

É de extrema importância que seja qual for a tecnologia utilizada estejamos seguros de que ela esta sendo corretamente armazenada, as unidades de *backup* não podem ser simplesmente guardadas em um caixa de papelão localizada sobre o monitor do servidor ou se quer na mesma sala. É idealizado o fato de se guardar em um cofre, em local distante e resistente a no mínimo fogo, umidade e tremores. Uma boa referencia são as salas cofres descritas na ISO 17799.

No mercado temos disponíveis para unidade de armazenamento fitas magnéticas, discos rígidos e discos ópticos, dentro de cada categoria existem prós e contras.

20.4.1 Discos Rígidos

Os discos rígidos são uma ótima unidade no ponto de vista do custo por MB, esta solução tem sido utilizada para realização do "*backup* incremental para sempre" (RAID). Possui alta capacidade de armazenamento e boa velocidade de acesso. Esta solução não se torna confiável como única forma de *backup* devido a sua grande sensibilidade e durabilidade baixa.

20.4.2 Unidades de Fitas

Certamente é a categoria mais indicada para *backups* "profissionais". Existem os mais variados modelos e fabricantes com as mais variadas capacidades de armazenamento e velocidades de acesso/gravação.

20.4.3 CD e DVD

Unidades com baixa capacidade de armazenamento. Sua vida esperada é de 75 anos em casos de extremos cuidados. Pode ser utilizado como forma de *backup* rápido e para pouco tempo de retenção. Adequado para uso doméstico.

20.5 Políticas de *backup*

Uma política de *backup* tem a função de formalizar todos os procedimentos técnicos e não técnicos de uma cópia de segurança dos dados. Dentro dele devem estar disponíveis informações sobre o que é feito no backup, dos tempos, validação dos dados e armazenagem.

Definir a política de *backup* não é um procedimento puramente técnico, no qual não compete somente a um administrador de redes a sua construção. Esse ponto é importante devido ao fato de que nesta política deverá estar descrito

exatamente o que deverá ser salvo.

Estamos agora entrando no maior ponto de problemas relacionado a *backup*, o que salvar e o que não salvar. Quando entramos nesta discussão fica claro que o melhor, para garantir, é aquela “vamos salvar tudo”, mas “tudo” é o que? Se pudermos salvar “tudo” ótimo, e cabe a uma política definir o “tudo”, informando caminhos completos destes arquivos, da forma lógica computacional, e então entra o administrador, é importante que haja completa descrição para que a empresa toda possa saber onde deixar seus arquivos seja na rede ou na sua estação tendo a certeza de que neste local as informações estão seguras contra perda.

Devido a limitações técnicas, é impossível que façamos a cópia de segurança dos dados a cada segundo, o mesmo falamos com relação à restauração, e ainda, seja por falta de espaço físico ou limitação financeira, nem todas estas cópias poderão ser mantidas e a tarefa quando falamos em tempo na política de *backup* é definir os seguintes pontos:

- Quando esta cópia será feita -Neste ponto, precisamos detalhar de quanta em quantas horas, dias, semanas ou meses os dados serão salvos, se existirem diferentes tempos de *backup* para os dados, estes também necessitam de especificação. Definir este tempo inclui novamente mais pessoas além do administrador, que será responsável por dar o aval de que é tecnicamente possível realizar a cópia no tempo em que gerentes definirem que a sua informação precisa ser enviada a um local de segurança.
- Quanto tempo demora esta cópia (*Backup Window*) - É necessário saber quanto tempo leva para o *backup* ser concluído, para que assim, em caso de uma perda no horário de *backup*, se saiba antes da restauração qual *backup* que contém dados íntegros. Hoje já consideramos adicionalmente para contornar este tamanho de janela o chamado *backup* incremental para sempre, a implementação destes é através de RAID ou dentro de sistemas de *storage* empresariais através de *snapshots*.
- Quanto tempo uma recuperação de dados irá levar - Além de saber os tempos relacionados ao procedimento de cópia, é importante que se tenha conhecimento do tempo que demora a efetuar a restauração dos dados do *backup*.
- Por quanto tempo uma cópia estará disponível - Seja o local que for armazenado este *backup*, sabe-se que haverá limitações, seja de espaço físico ou lógico ou realmente de quantidade de unidades de armazenamento. Isto nos leva a realizar um rodízio no *backup*, no qual para se salvar novos dados, abre-se mão de outros considerados mais antigos. A política então deve ser capaz de informar em quanto tempo um dado é antigo e por quanto tempo este dado antigo precisa ser mantido. O modelo de rotacionamento mais genérico e famoso que temos é o “*Grandfather-father-son* – GFS” que se define em realizar *backups* diários “filhos” (incrementais), *backups* semanais “pais” (full) e um mensal (full), o “avô”.

Estes itens são considerados como básicos a uma política genérica, em

determinados casos pode se tornar necessário adicionar mais pontos a esta, como por exemplo, segmentando um *backup* por setores, ou filiais, adicionando responsáveis ao processo. Salientamos que a política de *backup* assim como uma política de ética empresarial precisa ser ajustada de acordo com a empresa, os dados considerados valiosos de uma pode não ser o de outra, e principalmente, o tempo de retenção da cópia de dados.

20.6 O sistema Amanda

O **Amanda** (*Advanced Maryland Automatic Network Disk Archiver*) é um sistema de *backup* cliente/servidor, melhor que um único programa. Um servidor **Amanda** irá realizar numa única controladora de fita o *backup* de qualquer número de computadores que tenham o cliente do **Amanda** e uma conexão de com o servidor **Amanda**. Um problema comum em locais com um grande número de discos é que a quantidade de tempo requerida para o *backup* dos dados diretamente na fita excede a quantidade de tempo para a tarefa. O **Amanda** resolve este problema utilizando um disco auxiliar para realizar o *backup* de diversos sistemas de arquivos ao mesmo tempo. O **Amanda** cria "conjuntos de arquivos": um grupo de fitas utilizadas sobre o tempo para criar os *backups* completos de todos os sistemas de arquivos listados no arquivo de configuração do **Amanda**. O "conjunto de arquivos" também pode conter *backups* incrementais (ou diferenciais) noturnos de todos os sistemas de arquivos. Para restaurar um sistema de arquivos é necessário o *backup* completo mais recente e os incrementais, este controle é feito pelo próprio **Amanda**.

O arquivo de configurações provê um controle total da realização dos *backups* e do tráfego de rede que o **Amanda** gera. O **Amanda** utilizará qualquer programa de *backup* para gravar os dados nas fitas, por exemplo tar. O **Amanda** está disponível como pacote, porém ele não é instalado por padrão.

20.7 Configuração do servidor Amanda

http://wiki.zmanda.com/index.php/Main_Page
http://www.amanda.org/docs/
Para instalar o Amanda devemos nos certificar que as mídias remotas estão devidamente configuradas e proceder a instalação normal, com o comando:

```
urpmi -a amanda
```

Após a instalação o arquivo padrão /etc/amanda/DailySet1/amanda.conf, estará criado. Como exemplo configuraremos o sistema para um backup fictício, já que não dispomos de unidades de fita para um verdadeiro backup.
Para "*Backup* em HD com Amanda" pode-se ver um tutorial em http://www.sj.ifsc.edu.br/wiki/index.php/Backup_em_HD_usando_o_Amanda. Muito útil também o tutorial http://www.amanda.org/docs/howto-filedriver.html.

20.7.1 amanda.conf

O Amanda é bastante flexível. Poderíamos ter uma série de conjuntos independentes de backup no mesmo servidor. Para isto bastaria criar tantos

diretórios de configuração quanto tivéssemos necessidade, a exemplo do ifsc. Cada um tendo características independentes de periodicidade e dados a serem armazenados.
Em nosso exemplo vamos criar um único conjunto de *backups*. Para isso faça uma cópia do exemplo /etc/amanda/DailySet1 para ifsc com o comando:

```
cp -rf /etc/amanda/DailySet1 /etc/amanda/ifsc
```

Edite o amanda.conf do ifsc e modifique somente as linhas abaixo, as demais definições são as padrão:

```
org "ifsc"
mailto "suporte@sj.ifsc.edu.br odilson@sj.ifsc.edu.br"
tapecycle 21 tapes     #Número de fitas disponíveis.
tapedev "/dev/nst0"    #Dispositivo da fita.
tapetype DDS4          #Tipo de fita
labelstr "^ifsc[0-9][0-9]*$"    #Rótulos para fitas = ifsc01 à 99.
infofile "/var/lib/amanda/ifsc/curinfo"       #database filename
logdir   "/var/lib/amanda/ifsc"        #log directory
indexdir "/var/lib/amanda/ifsc/index"       #index directory
dumpcycle 4 weeks       # o número de dias num ciclo de dump normal
```

20.7.2 disklist

```
Edite o /etc/amanda/ifsc/disklist, vá ao final do arquivo e
acrescente todos os diretórios, de todas as máquinas, que deseja
backup, veja o exemplo:
     dk.uned.sj /scripts comp-root-tar
     dk.uned.sj /etc comp-root-tar
     [....]
     hendrix.sj.ifsc.edu.br /etc comp-root-tar
     hendrix.sj.ifsc.edu.br /root comp-root-tar
     [....]
     titas.uned.sj   /var/www comp-root-tar
     titas.uned.sj   /home comp-root-tar
     [....]
```

Neste caso estaríamos configurando o backup do diretório /scripts e /etc da máquina dk.uned.sj o /etc e /root da hendrix.sj.ifsc.edu.br e /var/www e /home da titas.uned.sj. É evidente que os clientes deverão responder por estes nomes (DNS) e ter o cliente **Amanda** configurado, conforme procedimento abaixo.

20.8 Configurando o cliente

Normalmente desejamos que o servidor **Amanda** também seja cliente. Em nosso caso já instalamos os pacotes necessários no item “Configuração do Servidor Amanda”. Num caso de cliente esplícito devemos instalar os pacotes com o seguinte comando:

```
urpmi -a libam amanda-client
```

Após isto devemos editar o arquivo /etc/xinetd.d/amanda e modificarmos uma única diretiva como abaixo:

```
disable = no
```

Reiniciamos o serviço com o comando:
```
service xinetd restart
```

Agora Vamos testar as configurações, no servidor, com o comando:
```
/usr/sbin/amcheck ifsc
```

Se tudo estiver correto obteremos uma mensagem do tipo:
```
Amanda Tape Server Host Check

-----------------------------

Holding disk /amanda: 22985136 kB disk space available, that's plenty

NOTE: skipping tape-writable test

Tape ifsc06 label ok

WARNING: tapecycle (21) <= runspercycle (21).

Server check took 0.023 seconds



Amanda Backup Client Hosts Check

--------------------------------

Client check: 3 hosts checked in 0.090 seconds, 0 problems found



(brought to you by Amanda 2.4.5)
```

20.9 Backups com o Amanda

Para fazermos backup dos diretórios usamos o **amdump** com a sintaxe:
```
amdump ifsc
```

Este também é o comando que deve ser programado na crontab para os *backups* periódicos.

20.10 Restaurando os *backups* com o Amanda

Uma forma de restaurar os *backups* é utilizando o '**amrecover**'. Ela é a opção mais poderosa, por isso deve ser dado preferência ao uso da mesma.
Na máquina servidora, como root, crie um diretório de restore:
```
mkdir restore
```

```
cd restore
```

Agora, chame o programa de recuperação:
```
amrecover ifsc
```

Dê o comando que determina a data que você quer restaurar, no formato AAAA-MM-DD, onde AAAA é o ano com quatro dígitos, MM é o mês com dois dígitos e DD é o dia com dois dígitos:
```
setdate 2006-09-05
```

O próximo passo é determinar de qual cliente se quer restaurar o *backup*:
```
sethost localhost
```

A seguir determina-se de qual "disco" se restaurará o *backup*. Abaixo, um exemplo, de como extrair somente um arquivo do diretório /etc:
```
setdisk /etc
```

Agora pode-se navegar pelos diretórios:
```
cd rc.d
```

O próximo passo é adicionar o arquivo a ser restaurado. Lembre-se que você pode usar coringas, como o * para adicionar todos os arquivos, ou pode adicionar um diretório:
```
add rc.local
```

Continue adicionando diretórios e arquivos, conforme o necessário. Depois disso resta extrair os arquivos:
```
extract
```

20.11 Comandos Extras do Amanda

Para descarregar o disco de suporte, temporário, devemos fazer uso do comando **amflush**.
Para limpeza geral no serviço Amanda execute o **amcleanup**.
Para gravar rótulos em fitas use o **amlabel**. Esta operação é muito importante para não haver sobreposição de dados pelo sistema amanda. As fitas, tendo rótulo, serão manipuladas devidamente pelo Amanda.
Para tarefas administrativas **amadmin**.
Amcheck verifica de você está utilizando a fita correta (esperada), se há espaço livre suficiente no disco de suporte e se as máquinas clientes estão configuradas adequadamente.
Para gerenciamento de empilhadores e trocadores de fita, se houver, use o **amtape**.
Para esboçar gráfico da atividade do Amanda em cada *dump* que for executado use o **amplot**.

21 Programação do Shell[7]

21.1 Introdução

- Por que programação shell?
  - Permite que administradores criem pequenos programas para automatizar a administração do sistema
  - Permite configurar melhor o sistema
    - Por exemplo, toda a configuração da seqüência de boot do UNIX envolve scripts shell
- O que é programação shell
  - Criação de um arquivo de texto contendo comandos do UNIX e comandos especiais do shell para:
    - Variáveis
    - Testes
    - Laços
    - Funções
    - Comentários
- Como executar?
  - Interpretação do arquivo de script por um shell
  - Interpretação pode ser lenta?
    - Sim! E lembre que cada comando que o script chama dispara um processo
    - Uma alternativa mais rápida: programar em C ou Java
    - Uma alternativa boa entre os dois extremos: perl

21.2 Scripts Shell

- O primeiro programa
  - O texto abaixo está no arquivo "alo"

```
#!/bin/bash
# Este programa diz alo
echo "Alo $LOGNAME, tenha um bom dia!"
```

- Execute o programa (Obs.: $ significa o *prompt* de comando):

```
$ ./alo
bash: ./alo: Permission denied
$ bash alo
Alo  jacques, tenha um bom dia!
$ chmod a+x alo
$ ./alo
Alo  jacques, tenha um bom dia!
$ PATH=$PATH:.
$ alo
Alo  jacques, tenha um bom dia!
```

7 Texto obtido de http://walfredo.dsc.ufpb.br/cursos/suporte20012/progsh/index.htm

21.3 Variáveis e Parâmetros

21.3.1 Algumas variáveis pré-definidas

- $LOGNAME, $HOSTNAME, $TERM, ...
- $$ (o process id do processo corrente - o shell)

```
$tempfile=/tmp/temp.$$
$ls > $tempfile
$echo $tempfile
      /tmp/temp.4171 (isto é um exemplo de saída do comando!)
$cat /tmp/temp.4171
      Desktop/
      Documentos/
      ....
$rm $tempfile
```

21.3.2 Substituição avançada de variáveis

Construção	Propósito
`${variavel:-valor}`	o valor da construção é o valor da variável, se houver, ou "valor" caso contrário. O valor da variável não muda
`${variavel:?mensagem}`	se variável não tiver valor exibe a mensagem de erro

21.3.3 Parâmetros

Quando um script é chamado, pode receber parâmetros na linha de comando

```
$ vi soma
      #!/bin/bash
      val=`expr ${1:-0} + ${2:-0} + ${3:-0}`
      echo A soma eh $val
      $ soma 2 3 5
      A soma eh 10
      $ soma 2 3
      A soma eh 5
      $ soma
      A soma eh 0
```

Isso é equivalente:

```
      #!/bin/bash
      echo A soma eh `expr ${1:-0} + ${2:-0} + ${3:-0}`
```

Tratamento de parâmetros

Variável	Propósito
$0	O nome do programa shell
$1 até $9	Os primeiros 9 parâmetros
$#	O número de parâmetros
$*	Todos os parâmetros do programa

Observe que $10 não é o parâmetro 10, é $1 concatenado com "0".
Isso não significa que só podemos ter 9 parâmetros: não há limite (quase).
Veremos como acessar os demais parâmetros depois.
Exemplo:

```
$ vi l
      #!/bin/bash
      ls -l $*
$ ./l
<mostra o conteúdo do diretório corrente>
$ ./l ./ ../
<mostra o conteúdo do diretório corrente e do diretório pai>
```

Como formar parâmetros

- Os exemplos abaixo parecem iguais mas o primeiro tem 4 parâmetros, o segundo tem 1 parâmetro e o terceiro tem 3 parâmetros

```
$ echo a b c d
      a b c d
$ echo "a b c d"
      a b c d
$ echo "a b" c d
      a b c d
```

21.4 Entrada-Saída Básica

A saída é feita basicamente da saída de outros comandos que estão no script.
Se precisar de um "print", usa-se o comando echo.
A leitura de informação da entrada padrão é feita com o comando do shell "read".

```
$ cat testaread
      #!/bin/bash
      read x
      echo Voce falou $x
$ cat lenome
      #!/bin/bash
      echo -n "Favor digitar seu nome: "
      read nome
      echo "Seu nome eh $nome"
```

21.5 Testes

21.5.1 Um problema a resolver

Você acabou de ser nomeado administrador de sistema numa empresa. O administrador anterior saiu correndo quando descobriu que um servidor principal da empresa estava sendo usado para guardar pornografia, warez (software pirata) e instruções para a fabricação de bombas caseiras. Suspeita-se que várias pessoas da empresa, além do ex-administrador de sistema, estavam envolvidas com a captação dessa informação. Pediu-se que você fizesse um programa para identificar pessoas que acessam sites "indesejáveis" e avisá-los da política de empresa acerca do uso apropriado da Internet com recursos da empresa.

Idealmente, você deve produzir um relatório informando quem acessa quais sites restritos e quantas visitas foram feitas. A informação a ser analisada está num arquivo "netwatch" que contém uma lista das pessoas e dos acessos que elas fizeram .
Um exemplo do arquivo segue abaixo:

```
ARQUIVO: netwatch
      jamiesob mucus.slime.com
      tonsloye xboys.funnet.com.fr
      tonsloye sweet.dreams.com
      root sniffer.gov.au
      jamiesob marvin.ls.tc.hk
      jamiesob never.land.nz
      jamiesob guppy.pond.cqu.edu.au
      tonsloye xboys.funnet.com.fr
      tonsloye www.sony.com
      janesk horseland.org.uk
      root www.nasa.gov
      tonsloye warez.under.gr
      tonsloye mucus.slime.com
      root ftp.ns.gov.au
      tonsloye xboys.funnet.com.fr
      root linx.fare.com
      root crackz.city.bmr.au
      janesk smurf.city.gov.au
      jamiesob mucus.slime.com
      jamiesob mucus.slime.com
```

Os sites proibidos estão contidos num arquivo "netproib":

```
ARQUIVO: netproib
      mucus.slime.com
      xboys.funnet.com.fr
      warez.under.gr
      crackz.city.bmr.au
      www.hotwarez.com.br
```

Desenvolva um script que satisfaça os requisitos. Você pode ignorar qualquer consideração de ética, privacidade e censura.

21.5.2 Comandos de testes no shell

```
if comando
then
  comandos executados se "comando" retornar status "ok" (0)
else
  comandos executados se "comando" retornar status "não ok" (diferente de 0)
fi

if comando1
then
  comandos executados se "comando1" retornar status "ok" (0)
elif comando2
  comandos executados se "comando2" retornar status "ok" (0)
else
  comandos executados se não entrar nos "if" acima
fi
```

```
comando1 && comando2
# construção na linha acima eh equivalente a (porta E):
if comando1
then
  comando2
fi

comando1 || comando2
# construção na linha acima eh equivalente a (porta OU):
if comando1
then
  :
else
  comando2
fi
```

21.5.3 Início do script scan

```
#!/bin/bash
# ARQUIVO: scan

if [ $# -eq 0 ]; then
  echo "Falta parametro - Rode o script ./scan.sh usuario1 [usuario2 ...]"
  exit
fi

if ls netwatch && ls netproib
then
  echo "Achei netwatch e netproib"
else
  echo "Nao pode achar um dos arquivos de dados - caindo fora"
  exit 1
fi
```

O comando exit encerra o shell e retorna um status para o processo "pai".

21.5.4 Testes com [...]

O comando "test" e a construção [...] são equivalentes e permitem testar condições envolvendo strings e arquivos.
Exemplo, o último script poderia ser reescrito como:

```
#!/bin/bash
# ARQUIVO: scan
if [ -r netwatch ] && [ -r netproib ]
then
  echo "Achei netwatch e netproib"
else
  echo "Nao pode achar um dos arquivos de dados - caindo fora"
  exit 1
fi
```

As opções de [...] seguem:

- Testes de strings

Expressão	Verdeiro se
`-z string`	tamanho de string é 0
`-n string`	tamanho de string não é 0
`string1 = string2`	dois strings são iguais
`string != string2`	dois strings não são iguais
`string`	string não é nulo

- Testes numéricos

Expressão	Verdeiro se
`int1 -eq int2`	O primeiro inteiro é igual ao segundo
`int1 -ne int2`	O primeiro inteiro NÃO é igual ao segundo
`int1 -gt int2`	O primeiro inteiro é maior que segundo
`int1 -ge int2`	O primeiro inteiro é maior ou igual ao segundo
`int1 -lt int2`	O primeiro inteiro é menor que segundo
`int1 -le int2`	O primeiro inteiro é menor ou igual ao segundo

- Testes de arquivos

Expressão	Verdeiro se
`-r file`	file existe e é pode ser lido
`-w file`	file existe e é pode ser gravado
`-x file`	file existe e é pode ser executado
`-f file`	file existe e é normal (regular)
`-d file`	file existe e é diretório
`-h file`	file existe e é link simbólico
`-c file`	file existe e é dispositivo especial a
`-b file`	file existe e é dispositivo especial a bloco
`-p file`	file existe e é pipe
`-u file`	file existe e tem bit setuid ligado
`-g file`	file existe e tem bit setgid ligado
`-k file`	file existe e tem bit sticky ligado
`-s file`	file existe e tem tamanho diferente de zero

- Operadores adicionais

Expressão	Propósito
`!`	inverte expressão lógica
`-a`	operador AND

-o	operador OR
(expr)	agrupar expressões

21.5.5 A construção case

```
echo -n "Sua reposta: "
read resp
case "$resp" in
Y* | y*)
  resp="sim"
  ;;
N* | n*)
  resp="nao"
  ;;
*)
  resp="talvez"
  ;;
esac
echo $resp
```

Observe o casamento de expressões regulares, acima.

21.6 Laços

21.6.1 O comando while

```
while comando
do
  comandos executados enquanto "comando" retornar status "ok"
done
```

O que o seguinte programa faz?

```
#!/bin/bash
# ARQUIVO: list
#
numLinha=1
while read linha
do
  echo "$numLinha $linha"
  numLinha=`expr $numLinha + 1`
done < $1
```

R. Lista o arquivo com numeração de linhas!

O que o seguinte programa faz?

```
while read linha
do
  usuario=`echo $linha | cut -d" " -f1`
  site=`echo $linha | cut -d" " -f2`
  if [ "$usuario" = "$1" ]
  then
    echo "$usuario visitou $site"
```

```
  fi
done < netwatch
```

R. Dado um usuário informa quais sites visitados pelo mesmo!

21.6.2 O comando for

```
for var in lista_de_palavras
(também pode ser usado: for var in `cat /root/lista_de_palavras)
do
  no corpo do loop, temos $var = a próxima palavra da lista
done
```

O que o seguinte programa faz?

```
for verifUsuario in $*
do
  while read linha
  do
    while read verifSite
    do
      usuario=`echo $linha | cut -d" " -f1`
      site=`echo $linha | cut -d" " -f2`
      if [ "$usuario" = "$verifUsuario" -a "$site" = "$verifSite" ]
      then
        echo "$usuario visitou o site proibido $site"
      fi
    done < netproib
  done < netwatch
done
```

R. Dado um, ou mais usuários, informa quais os sites proibidos foram visitados pelo(s) mesmo(s).

21.7 Funções

Uma nova versão do nosso trabalho.

```
#!/bin/bash
# ARQUIVO: scan
#

verificaArquivos() {
  if [ -r netwatch -a -r netproib ]
  then
    return 0
  else
    return 1
  fi
}

# Programa principal

if verificaArquivos
then
  echo "Arquivos de dados achados"
else
```

```
  echo "Nao pode achar um dos arquivos de dados - caindo fora"
  exit 1
fi

# o resto do trabalho
```

Outra versão

```
#!/bin/bash
# ARQUIVO: scan
#

verificaArquivos() {
  [ -r netwatch -a -r netproib ]
  return $?
}

# Programa principal

if verificaArquivos
then
  echo "Arquivos de dados achados"
else
  echo "Nao pode achar um dos arquivos de dados - caindo fora"
  exit 1
fi

# o resto do trabalho
```

21.8 Sinais e Traps

Sinais são enviados para processos de várias formas:

- Pelo kernel, quando um processo faz besteira
- Pelo usuário, usando o teclado (^C, ^\, encerrando a sessão)
- Usando o comando kill
- Ação normal: o processo morre

Porém, um processo pode ignorar os sinais ...
... ou captura-los para fazer algo
Quais são os sinais?

- Os sinais em negrito merecem discussão aqui, pois o administrador deve entendê-los

Sinal	Significado
0	**Fim do shell**
1	**Hangup**
2	**Interrupt**(^C)
3	**Quit**(^\)
4	Illegal Instruction
5	Trace trap
6	IOT instruction

7	EMT instruction
8	**Floating point exception**(bug de programa)
9	**Morte certa** (kill -9)
10	**Bus error**(bug de programa)
11	**Violação de segmentação**(bug de programa)
12	Bad argument
13	Pipe write error
14	Alarm
15	**Software termination signal** (kill sem

Sintaxe de captura de sinais

```
trap "comandos" sinais ...
```

Exemplo simples de captura de sinais

```
#!/bin/bash
tempfile=/tmp/temp.$$
trap "rm -f $tempfile" 0 1 2
ls -l > $tempfile
cat $tempfile
# não precisa remover temp no fim: o trap se encarrega disso
```

21.9 Depuração

- O comando "set -x" liga o rastreamento da execução de comandos
- O comando "set +x" desliga o rastreamento

21.10 Técnicas Avançadas

Por enquanto, nosso código de "scan" é o seguinte:

```
#!/bin/bash
# ARQUIVO: scan
#

if [ $# -eq 0 ]; then
  echo "Falta parametro - Rode o script ./scan.sh usuario1 [usuario2 ...]"
  exit
fi

verificaArquivos() {
  [ -r netwatch -a -r netproib ]
  return $?
}

# Programa principal

if verificaArquivos
then
  echo "Arquivos de dados achados"
else
```

```
  echo "Nao pode achar um dos arquivos de dados - caindo fora"
  exit 1
fi

for verifUsuario in $*
do
  while read linha
  do
    while read verifSite
    do
      usuario=`echo $linha | cut -d" " -f1`
      site=`echo $linha | cut -d" " -f2`
      if [ "$usuario" = "$verifUsuario" -a "$site" = "$verifSite" ]
      then
        echo "$usuario visitou o site proibido $site"
      fi
    done < netproib
  done < netwatch
done
```

Ainda não satisfizemos todos os requisitos, pois falta a contagem do número de visitas ...

21.10.1 eval

Passa duas vezes numa linha de comando. Só executa depois da segunda passagem

```
$ pipe=\|
$ eval ls $pipe more
<mostra o mesmo conteúdo que ls | more, pois o comando eval faz o shell ler o
conteúdo da variável "|", que é um caractere especial>
$ ls $pipe more
<erro, pois o valor de $pipe é | então o shell tenta fazer um ls nas strings "|" e
"more" que não existem>
```

21.10.2 Voltando ao programa scan

Como fazer a contagem de visitas?
Se um usuário jacques visitar www.hotwarez.com.br, teremos a contagem de visitas feita na variável $jacqueswwwhotwarezcombr
Segue o programa final eliminando a necessidade de parâmetros de entrada:

```
#!/bin/bash
logFile=netwatch
proibFile=netproib
passwdFile=$passwdFile

processaLogFile() {
  # Varre o arquivo $logFile e salva combinacoes usuario/site
  # para sites que estao na lista de sites proibidos

  while read linha
  do
    nomeUsuario=`echo $linha | cut -d" " -f1`
    site=`echo $linha | cut -d" " -f2 | sed s/\\\.//g`
    while read verifSite
```

```
    do
      verifSite=`echo $verifSite | sed s/\\\.//g`
      # echo $verifSite $site
      if [ "$site" = "$verifSite" ]
      then
        usuarioSite="$nomeUsuario$verifSite"
        if eval [ \$$usuarioSite ]
        then
          eval $usuarioSite=\`expr \$$usuarioSite + 1\`
        else
          eval $usuarioSite=1
        fi
      fi
    done < $proibFile
  done < $logFile
}

relatorio() {
  # Ve todas as combinaçes de usuários e sites restritos
  # Se uma variavel de visitas existir, inclua no relatório
  for usuario in $*
  do
    while read siteVisitado
    do
      verifSite=`echo $siteVisitado | sed s/\\\.//g`
      usuarioSite="$usuario$verifSite"
      if eval [ \$$usuarioSite ]
      then
        eval echo "$usuario: $siteVisitado \$$usuarioSite"
        numUsuarios = `expr $numUsuarios + 1`
      fi
    done < $proibFile
  done
}

usuariosEmPasswd() {
  # cria uma lista de usuarios a partir do arquivo de passwd
  cut -d":" -f1 < $passwdFile
}

verificaArquivos() {
  [ -r netwatch -a -r netproib ]
  return $?
}

# Programa principal
# remover comentario na linha seguinte para depurar
#set -x


if verificaArquivos
then
  echo "Arquivos de dados achados"
else
  echo "Nao pode achar um dos arquivos de dados - caindo fora"
  exit 1
fi
```

```
numUsuarios=0

if [ $# -gt 0 ]
then
  listaUsuarios=$*
else
  listaUsuarios=`usuariosEmPasswd`
fi

echo
echo "*** Relatorio de Acesso a Sites Proibidos ***"
echo
echo "

Segue uma lista de usuarios que visitaram sites proibidos,
o site visitado e o numero de visitas

"
processaLogFile
relatorio $listaUsuarios

echo

if [ $numUsuarios -eq 0 ]
then
  echo "Nao houve acesso a sites proibidos"
else
  echo "$numUsuarios combinacoes de usuario/site descobertas"
fi
```

Na realidade, este programa seria mais simples usando perl.
O truque do "eval" é muito feio!
Aqui, serviu para mostrar construções avançadas do shell.
Poderíamos ainda tratar uma opção do programa "-d" para ligar o debug.

```
# Programa principal
if [ "$1" = "-d" ]
then
  set -x
  shift
}
...
```

21.11 Um Exemplo Final

21.11.1 Definição do Problema

Seu servidor FTP provê acesso a arquivos na área /pub do sistema de arquivos.
Você desconfia que poucas pessoas estão usando o serviço FTP (devido à WWW) e você gostaria de examinar o log FTP usando uma ferramenta útil).
Você quer um programa que examine o log FP e produza estatísticas de uso sobre um certo tópico. As estatísticas devem incluir:

- O número de acessos por usuário
- O número de bytes transferidos
- O número de hosts que acessaram o serviço

O script deve aceitar uma palavra chave (o tópico de interesse) e um comando dizendo qual é a estatística desejada

- O comando pode ser: "users", "bytes" ou "hosts"

21.11.2 Informação adicional necessária

Suponha que o arquivo de log tenha o seguinte conteúdo, um acesso por linha

- nomeUsuário poderá ser uma identificação fornecida se o FTP for anônimo

```
hostRemoto tamanhoEmBytes nomeDoArquivo nomeUsuário
```

Exemplo do arquivo de log /var/log/ftp.log

```
aardvark.com 2345 /pub/85349/lectures.tar.gz flipper@aardvark.com
138.77.8.8 112 /pub/81120/cpu.gif sloth@topaz.cqu.edu.au
```

Podemos supor que o nome do arquivo sempre iniciará com /pub.
Exemplos de interação com o programa:

```
$ scanlog 85321 users
jamiesob@jasper.cqu.edu.au 1
b.spice@sworld.cqu.edu.au 22
jonesd 56

$ scanlog 85321 bytes
2322323

$ scanlog 85321 hosts
5

$ scanlog 85321 bytes users
2322323
jamiesob@jasper.cqu.edu.au 1
b.spice@sworld.cqu.edu.au 22
jonesd 56
```

O programa segue:

```
#!/bin/sh

LOGFILE="ftp.log"
TEMPFILE="/tmp/scanlog.$$"
trap "rm $TEMPFILE" 0 1 2
# funcoes

#---------------------------------------
# printNumHosts
# - mostra o numero de maquinas unicas que acessaram o topico

printNumHosts() {
  cut -f1 $TEMPFILE | uniq | wc -l
}

#-----------------------------------------------------
# printUsers
```

```
# - mostra os usuarios que acessaram o topico

printUsers() {
  for user in `cut -f4 $TEMPFILE | sort | uniq`
  do
    echo $user `grep $user $TEMPFILE | wc -l`
  done
}

#-----------------------------------------------------
# printBytes
# - mostra o numero de bytes transferidos

printBytes() {
  # se a entrada for 2 3 4 (em 3 linhas), queremos fazer 2 + 3 + 4
  numstr=`cut -f2 $TEMPFILE | sed "s/$/ + /g"`
  expr $numstr 0
}

#-----------------------------------------------------
# processaAcao
#

processaAcao() {
  # traduzir para minusculas
  acao=`echo $1 | tr [a-z] [A-Z]`
  case "$acao" in
    bytes) printBytes ;;
    users) printUsers ;;
    hosts) printNumHosts ;;
    *) echo "Comando desconhecido: $acao" ;;
  esac
}

#---- Programa principal

if [ $# -lt 2 ]
then
  echo "Sintaxe: $0 topico [users|bytes|hosts]" >&2
  exit 1
fi

topico=$1
shift
grep "/pub/$topico" $LOGFILE > $TEMPFILE

for aAcao in $*
do
  processaAcao "$aAcao"
done
```

Como exercício, refaça o comando acima usando as opções -u, -b e -h em vez do comando "users", "bytes" e "hosts"
Aceite qualquer combinação de opções:

- -u
- -u -b

- -ub
- -bu
- etc.

22 Configuração da interface de rede

22.1 Introdução

Até o momento trabalhamos com o sistema desconectado. A partir de agora trabalharemos com serviços em rede e a primeira tarefa é justamente configurar a(s) interface(s).
A configuração da(s) interface(s) de rede é um processo bastante simples, desde que se tenha um conhecimento prévio de protocolos TCP/IP e classes de redes.

22.2 Configuração

Para adicionarmos uma máquina à rede é obrigatória a configuração de no mínimo os seguintes parâmetros: endereço ip, máscara de rede. Com estes dois parâmetros a máquina já se comunica com outras máquinas da rede local.
Para uma configuração completa é necessário configurarmos ainda o nome de máquina, o servidor de nomes (DNS) e o roteador padrão (*default gateway*).
Todos estes parâmetros podem ser configurados estaticamente ou por meio de um servidor DHCP.
No caso de servidores de rede é praticamente obrigatório, para alguns serviços é obrigatório, que a configuração seja estática. Para a maioria dos clientes a configuração clássica é como cliente DHCP.

22.2.1 Configuração do ifcfg-ethN

O ifcfg-ethN é o arquivo de configuração de cada uma das interfaces de rede existentes na máquina. Edita-se o arquivo /etc/sysconfig/network-scripts/ifcfg-ethN, onde N é o número da interface. No caso de uma única interface este número é 0 (zero). Este arquivo tem os seguintes parâmetros mínimos:

```
DEVICE=eth0       # Nome do dispositivo
BOOTPROTO=static      # Configuração estática. Dinâmico seria dhcp
IPADDR=192.168.2.X   # Endereço ip
NETMASK=255.255.255.0      # Máscara de rede
BROADCAST=192.168.2.255 # Endereço de broadcast – para todas as máquinas.
GATEWAY=192.168.2.1        # Roteador padrão
ONBOOT=yes      # Interface inicializa no boot da máquina
MS_DNS1=172.18.0.1  # Endereço do servidor DNS primário
MS_DNS2=200.135.37.65      #  Endereço do servidor DNS secundário
```

22.2.2 Configuração do network

Este arquivo define o nome da máquina e quem é o roteador padrão da mesma. Edita-se o arquivo /etc/sysconfig/network, que tem por exemplo os seguintes

parâmetros:

NETWORKING=yes # Trabalha ou não em rede
GATEWAY=192.168.2.1 # Roteador padrão
GATEWAYDEV=eth0 # Interface de acesso ao roteador padrão
HOSTNAME=m**X** # Nome da máquina

22.2.3 Configuração do resolv.conf

Este arquivo define qual(is) é(são) o(s) servidore(s) DNS da máquina. Edita-se o arquivo /etc/resolv.conf, que tem por exemplo os seguintes parâmetros:

nameserver 172.18.0.1
nameserver 200.135.37.65

22.2.4 Finalizando

Observe que o parâmetro GATEWAY aparece no arquivo ifcfg-ethN e no network. Caso os dois estejam ajustados para valores diferentes o que será válido é o do arquivo ifcfg-ethN. O mesmo vale para os parâmetros MS_DNS e nameserver nos arquivos ifcfg-ethN e resolv.conf, que identificam os servidores de nomes, o que será válido é(são) o(s) parâmetro(s) contido(s) no ifcfg-ethN.
Uma vez ajustados estes parâmetros basta reiniciar a interface de rede com o comando:

```
service network restart
```

Os parâmetros ajustados acima serão assumidos como o padrão da máquina e, sempre que a máquina ou interfaces de rede forem reiniciadas, os valores assumidos serão dados por estes arquivos.
O comando ifconfig, sem parâmetros, serve para mostrar a atual configuração da(s) interface(s) de rede. Se desejarmos, para um teste por exemplo, podemos mudar os parâmetros da interface de rede sem modificar estes arquivos. Para isto usamos o comando ifconfig com a seguinte sintaxe:

```
ifconfig interface ip/mascara up/down
```

```
ifconfig eth0 192.168.2.X/24 up
```

No comando acima estamos mudando a configuração da interface de rede eth0 para assumir o endereço ip 192.168.2.**X** com máscara de rede 255.255.255.0. Isto passará a valer imediatamente mas ao reiniciarmos a interface ou máquina os arquivos acima serão lidos e “configurarão” a interface.

22.3 Apelidos de ip

No Linux, a mesma interface de rede pode responder por mais de um endereço ip. Isto pode ser útil em algumas configurações especiais de rede, por exemplo, duas sub-redes no mesmo domínio de colisão ou um servidor Apache atendendo a domínios virtuais. Para fazermos isto basta criarmos interfaces virtuais com nomes ethN:0, ethN:1 etc. Por exemplo:

```
ifconfig eth0:0 192.168.1.65/24 up
```

Ou criando um arquivo /etc/sysconfig/network-scripts/ifcfg-eth0:0 com somente o

conteúdo abaixo e reiniciarmos a rede.

IPADDR=192.168.2.1**X**
NETMASK=255.255.255.0

23 Roteadores e sub-redes

23.1 Introdução

Um roteador, por definição, é um equipamento com no mínimo duas interfaces de rede que encaminha os pacotes oriundos de uma das interfaces à outra, de acordo com regras pré-definidas. No mercado existem roteadores com uma interface ethernet e uma, duas ou três interfaces WAN (*Wide Area Network*), normalmente utilizados para conexão da rede local com a internet. Existem também ou chamados modem/router que além de roteadores são modens, comumente usados para conexão ADSL.
O roteamento é sem dúvida um dos principais serviços (protocolos) da rede TCP/IP, já que é por meio dele que é possível um pacote originado na Brasil chegar rapidamente ao Japão, por exemplo.
Uma máquina Linux, com duas ou mais interfaces de rede, também funcionar como um roteador. Esta pode ser uma opção interessante se desejarmos criar sub-redes na instituição e obrigatória na implementação de um *firewall* transparente.
O roteamento estático trabalha com uma tabela que é lida linha-a-linha de tal modo que quando for encontrada uma regra que atenda a "demanda" o sistema pára imediatamente. Analisemos uma tabela de roteamento de uma estação qualquer:

```
Destino         Roteador        MáscaraGen.     Opções Métrica Ref   Uso Iface

192.168.1.0     0.0.0.0         255.255.255.0   U     10     0        0 eth0

127.0.0.0       0.0.0.0         255.0.0.0       U     0      0        0 lo

0.0.0.0         192.168.1.254   0.0.0.0         UG    10     0        0 eth0
```

A primeira linha informa que, para pacotes destinados à rede 192.168.1.0/24, basta "jogar" os mesmos pela interface eth0, já que o roteador (0.0.0.0) não é definido.
A segunda linha trata da rota para a interface de *loopback*.
A terceira linha trata do roteador padrão. Para destino qualquer (0.0.0.0) deve-se encaminhar os pacotes para o endereço 192.168.1.254.
De um outro modo, esta tabela nos diz o seguinte: se um pacote for destinado à 192.168.1.0/24 o "roteamento" analisará somente a primeira linha e simplesmente "jogará" o pacote na interface eth0, se o pacote for destinado a qualquer endereço iniciado com 127 o pacote será "jogado" pela interface virtual lo e por último para qualquer outro destino (0.0.0.0) o pacote será encaminhado para 192.168.1.254 e este "que se vire". Observe que o endereço 192.168.1.254 é um endereço "atingível" pela interface eth0.

23.2 Entendendo Rotas

Para entendermos os princípios de roteamento vamos analisar a Ilustração 14.

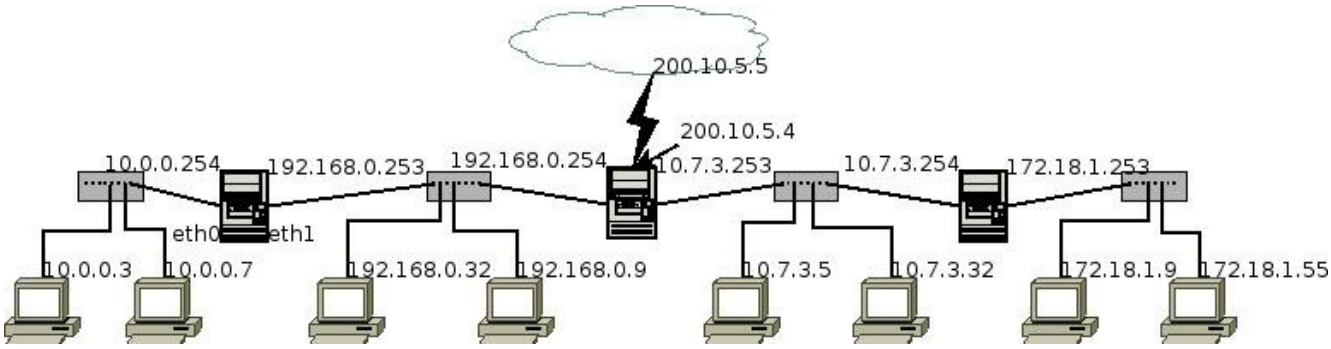


Ilustração 14: Exemplo de rede para construção de tabela de roteamento

Na Ilustração 14 observamos várias sub-redes interligadas por máquinas Linux, configuradas como roteadores, e uma delas conectada à Internet. Vamos montar uma tabela simplificada de roteamento para que todas a máquinas clientes possam "enxergar" as demais e à Internet. Suporemos que todas as máscaras de rede são 255.255.255.0.

Roteador 1	
Rede	**Roteador**
10.0.0.0	eth0
192.168.0.0	eth1
~~10.7.3.0~~	~~192.168.0.254~~
~~200.10.5.0~~	~~192.168.0.254~~
~~172.18.1.0~~	~~192.168.0.254~~
default	192.168.0.254

Roteador 2	
Rede	**Roteador**
192.168.0.0	eth0
10.7.3.0	eth1
200.10.5.0	eth2
10.0.0.0	192.168.0.253
172.18.1.0	10.7.3.254
default	200.10.5.5

Roteador 3	
Rede	**Roteador**
10.7.3.0	eth0
172.18.1.0	eth1
~~10.0.0.0~~	~~10.7.3.253~~
~~192.168.0.0~~	~~10.7.3.253~~
~~200.10.5.0~~	~~10.7.3.253~~
default	10.7.3.253

Podemos observar nas tabelas que todos os caminhos são contemplados e todos os roteadores passam a conhecer as demais redes e a Internet, muitas vezes usando o roteador padrão (*default*) para isto. Observe que as linhas canceladas (cortadas) não são necessárias já que apontam para o roteador que é o *default*. Isto possibilita que qualquer pacote de qualquer cliente pode "ir e voltar".

23.3 Configurando o roteador

Para transformarmos nossa máquina, de uma estação com duas interfaces de rede, em um roteador basta setarmos o bit ip_forward para 1. Isto pode ser feito com o comando:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

ou `sysctl -w netipv4 ip_forward=1`

E imediatamente nossa máquina passará a rotear pacotes de uma interface à outra. Este roteamento ocorrerá somente se os pacotes tiverem um destino explícito à outra interface, caso contrário os pacotes não serão roteados, ou seja, um roteador segmenta a rede, e seu tráfego, criando sub-redes distintas.

23.4 Configurando sub-redes

Como caso de estudos vamos montar a estrutura de sub-redes mostrada no diagrama esquemático abaixo, Ilustração 15.
Neste diagrama percebemos que, após as configurações necessárias, teremos 6 sub-redes compostas de 6 roteadores e 6 clientes para testes. A máquina "Professor", que também é um roteador, interligará estas sub-redes à rede da Campus São José do IFSC. Esta máquina será o roteador padrão de cada um dos 6 roteadores.

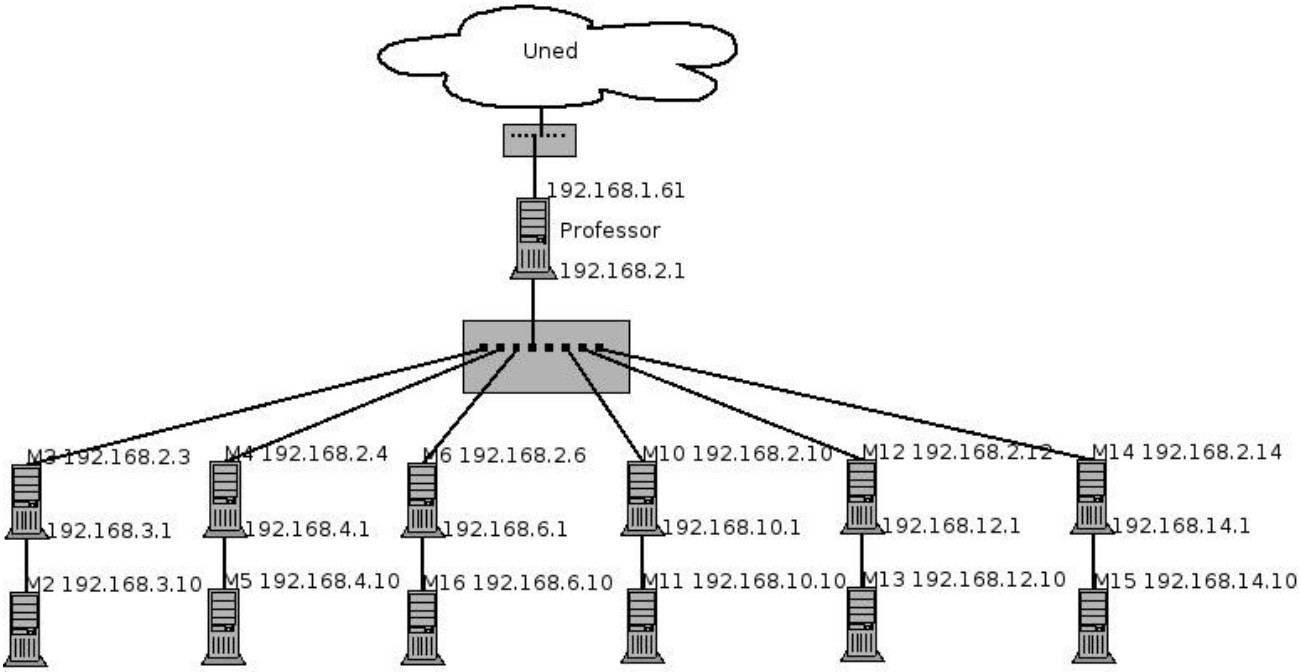


Ilustração 15: Diagrama de sub-redes

23.5 Caso de estudo

23.5.1 Roteadores

No roteador devemos, em primeiro lugar, definir os parâmetros da segunda interface de rede. Por questões de facilidade, vamos usar *ip aliases,* ou seja, uma única interface de rede do roteador responderá pelos dois ip's, mdesmo sendo de classes diferentes. Isto permitirá que não seja necessário reestruturar o cabeamento.
Devemos adotar a numeração do diagrama esquemático e seguir o modelo abaixo:

```
vi /etc/sysconfig/network-scripts/ifcfg-eth0:1
```

IPADDR=192.168.**X**.**X**
NETMASK=255.255.255.0

Reiniciamos o serviço de rede com o comando:
```
service network restart
```

Adicionamos as rotas para as 5 demais sub-redes com 5 comandos baseados no modelo abaixo:
```
route add -net 192.168.X.0/24 gw 192.168.2.X
```

Onde o parâmetro net identificará a rede a ser atingida e o gw identificará qual a interface "conhecida" (endereço) do roteador da respectiva rede.
Configuramos o roteamento do roteador com o comando:
```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Caso necessitemos que nossa máquina permaneça configurada como roteador e com suas rotas estáticas mesmo após uma reinicialização, devemos editar o arquivo /etc/rc.local e acrescentar ao final do mesmo as rotas e o ip_forward, com a exata sintaxe descrita acima.

23.5.2 Configuração do Cliente

No cliente devemos redefinir os parâmetros de rede, conforme modelo abaixo:
```
vi /etc/sysconfig/network-scripts/ifcfg-eth0
```

DEVICE=eth0
BOOTPROTO=static
IPADDR=192.168.**X**.10
NETMASK=255.255.255.0
BROADCAST=192.168.**X**.255
ONBOOT=yes

```
vi /etc/sysconfig/network
```

GATEWAY=192.168.**X**.1

Reiniciamos o serviço de rede com o comando:
```
service network restart
```

23.5.3 Testes

1. A partir do cliente "pingar" a interface mais próxima do roteador. Se este ping não funcionar devemos revisar a configuração física e lógica entre este e o roteador.
2. A partir do cliente "pingar" a interface externa do roteador. Se não pingar será por que o roteador não está roteando.
3. A partir do roteador "pingar" para 192.168.2.101. Se não pingar é por que tem algum erro de configuração física ou lógica na interface externa do roteador.
4. A partir do roteador "pingar" para a interface interna de um outro roteador,

por exemplo 192.168.4.1. Se houver problemas os motivos podem ser dois: não foi escrita uma rota adequada para tal rede, verificamos com o comando route -n, e/ou porque o roteador "pingado" está mal configurado. Lembre-se que os pacotes devem rota para ida e volta.

5. A partir do cliente pingar para outro cliente. Se houver problemas pode ser por má configuração do roteador "local" ou do roteador da rede "pingada".

Obs.: desfaça somente as tabelas de roteamento para podermos implementar NAT.

24 NAT - *Network Address Translator*

A tradução de endereço de rede é um procedimento que objetiva criar sub-redes e também a segurança das mesmas.
Podemos implementar um NAT de diversos modos mas a mais recomendada por facilidades e segurança é o mascaramento da seguinte forma:

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth0 -j MASQUERADE
```

Esta regra diz o seguinte: todos os pacotes que passarem (POSTROUTING) por esta máquina com origem de 192.168.1.0/24 e sairem pela interface eth0 serão mascarados, ou seja sairão desta máquina com o endereço de origem como sendo da eth0.
Com estas configurações o cliente acessa qualquer site na internet mas não pode ser acessado. A partir do cliente faça testes “pingando” para sites externos, roteadores vizinhos e tente pingar nos clientes vizinhos.
Obs.: desfaça todas as alterações, NAT, cabeamento etc para poder prosseguir.

25 Servidor DNS - *Domain Name System com Bind*

25.1 Introdução

Se o roteamento mantém a "conexão" entre as milhares de máquinas ligadas a internet o DNS faz o papel de dar nome às mesmas já que, para nós seres humanos, é difícil guardar números mas fácil gravar nomes. Já para as máquinas vale o oposto, entendem bem números mas nem tanto nomes. A interface entre estes dois mundos é feita pelo DNS.
O DNS é portanto uma tabela relacionando nomes e números ip. O DNS direto relaciona o número ip de uma máquina e seu nome. O DNS reverso relaciona o nome e seu ip.
O processo de resolução de nomes segue o esquema da Ilustração 16.

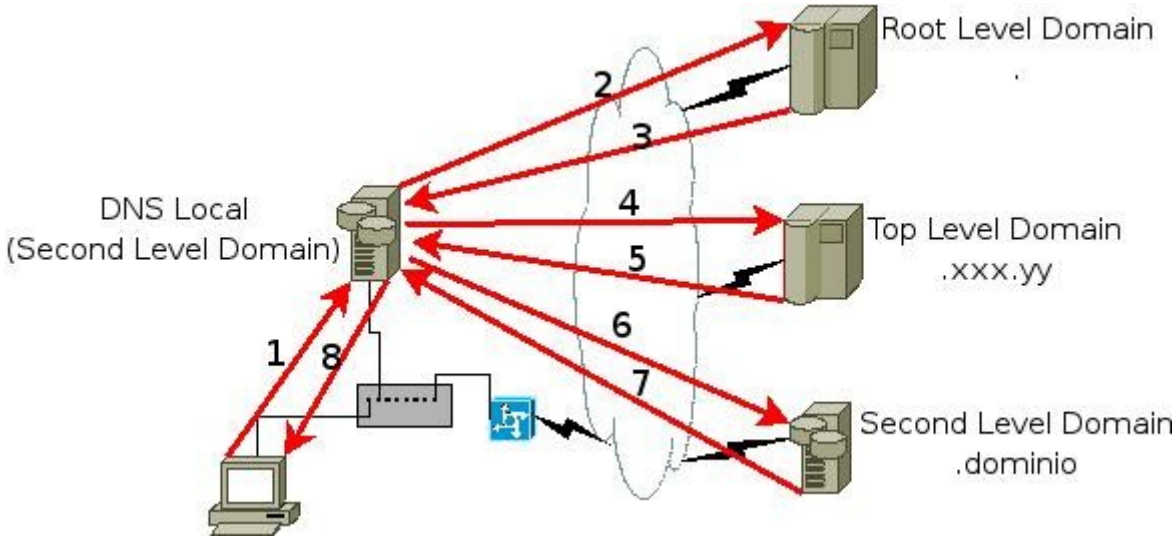


Ilustração 16: Processo de resolução de nomes

Vamos detalhar as setas em destaque:

1. O cliente, host A, deseja acessar algum serviço da máquina www.ifsc.edu.br. Este "olha" em sua tabela de *cache* DNS e verifica que não tem o ip desta máquina então solicita esta informação ao DNS local, seu "nameserver".
2. O DNS local busca em sua *cache* o ip de tal máquina, caso não encontre pede diretamente ao *Root Level Domain*.
3. Este responde que não conhece explicitamente o endereço da máquina mas sabe quem é o *Top Level Domain* responsável por aquele endereço.
4. O DNS local pede então ao *Top Level Domain* qual o ip da máquina.
5. Este responde que não conhece explicitamente o endereço da máquina mas sabe quem é o *Second Level Domain* responsável por aquele endereço.
6. O DNS local pede então ao *Second Level Domain* qual o ip da máquina.
7. O *Second Level Domain*, responsável pelo domínio "ifsc.edu.br." informa então o ip da máquina "www.ifsc.edu.br." ao DNS Local.
8. O DNS local armazena na tabela DNS *cahe* a correspondência entre ip e nome e entrega a informação ao cliente – Host A – que também guarda em seu cache.

25.2 Configuração de um servidor DNS

Como primeiro passo devemos instalar o pacote bind com o comando:

```
urpmi bind
```

25.2.1 Caso de estudo

Para podermos verificar o funcionamento do DNS vamos montar a estrutura lógica mostrada na Ilustração 17. **A máquina "professor" terá uma cópia de todos os arquivos de todos os domínios diretos e será a única a ter domínio reverso**. Ela será o servidor DNS de todos.

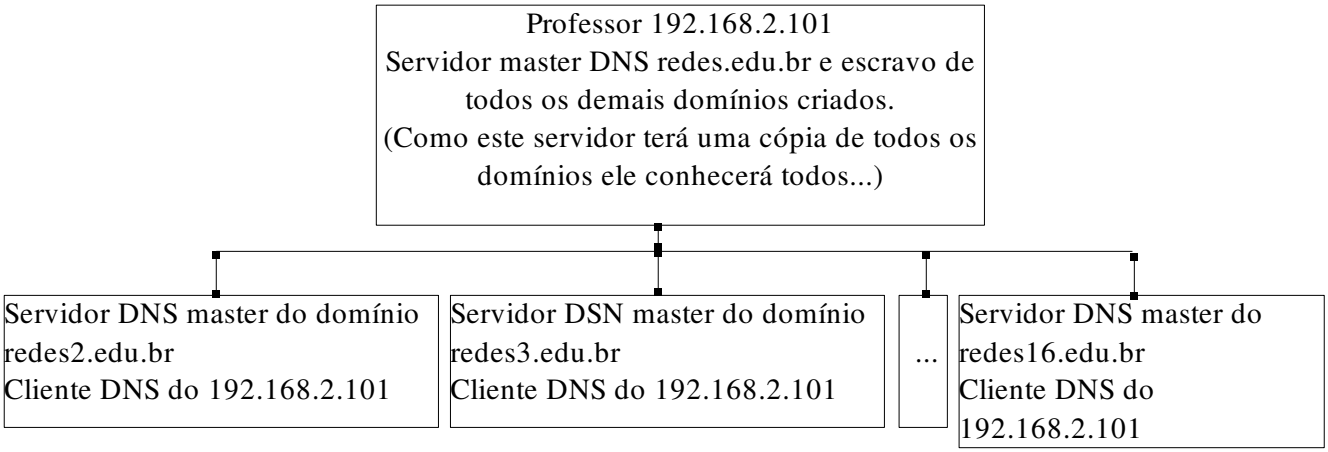


Ilustração 17: Estrutura para estudo do DNS

Para isto edite seu arquivo /etc/named.conf e crie o domínio redes**X**.edu.br incluindo no final do arquivo as seguintes linhas:

```
zone "redesX.edu.br" IN {                    # nome do domínio
      type master;                           # servidor master (slave etc)
      file "master/redesX.zone";             # arquivo de definição do domínio
      allow-update { none; };                # sem atualizações dinâmicas
};
```

Agora edite o arquivo /var/lib/named/var/named/master/redes**X**.zone, se necessário crie o subdiretório master, de acordo com o modelo abaixo.

```
$TTL    86400
@            IN SOA  mX.redesX.edu.br root (
                                2007032000          ; serial
                                3H          ; refresh
                                15M          ; retry
                                1W          ; expiry
                                1D )        ; minimum
             IN NS          mX.redesX.edu.br.
             IN MX   0      mX.redesX.edu.br.
localhost      IN A          127.0.0.1
$ORIGIN redesX.edu.br.
mX             A      192.168.2.X
www            A      192.168.2.X
```

Inicie o serviço de DNS com o comando:

```
service named start
```

Verifique o log em busca de possíveis problemas, neste log serão indicados os arquivo e linhas do mesmo que estão com problemas. Use o comando:

```
tail /var/lib/named/var/log/default.log
```

Como **primeiro teste** configure a sua máquina para ser sua própria cliente editando o /etc/resolv.conf e adicionando a diretiva “nameserver 192.168.2.**X**” no início do arquivo. Em seguida dê um ping para m**X**.redes**X**.edu.br (sua própria máquina). Se “pingar” é sinal de que o próprio servidor está funcionando.

Para podermos “enxergar” as demais máquinas devemos configurar a nossa máquina para ser cliente DNS da máquina “professor”, editando o arquivo /etc/resolv.conf e deixando-o somente com o conteúdo:

nameserver 192.168.2.101

25.2.2 Testes

Após as devidas configurações poderemos “pingar” para nomes de máquinas, por exemplo:

```
ping www.redes3.edu.br

nslookup m5.redes5.edu.br
```

Neste caso, antes de efetivamente “pingar” a máquina, o sistema converte o nome para número e mostra na tela, em seguida inicia-se o processo de ping. Se não funcionar a fonte do problema pode ser duas, ou o cliente mau configurado ou o servidor.
Para testar a resolução reversa de nomes usamos o seguinte comando, por exemplo:

```
host 192.168.2.10
```

Este comando deverá retornar o nome da máquina em questão.

26 Servidor de páginas Apache

26.1 Introdução[8]

O servidor Apache (*Apache server*) é o mais bem sucedido servidor web livre. Foi criado em 1995 por Rob McCool, então funcionário do NCSA (National Center for Supercomputing Applications), Universidade de Illinois. Numa pesquisa realizada em dezembro de 2005, foi constatado que a utilização do Apache supera 60% nos servidores ativos no mundo.
O servidor é compatível com o protocolo HTTP versão 1.1. Suas funcionalidades são mantidas através de uma estrutura de módulos, podendo inclusive o usuário escrever seus próprios módulos — utilizando a API do software.
É disponibilizado em versões para os sistemas Windows, Novell Netware, OS/2 e diversos outros do padrão POSIX (Unix, GNU/Linux, FreeBSD, etc).

26.2 Instalação e configuração

Como primeiro passo devemos instalar o pacote com o comando:

```
urpmi apache

urpmi apache-doc
```

Obs.: se for o caso escolha a opção 1, **estável**.
Em seguida inicia-se o servidor com o comando:

```
service httpd start
```

8 Texto obtido a partir de http://pt.wikipedia.org/wiki/Servidor_Apache

O servidor Apache já estrá rodando e pode ser testado usando um navegador com o endereço http://localhost/.
Acesse também o endereço http://localhost/manual/, que contém o manual (apache-doc) do servidor com uma série textos e *links* importantes.
Os principais arquivos de configuração do Apache são:

- /etc/httpd/conf/httpd.conf que está dividido em três seções: global, opções do servidor e máquinas virtuais. Esta última na verdade remete ao arquivo comentado no próximo item.
- /etc/httpd/conf/vhosts.d/vhosts.conf. Este arquivo define os domínios virtuais.

Como primeira configuração vamos mudar a página apresentada por nosso servidor. Para isto criamos um arquivo index.html com o conteúdo:

<html><body><h1>Esta é minha página de testes. Servidor 192.168.2.**X**</h1></body></html>

E copiamos este arquivo para /var/www/html/, que é o diretório padrão de hospedagem de páginas, com o comando:

```
cp -f index.html /var/www/html/
```

Acessamos novamente a página http://localhost/ e observamos o resultado. Obs.: pode ser necessário fazer uma atualização da página, no navegador, para que o mesmo releia o arquivo.

26.3 Domínios virtuais

O recurso de domínios virtuais é muito interessante pois permite que um servidor Apache responda pelas páginas de vários domínios de maneira independente. A princípio o usuário que acessará estes domínios não saberá que se trata do mesmo servidor. Os domínios podem ser tanto por nomes, desde que se tenha o registro formal do domínio, como por ips.
Para não ficarmos só na configuração básica vamos criar um domínio virtual baseado em ip. Como primeiro passo devemos conferir se nosso apelido de ip está ativo, com o comando:

```
ifconfig
```

Onde deve aparacer a configuração da interface eth0 e eth0:0.
Agora criamos o arquivo /etc/httpd/conf/vhosts.d/vhosts.conf com o seguinte conteúdo:

<VirtualHost 192.168.2.1**X**>
DocumentRoot /var/www/html/virtual
</VirtualHost>

Criamos o diretório virtual com o comando:

```
mkdir /var/www/html/virtual
```

Criamos mais um arquivo index.html dentro deste diretório com o conteúdo:

<html><body><h1>Esta é minha página virtual. Servidor 192.168.2.1**X**</h1></body></html>

Reiniciamos o servidor para que ele releia as configurações de domínios virtuais com o comando:

```
service httpd restart
```

Agora podemos testar e observar que possuímos duas páginas independentes:
http://192.168.2.**X/**
http://192.168.2.1**X/**

26.4 Páginas de Usuários

Para permitir que os usuários tenham sua página pessoal, seja em formato html ou simplesmente como repositório de arquivos (Indexes) procedemos do seguinte modo. Primeiramente instalamos o módulo userdir com o comando:

```
urpmi apache-mod_userdir
```

Em seguida editamos o arquivo /etc/httpd/conf/httpd.conf e acrescentamos ao final do mesmo o seguinte conteúdo (contêiner):

```
<Directory /home/*/public_html>    #Contêiner diretório home dos usuários

    AllowOverride All   #Aceita todo tipo de diretivas de autenticação

    Allow from all            #Permite acesso a todos

    Options Indexes FollowSymLinks MultiViews  #Indexes, se não houver o arquivo
                                               index.html mostra em formato de
                                               diretório. FollowSymLinks, pemite
                                               seguir os links da  página.
                                               MultiViews, tenta servir a página
                                               na língua do usuário.

    <IfModule mod_access.c>   #Se o módulo de controle de acesso, access,
existir...

      Order allow,deny  #Ordem de avaliação das diretivas para permitir ou negar
                        acesso ao recurso.

      Allow from all    #Permite para todos

    </IfModule>         #Fim do if

</Directory>                  #Fim do contêiner
```

Reiniciamos o apache com o comando:

```
service httpd restart
```

Assim qualquer usuário, que tiver um diretório public_html dentro de seu diretório de entrada, terá uma página no ar. Dentro do public_html pode ser colocado um arquivo index.html, em linguagem html, ou simplesmente arquivos para *download* externo.

27 Servidor de correio eletrônico Postfix

27.1 Introdução[9]

Um servidor de correio eletrônico gerencia os e-mails que são enviados e recebidos. Os servidores de e-mail podem ser servidores Internet, onde e-mails enviados e recebidos podem ser transitados para qualquer lugar do mundo, ou servidores de correio de intranet onde as mensagens trafegam apenas dentro da empresa. Através do correio eletrônico podem ser criados grupos de discussão sobre quaisquer assuntos. Estes grupos são chamados de listas ou refletores. Um refletor é uma caixa postal eletrônica falsa. Todas as mensagens enviadas para esta caixa postal, são transmitidas para as pessoas cadastradas na lista deste refletor. Desta forma cada membro do grupo passa a dispor das mensagens enviadas para o refletor em sua caixa postal ou mailbox. Cada membro, pode ler as mensagens e dar a sua opinião sobre elas enviando uma nova mensagem para o refletor.
Como exemplo de sistemas de correio eletrônico livres podemos citar o Postfix, que é um dos candidatos a substituir o SendMail. O Postfix é hoje uma das melhores alternativas para todas as empresas que desejam utilizar um servidor de email sem ter grandes gastos, ele foi escrito de forma direta e clara e visa facilitar e ajudar o Administrador Linux já que esse software é muito fácil de utilizar, além de ser um agente de transporte de email muitas vezes chamado simplesmente de servidor de email. Além de apresentar grande facilidade para sua configuração ele é um servidor de email robusto e com vários recursos.

27.2 Funcionamento do Correio Eletrônico[10]

Antes de implementar um serviço de correio eletrônico é importante que o administrador entenda como funciona a troca de mensagens, seja na Internet, seja em uma rede local. Para uma simples troca de mensagens entre dois usuários, pode ser necessária a utilização de vários protocolos e de várias aplicações. Será visto a seguir como isso acontece.
Um usuário que queira enviar uma mensagem para outro utilizará um aplicativo cliente de e-mail, também conhecido como MUA, ou Agente de Mensagens do Usuário. Ao terminar de redigir a sua mensagem o MUA enviará a mensagem a um MTA (Agente Transportador de Mensagens) que se encarregará então de entregar a mensagem ao MTA do destinatário, caso ele se encontre em outra máquina ou simplesmente colocar a mensagem na caixa postal do destinatário, caso ele se encontre no mesmo servidor. A transferência da mensagem entre o MUA e o MTA se efetua utilizando um protocolo chamado SMTP ou Protocolo Simples de Transferência de Mensagens. O protocolo SMTP será utilizado também entre o MTA do remetente e o MTA do destinatário.

O servidor de e-mail do destinatário, ao receber uma mensagem para um dos seus usuários, simplesmente a coloca na caixa postal deste usuário. Se o usuário

9 Texto obtido de http://pt.wikipedia.org/wiki/Sistema_de_correio_eletr%C3%B4nico
10 Texto obtido de http://www.conectiva.com/doc/livros/online/10.0/servidor/pt_BR/ch11.html

possui uma conta *shell* neste servidor, ele poderá ler os seus e-mails direto no servidor, caso contrário o usuário deverá transferir suas mensagens para sua máquina a fim de lê-las com o seu cliente de e-mail. A transferência de mensagens recebidas entre o servidor e o cliente de e-mail requer a utilização de outros programas e protocolos. Usualmente é utilizado para este fim o protocolo POP, Protocolo de "Agência" de Correio, que recebe este nome por agir como uma agência de correios mesmo, que guarda as mensagens dos usuários em caixas postais e aguarda que estes venham buscar suas mensagens. Outro protocolo que pode ser utilizado para este mesmo fim é o IMAP, Protocolo para Acesso de Mensagens via Internet, que implementa, além das funcionalidades fornecidas pelo POP, muitos outros recursos. Os protocolos POP e IMAP são protocolos para recebimentos de mensagens, ao contrário do protocolo SMTP, que serve para enviar mensagens, logo, possuem funcionalidades diferenciadas, como por exemplo, autenticação do usuário.
Para a utilização dos protocolos POP e IMAP é necessária a instalação do servidor apropriado, que vai ser o responsável por atender as solicitações do cliente de e-mail por novas mensagens. O recebimento de mensagens pelo cliente se dá através da solicitação do MUA do usuário ao seu servidor de e-mail, que após a autenticação do usuário vai informar se existem mensagens em sua caixa postal e quantas são. A seguir o MUA solicita a transferência das mensagens para a máquina local, finalizando assim o processo de troca de mensagens entre dois usuários. A Ilustração 18 resume todo esse processo.

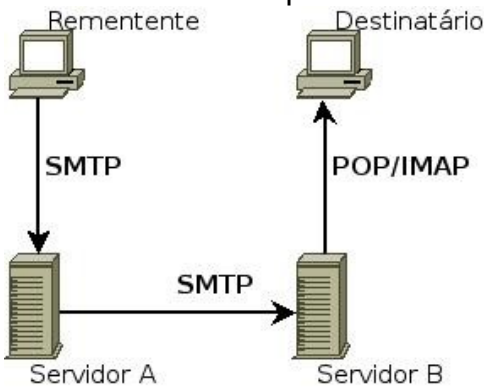


Ilustração 18: Protocolos de correio

27.3 Instalação e configuração

Devemos instalar os pacotes com o comando:

```
urpmi postfix
```

Configuramos o “segundo bloco” do arquivo /etc/postfix/main.cf, acrescentando/mudando somente os seguintes parâmetros:

```
# User configurable parameters
myhostname = mX.redesX.edu.br          # Nome da máquina
mydomain = redesX.edu.br               # Nome do domínio
myorigin = $mydomain                   # Especifica o domínio que
                                       aparece quando se envia um email
```

```
inet_interfaces = all                          # Interfaces que o servidor usa
mynetworks_style = subnet                      # A rede do servidor
mydestination = $myhostname, $mydomain    #O domínio que será
                                               especificado na entrega
```

Obs.: Outra diretiva interessante a se acrescentar em um caso real é a seguinte: **home_mailbox = Maildir/**. Esta diretiva salvará os emails destinados a determinado usuário em seu diretório home, dentro de uma pasta chamada Maildir. Contabilizando os arquivos em sua cota e deixando os mesmos fisicamente separados dos demais usuários.
Finalmente inicializamos o serviço com:

```
service postfix start
```

27.4 Testes

Para procedermos alguns testes em nosso servidor devemos ter uma ferramenta cliente de email. Para isto vamos usar uma ferramente a nível de linha de comando, mail (se não existir deve-se instalar o pacote mailx).

Para enviar uma mensagem proceda do seguinte modo:

- mail usuario@redesX.edu.br <Enter>,
- inserir o subjet <Enter>,
- inserir a mensagem, <Enter>
- <Ctrl>+<d>.

Como primeiro teste podemos enviar uma mensagem para um usuário da própria máquina e monitorar com o comando:

```
tail -f  /var/log/mail/info
```

Se aparecer algo do tipo:

```
Mar 23 09:56:29 professor postfix/pickup[15108]: 81A7C2C44C80:
uid=1572 from=<odilson>
Mar 23 09:56:29 professor postfix/cleanup[15113]: 81A7C2C44C80:
message-id=<20070323125629.81A7C2C44C80@professor.redes.edu.br>
Mar 23 09:56:29 professor postfix/qmgr[15109]: 81A7C2C44C80:
from=<odilson@redes.edu.br>, size=454, nrcpt=1 (queue active)
Mar 23 09:56:29 professor postfix/local[15115]: 81A7C2C44C80:
to=<root@redes.edu.br>, orig_to=<root>, relay=local, delay=0.29,
delays=0.23/0.01/0/0.04, dsn=2.0.0, status=sent (delivered to mailbox)
Mar 23 09:56:29 professor postfix/qmgr[15109]: 81A7C2C44C80: removed
```

é porque está tudo certo. O principal aviso é o status=sent (em negrito).
Uma vez que esteja funcionando localmente, pode-se enviar email para os colegas e, inclusive, emails externos. Lembrando que os emails externos não chegarão/retornarão ao nosso servidor, pois não temos um domínio válido.
Para ler mensagens basta digitar mail (logado na conta “certa”), aparecerá uma listagem de emails, e em seguida o número da mensagem.

28 Servidor SMB, *Server Message Block,* Samba

28.1 Introdução

O SAMBA é um software criado por Andrew Tridgell, que veio para facilitar a integração do mundo UNIX e o mundo Windows, integrando-os por meio do protocolo SMB (*Service Message Blocks*). Tem como função principal o compartilhamento de arquivos e impressoras com a família Windows©.
Um domínio Windows é um conjunto de computadores que residem na mesma sub-rede e pertencem ao mesmo grupo de trabalho, e um deles atua como controlador de domínio.
O PDC, *Primary Domain Controller*, é o controlador de domínio primário, onde está contido o banco de dados SAM, *Security Account Manager*, que é o banco de dados dos usuários do domínio Windows. O SAM é usado para validar os usuários no domínio. As mudanças, que por ventura ocorrerem, são propagadas para o BDC.
O BDC, *Backup Domain Controller*, é o reserva do controlador de domínio. Pode haver nenhum, um ou mais de um BDC num domínio mas um único PDC.
Pelo ambiente de rede Windows podemos navegar pelos computadores que estão disponíveis pela sub-rede e acessar seus recursos compartilhados.
O servidor WINS, *Windows Internet Name Server*, é uma implementação do servidor de nomes NetBIOS, *Network Basic Input/Output System.* O WINS é dinâmico: quando um cliente é iniciado são requeridos seu nome, endereço e grupo de trabalho. Este servidor manterá estas informações para futuras consultas e atualizações.
O Samba tem condições de exercer todos os papéis de uma rede Windows, com exceção do BDC. Mais especificamente o Samba pode ser: servidor de arquivo, servidor de impressão, PDC e servidor WINS, entre outros.

28.2 Instalação e configuração

Para instalar o samba basta executarmos o comando:

```
urpmi samba
```

O principal arquivo de configuração do samba é o /etc/samba/smb.conf. Este arquivo é dividido em duas secções, global e compartilhamentos. Devemos editá-lo e modificarmos/acrescentarmos as diretivas segundo o modelo abaixo:

```
[global]
    workgroup = redesX          # nome do grupo de trabalho ou do domínio
    netbios name = redesX       # nome da máquina no “formato” Microsoft
```

Para um caso real devemos nos preocupar ainda com mais mais algumas diretivas, as mais importantes são:

```
[global]
    security = user             # Modo de segurança: user, share
    encrypt passwords = yes     # Criptografia de senha: sim, não
    smb passwd file = /etc/samba/smbpasswd  # arquivo com senhas SMB
```

```
    load printers = yes          # Compartilhamento de impressora: sim,
não
    printcap name = cups         # Servidor de impressão, cups ou lprng
 [homes]
    .....                        # mantenha o original
```

Para criarmos outros compartilhamentos devemos criar “contêiners”, abaixo do [homes], com a seguinte sintaxe:

```
[software]                       # Nome do compartilhamento
    comment = Softwares          # Comentário
    path = /dados/software       # O diretório a ser compartilhado
    guest ok = no                # Convidado: sim, não
    public = no                  # Público: sim, não
    writable = yes               # Permissão de escrita: sim, não
    browseable = yes             # Navegação: sim, não
    force create mode = 0555     # Tipo de permissão dos arquivos criados
    force directory mode = 0555    # Tipo de permissão dos diretórios
criados
    veto files = /*.mp3/         # Arquivos que não poderão ser salvos
    valid users = user1,user2    # Grupo (de usuários) que tem acesso
    force group = admin          # Grupo atribuído ao criar
arquivos/diretórios
```

Como o padrão de senhas do SMB é diferente do Linux devemos criar as senhas “SMB” para os usuários de nossa máquina. Isto é feito com o comando:

```
smbpasswd -a usuario
```

Após a configuração devemos iniciar ou reiniciar o serviço com o comando:

```
service smb restart
```

28.3 Testes

A primeira verificação é feita pelo comando:

```
testparm
```

que verifica a integridade e coerência do arquivo /etc/samba/smb.conf.

Podemos também testar os compartilhamentos disponíveis com o comando:

```
smbclient -L redesX -U nome_de_usuário
```

Para fazermos os testes devemos ir à máquina Windows, logar com um usuário e senha cadastrado em nosso servidor, e fazer um mapeamento de rede apontando para //redes**X**/user ou //192.168.2.**X**/user, como abaixo:

```
smbmount //192.168.2.X/aluno ./diretorio_local_de_montagem -o username=aluno
```

Fazer testes criando/copiando/removendo arquivos e diretórios.

29 Servidor LDAP *Lightweight Directory Access Protocol* com OpenLdap.

29.1 Introdução

O LDAP é um protocolo leve para acessar serviços de diretório.

Diretórios são bancos de dados que armazenam informações mas com diferenças importantes em relação ao modelo tradicional. São hierárquicos, utilizando uma estrutura em árvore ao invés de tabelas. Otimizados para leitura. Permitem distribuição de dados entre servidores, já que ramos podem estar localizados remotamente.

Em redes locais, Ilustração 19, recomenda-se o uso do LDAP para centralizar bases de usuários e grupos, já que a maioria dos servidores atuais já disponibiliza integração com esta base, como por exemplo o Samba, Postfix, Apache, *Helpdesks* etc. Isto irá facilitará e muito o gerenciamento da rede já que toda a base de usuários será única, não ocorrendo problemas de inconsistência de senhas, por exemplo.

Cabe salientar que não necessariamente deve-se ter um servidor exclusivo para o LDAP, pode-se colocá-lo junto com o Samba por exemplo.

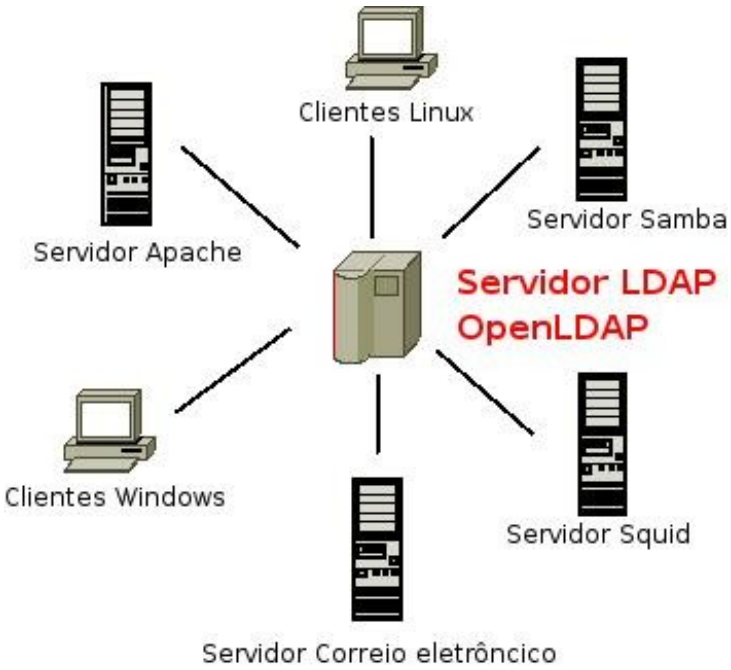


Ilustração 19: Exemplo de interligação LDAP e outros serviços

A árvore de diretórios pode ser organizada baseando-se nos domínios de nomes, DNS, que é uma das formas mais populares atualmente. Por exemplo: exemplo.com.br ==> dc=exemplo,dc=com,dc=br (dc, *domain component*).

29.2 Instalação e configuração básica

Para instalar o openldap devemos usar o comando:

```
urpmi openldap-server
```

Editar o arquivo /etc/openldap/slapd.conf e modificar os seguintes parâmetros:

> Trocar todas as ocorrências de dc=example,dc=com por dc=exemplo,dc=com,dc=br. No vi faz-se assim, no modo de comando, :%s/dc=example,dc=com/ dc=exemplo,dc=com,dc=br
>
> rootpw {SSHA}RUHmz+yqoZrZiC7o+XzfNCN5ewu/Ol77 #Senha criptografada: sair do editor e copiar a saída do comando “slappasswd -c crypt” no lugar desta senha

Editar o arquivo /etc/openldap/ldap.conf e modificar os seguintes parâmetros:

> BASE dc=exemplo, dc=com, dc=br
>
> HOST localhost
>
> URI ldap://localhost

Para inserção de dados no Ldap e integração com o Samba recomendamos as ferramentas do pacote smbldap-tools, que podem ser instaladas pelo comando:

```
urpmi smbldap-tools
```

Inicia-se os serviços com os comandos:

```
service smb (re)start

service ldap (re)start
```

Após a instalação o modo mais fácil de integrar estas ferramentas ao Ldap é usar um *script* já desenvolvido que é o configure.pl. Esta ferramenta deve ser executada com o Samba e Ldap pré-configurados e rodando. Digite o comando:

```
/usr/share/doc/smbldap-tools/configure.pl
```

Em seguida serão requisitados uma série de parâmetros, na maioria dos casos basta teclar Enter. Sempre que for requisitada uma senha utilize a mesma já informada no arquivo /etc/openldap/slapd.conf, para evitar problemas. No exemplo abaixo basta modificar/personalizar o que está em **negrito**:

```
Use of $# is deprecated at /usr/share/doc/smbldap-tools-0.9.2/configure.pl line 314.
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
    smbldap-tools script configuration
    -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
Before starting, check
 . if your samba controller is up and running.
 . if the domain SID is defined (you can get it with the 'net getlocalsid')

 . you can leave the configuration using the Crtl-c key combination
 . empty value can be set with the "." character
```

-=
Looking for configuration files...

Samba Configuration File Path [/etc/samba/smb.conf] >

The default directory in which the smbldap configuration files are stored is shown.
If you need to change this, enter the full directory path, then press enter to continue.
Smbldap-tools Configuration Directory Path [/etc/smbldap-tools/] >
-=
Let's start configuring the smbldap-tools scripts ...

. workgroup name: name of the domain Samba act as a PDC
  workgroup name [redes] >
. netbios name: netbios name of the samba controler
  netbios name [redes] >
. logon drive: local path to which the home directory will be connected (for NT Workstations). Ex: 'H:'
  logon drive [] >
. logon home: home directory location (for Win95/98 or NT Workstation).
  (use %U as username) Ex:'\\redes\%U'
  logon home (press the "." character if you don't want homeDirectory) [\\redes\%U] >
. logon path: directory where roaming profiles are stored. Ex:'\\redes\profiles\%U'
  logon path (press the "." character if you don't want roaming profile) [\\redes\profiles\%U] >
. home directory prefix (use %U as username) [/home/%U] >
. default users' homeDirectory mode [700] >
. default user netlogon script (use %U as username) [] >
  default password validation time (time in days) [45] >
. **ldap suffix [] > dc=exemplo,dc=com,dc=br**
. ldap group suffix [] > ou=Group
. ldap user suffix [] > ou=People
. ldap machine suffix [] > ou=Computer
. Idmap suffix [ou=Idmap] >
. sambaUnixIdPooldn: object where you want to store the next uidNumber
  and gidNumber available for new users and groups
  sambaUnixIdPooldn object (relative to ${suffix}) [sambaDomainName=redes] >
. ldap master server: IP adress or DNS name of the master (writable) ldap server
  ldap master server [127.0.0.1] >
. ldap master port [389] >
. **ldap master bind dn [] > cn=Manager,dc=exemplo,dc=com,dc=br**
. **ldap master bind password [] > insira_a_senha**
. ldap slave server: IP adress or DNS name of the slave ldap server: can also

be the master one
ldap slave server [127.0.0.1] >
. ldap slave port [389] >
. **ldap slave bind dn [] > cn=Manager,dc=exemplo,dc=com,dc=br**
. **ldap slave bind password [] > insira_a_senha**
. ldap tls support (1/0) [0] >
. SID for domain redes: SID of the domain (can be obtained with 'net getlocalsid redes')
SID for domain redes [S-1-5-21-1066659121-185135820-1519059970] >
. unix password encryption: encryption used for unix passwords
unix password encryption (CRYPT, MD5, SMD5, SSHA, SHA) [SSHA] >
. default user gidNumber [513] >
. default computer gidNumber [515] >
. default login shell [/bin/bash] >
. default skeleton directory [/etc/skel] >
. default domain name to append to mail adress [] >
-=
Use of uninitialized value in concatenation (.) or string at /usr/share/doc/smbldap-tools-0.9.2/configure.pl line 314, <STDIN> line 33.
backup old configuration files:
/etc/smbldap-tools/smbldap.conf->/etc/smbldap-tools/smbldap.conf.old
/etc/smbldap-tools/smbldap_bind.conf->/etc/smbldap-tools/smbldap_bind.conf.old
writing new configuration file:
/etc/smbldap-tools/smbldap.conf done.
/etc/smbldap-tools/smbldap_bind.conf done.

Em seguida devemos povoar o banco de dados com o comando:

```
smbldap-populate
```

Este comando gerará uma saída parecida com a que está abaixo, informando todas as entradas que estão sendo feitas no banco de dados. Ao final será requisitada um senha, recomenda-se o uso da mesma anteriormente informada.

Populating LDAP directory for domain gipsy (S-1-5-21-323509653-3845939124-2016821157)
(using builtin directory structure)

entry dc=exemplo,dc=com,dc=br already exist.
entry ou=People,dc=exemplo,dc=com,dc=br already exist.
adding new entry: ou=Group,dc=exemplo,dc=com,dc=br
adding new entry: ou=Computer,dc=exemplo,dc=com,dc=br
entry ou=Idmap,dc=exemplo,dc=com,dc=br already exist.
adding new entry: uid=root,ou=People,dc=exemplo,dc=com,dc=br
adding new entry: uid=nobody,ou=People,dc=exemplo,dc=com,dc=br
adding new entry: cn=Domain Admins,ou=Group,dc=exemplo,dc=com,dc=br
adding new entry: cn=Domain Users,ou=Group,dc=exemplo,dc=com,dc=br
adding new entry: cn=Domain

Guests,ou=Group,dc=exemplo,dc=com,dc=br
adding new entry: cn=Domain
Computers,ou=Group,dc=exemplo,dc=com,dc=br
adding new entry:
cn=Administrators,ou=Group,dc=exemplo,dc=com,dc=br
adding new entry: cn=Account
Operators,ou=Group,dc=exemplo,dc=com,dc=br
adding new entry: cn=Print
Operators,ou=Group,dc=exemplo,dc=com,dc=br
adding new entry: cn=Backup
Operators,ou=Group,dc=exemplo,dc=com,dc=br
adding new entry: cn=Replicators,ou=Group,dc=exemplo,dc=com,dc=br
entry sambaDomainName=gipsy,dc=exemplo,dc=com,dc=br already exist.
Updating it...

Please provide a password for the domain root:
Changing UNIX and samba passwords for root
New password: insira_a_senha
Retype new password: insira_a_senha

Podemos verificar se está tudo correto com o comando:

```
slapcat
```

Agora podemos acrescentar ou modificar usuários com os comandos: smbldap-useradd, smbldap-usermod smbldap-passwd etc. As *flags* para os comandos são praticamente as mesmas dos comandos similares para criação e modificação de contas de usuários no Linux.

A partir de agora podemos gerenciar a base de dados com ferramentas gráficas. Uma delas é a phpLDAPAdmin, http://phpldapadmin.sourceforge.net/, que funciona por meio de um navegador qualquer. Temos também a ferramente LDAP Admin para Windows, uma ferramente gratuita que roda na plataforma Windows.

29.3 Para configurar um cliente Linux

Instale os pacotes necessários para o cliente ldap com o seguinte comando como root:

```
urpmi nss_ldap openldap-client pam_ldap perl-ldap autofs
```

Para o caso do Mandriva existe a opção de configuração com o programa drakauth, seguindo os passos recomendados pelo mesmo. As ilustrações 20 e 21 mostram o procedimento necessário para configurar um cliente.

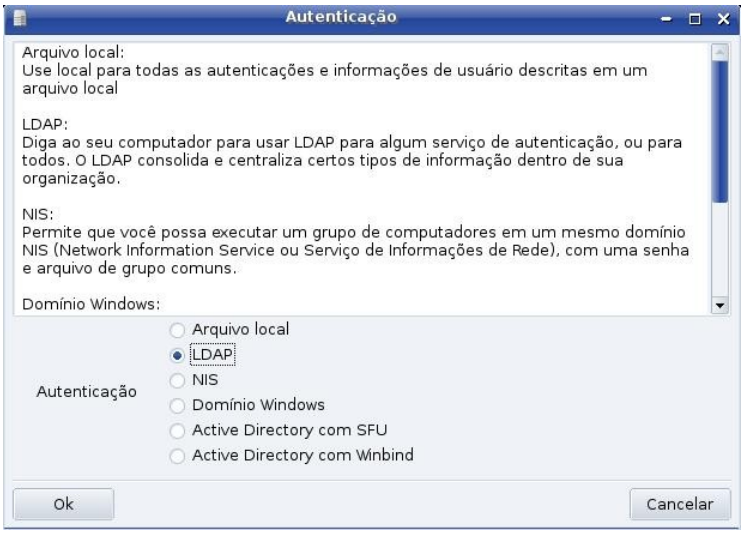


Ilustração 20: Primeira janela do DrakAuth

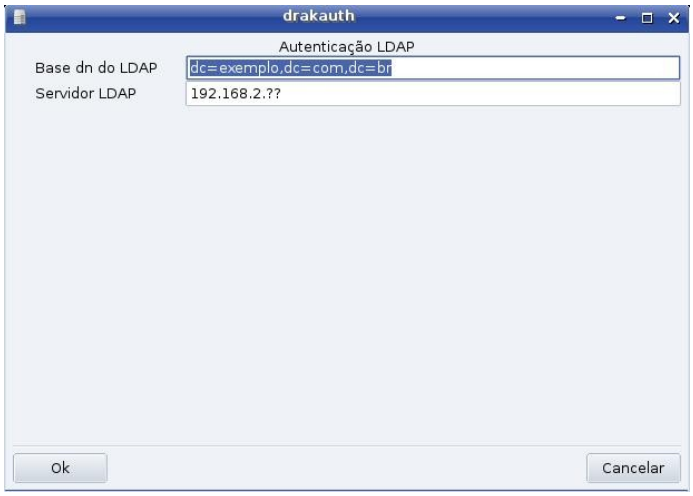


Ilustração 21: Segunda janela do DrakAuth

Outra forma é editarmos diretamente os arquivos de configuração de autenticação do Linux. Abaixo mostramo uma possibilidade de configuração dos mesmos.

Edite o arquivo /etc/openldap/ldap.conf com o seguinte conteúdo:

```
BASE    dc=exemplo,dc=com,dc=br
URI     ldaps://ip_do_servidor:636
SIZELIMIT       4000
TLS_REQCERT     allow
```

Edite o arquivo /etc/ldap.conf modificando o seguinte conteúdo:

```
host   ip_do_servidor
base   dc==exemplo,dc=com,dc=br
nss_base_passwd ou=People,dc=exemplo,dc=com,dc=br?sub
```

nss_base_shadow ou=People,dc=exemplo,dc=com,dc=br?sub

nss_base_group ou=Group,dc=exemplo,dc=com,dc=br?sub

Edite o arquivo /etc/nsswitch.conf modificando o seguinte conteúdo:

```
passwd:         files ldap
shadow:         files ldap
group:          files ldap
hosts:          files nis dns
networks:       files
services:       files
protocols:      files
rpc:            files
ethers:         files
netmasks:       files
netgroup:       files
publickey:      files
bootparams:     files
automount:      files ldap
aliases:        files
```

Edite o arquivo /etc/ldap.secret modificando o seguinte conteúdo:

senha_do_banco_ldap #para poder ter acesso ao mesmo.

Edite o arquivo /etc/pam.d/system-auth modificando o seguinte conteúdo:

```
auth        required      pam_env.so
auth        sufficient    pam_unix.so likeauth nullok
auth        sufficient    pam_ldap.so use_first_pass
auth        required      pam_deny.so
account     sufficient    pam_unix.so
account     sufficient    pam_ldap.so use_first_pass
account     required      pam_deny.so
password    required      pam_cracklib.so retry=3 minlen=2  dcredit=0
ucredit=0 ucredit=0
password    sufficient    pam_unix.so nullok use_authtok md5 shadow
password    sufficient    pam_ldap.so
password    required      pam_deny.so
session     optional      pam_mkhomedir.so skel=/etc/skel/ umask=0022
session     required      pam_limits.so
session     required      pam_unix.so
```

29.4 Testes

O primeiro teste é fazer uma consulta ao banco de usuários, agora teremos todos os usuários locais mais o do servidor LDAP. Use o comando:

```
getent passwd
```

Em seguida faça testes logando com um usuário devidamente cadastrado. Se a máquina já estava “na ativa” o teste deve ser com um usuário que ainda não

tenha logado na mesma.

30 Servidor NFS *Network File System*

30.1 Introdução

NFS é o sistema nativo Linux (Unix) para compartilhamento de arquivos em rede local. Permite que os diretórios remotos (servidor) sejam montados localmente (cliente) passando a impressão ao usuário de que o sistema de arquivos é local.

A segurança aos arquivos e diretórios é dada pelo permissionamento de arquivos e diretório padrão do Linux, sobreposto à uma “máscara” configurada no servidor de arquivos. Deve-se tomar o cuidado para garantir que todos os usuários tenham a mesma identificação (UID e GID) no servidor e cliente para não gerar “furos” na segurança de arquivos.

30.2 Instalação e configuração

Para instalar o servidor NFS usamos o comando

```
urpmi nfs-utils
```

A configuração dos diretórios a serem exportados é feita por meio do arquivo /etc/exports, que por padrão não existe e deverá ser criado com o primeiro compartilhamento. O formato deste arquivo é bastante simples. Nele devem ser informados todos os diretórios, um por linha, a serem exportados seguindo o formato:

diretório [cliente(s) opções]

Onde diretório é o próprio diretório a ser exportado/compartilhado. Se informarmos simplesmente o diretório, todos as máquinas terão permissão de escrita e leitura no dito diretório.

Em cliente deve pode ser informado:

- o nome da máquina cliente ou ip
- curingas de domínio como *uned.sj, todas as máquinas na uned.sj
- pares de endereço ip/máscara, 192.168.2.0/24.
- *, qualquer máquina

Opções pode ser ro (*read only*) rw (*read and write*) e *no_root_squash. Nesta última* o root do cliente passa a ter permissões de root no servidor. Não recomenda-se o uso desta opção a não ser entre servidores e com muito cuidado. Devemos observar que se compartilharmos um diretório com opção rw e as permissões do diretório em si são somente leitura o usuário terá permissão de somente leitura. A regra geral é que o que vale é a permissão mais restritiva.

Exemplos reais:

/home **192.168.2.0/24(rw)** # a rede 192.168.2.0 terá acesso para

escrita e leitura ao diretório /home.

/usr 192.168.2.7(ro) 192.168.2.1(rw,no_root_squash) # a máquina 192.168.2.7 poderá ler e a máquina 192.168.1.1 ler e escrever e o usuário root do cliente será “replicado”, tendo permissões de root no servidor.

/var/www/html www.sj.ifsc.edu.br(rw) # a máquina www.sj.ifsc.edu.br terá acesso de leitura e escrita ao dito diretório.

Após definirmos o que queremos compartilhar devemos informar ao sistema as nossas atualizações com os comandos:

```
service portmap restart

service nfs-server start

exportfs -a
```

30.3 Testes

Para verificar os compartilhamentos atuais em um determinado servidor usamos o comando:

```
showmount --exports ip_do_servidor
```

Podemos fazer isto com nossa própria máquina trocando servidor pelo nosso ip. O comando retornará a listagem de todos os diretórios compartilhados.

Agora podemos montar o diretório compartilhado no cliente. Para fins de testes o cliente pode ser nossa própria máquina. Sendo assim criamos um diretório com o comando:

```
mkdir nfs
```

e montamos o compartilhamento neste diretório com o comando:

```
mount 192.168.2.X:/usr nfs
```

Podemos conferir listando o conteúdo do diretório nfs e/ou com o comando df.

31 Servidor DHCP *Dynamic Host Configuration Protocol*

31.1 Introdução[11]

Toda máquina que for participar de uma rede, deve primeiro, ter um endereço IP. Em uma rede pequena (até 20 máquinas), a tarefa de configurar IPs é relativamente simples. Mas em uma rede grande com centenas de máquinas, esta tarefa de endereçamento torna-se trabalhosa.

11 Texto obtido de http://marcio.katan.googlepages.com/dhcp-mandriva

Para facilitar as coisas, foi criado um mecanismo de endereçamento automático de IP para máquinas em uma rede TCP/IP: o DHCP (*Dynamic Host Configuration Protocol* – Protocolo de configuração de máquinas dinâmico).

Um servidor DHCP pode facilitar muito a vida do administrador da rede. Dentre as configurações de serviços que podem ser passadas ao host cliente por dhcp são:

- Endereçamento IP, máscara de subrede, Gateway, Servidor(es) DNS, nome de host e/ou de domínio;
- Servidores e domínio NIS (autenticação);
- Servidores WINS (para redes Microsoft®);
- Servidores NTP (Hora);
- Imagens de boot para Terminais burros;

Como podemos observar, tudo o que é necessário para que uma máquina esteja em condições de ingressar em uma rede e usufruir de tudo o que ela possa oferecer, o DHCP se faz útil para sua configuração automática.

31.2 O protocolo DHCP

Entenda, com a explicação a seguir, como funciona o protocolo DHCP.
a) DHCP Discover – Quando uma máquina é ligada, ela tem um serviço (daemon) cliente do DHCP configurado para localizar o servidor neste momento. Este cliente DHCP envia um pacote UDP com destino à porta 67 do servidor chamado "***DHCP Discover"***. Este pacote *broadcast* tem o endereço IP de destino 255.255.255.255 e mac address de destino ff:ff:ff:ff:ff:ff – Ilustração 22.

Ilustração 22: DHCP Discover. Pacote enviado pela estação

b) DHCP Offer – O servidor ao receber o referido pacote em sua porta ethernet, irá analisá-lo e, em sua tabela de IPs, reservar um endereço e preparar um pacote de resposta ao cliente solicitante. Este pacote de resposta chama-se DHCP Offer – Ilustração 23.

Ilustração 23: DHCP Offer. Resposta do Servidor DHCP à estação

O único meio de a estação cliente saber que o pacote DHCP Offer se destina à ela, é através do mac address.

c) DHCP Request – O cliente ao receber o pacote do servidor, decide se aceita a configuração oferecida pois pode receber mais de uma oferta. Em caso positivo, retorna um novo pacote ao servidor, comunicando o aceitamento da oferta. Este pacote chama-se DHCP Request - Ilustração 24.

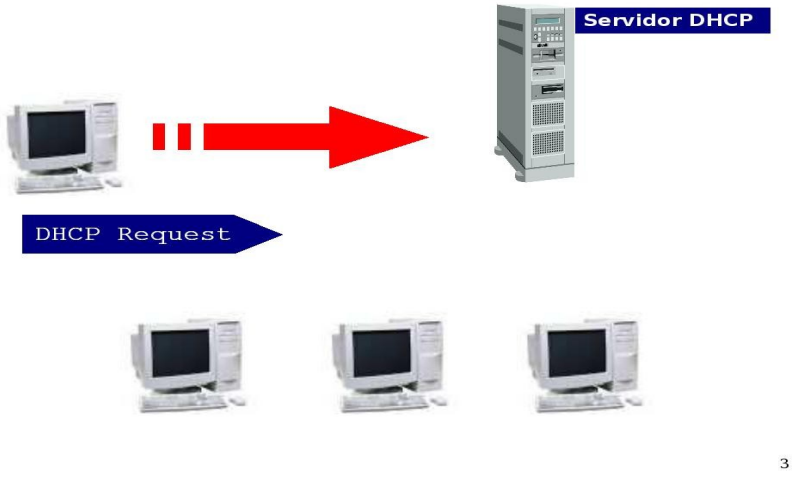


Ilustração 24: DHCP request. Confirmação da estação

d) DHCP Ack – Para finalizar a “conversação” entre cliente e servidor DHCP, este finaliza (efetiva) o aluguel (lease) do endereço ao cliente em sua tabela de IPs, e envia àquele, um pacote DHCP Ack para que ele ajuste suas configurações – Ilustração 25.

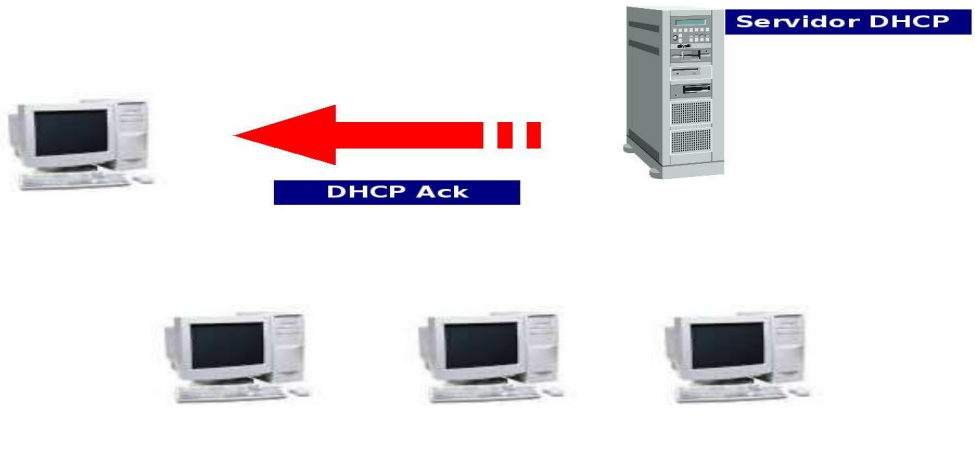


Ilustração 25: DHCP Ack. Confirmação do lease do endereço pelo Servidor

31.3 Instalação e configuração

Para instalarmos o servidor DHCP devemos executar o seguinte comando:

```
urpmi dhcp-server
```

O servidor DHCP atende somente rede(s) à(s) qual(is) pertence, portanto a sub-rede (subnet) declarada deve ser compatível (pertencer à mesma rede) com a faixa de Ips a serem alugados.
Agora editamos o arquivo /etc/dhcpd.conf de acordo com o modelo:

```
ddns-update-style none;        # Esta opção serve para indicar se o servidor DNS será atualizado quando um aluguel de ip for solicitado.

subnet 192.168.2.0 netmask 255.255.255.0 {  # Todas as diretivas entre as chaves ({}) serão explicitamente aplicadas aos clientas da rede (subnet) declarada.
        option routers 192.168.2.1;     # Roteador padrão
        option subnet-mask 255.255.255.0; # Máscar de rede
        option domain-name "redesX.edu.br";      # Nome do domínio
        option domain-name-servers mX.redesX.edu.br;        # Servidor DNS
        range dynamic-bootp 192.168.2.128 192.168.2.254;  # A faixa de endereços que serão alugados
        default-lease-time 21600;       # Tempo padrão de aluguel. Após este tempo o cliente tenta alugar novamente o mesmo ou um novo endereço.
        max-lease-time 43200;           # Tempo máximo de aluguel. Se este tempo for excedido o cliente sairá de rede.
        host novo {  # Podemos fornecer um ip fixo a um determinado cliente que receberá o nome “novo”.
                hardware ethernet 12:34:56:78:AB:CD;      # O endereço mac do cliente
                fixed-address 192.168.2.100;   # O ip que será fixado a ele.
```

```
        }
    }
```

No arquivo de configuração poderíamos ter a declaração de mais de uma sub-rede (subnet), sendo que poderíamos ter parâmetros globais, declarados acima da sub-redes, e parâmetros específicos, dentro das sub-redes.
Após a configuração do arquivo (re)iniciamos o servidor com o comando:

```
service dhcpd restart
```

31.4 Testes

Para fazermos um teste podemos usar o comando dhclient eth? na máquina sobressalente. Se o servidor estiver corretamente configurado este alugará algum ip e demais características para esta máquina.
Todas as trocas de mensagem para negociação do aluguel de ip podem ser observadas no arquivo de log /var/log/messages.

32 Servidor FTP *File Transfer Protocol*

32.1 Introdução

FTP significa Protocolo de Transferência de Arquivos, e é uma forma bastante rápida e versátil de transferir arquivos e diretórios, sendo uma das mais usadas na internet.
Pode referir-se tanto ao protocolo quanto ao programa que implementa este protocolo (neste caso, tradicionalmente aparece em letras minúsculas, por influência do programa de transferência de arquivos do Unix).
A transferência de dados em rede de computadores envolve normalmente transferência de arquivos e acesso a sistemas de arquivos remotos (com a mesma interface usada nos arquivos locais). É o padrão da pilha TCP/IP para transferir arquivos, é um protocolo genérico independente de hardware e do sistema operacional e transfere arquivos por livre arbítrio, tendo em conta restrições de acesso e propriedades dos arquivos.

32.2 Instalação e configuração

Para instalar o servidor FTP devemos primeiramente escolher um dos servidores disponíveis na distribuição dentre proftpd, pure-ftpd, vsftpd e wu-ftpd. Um dos mais usados na distribuição Mandriva é o proftpd. Cabe salientar que todos são similares na funcionalidade e configuração. Para instalá-lo basta executarmos o comando:

```
urpmi proftpd
```

O Proftpd vem absolutamente pronto para uso, não sendo necessária nenhuma configuração preliminar. Mas se desejarmos alguma configuração especial devemos editar o arquivo /etc/proftpd.conf.
Agora devemos (re)iniciar o serviço com o comando:

```
service proftpd restart
```

Uma boa medida de segurança é “enjaular” a conexão do usuário, isto significa dizer que após a conexão o usuário não poderá subir na árvore de diretórios, ele ficará restrito ao seu diretório e seus subdiretórios. Para isto devemos descomentar a opção:

```
DefaultRoot ~
```

32.3 Testes

Para testar podemos usar nossa própria máquina como cliente. Para isto basta executarmos o comando:

*ftp 192.168.2.**X***

E surgirá um prompt com algo parecido com:

```
Connected to 192.168.2.1.
220 (vsFTPd 2.0.2)
530 Please login with USER and PASS.
530 Please login with USER and PASS.
KERBEROS_V4 rejected as an authentication type
Name (192.168.2.1:odilson):
```

Então devemos informar o usuário ou <Enter> para o padrão, nesta caso odilson. Em seguida será requisitada a senha e após esta estaremos conectado no servidor remoto e poderemos usar praticamente todos os comandos normais do shell para visualizar, criar, modificar ou apagar arquivos e diretórios. Lembramos que todos os comandos podem ser executados no servidor remoto ou na máquina local, neste caso iniciando o comando com “!”. Para transferências de arquivos usa-se o comando put “arquivo”, para enviar da máquina local ao servidor, e get “arquivo”, no caso contrário. Para sairmos da aplicação digitamos o comando “bye”.

33 Servidor SSH *Secure Shell* com OpenSSH

33.1 Introdução[12]

Em informática, o *Secure Shell* ou SSH é, simultaneamente, um programa de computador e um protocolo de rede que permite a conexão com outro computador na rede, de forma a executar comandos de uma máquina remota. Possui as mesmas funcionalidades do TELNET, com a vantagem da conexão entre o cliente e o servidor ser criptografada.
O SSH faz parte da suíte de protocolos TCP/IP que torna segura a administração remota de um servidor Linux/Unix.
O scp (*Secure Copy*) é uma maneira segura de fazer cópias de arquivos e diretórios usando o protocolo SSH.

33.2 Instalação e configuração

Para instalarmos o servidor SSH devemos executar o seguinte comando:

12 Texto obtido de http://pt.wikipedia.org/wiki/SSH

```
urpmi openssh-server ou urpmi ssh
```

Por padrão o servidor OpenSSH já vem completamente configurado, não sendo necessário nenhuma ajuste de configuração para as operações padrão. Mas, se quisermos fazer alguns ajustes devemos editar o arquivo /etc/ssh/sshd_config. A recomendação para este arquivo é descomentar somente o que pretende-se mudar do padrão. Se descomentarmos uma linha e deixarmos o valor padrão podem ocorrem instabilidades no serviço. Os principais parâmetros que podem ser modificados são:

PermitRootLogin no # yes ou no. O usuário root poderá abrir um conexão ssh diretamente? Por questões de segurança recomenda-se deixar no, logar como usuário normal e em seguida dar o comando su.
X11Forwarding yes # yes ou no. Se no servidor e cliente existirem as bibliotecas gráficas ativas o usuário poderá executar um programa em modo gráfico remotamente, sendo que o processo estará rodando no servidor e a janela será exibida no terminal do cliente.

Além dos parâmetros “normais” podemos acrescentar alguns como por exemplo:

AllowUsers root user1 user2 # Se esta diretiva existir somente os usuários listados poderão abrir conexão ssh.
DenyUsers root user1 user2 # Se esta diretiva existir os usuários listados NÃO poderão fazer conexão ssh.
AllowGroups grupo # Idem AllowUsers para grupo
DenyGroups grupo # Idem DenyUsers para grupo
UsePAM yes # Habilita o login utilizando contas de máquinas remotas devidamente cadastradas, ex: LDAP

Feitas as configurações podemos (re)iniciar os serviço com o comando:

```
service sshd (re)start
```

33.3 Testes

Para testar podemos usar nossa própria máquina como cliente. Para isto basta executarmos o comando:

*ssh usuario@192.168.2.**X***

Com isto abriremos uma conexão com o servidor, podendo executar todos os comandos, como se estivéssemos logados localmente.
Podemos também fazer cópias de arquivos com o scp. Por exemplo:

```
scp -r usuario@192.168.2.X:diretorio ./
```

Com este comando copiamos recursivamente o diretório “diretorio” do servidor 192.168.2.X para o diretório corrente local.

Segurança e Monitoramento de Redes

34 Servidor *cache/proxy* Squid

34.1 Introdução[13]

Podemos resumir o significado de servidor proxy como uma espécie de "cache comunitário", onde toda página que um usuário visualizar ficará armazenada e quando outro (ou o mesmo) usuário requisitar a mesma página, ela não será trazida da Internet novamente, simplesmente será lida do disco e entregue, economizando tráfego de rede (isso se a página não tiver sido modificada na origem, caso isto tenha acontecido, ela será trazida da Internet novamente),Ilustração 26. Um proxy pode, além disso, fazer o controle de conteúdo, barrando o acesso a certos sites, por exemplo.

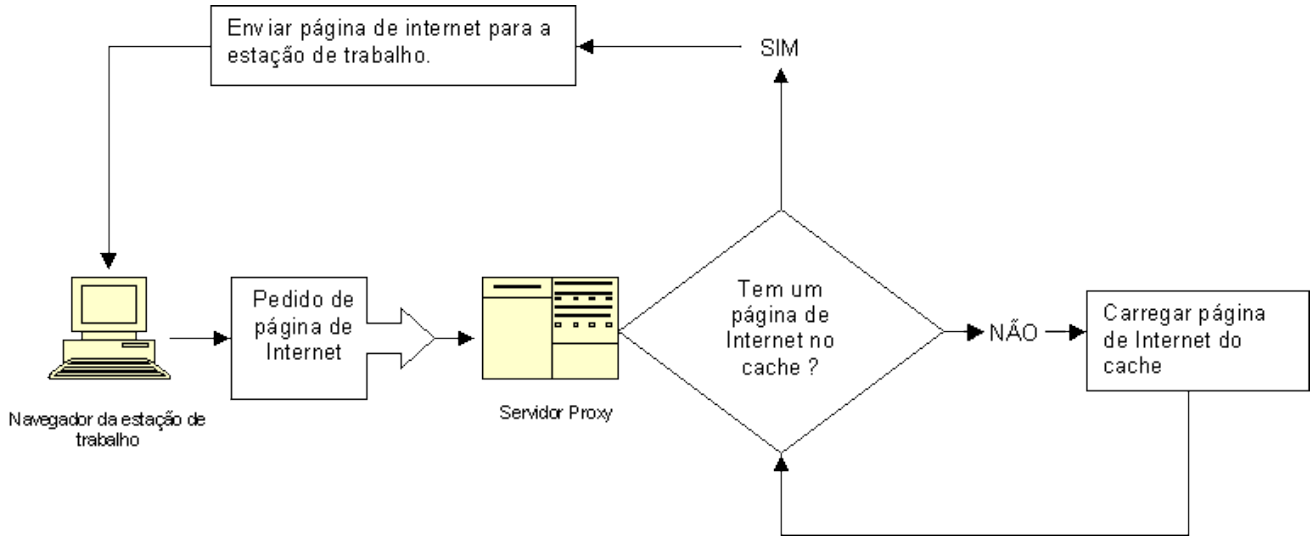


Ilustração 26: Funcionamento do Squid

34.2 Instalação e configuração

Para instalar o squid basta digitarmos o comando:

```
urpmi squid
```

Em seguida iniciamos o serviço com o comando:

```
service squid start
```

Para funcionar como cache, armazenamento centralizado de páginas, basta isto. Já a função de filtro (*firewall*) será estudada no próximo módulo, Segurança e Monitoramento de Redes.

O principal arquivo de configuração é o /etc/squid/squid.conf. As principais diretivas deste arquivo são:

13 Texto obtido de http://www.fundao.wiki.br/articles.asp?cod=199

http_port 3128	# Porta à qual o squid atenderá
cache_dir ufs /var/spool/squid 100 16 256	# Diretório de cache do tipo ufs, com armazenamento em /var/spool/squid, tamanho total de 100 MB, com 16 subdiretórios e cada um deles com 256 subdiretórios. Obs.: recomenda-se aumentar o tamanho total e mais nenhum outro parâmetro.
cache_mem 8 MB	# Tamanho da cache em RAM. Dependendo do uso da máquina, ocupe metade da RAM total.
maximum_object_size 4096 KB	# Tamanho máximo de um único objeto. Recomenda-se 16384 (16 MB). Isto é interessante quando faz-se *downloads* de arquivos.
visible_hostname m**X**.redes**X**.edu.br	# Nome real do servidor.

Se mudar algum parâmetro lembre-se de reiniciar o serviço (service squid restart).
As mensagens de erro que o squid envia aos usuários, por exemplo que determinado sítio tem acesso proibido, podem ser personalizadas. Existem várias e ficam no diretório /etc/squid/errors/.

34.2.1 Testes

Para testar devemos configurar nosso navegador para usar o nosso próprio proxy. Para isto abra o Firefox e clique em Editar, Preferências, Avançado, Aba Rede, Configurações, ajuste para Configuração manual de proxy e acrescente em HTTP: localhost, Porta: 3128 e clique em “Usar este proxy para todos os protocolos”. Feche o navegador para que a configuração se torne válida, veja Ilustração 27.

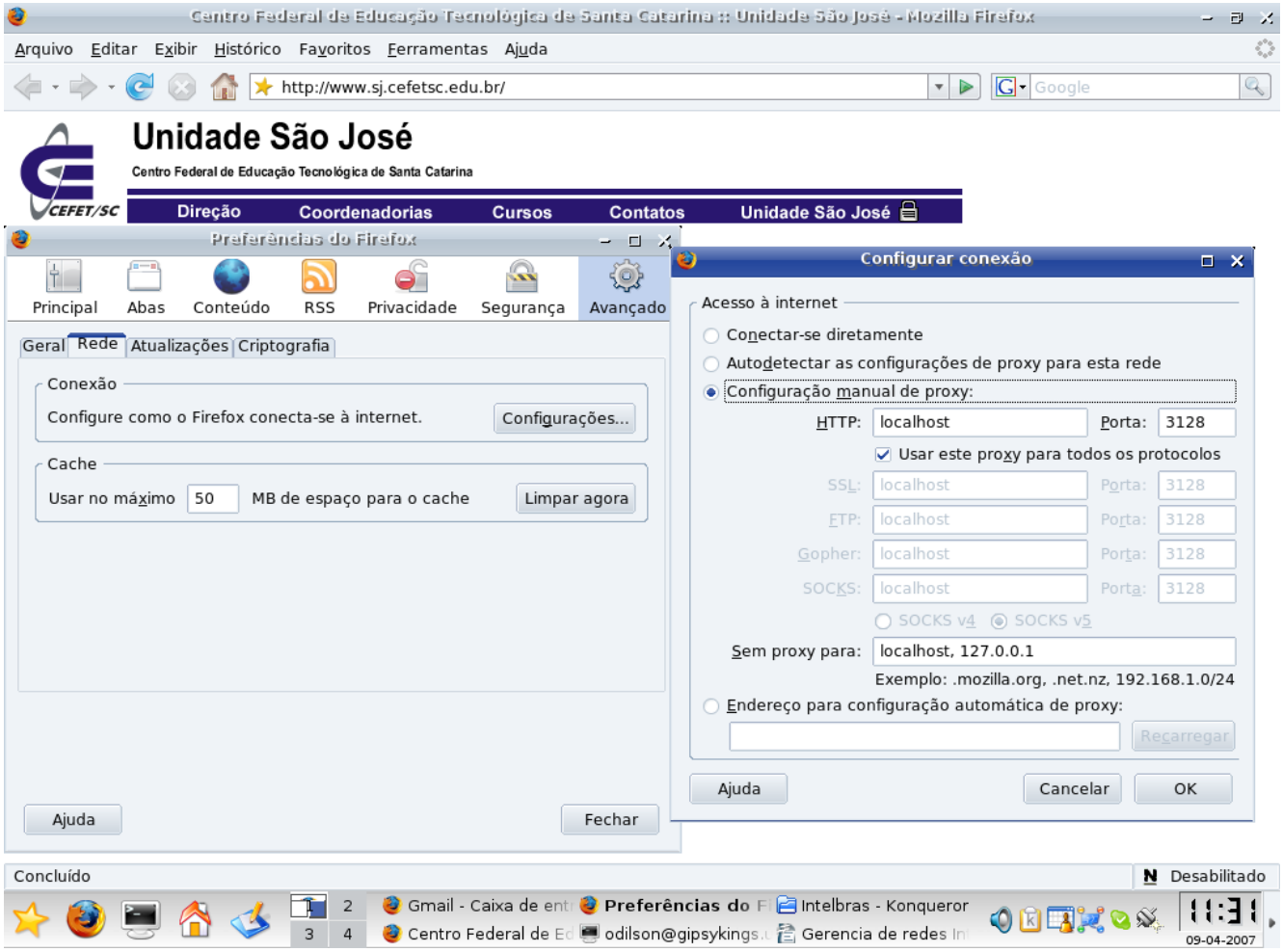


Ilustração 27: Configuração manual do proxy no Firefox

Em seguida acesse o site http://rpm.pbone.net/ e baixe algum arquivo, por exemplo amsn. Meça o tempo.
Vasculhe o diretório /var/spool/squid, por exemplo com du -s /var/spool/squid/*, e procure por um diretório/arquivo de mais ou menos o tamanho que você baixou. Baixe novamente o mesmo arquivo e do mesmo lugar. Meça o tempo e compare com o anterior. Se tudo correu bem a velocidade agora foi muito maior, já que o navegador buscou o arquivo no próprio disco.

34.3 Listas de controle de acesso

Com o Squid é possível o bloqueio de acesso a determinados sites.
Para operacionalizar esta função primeiramente devemos criar as chamadas acls (*access control list*) que são simplesmente listas de sites ou ips. O nome destas listas podem ser inventadas a vontade.
Após a criação destas listas deve-se criar as regras de acesso que incluirão as listas criadas.
A análise das regras, por parte do Squid, é seqüencial, ou seja, o Squid vai lendo as regras uma a uma e, assim que encontrar uma regra que enquadre o "pacote", pára a análise.

Portanto, nas regras podemos ter duas políticas diversas: libera alguns e proíbe todo o restou ou proíbe alguns e libera o restante. A adoção de uma ou outra depende da política pretendida.

34.3.1 Exemplos

- Liberando o acesso à internet a um único computador ou à uma rede:
  acl permitir_computadores src 192.168.2.5/255.255.255.0
  192.168.2.14/255.255.255.0 192.168.5.0/255.255.255.0
  http_access allow permitir_computadores
- Bloqueando o acesso à internet a um único computador ou à uma rede:
  acl proibir_computadores src 192.168.2.7/255.255.255.0
  192.168.9.0/255.255.255.0
  http_access deny proibir_computadores
- Bloqueando acesso à sites indesejados:
  acl proibir_sites dstdomain “/etc/squid/listas/proibidos”
  http_access deny proibir_sites
  Aqui devemos criar o arquivo /etc/squid/listas/proibidos e inserir os sites indesejados, um por linha.
- Bloqueando ou liberando o acesso a sites com palavras chaves:
  acl proibir_palavras url_regex -i “/etc/squid/listas/palavras”
  http_access deny proibir_palavras
  Aqui devemos criar o arquivo /etc/squid/listas/palavras e inserir as palavras indesejadas, uma por linha.
- Restringindo o horário de acesso:
  acl horario time MTWHF 08:00-18:00
  http_access allow permitir_computadores horario
  http_access deny permitir_computadores
  Obs.: S=domingo, M=segunda, T=terça, W=quarta, H=quinta, F=sexta e A=sábado

35 *Firewall* com iptables

35.1 Introdução

O *Firewall* é um programa que tem como objetivo proteger a máquina contra acessos indesejados, tráfego indesejado, proteger serviços que estejam rodando na máquina e bloquear a passagem de coisas que não se deseja receber (como conexões vindas da Internet para sua rede local segura, evitando acesso aos dados corporativos de uma empresa ou a seus dados pessoais). No kernel do Linux 2.4, foi introduzido o *firewall* `iptables’ (também chamado de netfilter) que substitui o `ipchains’ dos kernels da série 2.2. Este novo *firewall* tem como vantagem ser muito estável (assim como o `ipchains’ e `ipfwadm’), confiável, permitir muita flexibilidade na programação de regras pelo administrador do sistema, mais opções disponíveis ao administrador para controle de tráfego, controle independente do tráfego da rede local/entre redes/interfaces devido a nova organização das etapas de roteamento de pacotes.

O `iptables' é um ***firewall* a nível de pacotes** e funciona baseado no endereço/porta de origem/destino do pacote, prioridade, etc. Ele funciona através da comparação de regras para saber se um pacote tem ou não permissão para passar. Em *firewalls* mais restritivos, o pacote é bloqueado e registrado para que o administrador do sistema tenha conhecimento sobre o que está acontecendo em seu sistema.

Ele também pode ser usado para modificar e monitorar o tráfego da rede, fazer NAT (*masquerading, source* nat, *destination* nat), redirecionamento de pacotes, marcação de pacotes, modificar a prioridade de pacotes que chegam/saem do seu sistema, contagem de bytes, dividir tráfego entre máquinas, criar proteções anti-*spoofing*, contra syn flood, DoS, etc. O tráfego vindo de máquinas desconhecidas da rede pode também ser bloqueado/registrado através do uso de simples regras. As possibilidades oferecidas pelos recursos de filtragem `iptables' como todas as ferramentas UNIX maduras dependem de sua imaginação, pois ele garante uma grande flexibilidade na manipulação das regras de acesso ao sistema, precisando apenas conhecer quais interfaces o sistema possui, o que deseja bloquear, o que tem acesso garantido, quais serviços devem estar acessíveis para cada rede, e iniciar a construção de seu *firewall*.

O `iptables' ainda tem a vantagem de ser modularizável, funções podem ser adicionadas ao *firewall* ampliando as possibilidades oferecidas. Afirma-se que este é um *firewall* que tem possibilidades de gerenciar tanto a segurança em máquinas isoladas como roteamento em grandes organizações, onde a passagem de tráfego entre redes deve ser minuciosamente controlada.

Um *firewall* não funciona de forma automática (instalando e esperar que ele faça as coisas por você), é necessário pelo menos conhecimentos básicos de rede tcp/ip, roteamento e portas para criar as regras que farão a segurança de seu sistema. A segurança do sistema depende do controle das regras que serão criadas por você, as falhas humanas são garantia de mais de 95% de sucesso nas invasões.

Enfim o `iptables' é um *firewall* que agrada tanto a pessoas que desejam uma segurança básica em seu sistema, quando administradores de grandes redes que querem ter um controle minucioso sobre o tráfego que passam entre suas interfaces de rede (controlando tudo o que pode passar de uma rede a outra), controlar o uso de tráfego, monitoração, etc.

35.1.1 Características do firewall iptables

- ✔ Especificação de portas/endereço de origem/destino
- ✔ Suporte a protocolos TCP/UDP/ICMP (incluindo tipos de mensagens icmp)
- ✔ Suporte a interfaces de origem/destino de pacotes
- ✔ Manipula serviços de *proxy* na rede
- ✔ Tratamento de tráfego dividido em *chains* (para melhor controle do tráfego que entra/sai da máquina e tráfego redirecionado.
- ✔ Permite um número ilimitado de regras por *chain*
- ✔ Muito rápido, estável e seguro
- ✔ Possui mecanismos internos para rejeitar automaticamente pacotes duvidosos ou mal formados.

- Suporte a módulos externos para expansão das funcionalidades padrões oferecidas pelo código de *firewall*
- Suporte completo a roteamento de pacotes, tratadas em uma área diferente de tráfegos padrões.
- Suporte a especificação de tipo de serviço para priorizar o tráfego de determinados tipos de pacotes.
- Permite especificar exceções para as regras ou parte das regras
- Suporte a detecção de fragmentos
- Permite enviar alertas personalizados ao `syslog' sobre o tráfego aceito/bloqueado.
- Redirecionamento de portas
- *Masquerading*
- Suporte a SNAT (modificação do endereço de origem das máquinas para um único IP ou faixa de IP's).
- Suporte a DNAT (modificação do endereço de destino das máquinas para um único IP ou fixa de IP's)
- Contagem de pacotes que atravessaram uma interface/regra
- Limitação de passagem de pacotes/conferência de regra (muito útil para criar proteções contra, *syn flood*, ping *flood*, DoS, etc).

35.1.2 Como funciona um firewall ?

O FILTRO DE PACOTES do Linux funciona mediante regras estabelecidas. Todos os pacotes entram no kernel para serem analisados. As CHAINS (correntes) são as situações possíveis dentro do kernel. Quando um pacote entra no kernel, este verifica o destino do pacote e decide qual *chain* irá tratar do pacote. Isso se chama roteamento interno. Os tipos de *chains* irão depender da tabela que estaremos utilizando no momento. Existem 3 tabelas possíveis:

- *filter*: é a tabela *default*. Quando não especificarmos a tabela, a *filter* será utilizada. Refere-se às atividades normais de tráfego de dados, sem a ocorrência de NAT. Admite as *chains* INPUT, OUTPUT e FORWARD.
- nat: utilizada quando há NAT. Exemplo: passagem de dados de uma rede privada para a Internet. Admite as *chains* PREROUTING, OUTPUT e POSTROUTING.
- mangle (despedaçar): Usada para marcar pacotes permitindo por exemplo o controle de fluxo e interfaces.

35.2 Cadeias iptables

Na Ilustração 28 temos a representação das principais cadeias (*chain*) do iptables. A INPUT e FORWARD pertencem a tabela *filter* e a PREROUTING e POSTROUTING a tabela NAT.

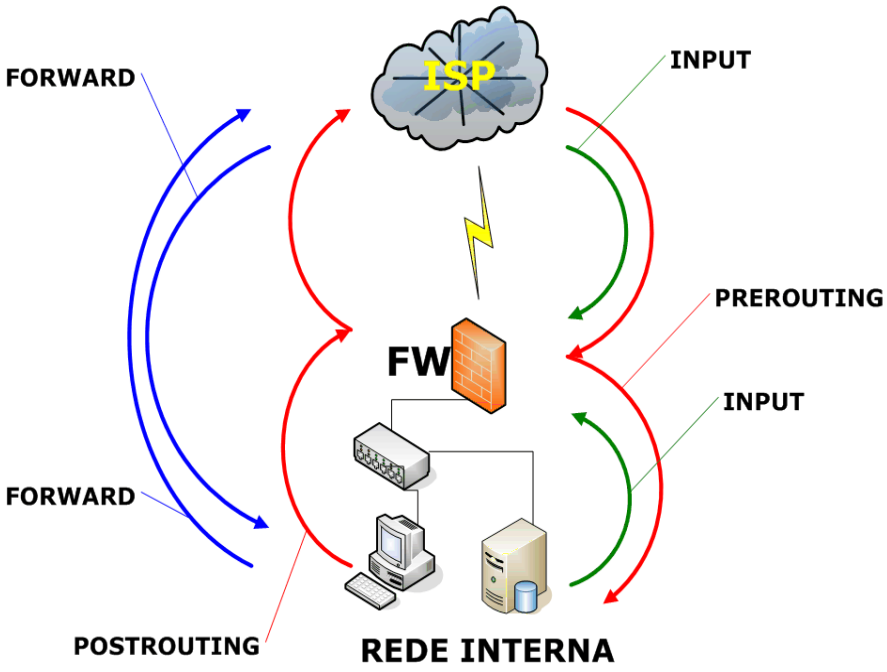


Ilustração 28: Cadeias mais usadas do iptables

As regras (*rules*) de *firewall*, geralmente, são compostas assim:

```
iptables [-t tabela] [opção] [chain] [dados] -j [ação]
```

Exemplo:

```
iptables -A FORWARD -d 192.168.1.1 -j DROP
```

A linha acima determina que todos os pacotes destinados (-d) à máquina 192.168.1.1 devem ser descartados. No caso:

- tabela: *filter* (é a *default*)
- opção: -A
- chain: FORWARD
- dados: -d 192.168.1.1
- ação: DROP

O iptables monta 3 tabelas distintas, ***Filter***, **NAT** e **Mangle**. Sendo que, em cada uma delas, o processamento da regras é seqüencial, ou seja, caso o iptables encontre uma regra onde o pacote se enquadre, ele pára, e encaminha o pacote seguindo esta regra. A única exceção são os casos de log, onde o log é armazenado e continua o processamento do iptables.

Agora vamos detalhar as duas principais tabelas.

35.3 Tabela Filter

35.3.1 São três, as possíveis chains:

- INPUT: utilizada quando o destino final é a própria máquina *firewall*;
- OUTPUT: qualquer pacote gerado na máquina *firewall* e que deva sair para a rede será tratado pela *chain* OUTPUT;
- FORWARD: qualquer pacote que atravessa o *firewall*, oriundo de uma máquina e direcionado a outra, será tratado pela *chain* FORWARD.

35.3.2 As principais opções são:

-P - *Policy* (política). Altera a política da *chain*. A política inicial de cada *chain* é ACCEPT. Isso faz com que o *firewall*, inicialmente, aceite qualquer INPUT, OUTPUT ou FORWARD. A política pode ser alterada para DROP, que irá negar o serviço da *chain*, até que uma opção -A entre em vigor. O -P não aceita REJECT ou LOG. Exemplos:

```
iptables -P FORWARD DROP

iptables -P INPUT ACCEPT
```

-A - *Append* (anexar). Acresce uma nova regra à *chain*. Tem prioridade sobre o -P. Geralmente, como buscamos segurança máxima, colocamos todas as *chains* em política DROP, com o -P e, depois, abrimos o que é necessário com o -A. Exemplos:

```
iptables -A OUTPUT -d 172.20.5.10 -j ACCEPT #iptables -A FORWARD -s 10.0.0.1 -j
DROP

iptables -A FORWARD -d www.chat.com.br -j DROP
```

-D - *Delete* (apagar). Apaga uma regra. A regra deve ser escrita novamente, trocando-se a opção para -D. Exemplos: Para apagar as regras anteriores, usa-se:

```
iptables -D OUTPUT -d 172.20.5.10 -j ACCEPT #iptables -D FORWARD -s 10.0.0.1 -j
DROP

iptables -D FORWARD -d www.chat.com.br -j DROP
```

Também é possível apagar a regra pelo seu número de ordem. Pode-se utilizar o -L para verificar o número de ordem. Verificado esse número, basta citar a *chain* e o número de ordem. Exemplo:

```
iptables -D FORWARD 4
```

Isso deleta a regra número 4 de *forward*.
-L - *List* (listar). Lista as regras existentes. Exemplos:

```
iptables -L

iptables -L FORWARD
```

-F - *Flush* (esvaziar). Remove todas as regras existentes. No entanto, não altera a política (-P). Exemplos:

```
iptables -F

iptables -F FORWARD
```

35.3.3 Chains

As *chains* já são conhecidas:
INPUT - Refere-se a todos os pacotes destinados à máquina *firewall*.
OUTPUT - Refere-se a todos os pacotes gerados na máquina *firewall*.
FORWARD - Refere-se a todos os pacotes oriundos de uma máquina e destinados a outra. São pacotes que atravessam a máquina *firewall*, mas não são destinados a ela.

35.3.4 Dados

Os elementos mais comuns para se gerar dados são os seguintes:
-s - *Source* (origem). Estabelece a origem do pacote. Geralmente é uma combinação do endereço IP com a máscara de sub-rede, separados por uma barra. Exemplo:

```
-s 172.20.0.0/255.255.0.0
```

No caso, vimos a sub-rede 172.20.0.0. Para *hosts*, a máscara sempre será 255.255.255.255. Exemplo:

```
-s 172.20.5.10/255.255.255.255
```

Agora vimos o *host* 172.20.5.10. Ainda no caso de *hosts*, a máscara pode ser omitida. Caso isso ocorra, o iptables considera a máscara como 255.255.255.255. Exemplo:

```
-s 172.20.5.10
```

Isso corresponde ao *host* 172.20.5.10. Há um recurso para simplificar a utilização da máscara de sub-rede. Basta utilizar a quantidade de bits 1 existentes na máscara. Assim, a máscara 255.255.0.0 vira 16. A utilização fica assim:

```
-s 172.20.0.0/16
```

Outra possibilidade é a designação de *hosts* pelo nome. Exemplo:

```
-s www.chat.com.br
```

Para especificar qualquer origem, utilize a rota *default*, ou seja, 0.0.0.0/0.0.0.0, também admitindo 0/0.
-d - *Destination* (destino). Estabelece o destino do pacote. Funciona exatamente como o -s, incluindo a sintaxe.
-p - *Protocol* (protocolo). Especifica o protocolo a ser filtrado. O protocolo IP pode ser especificado pelo seu número (vide /etc/protocols) ou pelo nome. Os protocolos mais utilizados são udp, tcp e icmp. Exemplo:

```
-p icmp
```

-i - *In-Interface* (interface de entrada). Especifica a interface de entrada. As interfaces existentes podem ser vistas com o comando #ifconfig. O -i não pode ser utilizado com a *chain* OUTPUT. Exemplo:

```
-i ppp0
```

O sinal + pode ser utilizado para simbolizar várias interfaces. Exemplo:

```
-i eth+
```

eth+ refere-se à eth0, eth1, eth2 etc.
-o - *Out-Interface* (interface de saída). Especifica a interface de saída. Similar a -i, inclusive nas flexibilidades. O -o não pode ser utilizado com a *chain* INPUT.
! - Exclusão. Utilizado com -s, -d, -p, -i, -o e outros, para excluir o argumento. Exemplo:

```
-s ! 10.0.0.1
```

Isso refere-se a qualquer endereço de entrada, exceto o 10.0.0.1.

```
-p ! tcp
```

Todos os protocolos, exceto o TCP.
--sport - *Source Port*. Porta de origem. Só funciona com as opções -p udp e -p tcp. Exemplo:

```
-p tcp --sport 80
```

Refere-se à porta 80 sobre protocolo TCP.
--dport - *Destination Port*. Porta de destino. Só funciona com as opções -p udp e -p tcp. Similar a --sport.

35.3.5 Ações

As principais ações são:
ACCEPT - Aceitar. Permite a passagem do pacote.
DROP - Abandonar. Não permite a passagem do pacote, descartando-o. Não avisa a origem sobre o ocorrido.
REJECT - Igual ao DROP, mas avisa a origem sobre o ocorrido (envia pacote icmp unreachable).
LOG - Cria um log referente à regra, em /var/log/messages. Usar antes de outras ações.

35.3.6 Exemplos comentados de regras de firewall (tabela filter)

```
iptables -L
```

Lista todas as regras existentes.

```
iptables -F
```

Apaga todas as regras sem alterar a política.

```
iptables -P FORWARD DROP
```

Estabelece uma política de proibição inicial de passagem de pacotes entre sub-redes.
```
iptables -A FORWARD -j DROP
```

Todos os pacotes oriundos de qualquer sub-rede e destinados a qualquer sub-rede deverão ser descartados.
```
iptables -A FORWARD -j ACCEPT
```

Todos os pacotes oriundos de qualquer sub-rede e destinados a qualquer sub-rede deverão ser aceitos.
```
iptables -A FORWARD -s 10.0.0.0/8 -d www.chat.com.br -j DROP
```

Os pacotes oriundos da sub-rede 10.0.0.0 (máscara 255.0.0.0) e destinados ao *host* www.chat.com.br deverão ser descartados.
```
iptables -A FORWARD -s 10.0.0.0/8 -d www.chat.com.br -j REJECT
```

Os pacotes oriundos da sub-rede 10.0.0.0 (máscara 255.0.0.0) e destinados ao *host* www.chat.com.br deverão ser descartados. Deverá ser enviado um ICMP avisando à origem.
```
iptables -A FORWARD -d www.chat.com.br -j DROP
```

Os pacotes oriundos de qualquer lugar e destinados ao *host* www.chat.com.br deverão ser descartados.
```
iptables -A FORWARD -d 10.0.0.0/8 -s www.chat.com.br -j DROP
```

Os pacotes destinados à sub-rede 10.0.0.0 (máscara 255.0.0.0) e oriundos do *host* www.chat.com.br deverão ser descartados.
```
iptables -A FORWARD -s www.chat.com.br -j DROP
```

Os pacotes oriundos do host www.chat.com.br e destinados a qualquer lugar deverão ser descartados.
```
iptables -A FORWARD -s 200.221.20.0/24 -j DROP
```

Os pacotes oriundos da sub-rede 200.221.20.0 (máscara 255.255.255.0) e destinados a qualquer lugar deverão ser descartados.
```
iptables -A FORWARD -s 10.0.0.5 -p icmp -j DROP
```

Os pacotes icmp oriundos do *host* 10.0.0.5 e destinados a qualquer lugar deverão ser descartados.
```
iptables -A FORWARD -i eth0 -j ACCEPT
```

Os pacotes que entrarem pela interface eth0 serão aceitos.
```
iptables -A FORWARD -i ! eth0 -j ACCEPT
```

Os pacotes que entrarem por qualquer interface, exceto a eth0, serão aceitos.
```
iptables -A FORWARD -s 10.0.0.5 -p tcp --sport 80 -j LOG
```

O tráfego de pacotes TCP oriundos da porta 80 do *host* 10.0.0.5 e destinados a qualquer lugar deverá ser gravado em log. No caso, /var/log/messages.

```
iptables -A FORWARD -p tcp --dport 25 -j ACCEPT
```

Os pacotes TCP destinados à porta 25 de qualquer host deverão ser aceitos.

35.3.7 Impasses

Ao se fazer determinadas regras, devemos prever o retorno. Assim, digamos que exista a seguinte situação:

```
iptables -P FORWARD DROP

iptables -A FORWARD -s 10.0.0.0/8 -d 172.20.0.0/16 -j ACCEP
```

Com as regras anteriores, fechamos todo o FORWARD e depois abrimos da sub-rede 10.0.0.0 para a sub-rede 172.20.0.0. No entanto, não tornamos possível a resposta da sub-rede 172.20.0.0 para a sub-rede 10.0.0.0. O correto, então, seria:

```
iptables -P FORWARD DROP

iptables -A FORWARD -s 10.0.0.0/8 -d 172.20.0.0/16 -j ACCEPT

iptables -A FORWARD -d 10.0.0.0/8 -s 172.20.0.0/16 -j ACCEPT
```

35.3.8 Extensões

As extensões permitem filtragens especiais, principalmente contra ataques de *hackers*. Quando necessárias, devem ser as primeiras linhas do *firewall*. As mais importantes são:

Contra Ping

```
iptables -A FORWARD -p icmp --icmp-type echo-request -j DROP
```

Contra Ping *of Death*

```
iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s -j
ACCEPT
```

Contra ataques Syn-flood

```
iptables -A FORWARD -p tcp -m limit --limit 1/s -j ACCEPT
```

Contra Port scanners avançados (nmap)

```
iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST -m limit --limit 1/s -j
ACCEPT
```

35.4 Tabela NAT - Network Address Translator

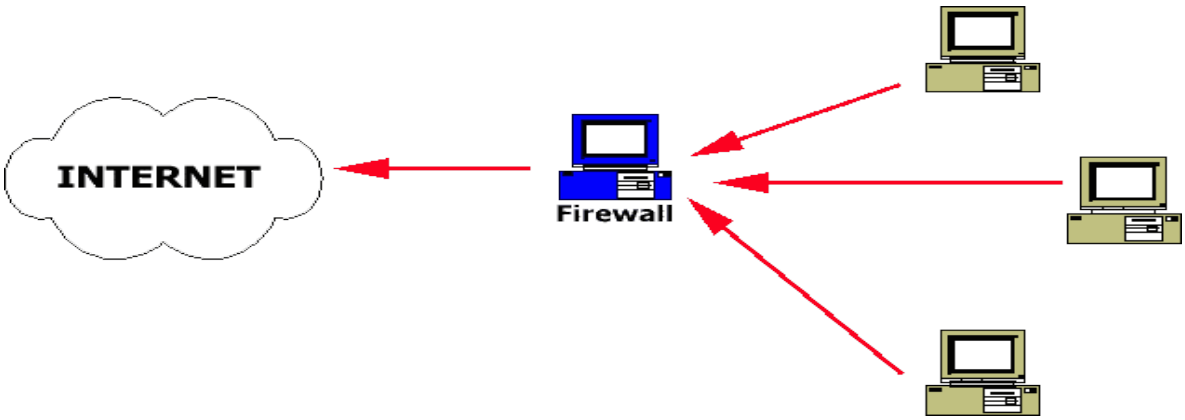
Existem vários recursos que utilizam NAT. Os mais conhecidos são:

- ✔ Mascaramento (*masquerading*)
- ✔ Redirecionamento de portas (*port forwarding*)
- ✔ Redirecionamento de servidores (*forwarding*)
- ✔ Proxy transparente (*transparent proxy*)
- ✔ Balanceamento de carga (*load balance*)
- ✔ Aumentar a capacidade de numeração dos números IP, assim mais *hosts*

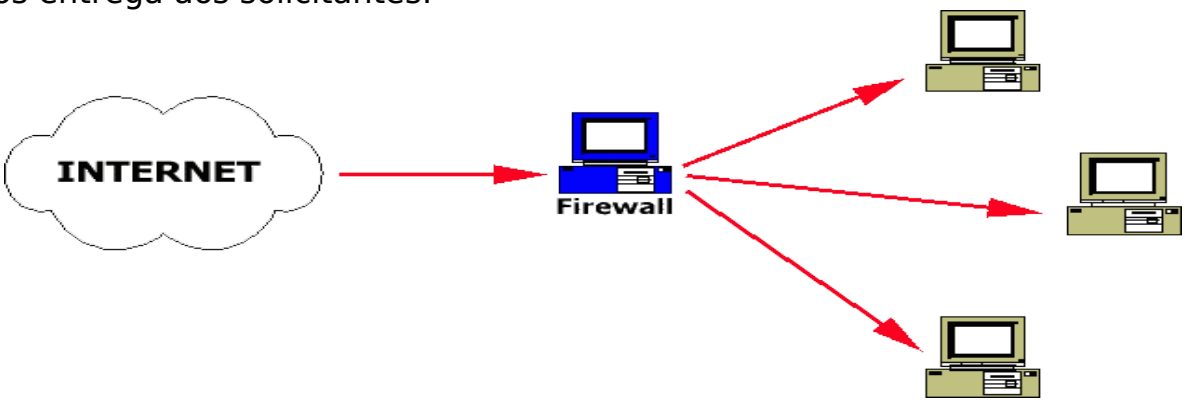
serão possíveis na internet sem precisar aumentar a capacidade de numeração dos atuais 32 bits, ou 4 bytes.

35.4.1 Mascaramento

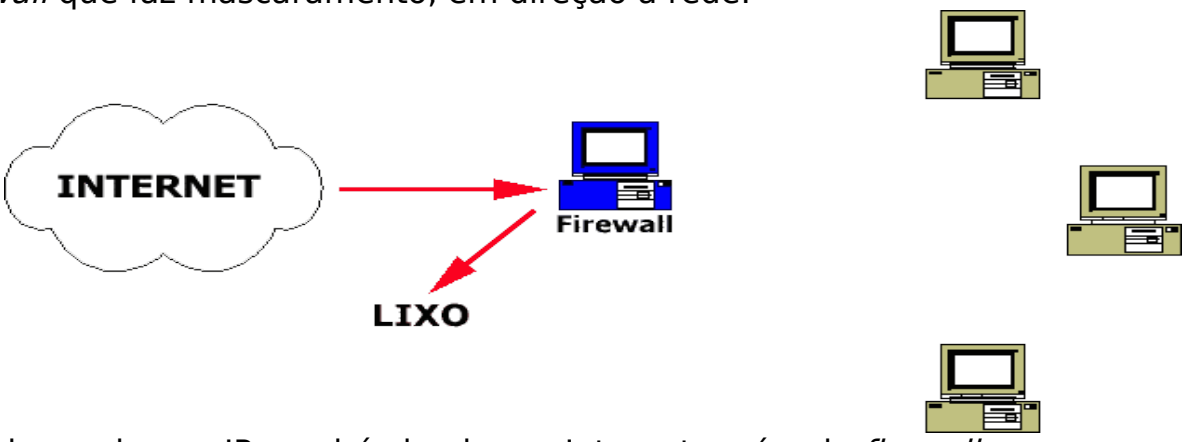
O mascaramento é uma forma de fazer NAT (*Network Address Translation*). Com isso, é possível fazer uma rede inteira navegar na Internet com segurança. A rede solicita os dados para a máquina que faz o mascaramento. Essa busca tais dados na Internet...



...e os entrega aos solicitantes:



No entanto, um *host* da Internet, por vontade própria, não consegue ultrapassar o *firewall* que faz mascaramento, em direção à rede:



O único endereço IP que irá circular na Internet será o do *firewall*.
O mascaramento também possui um esquema de funcionamento. Como haverá trocas de endereços, deveremos utilizar a tabela NAT para fazer isso.

35.4.2 Redirecionamento de portas

O redirecionamento de portas ocorre quando desejamos alterar a porta de destino de uma requisição. Exemplo: tudo que for destinado à porta 23 de qualquer máquina, quando passar pela máquina *firewall*, será redirecionado para a porta 10000 de outro *server*.

35.4.3 Redirecionamento de servidores

Todos os pacotes destinados a um servidor ou porta do mesmo, serão redirecionados para outro servidor ou porta de outro servidor.

35.4.4 Proxy transparente

É a técnica que força o uso de um servidor *proxy* na rede.

35.4.5 Balanceamento de carga

O balanceamento de carga (*load balance*) é uma técnica utilizada para distribuir carga em *clusters* servidores. Entende-se por *cluster*, uma série de servidores grupados e sincronizados, a fim de conterem os mesmos dados. O *load balance* é o ato de distribuir os clientes aos servidores mais desocupados. Esse trabalho também pode ser feito por servidores DNS.

35.4.6 Divisão do NAT

- SNAT: aplica-se quando desejamos alterar o endereço de origem do pacote. Somente a chain POSTROUTING faz SNAT. O mascaramento é um exemplo de SNAT.
- DNAT: aplica-se quando desejamos alterar o endereço de destino do pacote. As chains PREROUTING e OUTPUT fazem DNAT. O redirecionamento de porta, o redirecionamento de servidor, o load balance e o proxy transparente são exemplos de DNAT.

35.4.7 Regras de NAT

As regras mais utilizadas, além da maioria dos recursos descritos para uso com a tabela *filter*, contêm o seguinte:

35.4.7.1 Chains

Existem as seguintes *chains*:

- **PREROUTING**: utilizada para analisar pacotes que estão entrando no kernel para sofrerem NAT. O PREROUTING pode fazer ações de NAT com o endereço de destino do pacote. Isso é conhecido como DNAT (*Destination* NAT);
- **POSTROUTING**: utilizada para analisar pacotes que estão saindo do kernel, após sofrerem NAT. O POSTROUTING pode fazer ações de NAT com o endereço de origem do pacote. Isso é conhecido como SNAT (*Source* NAT);
- **OUTPUT**: utilizada para analisar pacotes que são gerados na própria

máquina e que irão sofrer NAT. O OUTPUT pode fazer ações de NAT com o endereço de destino do pacote. Também é DNAT.

35.4.7.2 *Opções*

-A - *Append* (anexar).
-D - Deletar.

35.4.7.3 *Dados*

-t - *Table* (tabela). Estabelece a tabela a ser utilizada. A tabela *default*, por omissão, é *filter*. Para o mascaramento ou NAT será nat. Exemplo:

```
iptables -t nat -A ...
```

--to - utilizado para definir IP e porta de destino, após um DNAT, ou de origem, após um SNAT. Deve ser utilizado após uma ação (-j ação). Assim:

```
-j DNAT --to 10.0.0.2

-j DNAT --to 10.0.0.2:80

-j SNAT --to 172.20.0.2
```

--dport - assim como -d define um *host* de destino, --dport define uma porta de destino. Deve ser utilizado antes de uma ação (-j ação). Antes de --dport, deve ser especificado um protocolo (-p). Exemplo:

```
-d 127.20.0.1 -p tcp --dport 80 -j DNAT --to 10.0.0.2
```

--sport - assim como -s define um *host* de origem, --sport define uma porta de origem. Deve ser utilizado antes de uma ação (-j ação).
--to-port - define uma porta de destino, após um REDIRECT.
Obs: A maioria dos dados básicos apresentados para a tabela *filter* continuam valendo. Exemplo: -p servirá para definir um protocolo de rede; -d define um *host* de destino.

35.4.7.4 *Ações*

SNAT - Utilizado com POSTROUTING para fazer ações de mascaramento da origem.
DNAT - Utilizado com PREROUTING e OUTPUT para fazer ações de redirecionamento de portas e servidores, balanceamento de carga e *proxy* transparente. Caso a porta de destino não seja especificada, valerá a porta de origem. No *firewall*, a porta que será redirecionada não pode existir ou estar ocupada por um *daemon*.
MASQUERADE - Faz mascaramento na saída de dados.
REDIRECT - Redireciona uma requisição para uma porta local do firewall.

35.4.8 Exemplos comentados de regras de firewall (tabela nat)

```
iptables -t nat -L
```

Mostra as regras de NAT ativas.

```
iptables -t nat -F
```

Apaga todas as regras de NAT existentes.

```
iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

Todos os pacotes que saírem pela interface ppp0 (modem) serão mascarados. Isso dá um nível de segurança elevado à rede que está atrás da ppp0. É uma boa regra para navegação na Internet. Note que esse tipo de mascaramento não usa SNAT.

```
iptables -t nat -A POSTROUTING -d 0/0 -j MASQUERADE
```

Tem o mesmo efeito da regra anterior. No entanto, parece ser menos segura, pois estabelece que qualquer pacote destinado a qualquer outra rede, diferente da interna, será mascarado. A regra anterior refere-se aos pacotes que saem por determinada interface. A opção -d 0/0 poderia ser -d 0.0.0.0/0 também. É uma outra regra para navegação na Internet.

```
iptables -t nat -A PREROUTING -t nat -p tcp -d 10.0.0.2 --dport 80 -j DNAT --to
172.20.0.1
```

Redireciona todos os pacotes destinados à porta 80 da máquina 10.0.0.2 para a máquina 172.20.0.1. Esse tipo de regra exige a especificação do protocolo. Como não foi especificada uma porta de destino, a porta de origem (80) será mantida como destino.

```
iptables -t nat -A OUTPUT -p tcp -d 10.0.0.10 -j DNAT --to 10.0.0.1
```

Qualquer pacote TCP, originado na máquina *firewall*, destinado a qualquer porta da máquina 10.0.0.10, será desviado para a máquina 10.0.0.1 .

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 200.20.0.1
```

Essa regra faz com que todos os pacotes que irão sair pela interface eth0 tenham o seu endereço de origem alterado para 200.20.0.1 .

```
iptables -t nat -A PREROUTING -i eth0 -j DNAT --to 172.20.0.1
```

Todos os pacotes que entrarem pela eth0 serão enviados para a máquina 172.20.0.1

```
iptables -t nat -A PREROUTING -i eth0 -j DNAT --to 172.20.0.1-172.20.0.3
```

Aqui haverá o *load balance*. Todos os pacotes que entrarem pela eth0 serão distribuídos entre as máquinas 172.20.0.1 , 172.20.0.2 e 172.20.0.3 .

```
iptables -t nat -A PREROUTING -s 10.0.0.0/8 -p tcp --dport 80 -j REDIRECT --to-
port 3128
```

Todos os pacotes TCP que vierem da rede 10.0.0.0, com máscara 255.0.0.0, destinados à porta 80 de qualquer *host*, não sairão; serão redirecionados para a porta 3128 do *firewall*. Isso é o passo necessário para fazer um *proxy* transparente. O *proxy* utilizado deverá aceitar esse tipo de recurso. No caso, o Squid, que aceita transparência, deverá estar instalado na máquina *firewall*, servindo na porta 3128.

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24 -o eth1 -j SNAT 200.20.5.0/24
```

Uma situação interessante: todos os pacotes que saírem da rede 192.168.1.0 serão transformados em 200.20.5.0.
Apesar de estarmos lidando com um *firewall*, que é um roteador controlado, há a possibilidade de fazermos algumas operações dentro da mesma sub-rede. No entanto, isso tem que ser bem estudado. Muitas vezes irá exigir regras especiais de roteamento estático (comando #route).

35.5 Salvando e recuperando tudo

As regras iptables poderão ser salvas com o comando:

```
iptables-save > /caminho_do_arquivo/nome_do_arquivo
```

A recuperação poderá ser feita pelo comando:

```
iptables-restore < /caminho_do_arquivo/nome_do_arquivo
```

Um típico exemplo de carregamento de regras de iptables, após a inicialização do sistema, seria:

```
echo 1 > /proc/sys/net/ipv4/ip_forward

iptables-restore < /etc/firewall
```

Isso pode ser inserido no fim do arquivo /etc/rc.local.

35.6 Aumentando o nível de segurança

Caso deseje aumentar o nível de segurança, evitando ataques diversos, digite COMO PRIMEIRAS regras:

```
iptables -A FORWARD -p icmp --icmp-type echo-request -j DROP

iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --limit 1/s -j
ACCEPT

iptables -A FORWARD -p tcp -m limit --limit 1/s -j ACCEPT

iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST -m limit --limit 1/s -j
ACCEPT
```

35.7 Instalando e configurando

Para instalarmos o firewall iptables basta digitarmos o comando:

```
urpmi iptables
```

As configurações são feitas com as inserções das regras, conforme indicado nos itens acima.

36 Firewall com Shorewall

36.1 Introdução[14]

O Shoreline *Firewall*, comumente conhecido como Shorewall, é uma ferramente de “alto nível” para configurar o iptables. Você descreve as necessidades de seu *firewall* usando entradas/diretivas num conjunto de arquivos de configuração. O Shorewall lê estes arquivos e monta as regras do iptables para atender suas demandas. O Shorewall pode usar um hardware dedicado de *firewall*, um sistema “multifuncional” de gateway/roteador/servidor ou uma máquina com sistema GNU/Linux.
Shorewall não é um *daemon*. Uma vez que o Shorewall tenha configurado o Netfilter (iptables), seu serviço finaliza e não permanece um processo Shorewall rodando no sistema. O programa /sbin/shorewall pode ser usado a qualquer momento para monitorar/ajustar o Netfilter *firewall*.
Características:

- Sofisticado.
- Flexível, provavelmente é a ferramenta de configuração do Netfilter mais flexível da atualidade.
- Extensamente documentado.
- Vem incluso nas principais distribuições.
- É constantemente atualizado.

Algumas alternativas ao Shorewall:

- **Firestarter** - http://www.fs-security.com
- **Firewall Builder** - http://www.fwbuilder.org
- **Arno's IPTABLES Firewall Script** - http://rocky.eld.leidenuniv.nl
- **Webmin** - http://www.webmin.com
- **FireHOL** - http://firehol.sourceforge.net

36.2 Zonas

36.2.1 Arquitetura de zonas

Diferentemente de quase todas as outras ferramentas de configuração do Netfilter disponíveis, Shorewall trabalha com conceito de ZONAS. Shorewall não assume nenhum papel específico a cada zona (diferente de outras ferramentas que explicitamente definem zonas Internet, DMZ, etc.), característica sobre a qual recaem a maior parte de sua flexibilidade.
Uma zona é um conceito abstrato que identifica a origem e o destino de um pacote. As mesmas se utilizam para a definição de regras de aceitação ou recusa de pacotes em função de sua procedência ou destino (junto com outras características do mesmo). As zonas podem ou não estar associadas a interfaces de rede, subredes de IP ou conjuntos de IPs (mediante o uso do patch ipset).

14 Texto obtido de http://www.grulic.org.ar/eventos/charlas/shorewall-2005-09.html

36.2.2 Zona "fw"

A única zona explicitamente associada a um equipamento/IP é a zona "fw", a qual "contém" o firewall que se está configurando. É importante no momento de configurar o Shorewall assegura-se de que foram configuradas vias de aceso suficientes à zona "fw" do lugar onde se está trabalhando pois, em caso contrário, pode-se perder o acesso ao mesmo.

36.3 Arquivos de configuração

Todos os arquivos de configuração do Shorewall são documentados e contém uma série de exemplos. Abaixo veremos as principais funcionalidades dos principais arquivos de configuração do mesmo.

36.3.1 shorewall.conf

É o principal arquivo de configuração do Shorewall. Algumas das variáveis importantes a se configurar neste arquivo são:

- STARTUP_ENABLED, deve-se ajustar para *Yes*, se não não “teremos” *firewall*.
- ADMINISABSENTMINDED, se está ajustada para *No*, somente o tráfego listado em /etc/shorewall/routestopped serão aceitos quando o Shorewall for parado. Caso esteja em *Yes*, todas as conexões serão aceitas.
- LOGRATE e LOGBURST, estes parâmetros indicam/limitam a rotação dos logs e a taxa de pacotes que são registrados. É útil quando se geram muitos pacotes inválidos na rede o que poderia gerar muitíssimo consumo do processador no *firewall*.
- IP_FORWARDING, este parâmetro determina se o Shorewall deve ou não habilitar o encaminhamento de pacotes IPV4.

36.3.2 zones

É neste arquivo que se definem as zonas do firewall (exceto a zona “fw” que sempre é definida). Exemplos:

```
#ZONE   TYPE

fw      firewall

net     ipv4

loc     ipv4

dmz     ipv4
```

36.3.3 interfaces

Neste arquivo se definem as interfaces que o *firewall* tomará conta e sua possível associação a uma zona (pode não ser necessário). Além disto se identificam certas propriedades a respeito da “interpretação” dos pacotes que entram ou

saem pelas mesmas. O formato de cada definição é:

```
ZONE    INTERFACE       BROADCAST       OPTIONS
```

Exemplos:

```
dmz     eth1            detect

net     eth1        detect          tcpflags,dhcp,routefilter,nosmurfs,logmartians

loc     eth0            detect          tcpflags,detectnets,nosmurfs
```

36.3.4 policy

Neste arquivo se definem as políticas para os pacotes que trafegam entre uma zona e outra. O formato das definições é:

```
SOURCE          DEST            POLICY          LOG LEVEL           LIMIT:BURST
```

As políticas possíveis são:

- ACCEPT, aceita-se a conexão.
- DROP, ignora-se a conexão.
- REJECT, rejeita-se explicitamente a conexão.
- QUEUE, envia o pedido à uma aplicação com o "*target*" QUEUE.
- CONTINUE, o pedido de conexão continua a ser analisado por outras regras.
- NONE, assume que nunca haverá conexão da origem para o destino.

Exemplos:

```
#SOURCE         DEST            POLICY          LOG LEVEL           LIMIT:BURST

$FW             net             ACCEPT

$FW             dmz             ACCEPT

$FW             loc             ACCEPT

$FW             all             ACCEPT

## Policies for traffic originating from the Internet zone (net)

net             dmz             ACCEPT

net             $FW             ACCEPT

net             loc             DROP    info

net             all             DROP    info

## THE FOLLOWING POLICY MUST BE LAST

all             all             ACCEPT
```

36.3.5 rules

Provavelmente este é o arquivo de configuração mais importante. Aqui se definem as regras que permitem ou negam o acesso a serviços e portas deste e a determinadas zonas do *firewall*. Também se definem as regras de DNAT e registro de certos pacotes. O formato das regras é:

```
ACTION  SOURCE  DEST_PROTO  DEST_PORT  SOURCE_PORT(S)  ORIGINAL_DEST  RATE_LIMIT  USER/GROUP
```

As ações podem ser: ACCEPT, DROP, REJECT, DNAT, DNAT-, REDIRECT, CONTINUE, LOG, QUEUE ou uma ação definida pelo usuário.
Exemplos:

```
REDIRECT   loc   8080   tcp   www   -   !200.135.233.1

ACCEPT:info   loc   net   tcp   25

ACCEPT   net:200.135.233.1   loc

## Redirecionamento de ip e portas

DNAT    net     loc:192.168.1.9 tcp     20,21,22,53,80  -       200.135.233.9

DNAT    net     loc:192.168.1.7:22 tcp  2222    -       200.135.233.9
```

36.3.6 masq

Este é o arquivo utilizado para definir mascaramento e SNAT. É essencial para as redes locais que desejam se conectar à internet via *firewall*. O formato das definições é:

```
INTERFACE               SUBNET          ADDRESS         PROTO   PORT(S) IPSEC
```

Exemplo:

```
eth0 eth1
```

36.3.7 Outros

Outros arquivos de configuração (lista incompleta) úteis são:

- **hosts**: Utiliza-se para associar grupos de *hosts* a uma zona. Em especial para definir múltiplas zonas sobre uma interface.
- **tunnels**: Utiliza-se para configurar automaticamente regras de Netfilter para tipos distintos de tuneis (IPSEC, OpenVPN, etc.)
- **tcrules**: Utiliza-se para carregar regras de tc (ferramenta para a configuração dos serviços de *traffic shaping* do kernel) desde o *firewall*.
- Muitos mais... (ver /etc/shorewall)

36.4 Alguns exemplos “reais”

Retirado de http://www.grulic.org.ar/eventos/charlas/shorewall-2005-09.html.

36.4.1 Firewall standalone

Um firewall conectado a Internet mediante uma interface ppp. Se considera que há somente uma zona (Net) sobre a qual “vivem” os possíveis clientes.
/etc/shorewall/shorewall.conf

```
STARTUP_ENABLED=Yes

IP_FORWARDING=Off

#LAST LINE -- DO NOT REMOVE
```

/etc/shorewall/zones

```
net Net Internet

#LAST LINE - ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

/etc/shorewall/interfaces

```
net ppp+ - norfc1918,nobogons,routefilter,logmartians,blacklist,tcpflags,nosmurfs

#LAST LINE - ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

/etc/shorewall/policy

```
fw all ACCEPT

net all DROP info

all all REJECT info

#LAST LINE -- DO NOT REMOVE
```

/etc/shorewall/rules

```
AllowWeb net fw   # Servidor http,https

AllowSSH net fw   # Servidor ssh

AllowPing net:172.16.0.0/16 fw

ACCEPT net:172.16.0.0/16 fw tcp 10000

#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

36.4.2 Firewall numa típica rede de zonas e interfaces

Este caso é corresponde ao de qualquer rede típica, onde se tem uma rede interna que se deseja conectar à (e proteger da) Internet. Neste caso assumimos que a rede interna está conectada a uma interface ethernet eth0 e a Internet por uma interface ppp. Assumimos também que deseja-se fazer *masquerading* da rede interna e que o servidor possui um domínio DHCP para a auto configuração dos *hosts* na rede local.
/etc/shorewall/shorewall.conf

```
STARTUP_ENABLED=Yes

#LAST LINE -- DO NOT REMOVE
```

/etc/shorewall/zones

```
loc Local Local networks

net Net Internet

#LAST LINE - ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

/etc/shorewall/interfaces

```
net ppp+ - norfc1918,nobogons,routefilter,logmartians,blacklist,tcpflags,nosmurfs

loc eth0 detect dhcp

#LAST LINE - ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

/etc/shorewall/policy

```
loc net ACCEPT

#loc fw ACCEPT

net all DROP info

all all REJECT info

#LAST LINE -- DO NOT REMOVE
```

/etc/shorewall/masq

```
ppp+ eth0

#LAST LINE -- ADD YOUR ENTRIES ABOVE THIS LINE -- DO NOT REMOVE
```

/etc/shorewall/rules

```
AllowWeb fw all

AllowWeb all fw   # Servidor http,https (para ambas zonas)

AllowSSH fw all

AllowSSH loc fw              # Servidor ssh para rede interna

ACCEPT:info net fw tcp 22000 # Servidor ssh para Internet (Informando sobre
conexões estabelecidas)

AllowSMB loc fw # Servidor samba

AllowSMB fw loc # Servidor samba

ACCEPT loc fw tcp 3128
```

```
ACCEPT fw all tcp 6667:6669

#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

36.4.3 Múltiplas zonas sobre uma interface

Neste exemplo vamos supor uma configuração de hardware e rede idêntica ao caso anterior, ou seja, um *firewall*/roteador com duas interfaces, uma ethernet eth0 conectada à rede interna que desejamos conectar à internet e uma interface ppp que conecta o *host* a dita rede. A diferença é que agora desejamos discriminar entre duas classes de *hosts* que podem conectar-se desde a rede local: um com acesso a todos os serviços de rede disponíveis e outro que pode utilizar somente http sobre a internet e smtp no *firewall* onde está instalado um MTA (*Mail Transfer Agent*). Também desejamos que os hosts da rede interna sejam configurados via dhcp.

/etc/shorewall/shorewall.conf

```
STARTUP_ENABLED=Yes

#LAST LINE -- DO NOT REMOVE
```

/etc/shorewall/zones

```
loc2 Local Local networks

loc1 Local Local networks

net Net Internet

#LAST LINE - ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

/etc/shorewall/interfaces

```
net ppp+ - norfc1918,nobogons,routefilter,logmartians,blacklist,tcpflags,nosmurfs

- eth0 detect dhcp

#LAST LINE - ADD YOUR ENTRIES ABOVE THIS ONE - DO NOT REMOVE
```

/etc/shorewall/host (primeira opção)

```
loc1 eth0:192.168.0.0/25

loc2 eth0:192.168.0.128/25

net eth0:0.0.0.0/0

#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS LINE -- DO NOT REMOVE
```

/etc/shorewall/host (segunda opção)

```
loc2 eth0:192.168.0.100/32,192.168.0.101/32,192.168.0.102/32

loc1 eth0:192.168.0.0/24
```

```
net eth0:0.0.0.0/0

#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS LINE -- DO NOT REMOVE
```

/etc/shorewall/policy

```
loc1 net ACCEPT

loc2 net REJECT

loc1 loc2 ACCEPT

loc2 loc1 ACCEPT

net all DROP info

all all REJECT info

#LAST LINE -- DO NOT REMOVE
```

/etc/shorewall/rules

```
AllowWeb loc2 net

AllowSMTP loc2 fw

AllowWeb fw all

AllowWeb loc1 fw   # Servidor http,https (para ambas zonas)

AllowSSH fw all

AllowSSH loc1 fw             # Servidor ssh para rede interna

ACCEPT:info net fw tcp 22000 # Servidor ssh para Internet (Informando sobre
conexões estabelecidas)

AllowSMB loc1 fw # Servidor samba

AllowSMB fw loc1 # Servidor samba

AllowSMB fw loc2 # Servidor samba

ACCEPT loc fw tcp 3128

ACCEPT fw all tcp 6667:6669

#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

36.4.4 Proxy transparente com Squid

/etc/squid/squid.conf

```
ACL local_network src 192.168.0.0/24
```

```
http_access allow local_network

httpd_accel_host virtual

httpd_accel_port 80

httpd_accel_with_proxy on

httpd_accel_uses_host_header on
```

/etc/shorewall/rules

```
REDIRECT  loc        3128     tcp      www

ACCEPT    fw         net      tcp      www

#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

36.4.5 Regras para P2P

/etc/shorewall/rules (p2p sobre o *firewall*)

```
ACCEPT net fw tcp 6881:6889,4661,4662,36711

ACCEPT net fw udp 6881:6889,4665,4672

#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

/etc/shorewall/rules (p2p sobre una máquina interna)

```
DNAT net loc:192.168.0.100 tcp 6881:6889,4661,4662,4711

DNAT net loc:192.168.0.100 udp 6881:6889,4665,4672

#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

36.4.6 Regras para DNAT

/etc/shorewall/rules

```
DNAT:info net loc:192.168.0.2 tcp www

DNAT:info net loc:192.168.0.2 tcp 443

DROP loc:!192.168.0.2 all tcp 25

DNAT:info net loc:192.168.0.2 tcp 25

#LAST LINE -- ADD YOUR ENTRIES BEFORE THIS ONE -- DO NOT REMOVE
```

36.5 Instalação e configuração

Para instalarmos o Shorewall digitamos o s comando:

```
urpmi shorewall
```

Editamos como mínimo os arquivos /etc/shorewall/

- zones
- interfaces
- policy e
- rules

segundo os modelos descritos anteriormente.

37 Anti-vírus

37.1 Introdução[15]

Praticamente não existem vírus que ataquem o sistema Linux. O motivo na verdade é a sobreposição de alguns fatores listados a seguir.
Para que um vírus infecte um programa executável num sistema com *kernel* Linux, numa distro GNU/Linux (Debian, Slackware, RedHat, Suse, Ubuntu, Kurumin, Mandriva, etc.) por exemplo, o executável precisa estar em arquivo com permissão de escrita para o usuário que esteja ativando o vírus. Tal situação é incomum. Numa instalação *desktop*, via de regra os arquivos executáveis têm como dono (*owner*) o administrador do sistema (*root*), e rodam em processo de usuário comum. Ou seja, a partir de uma conta não-privilegiada.
Além do que, quanto menos experiente for o usuário, menos provável que tenha ele mesmo feito a instalação do executável, e portanto, que seja o *owner* do arquivo correspondente. Assim, os usuários de Linux que menos entendem dos perigos de infecção viral são os que têm pastas pessoais (diretório *home*) menos férteis para isso. A medida que os usuários vão se “especializando” vão entendendo do assunto e sabendo as conseqüências dos vírus.
Prosseguindo, ainda que um vírus consiga infectar um programa executável, sua missão de proliferar-se esbarra em dificuldades das quais os limites nas permissões do dono do arquivo infectado são apenas o começo. As dificuldades continuam nos programas para conectividade, por serem esses no Linux construídos conservadoramente, sem os recursos de macros em alto nível que têm permitido, por exemplo, os recentes vírus de Windows propagarem-se tão rapidamente.
Esse conservadorismo não é uma característica do Linux, mas reflete diretamente importantes diferenças na base de usuários de plataformas livres e proprietárias. Diferenças na forma como essas bases atuam no processo de desenvolvimento, e na forma como a robustez e a popularidade dos programas é afetada por essa atuação, através dos respectivos modelos de licença e de negócio. Na forma, por exemplo, em que vacinas atuam. As lições aprendidas pela observação do que acontece no outro modelo servem, no modelo colaborativo, para vacinar não o software em si, mas o processo e a estratégia de desenvolvimento dos softwares livres, livres inclusive das estratégias de negócio de interessados que lhes sejam conflitantes.
Aplicativos e sistemas baseados em Linux são quase todos de código fonte

15 Texto obtido de http://www.cic.unb.br/docentes/pedro/trabs/virus_no_linux.html

aberto. Devido à quase totalidade desse mercado estar acostumado à disponibilidade do código-fonte, produtos distribuídos apenas em formato executável são ali raros, e encontram mais dificuldade para firmar presença. Isso tem dois efeitos no eco sistema viral, se considerarmos que a propagação ocorre em formato executável. Primeiro, programas com código fonte aberto são lugares difíceis para vírus se esconderem. Segundo, a (re)instalação por compilação do código-fonte corta completamente um dos principais vetores de propagação dos vírus.
Cada um desses obstáculos representa uma barreira significativa. Porém, é quando essas barreiras atuam em conjunto que a vida do vírus se complica. Um vírus de computador, da mesma forma que o biológico, precisa de uma taxa de reprodução maior do que a taxa de erradicação (morte), para se proliferar. Na plataforma Linux, cada um desses obstáculos reduz significativamente a taxa de reprodução. E, se a taxa de reprodução cai abaixo do nível necessário para substituir a população erradicada, o vírus está condenado à extinção nesse ambiente, mesmo antes das notícias alarmistas sobre o potencial de dano às vítimas.

37.1.1 Mas então, qual é a função do antivírus no Linux?

Há antivírus que rodam no Linux e você deve ouvir falar neles de vez em quando. Na verdade, esses programas permitem que uma máquina Linux procure vírus de computadores pessoais, máquinas Windows, Macintosh, etc, e não propriamente vírus para Linux. Esses antivírus são muito utilizados quando o Linux está rodando como servidor de email ou arquivos, permitindo que sejam pesquisadas todas as mensagens que forem recebidas, por exemplo.

37.2 Instalando e configurando o antivírus CLAMAV[16]

O Clan AntiVirus é um pacote de ferramentas antivírus sobre licença GPL desenhado especificamente para análise de correio eletrônico em servidores de correio. Este pacote contém várias utilidades nomeadamente um serviço com paralelismo (multi-threaded) flexível e escalável, um utilitário de análise para linha de comandos e uma ferramenta avançada para atualização automática da Base de Dados. O núcleo do pacote é um motor anti-viral disponível como biblioteca.
De seguida são listadas as suas principais funcionalidades:

- analisador de linha de comandos
- serviço rápido e com paralelismo com suporte para análise automática “ao acesso”
- interface milter para o sendmail
- atualizador avançado para a Base de Dados com suporte para scripts e assinaturas digitais
- biblioteca C para análise viral
- análise “ao acesso” (Linux® and FreeBSD®)

16 Texto obtido de http://www.clamav.net/about/

- múltiplas atualizações diárias da Base de Dados de vírus (consultar página oficial para verificar número total de assinaturas)
- suporte incorporado para vários formatos de arquivos, nomeadamente Zip, RAR, Tar, Gzip, Bzip2, OLE2, Cabinet, CHM, BinHex, SIS e outros
- suporte incorporado da maioria dos formatos de correio
- suporte incorporado de executáveis ELF e ficheiros executávei portáveis comprimidos com UPX, FSG, Petite, NsPack, wwpack32, MEW, Upack e ofuscados com SUE, Y0da Cryptor e outros
- suporte incorporado para formatos de documentos comuns nomeadamente arquivos MS Office e MacOffice, HTML, RTF e PDF.

A distribuição Mandriva adota nativamente a instalação do CLAMAV, mas caso o mesmo não esteja instalado basta executar o comando:

```
urpmi clamav
```

Agora que já instalamos o *clamav* em nosso sistema, é hora de configurarmos o software através de seu arquivo de configuração */etc/clamav.conf*:[17]

```
vi clamav.conf
```

```
# Exemplo
/var/log/clamav/clamav.log
LogFileMaxSize 0
LogTime
LogSyslog
PidFile /var/run/clamav/clamd.pid
TemporaryDirectory /tmp
DatabaseDirectory /var/lib/clamav
LocalSocket /var/run/clamav/clamd.sock
User clamav
ScanMail
ArchiveMaxFileSize 10M # este parâmetro diz o tamanho máximo da mensagem em anexo que será scaneada.
ClamukoScanOnLine
# Acrescente ainda:
ClamukoIncludePath /home
ClamukoIncludePath /var/spool # essa linha discrimina os diretórios a serem scaneados
```

Salve e pronto. Agora vamos criar os diretórios se necessário:

```
mkdir /var/log/clamav
chown clamav:root /var/log/clamav
mkdir /var/run/clamav
chown clamav:root /var/run/clamav
```

Depois de configurar o *clamav.conf*, agora iremos configurar o *freshclam.conf*, que é o responsável pelas atualizações do banco de dados de vírus do clamav.

17 Texto obtido de http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=945&pagina=1

No site da clamav (www.clamav.net) existe uma seção de *mirrors* (http://www.clamav.net/mirrors.html) que tem os endereços dos *mirrors* disponíveis para atualização.

```
vi /etc/freshclam.conf
```

```
UpdateLogFile /var/log/clamav/freshclam.log
DatabaseMirror clamav.sonic.net
DatabaseMirror clamav.xmundo.net
DatabaseMirror database.clamav.net
# você pode escolher o mirror que quiser no site da clamav
# Caso esteja comentado descomente ou altere o caminho
NotifyClamd /etc/clamav.conf
```

Feito as modificações, salve o arquivo e depois execute o *freshclam* no seu terminal:

```
freshclam
```

Ele irá baixar as últimas atualizações de vírus da internet.

37.3 Integrando o CLAMAV ao Postfix

Para que o Postfix escaneie todos os emails que entram ou saem do servidor devemos configurar o AMaViS – A *Mail Virus Scanner*, que irá usar o CLAMAV para escanear todas as mensagens de email. O Amavis é uma interface entre o MTA (*Message Tranfer Agent*) e um ou mais sistemas de antivírus. Primeiramente devemos instalar o AMaViS com o comando:

```
urpmi amavis
```

O serviço que já vem previamente configurado, caso desejemos alterar algum parâmetro devemos acessar o arquivo /etc/amavisd/amavisd.conf. Em seguida devemos inicializar o mesmo com o comando:

```
service amavisd start
```

Supondo que você já tem o Postfix instalado e funcionando não são necessários mais ajustes, pois os pacotes configuram todos os serviços e arquivos necessários.

37.4 Integrando o CLAMAV ao Samba

Para configurarmos o samba para escanear todos os arquivos que “passam” por ele devemos primeiramente instalar o pacote samba-vscan-clamav com o comando:

```
urpmi samba-vscan-clamav
```

Em seguida devemos criar um diretório .recycle em cada um dos diretórios compartilhados pelo Samba com permissão 777. E por último devemos descomentar (ou acrescentar) as seguintes linhas no arquivo /etc/samba/smb.conf:

```
vfs objects = vscan-clamav recycle
vscan-clamav: config-file = /etc/samba/vscan-clamav.conf
```

37.5 Escanear diretórios em busca de vírus

Como segurança adicional devemos programar em nossa cron uma atividade diária de escaneamento de arquivos em busca de vírus. Abaixo temos um pequeno scritp que faz esta tarefa e, caso encontre vírus, envia um ou mais emails para as pessoas responsáveis avisando sobre o ocorrido.

```
#!/bin/bash
#--------------------------------------------------------------------------
# Script que procura por vírus, move para /var/infectados, caso encontre, e
# envia e-mail.
# Odilson Tadeu Valle 03/08/2006
#--------------------------------------------------------------------------
#
# Escaneia as pastas /home e /dados. Caso encontre vírus move os
arquivos
# infectados para /var/infectados e cria uma lista dos mesmos em
# /root/lista_virus
clamscan -r -i /home/ /dados/ --move=/var/infectados/ --recursive >
/tmp/lista_virus 2>&1
# Verifica se o arquivo /tmp/lista_virus está vazio, se não estiver manda um
# e-mail de alerta
cat /tmp/lista_virus |grep FOUND > /tmp/virusencontrado.txt 2>&1
tam_arq=`ls -l /tmp/virusencontrado.txt |awk '{print $5}'`
if [ $tam_arq -gt 5 ]
then
        date >> /tmp/alerta.txt
        echo " " >> /tmp/alerta.txt
        echo "Atenção Gerente!" >> /tmp/alerta.txt
        echo " " >> /tmp/alerta.txt
        echo "Foram encontrados vírus no servidor." >> /tmp/alerta.txt
         echo "Os arquivos infectados foram movidos para o diretório
         /var/infectados." >> /tmp/alerta.txt
        echo "Segue a listagem dos mesmos." >> /tmp/alerta.txt
        echo " " >> /tmp/alerta.txt
        cat /tmp/lista_virus >> /tmp/alerta.txt
         /bin/mail -s "Vírus encontrado. Listagem em anexo" -a /tmp/alerta.txt
         admin1@dominio.edu.br
         /bin/mail -s "Vírus encontrado. Listagem em anexo" -a /tmp/alerta.txt
         admin2@dominio.edu.br
fi
exit
```

38 Redes Virtuais Privadas - VPN

O uso de Redes Privadas Virtuais representa uma alternativa interessante na racionalização dos custos de redes corporativas oferecendo "confidencialidade" e integridade no transporte de informações através de redes públicas.

38.1 Introdução[18]

A idéia de utilizar uma rede pública como a Internet em vez de linhas privativas para implementar redes corporativas é denominada de *Virtual Private Network* (VPN) ou Rede Privada Virtual. As VPNs são túneis de criptografia entre pontos autorizados, criados através da Internet ou outras redes públicas e/ou privadas para transferência de informações, de modo seguro, entre redes corporativas ou usuários remotos.
A segurança é a primeira e mais importante função da VPN. Uma vez que dados privados serão transmitidos pela Internet, que é um meio de transmissão inseguro, eles devem ser protegidos de forma a não permitir que sejam modificados ou interceptados.
Outro serviço oferecido pelas VPNs é a conexão entre corporações (Extranets) através da Internet, além de possibilitar conexões *dial-up* criptografadas que podem ser muito úteis para usuários móveis ou remotos, bem como filiais distantes de uma empresa.
Uma das grandes vantagens decorrentes do uso das VPNs é a redução de custos com comunicações corporativas, pois elimina a necessidade de *links* dedicados de longa distância que podem ser substituídos pela Internet. As LANs podem, através de *links* dedicados ou discados, conectar-se a algum provedor de acesso local e interligar-se a outras LANs, possibilitando o fluxo de dados através da Internet. Esta solução pode ser bastante interessante sob o ponto de vista econômico, sobretudo nos casos em que enlaces internacionais ou nacionais de longa distância estão envolvidos. Outro fator que simplifica a operacionalização da WAN é que a conexão LAN-Internet-LAN fica parcialmente a cargo dos provedores de acesso.

38.2 Aplicações para redes privadas virtuais

Abaixo, são apresentadas as três aplicações ditas mais importantes para as VPNs.

38.2.1 Acesso remoto via Internet

O acesso remoto a redes corporativas através da Internet pode ser viabilizado com a VPN através da ligação local a algum provedor de acesso (*Internet Service Provider* - ISP). A estação remota disca para o provedor de acesso, conectando-se à Internet e o software de VPN cria uma rede virtual privada entre o usuário remoto e o servidor de VPN corporativo através da Internet, Ilustração 29.

18 Texto obtido de http://www.rnp.br/newsgen/9811/vpn.html

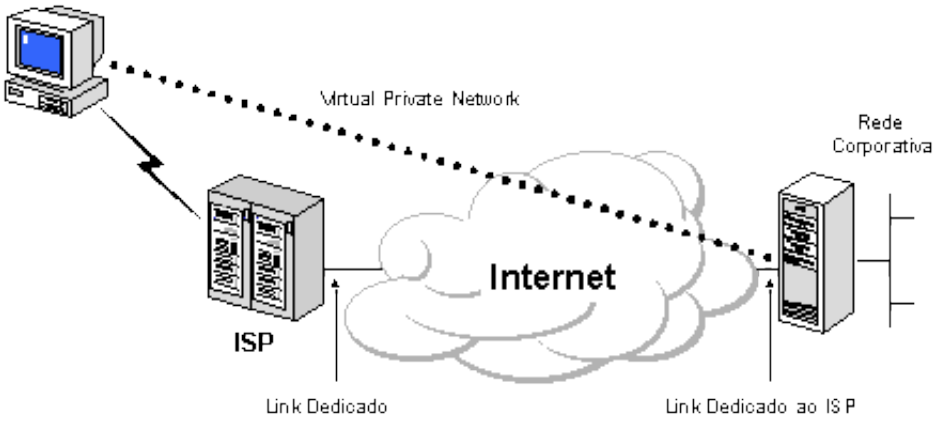


Ilustração 29: Acesso remoto via Internet

38.2.2 Conexão de LANs via Internet

Uma solução que substitui as conexões entre LANs através de circuitos dedicados de longa distância é a utilização de circuitos dedicados locais interligando-as à Internet, Ilustração 30. O software de VPN assegura esta interconexão formando a WAN corporativa.
A depender das aplicações também, pode-se optar pela utilização de circuitos discados em uma das pontas, devendo a LAN corporativa estar, preferencialmente, conectada à Internet via circuito dedicado local ficando disponível 24 horas por dia para eventuais tráfegos provenientes da VPN.

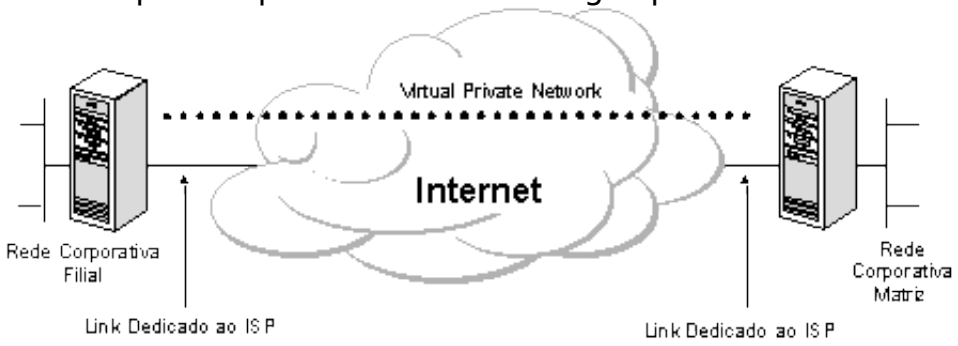


Ilustração 30: Conexão de LANs via Internet

38.2.3 Conexão de computadores numa intranet

Em algumas organizações, existem dados confidenciais cujo acesso é restrito a um pequeno grupo de usuários. Nestas situações, redes locais departamentais são implementadas fisicamente separadas da LAN corporativa. Esta solução, apesar de garantir a "confidencialidade" das informações, cria dificuldades de acesso a dados da rede corporativa por parte dos departamentos isolados.
As VPNs possibilitam a conexão física entre redes locais, restringindo acessos indesejados através da inserção de um servidor VPN entre elas, Ilustração 31. Observe que o servidor VPN não irá atuar como um roteador entre a rede departamental e o resto da rede corporativa uma vez que o roteador

possibilitaria a conexão entre as duas redes permitindo o acesso de qualquer usuário à rede departamental sensitiva. Com o uso da VPN o administrador da rede pode definir quais usuários estarão credenciados a atravessar o servidor VPN e acessar os recursos da rede departamental restrita. Adicionalmente, toda comunicação ao longo da VPN pode ser criptografada assegurando a "confidencialidade" das informações. Os demais usuários não credenciados sequer enxergarão a rede departamental.

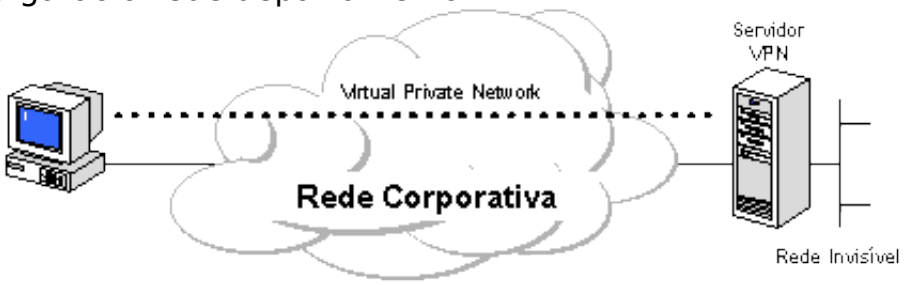


Ilustração 31: Conexão de computadores numa Intranet

38.3 Requisitos básicos

No desenvolvimento de soluções de rede, é bastante desejável que sejam implementadas facilidades de controle de acesso a informações e a recursos corporativos. A VPN deve dispor de recursos para permitir o acesso de clientes remotos autorizados aos recursos da LAN corporativa, viabilizar a interconexão de LANs de forma a possibilitar o acesso de filiais, compartilhando recursos e informações e, finalmente, assegurar privacidade e integridade de dados ao atravessar a Internet bem como a própria rede corporativa. A seguir são enumeradas características mínimas desejáveis numa VPN:

- ✔ **Autenticação de Usuários**

Verificação da identidade do usuário, restringindo o acesso às pessoas autorizadas. Deve dispor de mecanismos de auditoria, provendo informações referentes aos acessos efetuados - quem acessou, o quê e quando foi acessado.

- ✔ **Gerenciamento de Endereço**

O endereço do cliente na sua rede privada não deve ser divulgado, devendo-se adotar endereços fictícios para o tráfego externo.

- ✔ **Criptografia de Dados**

Os dados devem trafegar na rede pública ou privada num formato cifrado e, caso sejam interceptados por usuários não autorizados, não deverão ser decodificados, garantindo a privacidade da informação. O reconhecimento do conteúdo das mensagens deve ser exclusivo dos usuários autorizados.

- ✔ **Gerenciamento de Chaves**

O uso de chaves que garantem a segurança das mensagens criptografadas deve funcionar como um segredo compartilhado exclusivamente entre as partes envolvidas. O gerenciamento de chaves deve garantir a troca periódica das mesmas, visando manter a comunicação de forma segura.

- ✔ **Suporte a Múltiplos Protocolos**

Com a diversidade de protocolos existentes, torna-se bastante desejável que uma VPN suporte protocolos padrão de fato usadas nas redes públicas, tais como IP (*Internet Protocol*), IPX (*Internetwork Packet Exchange*), etc.

38.4 Tunelamento

As redes virtuais privadas baseiam-se na tecnologia de tunelamento cuja existência é anterior às VPNs. Ele pode ser definido como processo de encapsular um protocolo dentro de outro. O uso do tunelamento nas VPNs incorpora um novo componente a esta técnica: antes de encapsular o pacote que será transportado, este é criptografado de forma a ficar ilegível caso seja interceptado durante o seu transporte. O pacote criptografado e encapsulado viaja através da Internet até alcançar seu destino onde é desencapsulado e decriptografado, retornando ao seu formato original, Ilustração 32. Uma característica importante é que pacotes de um determinado protocolo podem ser encapsulados em pacotes de protocolos diferentes. Por exemplo, pacotes de protocolo IPX podem ser encapsulados e transportados dentro de pacotes TCP/IP.
O protocolo de tunelamento encapsula o pacote com um cabeçalho adicional que contém informações de roteamento que permitem a travessia dos pacotes ao longo da rede intermediária. Os pacotes encapsulados são roteados entre as extremidades do túnel na rede intermediária. Túnel é a denominação do caminho lógico percorrido pelo pacote ao longo da rede intermediária Após alcançar o seu destino na rede intermediária, o pacote é desencapsulado e encaminhado ao seu destino final. A rede intermediária por onde o pacote trafegará pode ser qualquer rede pública ou privada.
Note que o processo de tunelamento envolve encapsulamento, transmissão ao longo da rede intermediária e desencapsulamento do pacote.

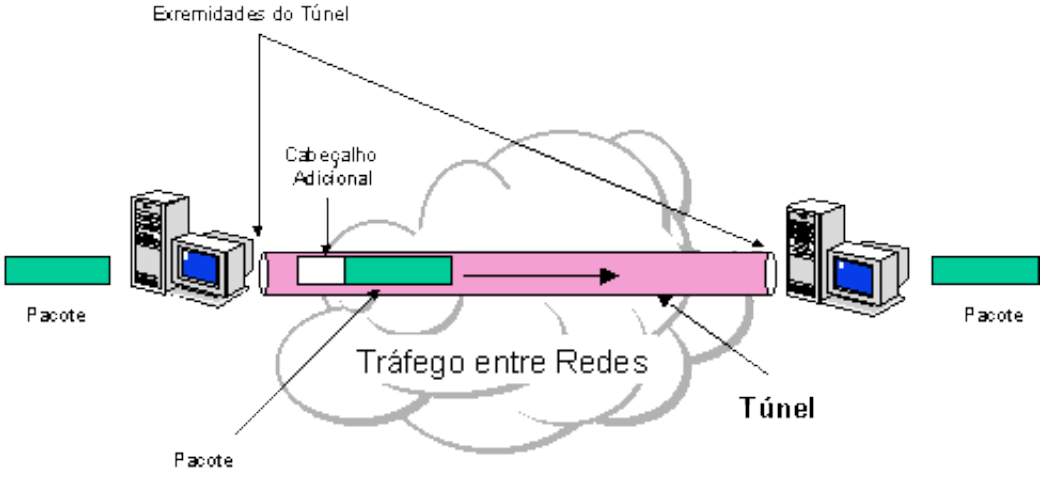


Ilustração 32: Tunelamento de pacotes

38.5 Protocolos de tunelamento

Para se estabelecer um túnel é necessário que as suas extremidades utilizem o mesmo protocolo de tunelamento.
O tunelamento pode ocorrer na camada 2 ou 3 (respectivamente enlace e rede)

do modelo de referência OSI (*Open Systems Interconnection*).

38.5.1 Tunelamento em Nível 2 - Enlace - (PPP sobre IP)

O objetivo é transportar protocolos de nível 3, tais como o IP e IPX na Internet. Os protocolos utilizam quadros como unidade de troca, encapsulando os pacotes da camada 3 (como IP/IPX) em quadros PPP (*Point-to-Point Protocol*). Como exemplos podemos citar:

- PPTP (*Point-to-Point Tunneling Protocol*) da Microsoft permite que o tráfego IP, IPX e NetBEUI sejam criptografados e encapsulados para serem enviados através de redes IP privadas ou públicas como a Internet.
- L2TP (*Layer 2 Tunneling Protocol*) da IETF (*Internet Engineering Task Force*) permite que o tráfego IP, IPX e NetBEUI sejam criptografados e enviados através de canais de comunicação de datagrama ponto a ponto tais como IP, X25, Frame Relay ou ATM.
- L2F (*Layer 2 Forwarding*) da Cisco é utilizada para VPNs discadas.

38.5.2 Tunelamento em Nível 3 - Rede - (IP sobre IP)

Encapsulam pacotes IP com um cabeçalho adicional deste mesmo protocolo antes de enviá-los através da rede.

- O *IP Security Tunnel Mode* (IPSec) da IETF permite que pacotes IP sejam criptografados e encapsulados com cabeçalho adicional deste mesmo protocolo para serem transportados numa rede IP pública ou privada. O IPSec é um protocolo desenvolvido para IPv6, devendo, no futuro, se constituir como padrão para todas as formas de VPN caso o IPv6 venha de fato substituir o IPv4. O IPSec sofreu adaptações possibilitando, também, a sua utilização com o IPv4.

38.6 O funcionamento dos túneis

Nas tecnologias orientadas à camada 2 (enlace), um túnel é similar a uma sessão, onde as duas extremidades do túnel negociam a configuração dos parâmetros para estabelecimento do túnel, tais como endereçamento, criptografia e parâmetros de compressão. Na maior parte das vezes, são utilizados protocolos que implementam o serviço de datagrama. A gerência do túnel é realizada através protocolos de manutenção. Nestes casos, é necessário que o túnel seja criado, mantido e encerrado. Nas tecnologias de camada 3, não existe a fase de manutenção do túnel.
Uma vez que o túnel é estabelecido os dados podem ser enviados. O cliente ou servidor do túnel utiliza um protocolo de tunelamento de transferência de dados que acopla um cabeçalho preparando o pacote para o transporte. Só então o cliente envia o pacote encapsulado na rede que o roteará até o servidor do túnel. Este recebe o pacote, desencapsula removendo o cabeçalho adicional e encaminha o pacote original à rede destino. O funcionamento entre o servidor e o cliente do túnel é semelhante.

38.7 Protocolos × Requisitos de tunelamento

Os protocolos de nível 2, tais como PPTP e L2TP, foram baseados no PPP, e, como conseqüência, herdaram muito de suas características e funcionalidades. Estas características e suas contrapartes de nível 3 são analisadas juntamente com alguns dos requisitos básicos das VPNs.

38.7.1 Autenticação de usuário

Os protocolos de tunelamento da camada 2 herdaram os esquemas de autenticação do PPP e os métodos EAP (*Extensible Authentication Protocol*). Muitos esquemas de tunelamento da camada 3 assumem que as extremidades do túnel são conhecidas e autenticadas antes mesmo que ele seja estabelecido. Uma exceção é o IPSec que provê a autenticação mútua entre as extremidades do túnel. Na maioria das implementações deste protocolo, a verificação se dá a nível de máquina e não de usuário. Como resultado, qualquer usuário com acesso às máquinas que funcionam como extremidades do túnel podem utilizá-lo. Esta falha de segurança pode ser suprida quando o IPSec é usado junto com um protocolo de camada de enlace como o L2TP.

38.7.2 Suporte a token card

Com a utilização do EAP, os protocolos de tunelamento de camada de enlace podem suportar uma variedade de métodos de autenticação, tais como senhas e cartões inteligentes (*smart cards*). Os protocolos de camada 3 também podem usar métodos similares, como, por exemplo, o IPSec que define a autenticação de chave pública durante a negociação de parâmetros feita pelo ISAKMP (*Internet Security Association and Key Management Protocol*).

38.7.3 Endereçamento dinâmico

O tunelamento na camada 2 suporta alocação dinâmica de endereços baseada nos mecanismos de negociação do NCP (*Network Control Protocol*). Normalmente, esquemas de tunelamento na camada 3 assumem que os endereços foram atribuídos antes da inicialização do túnel.

38.7.4 Compressão de dados

Os protocolos de tunelamento da camada 2 suportam esquemas de compressão baseados no PPP. O IETF está analisando mecanismos semelhantes, tais como a compressão de IP, para o tunelamento na camada 3.

38.7.5 Criptografia de dados

Protocolos de tunelamento na camada de enlace suportam mecanismos de criptografia baseados no PPP. Os protocolos de nível 3 também podem usar métodos similares. No caso do IPSec são definidos vários métodos de criptografia de dados que são executados durante o ISAKMP. Algumas implementações do protocolo L2TP utilizam a criptografia provida pelo IPSec para proteger cadeias de dados durante a sua transferência entre as extremidades do túnel.

38.7.6 Gerenciamento de chaves

O MPPE (*Microsoft Point-to-Point Encryption*), protocolo de nível de enlace, utiliza uma chave gerada durante a autenticação do usuário, atualizando-a periodicamente. O IPSec negocia uma chave comum através do ISAKMP e, também, periodicamente, faz sua atualização.

38.7.7 Suporte a múltiplos protocolos

O tunelamento na camada de enlace suporta múltiplos protocolos o que facilita o tunelamento de clientes para acesso a redes corporativas utilizando IP, IPX, NetBEUI e outros. Em contraste, os protocolos de tunelamento da camada de rede, tais como o IPSec, suportam apenas redes destino que utilizam o protocolo IP.

38.8 Tipos de túneis

Os túneis podem ser criados de 2 diferentes formas - voluntárias e compulsórias:

- Túnel Voluntário - um cliente emite uma solicitação VPN para configurar e criar um túnel voluntário. Neste caso, o computador do usuário funciona como uma das extremidades do túnel e, também, como cliente do túnel.
- Túnel Compulsório - um servidor de acesso discado VPN configura e cria um túnel compulsório. Neste caso, o computador do cliente não funciona como extremidade do túnel. Outro dispositivo, o servidor de acesso remoto, localizado entre o computador do usuário e o servidor do túnel, funciona como uma das extremidades e atua como o cliente do túnel.

38.8.1 Tunelamento voluntário

Ocorre quando uma estação ou servidor de roteamento utiliza um software de tunelamento cliente para criar uma conexão virtual para o servidor do túnel desejado. O tunelamento voluntário pode requerer conexões IP através de LAN ou acesso discado.
No caso de acesso discado, o mais comum é o cliente estabelecer a conexão discada antes da criação do túnel.
Nas LANs, o cliente já se encontra conectado à rede que pode prover o roteamento de dados encapsulados para o servidor de túnel selecionado. Este é o caso de clientes numa LAN corporativa que inicializa túneis para alcançar uma subrede privada na mesma rede.

38.8.2 Tunelamento compulsório

O computador ou dispositivo de rede que provê o túnel para o computador cliente é conhecido de diversas formas: FEP (*Front End Processor*) no PPTP, LAC (*L2TP Access Concentrator*) no L2TP ou *IP Security Gateway* no caso do IPSec.
Doravante, adotaremos o termo FEP para denominar esta funcionalidade - ser capaz de estabelecer o túnel quando o cliente remoto se conecta.

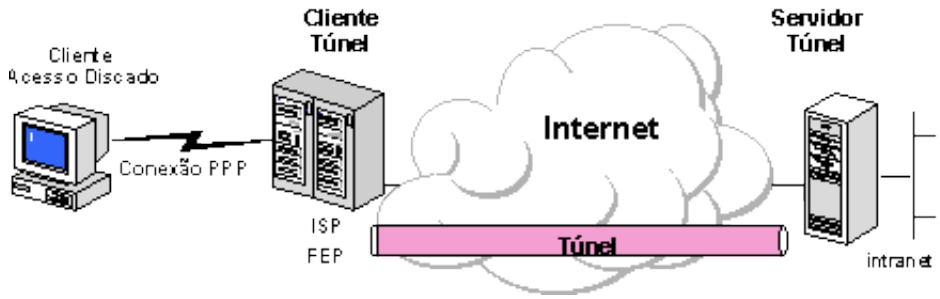


Ilustração 33: Tunelamento compulsório

No caso da Internet, o cliente faz uma conexão discada para um túnel habilitado pelo servidor de acesso no provedor (ISP), Ilustração 33. Por exemplo, uma companhia pode ter um contrato com uma ou mais provedores para disponibilizar um conjunto de FEPs em âmbito nacional. Estas FEPs podem estabelecer túneis sobre a Internet para um servidor de túnel conectado à rede corporativa privada, possibilitando a usuários remotos o acesso à rede corporativa através de uma simples ligação local.
Esta configuração é conhecida como tunelamento compulsório porque o cliente é compelido a usar um túnel criado pelo FEP. Uma vez que a conexão é estabelecida, todo o tráfego "de/para" o cliente é automaticamente enviado através do túnel. No tunelamento compulsório, o cliente faz uma conexão PPP. Um FEP pode ser configurado para direcionar todas as conexões discadas para um mesmo servidor de túnel ou, alternativamente, fazer o tunelamento individual baseado na identificação do usuário ou no destino da conexão.
Diferente dos túneis individualizados criados no tunelamento voluntário, um túnel entre o FEP e o servidor de túnel pode ser compartilhado por múltiplos clientes discados. Quando um cliente disca para o servidor de acesso (FEP) e já existe um túnel para o destino desejado, não se faz necessária a criação de um novo túnel redundante. O próprio túnel existente pode transportar, também, os dados deste novo cliente. No tunelamento compulsório com múltiplos clientes, o túnel só é finalizado no momento em que o último usuário do túnel se desconecta.

38.9 IPSEC – *Internet Protocol Security*

O IPSec é um protocolo padrão de camada 3 projetado pelo IETF que oferece transferência segura de informações fim a fim através de rede IP pública ou privada. Essencialmente, ele pega pacotes IP privados, realiza funções de segurança de dados como criptografia, autenticação e integridade, e então encapsula esses pacotes protegidos em outros pacotes IP para serem transmitidos. As funções de gerenciamento de chaves também fazem parte das funções do IPSec.
Tal como os protocolos de nível 2, o IPSec trabalha como uma solução para interligação de redes e conexões via linha discada. Ele foi projetado para suportar múltiplos protocolos de criptografia possibilitando que cada usuário escolha o nível de segurança desejado.
Os requisitos de segurança podem ser divididos em 2 grupos, os quais são independentes entre si, podendo ser utilizado de forma conjunta ou separada, de acordo com a necessidade de cada usuário:

- Autenticação e Integridade;
- Confidencialidade.

Para implementar estas características, o IPSec é composto de 3 mecanismos adicionais:

- AH - *Autentication Header*;
- ESP - *Encapsulation Security Payload*;
- ISAKMP - *Internet Security Association and Key Management Protocol*.

38.9.1 Negociação do nível de segurança

O ISAKMP combina conceitos de autenticação, gerenciamento de chaves e outros requisitos de segurança necessários às transações e comunicações governamentais, comerciais e privadas na Internet. Com o ISAKMP, as duas máquinas negociam os métodos de autenticação e segurança dos dados, executam a autenticação mútua e geram a chave para criptografar os dados. Trata-se de um protocolo que rege a troca de chaves criptografadas utilizadas para decifrar os dados. Ele define procedimentos e formatos de pacotes para estabelecer, negociar, modificar e deletar as SAs (*Security Associations*). As SAs contêm todas as informações necessárias para execução de serviços variados de segurança na rede, tais como serviços da camada IP (autenticação de cabeçalho e encapsulamento), serviços das camadas de transporte, e aplicação ou auto-proteção durante a negociação do tráfego. Também define pacotes para geração de chaves e autenticação de dados. Esses formatos provêm consistência para a transferência de chaves e autenticação de dados que independem da técnica usada na geração da chave, do algoritmo de criptografia e do mecanismo de autenticação.
O ISAKMP pretende dar suporte para protocolos de segurança em todas as camadas da pilha da rede. Com a centralização do gerenciamento dos SAs, o ISAKMP minimiza as redundâncias funcionais dentro de cada protocolo de segurança e também pode reduzir o tempo gasto durante as conexões através da negociação da pilha completa de serviços de uma só vez.

38.9.2 Autenticação e integridade

A autenticação garante que os dados recebidos correspondem àqueles originalmente enviados, assim como garante a identidade do emissor. Integridade significa que os dados transmitidos chegam ao seu destino íntegros, eliminando a possibilidade de terem sido modificados no caminho sem que isto pudesse ser detectado.
O AH é um mecanismo que provê integridade e autenticação dos datagramas IP. A segurança é garantida através da inclusão de informação para autenticação no pacote a qual é obtida através de algoritmo aplicado sobre o conteúdo dos campos do datagrama IP, excluindo-se aqueles que sofrem mudanças durante o transporte. Estes campos abrangem não só o cabeçalho IP como todos os outros cabeçalhos e dados do usuário. No IPv6, o campo *hop-count* e o *time-to-live* (TTL) do IPv4 não são utilizados, pois são modificados ao longo da transferência.
Para alguns usuários o uso da autenticação pode ser suficiente não sendo necessária a "confidencialidade".

No IPV6, o AH normalmente é posicionado após os cabeçalhos de fragmentação e *End-to-End,* e antes do ESP e dos cabeçalhos da camada de transporte (TCP ou UDP, por exemplo).

38.9.3 Confidencialidade

Propriedade da comunicação que permite que apenas usuários autorizados entendam o conteúdo transportado. Desta forma, os usuários não autorizados, mesmo tendo capturado o pacote, não poderão ter acesso às informações nele contidas. O mecanismo mais usado para prover esta propriedade é chamado de criptografia.
O serviço que garante a "confidencialidade" no IPSec é o ESP - *Encapsulating Security Payload*. O ESP também provê a autenticação da origem dos dados, integridade da conexão e serviço *anti-reply*. A "confidencialidade" independe dos demais serviços e pode ser implementada de 2 modos - transporte e túnel. No primeiro modo, o pacote da camada de transporte é encapsulado dentro do ESP, e, no túnel, o datagrama IP é encapsulado inteiro dentro da cabeçalho do ESP.

38.10 Algumas conclusões

As VPNs podem se constituir numa alternativa segura para transmissão de dados através de redes públicas ou privadas, uma vez que já oferecem recursos de autenticação e criptografia com níveis variados de segurança, possibilitando eliminar os *links* dedicados de longa distância, de alto custo, na conexão de WANs.
Entretanto, em aplicações onde o tempo de transmissão é crítico, o uso de VPNs através de redes externas ainda deve ser analisado com muito cuidado, pois podem ocorrer problemas de desempenho e atrasos na transmissão sobre os quais a organização não tem nenhum tipo de gerência ou controle, comprometendo a qualidade desejada nos serviços corporativos.
A decisão de implementar ou não redes privadas virtuais requer uma análise criteriosa dos requisitos, principalmente aqueles relacionados a segurança, custos, qualidade de serviço e facilidade de uso que variam de acordo com o negócio de cada organização.

38.11 Instalação e configuração

Como plataforma de testes vamos montar a estrutura da Ilustração 34. Deste modo qualquer cliente “enxergará” outro cliente, mesmo sendo da rede remota, mesmo sem saber que está usando VPN.

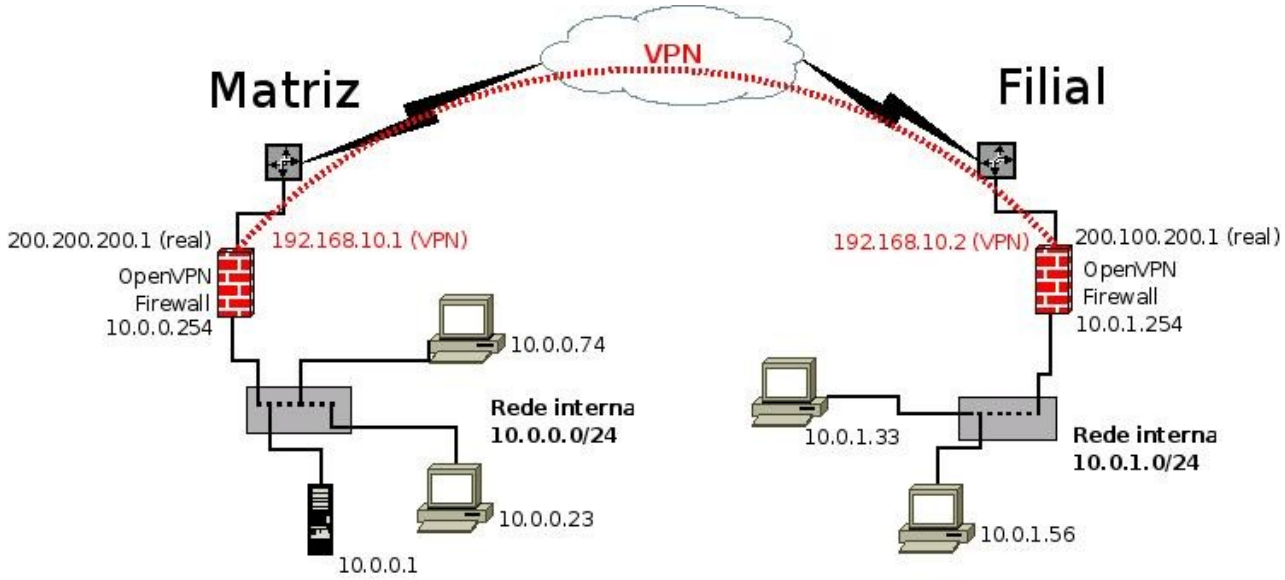


Ilustração 34: Exemplo de implementação VPN

Primeiramente devemos instalar o pacote OpenVPN, em ambos os *firewalls*, com o comando:

```
urpmi openvpn
```

Carregar o módulo tun, se necessário, com o comando:

```
modprobe tun
```

Instalar a biblioteca para compressão de dados, se necessário, com o comando:

```
urpmi liblzo
```

38.11.1 Configuração na Matriz

O OpenVPN pode operar com 3 tipos de criptografia. Nenhuma criptografia (apenas o túnel), criptografia com chaves estáticas e no modo TLS, em que as chaves são trocadas periodicamente. No nosso exemplo, usaremos criptografia com chaves estáticas, que é o padrão.
O diretório /etc/openvpn é onde ficarão todos os arquivos de configuração.

Vamos cria a chave estática com o comando:

```
openvpn --genkey --secret /etc/openvpn/chave
```

Podemos visualizar o conteúdo da chave com o comando:

```
cat /etc/openvpn/chave
```

Criamos/editamos o arquivo /etc/openvpn/matriz.conf, com o seguinte conteúdo:

```
# Usar como interface o driver TUN
dev tun
persist-tun
# 192.168.10.1 ip que será assumido na matriz
# 192.168.10.2 ip remoto, ou seja, esse será o ip da filial. Observe que cria-
```

```
#se uma nova rede para a VPN, independente das já existentes.
ifconfig 192.168.10.1 192.168.10.2
# Entra no diretório onde se encontram os arquivos de configuração
cd /etc/openvpn
# Indica que esse túnel possui uma chave de criptografia
secret chave
persist-key
# OpenVPN usa a porta 5000/UDP por padrão.
# Cada túnel do OpenVPN deve usar uma porta diferente.

port 5000
# Usuário que rodará o daemon do OpenVPN
user openvpn
# Grupo que rodará o daemon do OpenVPN
group openvpn
# Usa a biblioteca lzo
comp-lzo
# Envia um ping via UDP para a parte remota a cada 15 segundos para
# manter a conexão de pé em firewall statefull.
# Muito recomendado, mesmo se você não usa um firewall baseado em
# statefull.
ping-timer-rem
# Nível de log
verb 3
```

Em seguida, vamos iniciar a conexão no servidor, faltando apenas configurar a filial. Execute o seguinte comando:

```
service openvpn restart
```

Podemos conferir se está tudo em ordem verificando se foi criada uma interface tun0, com o comando:

```
ifconfig
```

Se aparecer algo como abaixo, ou parecido, o túnel na Matriz já está pronto e a espera da conexão da filial.

```
tun0      Link encap:Não Especificado  Endereço de HW 00-00-00-00-00-00-
00-00-00-00-00-00-00-00-00-00
          inet end.: 192.168.10.1  P-a-P:192.168.10.2  Masc:255.255.255.255
          UP POINTOPOINT RUNNING NOARP 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)
```

38.11.2 Configuração na filial

Copie a chave gerada pela matriz com o comando:

```
scp user@ip.do.servidor.da.matriz:/etc/openvpn/chave /etc/openvpn/
```

Criamos/editamos o arquivo /etc/openvpn/filial.conf, com o seguinte conteúdo:

```
# Usar como interface o driver TUN
dev tun
persist-tun
# 192.168.10.1 ip que será assumido na matriz
# 192.168.10.2 ip remoto, ou seja, esse será o ip da filial. Observe que cria-
#se uma nova rede para a VPN, independente das já existentes.
ifconfig 192.168.10.2 192.168.10.1
# Indica onde está o ip da Matriz (essa é a única linha que acrescentamos
# no arquivo de configuração da filial), o resto é tudo igual.
remote 200.200.200.1
# Entra no diretório onde se encontram os arquivos de configuração
cd /etc/openvpn
# Indica que esse túnel possui uma chave de criptografia
secret chave
persist-key
# OpenVPN usa a porta 5000/UDP por padrão.
# Cada túnel do OpenVPN deve usar uma porta diferente.

port 5000
# Usuário que rodará o daemon do OpenVPN
user openvpn
# Grupo que rodará o daemon do OpenVPN
group openvpn
# Usa a biblioteca lzo
comp-lzo
# Envia um ping via UDP para a parte remota a cada 15 segundos para
# manter a conexão de pé em firewall statefull.
# Muito recomendado, mesmo se você não usa um firewall baseado em
# statefull.
ping-timer-rem
# Nível de log
verb 3
```

Em seguida, vamos iniciar a conexão no servidor, faltando apenas configurar a filial. Execute o seguinte comando:

```
service openvpn restart
```

Podemos conferir se está tudo em ordem verificando se foi criada uma interface tun0, com o comando:

```
ifconfig
```

Se aparecer algo como abaixo, ou parecido, o túnel na Matriz já está pronto e a espera da conexão da filial.

```
tun0      Link encap:Não Especificado  Endereço de HW 00-00-00-00-00-00-
00-00-00-00-00-00-00-00-00-00
          inet end.: 192.168.10.2  P-a-P:192.168.10.1  Masc:255.255.255.255
          UP POINTOPOINT RUNNING NOARP 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)

38.11.3 Configurações nos firewalls

Caso esteja usando o **iptables puro** insira as seguintes regras*, no caso de uso da política DROP*:

```
iptables -A INPUT -p udp –dport 5000 -j ACCEPT

iptables -A INPUT -i tun+ -j ACCEPT

iptables -A FORWARD -i tun+ -j ACCEPT

iptables -A INPUT -i tap+ -j ACCEPT

iptables -A FORWARD -i tap+ -j ACCEPT
```

Caso esteja utilizando o **Shorewall** edite os arquivos abaixo acrescentando as linhas indicadas.

- /etc/shorewall/rules

  ACCEPT:info net fw udp 1194 -
  ACCEPT vpn fw udp - 53,137
  ACCEPT vpn fw udp 53,67,137,138,139,177 -
  ACCEPT vpn fw tcp
  20,21,22,25,53,80,110,123,137,138,139,143,443,445,631,901,1512,3128,10000 -
  ACCEPT vpn masq udp - 53
  ACCEPT vpn masq udp 53 -
  ACCEPT vpn masq tcp 20,21,22,25,53,80,110,143,443,5900 -
- /etc/shorewall/interfaces
  vpn tun0 detect
- /etc/shorewall/zones
  vpn VPN Teste VPN

39 SNMP - Simple Network Management Protocol e MRTG - The Multi Router Traffic Grapher

39.1 Introudção[19]

O **protocolo SNMP** (do inglês *Simple Network Management Protocol* - Protocolo de Gerência Simples de Rede) é um protocolo de gerência típica de redes TCP/IP, da camada de aplicação que facilita o intercâmbio de informação entre os dispositivos de rede. O SNMP possibilita aos administradores de rede gerenciar o

19 Texto obtido de http://pt.wikipedia.org/wiki/Simple_Network_Management_Protocol

desempenho da rede, encontrar e resolver problemas de rede, e planejar o crescimento desta.
O software de gerência de redes segue o modelo cliente-servidor convencional: uma aplicação 'Servidora' na máquina do gerente e uma aplicação 'cliente' no dispositivo de rede a ser analisado ou monitorado. Para evitar confusão com outras aplicações de rede, os sistema de gerência de redes evitam os termos 'cliente' e 'servidor'. Em vez disso, usam "Gerente" para a aplicação servidora e "Agente" para a aplicação cliente que corre no dispositivo de rede.
O **MRTG** é um programa feito em perl muito útil para analisar o tráfego utilizado em sua rede/link. Ele gera gráficos que te mostram o uso da banda em termos de velocidade.

39.1.1 Gerente da rede

O programa gerente da rede é a entidade responsável pelo monitoramento e controle dos sistemas de hardware e software que compõem a rede, e o seu trabalho consiste em detectar e corrigir problemas que causem ineficiência (ou impossibilidade) na comunicação e eliminar as condições que poderão levar a que o problema volte a surgir.
A gerência de uma rede pode não ser simples, dada sua heterogeneidade em termos de hardware e software, e de componentes da rede, por vezes incompatíveis. As falhas intermitentes, se não forem detectadas, podem afetar o desempenho da rede. Um software de gerência de redes permite ao gestor monitorar e controlar os componentes da sua rede.

39.1.2 Componentes Básicos do SNMP

Uma rede gerenciada pelo protocolo SNMP é formada por três componentes chaves:

1. Dispositivos Gerenciados
2. Agentes
3. Sistemas de Gerenciamento de Redes (NMS - *Network-Management Systems*)

Um Dispositivo Gerenciado é um nó de rede que possui um agente SNMP instalado e se encontra em uma rede gerenciada. Estes dispositivos coletam e armazenam informações de gerenciamento e mantém estas informações disponíveis para sistemas NMS através do protocolo SNMP. Dispositivos gerenciados, também às vezes denominados de dispositivos de rede, podem ser roteadores, servidores de acesso, impressoras, computadores, servidores de rede, switches, dispositivos de armazenamento, dentre outros.
Um Agente é um módulo de software de gerenciamento de rede que fica armazenado em um Dispositivo Gerenciado. Um agente tem o conhecimento das informações de gerenciamento locais e traduzem estas informações para um formato compatível com o protocolo SNMP.
Um sistema NMS é responsável pelas aplicações que monitoram e controlam os Dispositivos Gerenciados. Normalmente é instalado em um (ou mais de um) servidor de rede dedicado a estas operações de gerenciamento, que recebe informações (pacotes SNMP) de todos os dispositivos gerenciados daquela rede.

39.1.3 Arquitetura

O *framework* SNMP consiste de: Agentes Mestres (*Master Agents*), Sub-agentes (*Subagents*) e Estações de Gerenciamento (*Management Stations*).

39.1.3.1 Master Agent

O *Master Agent* em uma rede gerenciada é, na verdade, um software sendo executado em um dispositivo com suporte a SNMP, por exemplo, um roteador, que interage com uma estação de gerenciamento. É o equivalente a um servidor, na comunicação cliente/servidor, ou a um daemon, sob o ponto de vista de sistemas operacionais. Os subagentes são os responsáveis por passarem informações específicas para o *Masters Agent*.

39.1.3.2 Subagent

Os subagentes ou *subagents* são pequenos programas em execução no dispositivo com suporte a SNMP, responsáveis pelo monitoramento de recursos específicos naquele dispositivo, como por exemplo, o status de um link ethernet em um roteador, ou a quantidade de espaço livre em um disco de um servidor. Algumas características dos softwares subagentes são:

- Coletar informações de objetos gerenciados
- Configurar parâmetros destes objetos gerenciados
- Responder a solicitações do software de gerência da rede
- Gerar alarmes ou *traps* em determinadas situações

39.1.3.3 Management Station

O Gerente da Rede ou Estação de Gerenciamento ou ainda *Management Station* é o componente final da arquitetura de uma solução SNMP. Funciona como um cliente em uma comunicação cliente/servidor. Realiza requisições de informações aos dispositivos gerenciados, que podem ser temporárias ou através de comandos a qualquer tempo. E ainda é o responsável por receber alarmes gerados pelos agentes e gerar saídas para estes alarmes, tais como, alterar (SET) o valor de um determinado parâmetro gerenciado no equipamento, enviar mensagem para o celular do administrador da rede, dentre outras.

39.1.4 O SNMP e o ASN.1

O SNMP é um protocolo padrão usado para gerência de redes, que define os formatos dos pedidos que o *Gerente* envia para o *Agente* e os formatos das respostas que o *agente* retorna, assim como o significado exato de cada pedido e resposta. Uma mensagem SNMP é codificada com um padrão designado de ASN.1 (do inglês: *Abstract Syntax Notation.1*).
O ASN.1 para permitir a transferência de grandes inteiros, sem desperdiçar espaço em cada transferência, usa uma combinação de tamanho e valor para cada objeto a ser transferido....

39.1.5 Comandos do SNMP

O SNMP não define um grande número de comandos, em lugar disso define duas operações básicas:

- *fetch*, para obter um valor de um dispositivo
- *store*, para colocar um valor num dispositivo

O comando que especifica uma operação de *fetch* ou *store* deve especificar o nome do objeto, que é único.
Podemos definir objetos. No caso de um contador de erros de CRC e uma vez que o SNMP não inclui comandos específicos para fazer *reset* do contador, uma forma simples é colocar zero no contador. Neste caso, o Gerente faz o *fetch* (leitura) do parâmetro desejado para determinar o estado do dispositivo. As operações que controlam o dispositivo são definidas como efeitos secundários de *store* (alterar/gravar valores) em objetos.
[[Especifica (na versão 1) quatro unidades de dados do protocolo (PDU):

1. GET, usado para retirar um pedaço de informação de gerenciamento.
2. GETNEXT, usado interativamente para retirar seqüências de informação de gerenciamento.
3. SET, usado para fazer uma mudança no subsistema gerido.
4. TRAP, usado para reportar uma notificação ou para outros eventos assíncronos sobre o subsistema gerido.

39.1.6 Nomes de objetos e MIB

Todos os objetos acessados pelo SNMP devem ter nomes únicos definidos e atribuídos. Além disso, o *Gerente* e o *Agente* devem acordar os nomes e significados das operações *fetch* e *store*. O conjunto de todos os objetos SNMP é coletivamente conhecido como MIB (do inglês: *Management Information Base*). O standard SNMP não define o MIB, mas apenas o formato e o tipo de codificação das mensagens. A especificação das variáveis MIB, assim como o significado das operações *fetch* e *store* em cada variável, são especificados por um padrão próprio.
A definição dos objetos do MIB é feita com o esquema de nomes do ASN.1, o qual atribui a cada objeto um prefixo longo que garante a unicidade do nome, a cada nome é atribuído um número inteiro. Também, o SNMP não especifica um conjunto de variáveis, e que a definição de objetos é independente do protocolo de comunicação, permite criar novos conjuntos de variáveis MIB, definidos como standards, para novos dispositivos ou novos protocolos. Por isso, foram criados muitos conjuntos de variáveis MIB que correspondem a protocolos como UDP, IP, ARP, assim como variáveis MIB para hardware de rede como Ethernet ou FDDI, ou para dispositivos tais como bridges, switches ou impressoras.

39.1.7 SNMPv2 e SNMPv3

A versão 2 do SNMP é uma evolução do protocolo inicial. O SNMPv2 oferece uma boa quantidade de melhoramentos em relação ao SNMPv1, incluindo operações adicionais do protocolo, melhoria na performance, segurança, confidencialidade e comunicações Gerente-para-Gerente. A padronização de uma outra versão do SNMP - o SNMPv3 ainda está em desenvolvimento, definido nos RFC 3411 -RFC

3418.
Na prática, as implementações do SNMP oferecem suporte para as múltiplas versões (RFC 3584), tipicamente SNMPv1, SNMPv2c e SNMPv3.

39.2 Instalação e Configuração

Serão necessários os pacotes *SNMP*, *MRTG* e o *Apache* rodando. Instale e configure o Apache o SNMP e o MRTG nesta ordem. Tem que ser feito desta forma, pois é preciso que o SNMP esteja devidamente configurado para que o MRTG possa atuar sem problemas.

39.2.1 Instalando o SNMP

Primeiramente devemos instalar o SNMP

```
urpmi net-snmp
```

Depois de instalados, iremos configurar o *SNMP*. Edite o arquivo snmpd.conf:

```
vi /etc/snmp/snmpd.conf
```

Existem diversos tipos de configuração para monitoramento via MRTG, neste texto usaremos a configuração padrão, ou seja não é necessário mudar absolutamente nada.
Configurações que podem ser interessantes de serem feitas são obejtivando monitorar alguma partição e/ou serviço específico. Se este for o desejo edite o arquivo do seguinte modo, nas seções indicadas:

```
# disk checks

disk / 10000

#logo após a linha acima, já existente, acrescente a(s) partição(ões)que deseja
monitorar.

disk /home 1000   #Onde 1000 é o espaço em bytes mínimo a ser monitorado.

# Process checks.

proc portmap 100 0      #Monitora o serviço portmap e avisa caso tenha menos de
                        zero e mais de 100 processos portmap sendo executados.

proc syslogd 100 0

proc /usr/sbin/sshd 100 0

proc ntpd 100 0

proc /usr/bin/freshclam 100 0

proc crond 100 0

proc xinetd 100 0
```

```
proc /usr/bin/nxserver 100 0
```

Não altera mais nenhum parâmetro, salve o arquivo e inicie o serviço:

```
service snmpd start
```

39.2.1.1 Testes

Para testar se o snmp está ativo em nossa máquina, podemos fazer uma consulta. As consultas podem conter filtros por MIB etc. No nosso caso fazermos uma consulta geral com o comando:

```
snmpwalk -v1 -c public numero_do_ip
```

39.2.2 Instalando o MRTG

Instale o pacote *MRTG*. Depois de instalado, iremos criar o diretório onde serão gerados os documentos gráficos/HTML.

```
urpmi mrtg
```

Iremos criar agora o arquivo de configuração com o *cfgmaker*.

```
cfgmaker --global 'WorkDir: /var/lib/mrtg' -output /etc/mrtg/mrtg.local.cfg
public@localhost
```

Caso queira configurar mais algum *device*, use o mesmo comando com outro nome. Lembrando que depois do "@" deve-se especificar o nome na rede do *device* ou o IP. Exemplo, vamos analisar o tráfego do roteador ligado ao l*ink Frame Relay*:

```
cfgmaker --global 'WorkDir: /var/lib/mrtg' -output /etc/mrtg/mrtg.router.cfg
public@172.18.0.254
```

Edite o arquivo /etc/mrtg/mrtg.router.cfg e acrescente as seguintes linhas logo após a diretiva “EnableIPV6: no”:

```
WorkDir: /var/lib/mrtg          #Define qual será a pasta de trabalho do MRTG; ou
                                seja, a pasta onde serão salvos os arquivos gerados
                                pelo MRTG (logs, arquivos html e png, etc). É
                                recomendável criar uma sub-pasta para cada host.

Options[_]: growright, bits     #São duas opções em uma (mas podem ser configuradas
                                separadamente): o growright faz com que o gráfico
                                "caminhe" da direita para a esquerda, fazendo com
                                que o horário atual fique à direita no gráfico; já
                                o parâmetro bits define que o gráfico trará as
                                informações em bits (por padrão, as informações são
                                expressas em bytes).

Interval: 10                    #É o tempo, em minutos, em que o MRTG irá buscar
                                novas informações estatísticas junto ao host. Por
                                padrão, 5 minutos.

Refresh: 600                    #É o tempo, em segundos, em que o navegador irá
```

```
                                atualizar a página. Por padrão, 300 segundos (5
                                minutos).

Language: brazilian             #Linguagem que será utilizada nos arquivos HTML que
                                o MRTG gera.

RunAsDaemon: Yes
                                #Para rodar o MRTG como daemon (processo). Ou seja,
                                o MRTG ficará carregado, e vai buscar os dados do
                                host conforme o parâmetro Interval (ou nos 5
                                minutos padrão).
```

Em seguida, criaremos a sua página index, a a partir dos dois arquivos de configuração:

```
indexmaker --output=/var/lib/mrtg/index.html /etc/mrtg/mrtg.local.cfg
/etc/mrtg/mrtg.router.cfg
```

Por fim, mas não menos importante, vamos rodar o MRTG:

```
env LANG=C /usr/bin/mrtg /etc/mrtg/mrtg.local.cfg

env LANG=C /usr/bin/mrtg /etc/mrtg/mrtg.router.cfg
```

39.2.3 Otimizando e Protegendo

Caso você tenha configurado como *daemon* lembre-se de iniciar o processo sempre que iniciar a máquina, por exemplo inserindo o comando ao final do arquivo /etc/rc.local. Caso não esteja como *daemon* criaremos uma entrada no *cron* para gerar atualização a cada 5 min.

```
crontab -e
```

Inserimos as linhas (com “env LANG=C” se necessário):

```
*/5 * * * * /usr/bin/mrtg /etc/mrtg/mrtg.local.cfg

*/5 * * * * /usr/bin/mrtg /etc/mrtg/mrtg.router.cfg
```

Criando arquivo de autenticação para a página do MRTG, caso desejemos que somente pessoas autorizadas tenham acesso à página.

```
vi /var/www/html/mrtg/.htaccess
```

Dentro do arquivo ponha:

```
AuthName "MRTG Graphs/Html restricted access"

AuthType Basic

AuthUserFile /etc/httpd/conf/htpasswd

require user mrtgadmin
```

Depois, crie uma senha para autenticar:

```
htpasswd -c /etc/httpd/conf/htpasswd mrtgadmin
```

Mude o dono e grupo do arquivo de senhas para que o Apache possa acessá-lo:
```
chown apache:apache /etc/httpd/conf/htpasswd
```

Crie um host virtual no Apache, conforme secção 26.3, apontando para /var/lib/mrtg.

39.3 Testes

Acesse a página 192.168.2.**X**/mrtg e verifique o gráfico gerado que deve ser algo similar à Ilustração 35.

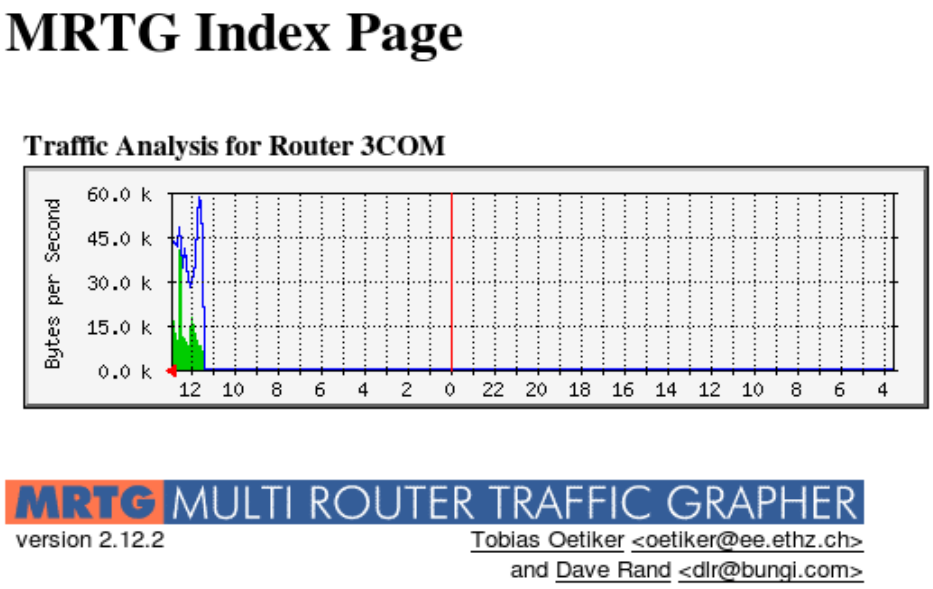


Ilustração 35: Gráfico típico do MRTG

Obs.: poderíamos personalizar os cabeçalhos dos gráficos da página editando os arquivos /etc/mrtg/mrtg.local.cfg e /etc/mrtg/mrtg.router.cfg

40 Nagios

40.1 Introdução

O Nagios é um aplicativo de monitoramento de sistemas e de redes. Ele checa clientes e serviços, por você especificados, alertando quando as coisas estão indo mal ou se restabelecendo.
O Nagios foi originalmente desenhado para rodar no Linux, apesar dele poder funcionar na maioria dos Unix. Para mais informações sobre em qual sistema operacional o Nagios irá, ou não, funcionar, veja a página de portabilidade em sistemas operacionais, acessível em http://www.nagios.org/ports.shtml.
Alguma das várias ferramentas do Nagios incluem:

- Monitoramento de rede e serviços (SMTP, POP3, HTTP, NNTP, PING, etc.);
- Monitoramento dos recursos de clientes (carga de processador, uso de disco, etc.);
- Organização simples de *plugins* que permite aos usuários desenvolverem

seus próprios serviços de checagem;
- Checagem paralela de serviços;
- Habilidade para definir hierarquia de redes de clientes usando clientes pais (*parent hosts*), permitindo a detecção e distinção entre clientes que estão desativados e aqueles que estão inalcançáveis;
- Notificação de contatos quando problemas em serviços e clientes ocorrerem ou forem resolvidos (via email, pager, ou métodos definidos pelo usuário);
- Habilidade para definir tratadores de eventos (*event handlers*) que serão executados durante eventos de serviços ou clientes na tentativa de resolução de problemas;
- Rotatividade automática de arquivos de logs;
- Suporte para implementação de clientes de monitoramento redundantes;
- Interface *web* para visualização do status atual da rede, histórico de notificações e problemas, arquivos de log, etc.

40.2 Instalando e configurando o Nagios

Para instalarmos o Nagios devemos ter previamente instalado um servidor Apache. Em seguida instalamos o Nagios propriamente dito com o comando:

```
urpmi nagios-www
```

Junto com estes pacotes são instalados todos os arquivos necessários para o funcionamento do Nagios e, inclusive, é reconfigurado automaticamente o Apache para poder acessar as páginas do Nagios. Sendo necessário somente criar uma senha para o acesso ao Nagios com o comando:

```
htpasswd /etc/nagios/passwd nagios
```

O Nagios vem pronto para o funcionamento, **não sendo necessária nenhuma alteração de arquivos** para o monitoramento das funções básicas da máquina local.
De qualquer modo destacamos os principais arquivos de configuração e suas aplicações que são os seguintes (/etc/nagios):

- nagios.cfg: arquivo principal de configuração do Nagios;
- command-old-style.cfg: compatibilidades com versão 2;
- group: usuários que podem configurar o Nagios;
- cgi.cfg: arquivo de configuração das CGIs;
- passwd e passwd.plaintext: senhas de usuários do Nagios;
- resource.cfg: arquivo contendo macros definidas pelo usuário;

Dentro do diretório do Nagios existe um sub-diretório – ***objetcs*** – que contém os principais arquivos de configuração dos monitoramentos a serem realizados.

- commands.cfg: definições dos comandos a serem executados pelo Nagios;
- contacts.cfg: indivíduos que, possivelmente, deverão ser notificados no caso de problemas na rede;
- localhost.cfg: definições do monitoramento do hospedeiro especial *localhost;*
- printer.cfg: definições do monitoramento de impressoras;
- switch.cfg: definições do monitoramento de switchs;

- templates.cfg: é um arquivo que contém vários exemplos de como utilizar as diretivas de monitoramento do NAGIOS;
- timeperiods.cfg: definições de horários considerados válidos para a realização de checagens e envio de notificações;
- windows.cfg: exemplos de configurações específicas para monitoramento de máquinas Windows.

Dentro do diretório do Nagios existe outro sub-diretório – ***plugins*** – que contém os principais arquivos de configuração dos plugins para monitoramentos específicos, como por exemplo plugins de monitoramento de discos, servidores http, processos etc.
Além disto o Nagios 3 já possui outros diretórios para organizar o monitoramento agrupando as máquinas por seu tipo. Por exemplo: roteadores, servidores, *switchs* etc.
Editamos um dos arquivos de configuração do Nagios, /etc/nagios/cgi.cfg, para ficar do seguinte modo:

```
use_authentication=0
```

Podemos fazer uma conferência inicial da configuração dos arquivos com o comando:

```
nagios -v /etc/nagios/nagios.cfg
```

Iniciamos o Nagios com o comando:

```
service nagios start
```

Reiniciando o servidor Apache já é possível acessar a página do Nagios, com o endereço http://localhost/nagios/. Observe que esta página estará acessível somente via localhost, caso desejemos que a mesma seja acessível de outras máquinas devemos editar o arquivo /etc/httpd/conf/webapps.d/12_nagios.conf e mudar as diretivas "*allow from*" para o(s) ip(s) desejados e reiniciar o Apache.
O Nagios já estará plenamente funcional e monitorando uma série de serviços e características do *hardware* da máquina *localhost*.
Vamos acrescentar o monitoramento de alguns dados de uma máquina remota.

40.2.1 Monitorando outras máquinas

Para o nosso modesto exemplo editamos o arquivo /etc/nagios/conf.d/sample.cfg e acrescentamos na seção "HOST DEFINITION" as seguintes linhas para monitorar uma máquina chamada dk:

```
define host{

        use             linux-server      ; tipo de máquina: linux-server,
                                          windows-server etc.

        host_name       dk                ; nome da máquina

        alias           dk                ; apelido da máquina

        address         172.18.0.1        ; ip da máquina

        }
```

Na secção “HOST GROUP DEFINITION”, acrescentamos o nome de nossa nova máquina:

```
      members         localhost,dk
```

Na secção “SERVICE DEFINITIONS”, definimos os parâmetros que queremos monitorar, como por exemplo:

```
# A definição abaixo monitora o serviço http, se quisermos desabilitar a mesma
basta mudar o valor do parâmetro  notifications_enabled para 0.

define service{

        use                             generic-service

        host_name                       dk

        service_description             HTTP

        check_command                   check_http

        notifications_enabled           1

        }

# Esta definição é basicamente para saber se a máquina está ativa ou não. No
check_command são definidos: Se tempo de resposta é maior que 100 ms e a perda de
pacotes é maior que 20% será gerado um alarme de aviso. Se o tempo de resposta é
500 ms e a perda de pacotes for 60% será gerado um alarme crítico.

define service{

        use                             generic-service

        host_name                       dk

        service_description             PING

        check_command                   check_ping!100.0,20%!500.0,60%

        }

# Monitora a partição raiz. Gera um alarme de aviso quando o espaço livre for
menor que 20% e um alarme crítico quando menor que 10%.

define service{

        use                             generic-service

        host_name                       dk

        service_description             Root Partition

        check_command                   check_local_disk!20%!10%!/
```

```
        }

#Monitora a quantidade de usuários logados: aviso 20 ou mais, crítico 50.

define service{

        use                             generic-service

        host_name                       dk

        service_description             Current Users

        check_command                   check_local_users!20!50

        }

# Monitora os processos. Aviso mais de 250 processos, crítico mais de 400. A flag
-s do comando ps mostra processos com estados específicos, por exemplo R = run, Z
= zombie etc. Se desejado podemos omitir !RSZDT e assim serão monitorados todos os
processos em todos os estados.

define service{

        use                             generic-service

        host_name                       dk

        service_description             Total Processes

        check_command                   check_local_procs!250!400!RSZDT

        }

# Monitora a carga da máquina.

define service{

        use                             generic-service

        host_name                       dk

        service_description             Current Load

        check_command                   check_local_load!5.0,4.0,3.0!10.0,6.0,4.0

        }

# Monitora a partição swap.

define service{

        use                             generic-service
```

```
        host_name                       dk

        service_description             Swap Usage

        check_command                   check_local_swap!20!10

        }

# Monitora o serviço ssh. Idem http.

define service{

        use                             generic-service

        host_name                       dk

        service_description             SSH

        check_command                   check_ssh

        notifications_enabled           1

        }
```

40.3 Testes

Podemos fazer um teste rápido da integridade da configuração indo para o diretório /etc/nagios com o comando:

```
nagios -v nagios.cfg
```

Agora devemos (re)iniciar o nagios com o comando:

```
service nagios (re)start
```

E podemos monitorar os novos serviços em: http://localhost/nagios/. Cabe salientar que o monitoramento do Nagios se dá em intervalos de tempo aleatórios, portanto haverá um certo retardo até as informações serem completadas.
Também é muito interessante atualizar o endereço do e-mail do(s) administrador(es) da rede, isto deve ser feito também no arquivo /etc/nagios/conf.d/sample.cfg mudando o campo “email” na secção “CONTACTS”.
Para os administradores de rede é muito interessante uma ferramenta associada ao Nagios, que é um plugin do navegador Firefox: **Nagios Checker**. Com este plugin instalado e configurado haverá avisos sonoros e visuais no próprio navegador sempre que for gerado algum alarme pelo Nagios.

41 Cacti

41.1 Introdução[20]

Cacti é uma ferramenta que recolhe e exibe informações sobre o estado de uma rede de computadores através de gráficos. Foi desenvolvido para ser flexível de modo a se adaptar facilmente a diversas necessidades, bem como ser robusto e fácil de usar. Monitora o estado de elementos de rede e programas bem como largura de banda utilizada e uso de CPU.
Trata-se de uma interface e uma infra-estrutura para o RRDTool, que é responsável por armazenar os dados recolhidos e por gerar os gráficos. As informações são repassadas para a ferramenta através de *scripts* ou outros programas escolhidos pelo usuário os quais devem se encarregar de obter os dados. Pode-se utilizar também o protocolo SNMP para consultar informações em elementos de redes e/ou programas que suportam tal protocolo.
Sua arquitetura prevê a possibilidade de expansão através de *plugins* que adicionam novas funcionalidades. Um destes *plugins* é o PHP Network Weathermap que mostra um mapa da rede e o estado de cada elemento.

41.2 Instalação e configuração

Para instalarmos o Cacti no Mandriva precisamos do Apache funcionando no sistema e em seguida digitarmos o comando:

```
urpmi cacti mysql

service mysqld start
```

Agora devemos criar a base de dados a partir de um modelo criado pelos pacotes do cacti, com os seguintes comandos:

```
mysql -u root -p                              #entraremos no Mysql

“requisição de senha”                         #em branco <Enter>

CREATE DATABASE cacti;                        #criamos a base cacti

use cacti                                     #”entraremos” na base cacti

source /usr/share/cacti/cacti.sql             #”povoaremos” a base a partir do
                                              modelo

exit                                          #saímos do Mysql
```

Editamos o arquivo /etc/cacti.conf e editamos a linha abaixo, compatibilizando com a senha criada no Mysql.

```
$database_username = "root";
$database_password = "senha_do_root (em_branco)";
```

Observe que esta não é a configuração mais segura. Num caso real devemos criar um usuário e senha na base Mysql e dar permissões ao mesmo de acesso à

20 Texto obtido de http://pt.wikipedia.org/wiki/Cacti

base cacti. Informamos então este usuário e senha no arquivo /etc/cacti.conf.
Iniciamos o serviço mysql com o comando:

```
service mysqld (re)start
```

Agora então podemos acessar, via navegador, a página http://localhost/cacti/ e seguir as orientações da mesma. Todas as opções estão na configuração padrão, aceite-as. Ao final será requisitado login e senha, entre com admin X admin e o Cacti automaticamente requisitará a troca de senha do usuário admin.
Após isto podemos clicar na aba GRAPHS poderemos observar algo parecido com o da Ilustração 36.
Podemos ainda configurar mais monitores, para isto basta clicarmos na aba CONSOLE, na opção New Graph e selecionarmos as opções desejadas.
Podemos ainda verificar as configurações do Cacti da aba SETTINGS e demais abas.

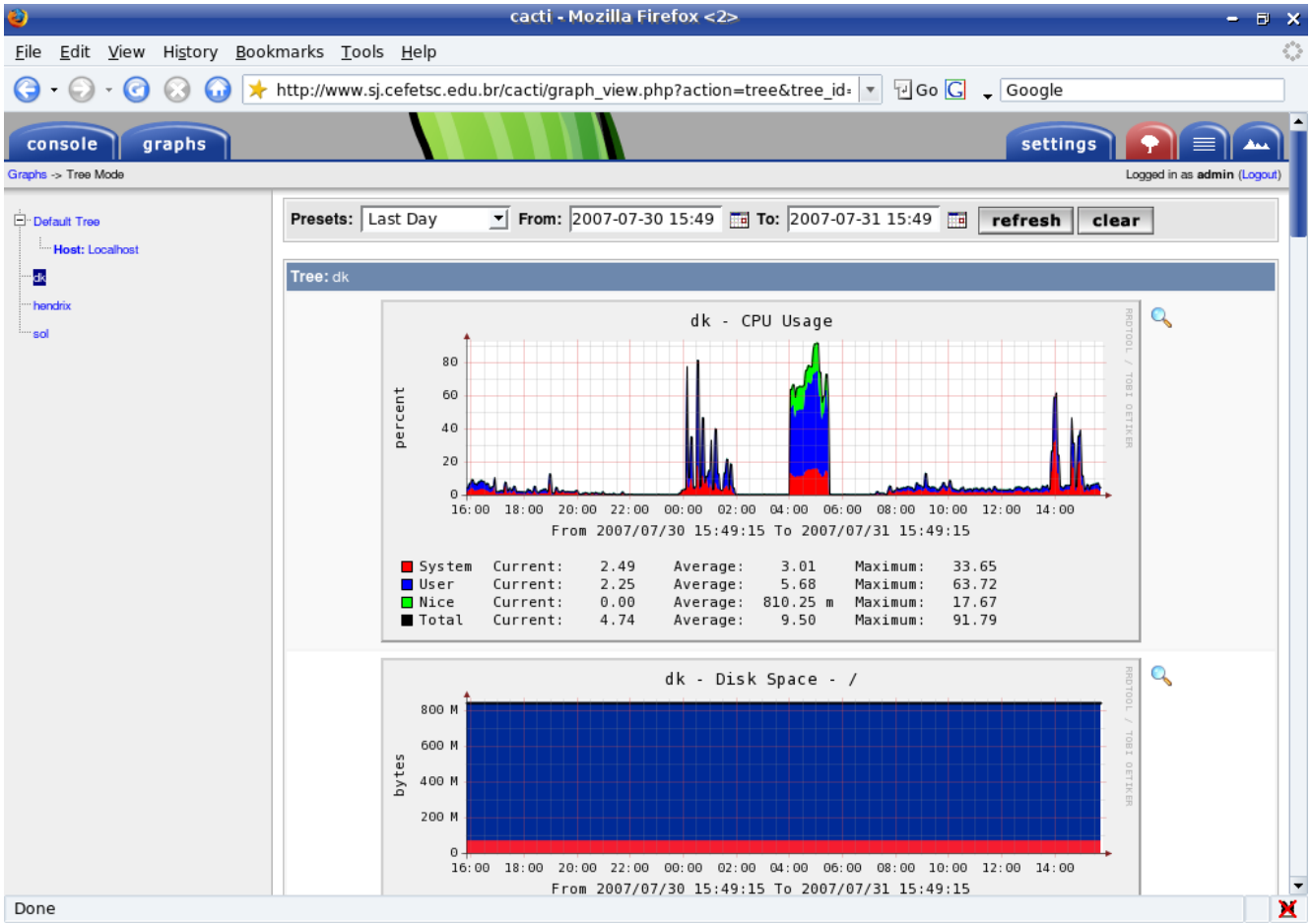


Ilustração 36: Gráficos do Cacti

42 DenyHosts

42.1 Introdução[21]

DenyHosts é um *script* escrito por Phil Schwartz para ajudar administradores de sistemas bloquear ataques de força bruta em seus servidores SSH. Ele monitora os arquivos de LOG do sistema (/var/log/secure no Redhat, /var/log/auth.log on Mandrake, etc...) e quando um ataque é detectado adiciona o IP do atacante no /etc/hosts.deny.
Quando executado pela primeira vez, o DenyHosts irá criar um diretório de trabalho para armazenar as informações coletadas dos arquivos de LOG em um formato que nós, humanos mortais, possamos ler, compreender e editar caso seja necessário.
O *script* possui uma grande variedade de configurações que podem ser exploradas, como por exemplo, configurar quantas tentativas inválidas devem ser consideradas um ataque, ou quantas tentativas erradas de usuários que não existem no seu sistema são aceitas... pode enviar emails com relatórios... essas configurações serão explicadas adiante.

42.2 Instalando o DenyHosts

O DenyHosts é dependente do Python v2.3 ou superior. Execute o seguinte comando para saber qual versão você tem, se é que tem:

```
rpm -q python
```

Caso você não possua o Python instalado, instale com o comando:

```
urpmi python
```

Agora vamos a instalação do DenyHosts propriamente dito. Faça download da última versão do DenyHosts na página oficial, existe um rpm noarch (para qualquer arquitetura) e um tar.gz que também é independente de plataforma, vamos optar por este.

http://denyhosts.sourceforge.net/

Desempacote o DenyHosts com o comando:

```
tar -zxvf DenyHosts-2.6.tar.gz
```

Para manter o sistema organizado, colocamos o DenyHosts dentro do /sbin

```
mv DenyHosts-2.6 /sbin/DenyHosts
```

Vamos criar um arquivo de configuração baseado no arquivo de configuração exemplo:

```
cd /sbin/DenyHosts

cp denyhosts.cfg-dist denyhosts.cfg
```

Agora vamos editar o arquivo de configuração. Veremos várias opções, as principais, que podem ou não ser utilizadas.

21 Texto obtido de http://www.drsolutions.com.br/exemplos/protegersshd.pdf

```
vi denyhosts.cfg
```

SECURE_LOG = /var/log/auth.log	#Deve apontar para o seu arquivo de Log de autenticação.
PURGE_DENY = 2w	#Depois de quanto tempo o bloqueio para aquele IP será removido. No exemplo está ajustado para 2 semanas. Se deixarmos em branco nunca será removido.
BLOCK_SERVICE = sshd	#Serviços que serão bloqueados. Pode ser **ALL**, para todos os serviços.
DENY_THRESHOLD_INVALID = 5	#Número de tentativas que um usuário inválido, ou seja, não está no /etc/passwd deve fazer para que seja bloqueado.
DENY_THRESHOLD_VALID = 10	#Número de tentativas que um usuário válido, ou seja, está no /etc/passwd deve fazer para que seja bloqueado.
DENY_THRESHOLD_ROOT = 1	# Número de tentativas erradas com o root.
WORK_DIR = /usr/share/denyhosts/data	#Diretório onde serão armazenados os arquivos do DenyHosts.
HOSTNAME_LOOKUP=YES	#Quando setado para "YES", todo IP x FQDN reportado ao DenyHosts tentará ser resolvido.
ADMIN_EMAIL = user@dominio.xxx.xx	# Email que irá receber os relatórios de segurança.
	#SMTP_*: Configura a conta de email que será usada para o envio dos emails de relatórios.
SMTP_HOST = smtp.dominio.xxx.xx	
SMTP_PORT = 25	
SMTP_FROM = DenyHosts <user@dominio.xxx.xx>	
SMTP_SUBJECT = DenyHosts Report	
SMTP_USERNAME = user	
SMTP_PASSWORD = senha.aqui	
DAEMON_LOG = /var/log/denyhosts	# Arquivo de LOG do DenyHosts.
DAEMON_SLEEP = 30s	#De quanto em quanto tempo o

	DenyHosts deve varrer o arquivo de logs do sistema. No exemplo deixamos 30s
DAEMON_PURGE = 6h	#De quanto em quanto tempo o DenyHosts deve reescrever o arquivo HOSTS_DENY.

Agora devemos rodar o programa. O modo mais recomendado é como *daemon*.
```
/sbin/DenyHosts/denyhosts.py --config=/sbin/DenyHosts/denyhosts.cfg --daemon
```

Pode ser necessário criar um arquivo vazio para o DenyHosts rodar pela primeira vez, caso ocorra uma mensagem de erro. Faça isto com o comando:
```
touch /var/log/secure
```

É interessante adicionarmos este comando ao final do arquivo /etc/rc.local, para rodar sempre que a máquina for reiniciada.

42.3 Testes

Agora podemos fazer alguns testes informando usuários inexistentes, usuários válidos mas com senha errada etc.
Em seguida observamos os arquivos criados/modificados no diretório /usr/share/denyhosts/data. Nestes arquivos teremos as relações de *hosts* e usuários “travados”, liberados e datas de acesso. Por exemplo:
```
tail hosts-restricted
```

```
89.119.134.50:0:Wed Jun  6 19:51:04 2007
89.121.0.99:0:Tue Oct 17 05:56:19 2006
89.137.189.2:0:Wed Apr  4 07:25:18 2007
89.171.160.18:0:Sun Jul 29 07:42:05 2007
89.212.5.25:0:Sun Jan 28 02:21:02 2007
89.250.246.112:0:Mon Apr 23 07:04:25 2007
89.96.238.226:0:Fri Oct  6 08:56:38 2006
89.97.246.138:0:Wed Apr 25 22:42:32 2007
91.192.213.196:0:Wed Jun 13 07:40:34 2007
91.92.222.198:0:Sun Jul 22 16:50:25 2007
```

43 Webmin

43.1 Introdução[22]

O Webmin é um gerenciador de sistema baseado numa interface web. Com este utilitário você pode administrar sua(s) máquina(s) pela rede através de um navegador comum. Ele é bem completo e tem módulos para configuração de várias e várias coisas. É uma mão e tanta para os administradores de sistema. Algumas das tarefas que você pode fazer com o Webmin atualmente:

- Mudar senhas, configurar o crontab, configurar scripts de inicialização,

22 Texto obtido de http://www.devin.com.br/eitch/webmin/

backup, configuração do pam, quotas, gerência de processos, pacotes, usuários e grupos.
- Configura e administrar servidores majordomo, cvs, sendmail, qmail, postfix, fetchmail, jabber, samba, postgresql, proftpd, ssh, squid, wu-ftpd, apache, dhcp, dns bind, MySQL.
- Configura rede, exportações NFS, NIS, PPP, túneis SSL.
- Administração de impressoras, gerenciadores de boot, cd-roms, raid, partições, lvm, clustering.
- Além de outras coisas como shell via web, gerenciador de arquivos, módulos perl, etc.

Então dá pra ver que o sistema é bem completo né? E ele é também amplamente usado. Vamos através deste tutorial saber como instalar e configurar de um modo bem prático e direto.

43.2 Instalação e configuração

Para instalar e rodar o Webmin devemos executar os comandos:

```
urpmi webmin

service webmin start
```

Uma vez instalado ele já estará absolutamente pronto para o uso, para isto basta acessar com um navegador qualquer o endereço https://192.168.2.**X**:10000/. Ou seja uma conexão segura, https, na porta 10000 de seu servidor. Agora devemos informar o usuário e senha, que são as mesmas cadastradas em nossa máquina. Se desejamos fazer manutenção nos serviços o ideal é usar o próprio root.
No primeiro acesso teremos uma janela do tipo mostrado na Ilustração 37:

Ilustração 37: Primeira janela do Webmin

Como primeira configuração devemos alterar a linguagem de apresentação clicando no ícone "*Change Language and Theme*". Escolhemos "*Portuguese (Brasilian) (PT_BR)*" setamos a opção "*Personal Choice*"e clicamos em "*Make Change*". Agora teremos a interface em português, Ilustração 38.
Agora podemos dar uma navegada nas diversas janelas, principalmente em "Servidores", onde termos acesso a todas as configurações dos servidores que já instalamos configuramos. Ou seja, podemos fazer a manutenção/configuração do nosso servidor remotamente, através de um navegador qualquer e de maneira bastante amigável.

Ilustração 38: Webmin em português

44 Referências bibliográficas

44.1 Livros/apostilas

Tibet, Chuck V. Linux: Administração e Suporte. Novatec Editora. ISBN: 85-85184-95-7. 2001.
Ferreira, Rubens E. Linux: Guia do Administrador do Sistema. Novatec Editora. ISBN: 85-7522-038-1. 2003.
Hunt, Craig. Linux: Servidores de rede. Editora Ciência Moderna. ISBN: 85-7393-321-6. 2004.
Stanger, James; Lane, Patrick T.; Danielyan, Edgar. Rede Segura Linux. Editora Alta Books. ISBN: 85-88745-10-0. 2002.
Nemeth, Evi; Snyder, Garth; Seeebas, Scott; Hein, Trnt T. Manual de Administração do Sistema Unix. Editora Boojman. ISBN: 85-7307-979-7. 2002.
Curso de Introdução ao Linux. Marco Álvarez, Cláudia Nasu, Alfredo Lanari, Luciene Marin. Universidade Federal de Mato Grosso do Sul.

44.2 Página de Gerência de Redes do IFSC - Campus São José.

http://www.sj.ifsc.edu.br/wiki/index.php/Ger%C3%AAncia_de_Redes_%28p

%C3%A1gina%29

44.3 Sites de dicas Linux:

http://pt.wikipedia.org/wiki/TAR
http://www.dicas-l.com.br/dicas-l/19980517.php
http://pt.wikipedia.org/wiki/RPM
http://pt.wikipedia.org/wiki/Sistema_de_ficheiros
http://www.linuxbsd.com.br/phpLinuxBSD/modules/artigos_tecnicos/fstab.htm
Multiterminais/Multiterminal com Ruby – Wikibooks - http://pt.wikibooks.org/wiki/Multiterminais/Multiterminal_com_Ruby
Index of /MPlayer/releases/codecs - http://www4.mplayerhq.hu/MPlayer/releases/codecs/
Roger Lovato Â» Webcam DSB-C110 no Mandriva Linux - http://www.roger.lovato.com.br/artigos/11
Piter Punk's HomePage - dicas - http://piterpunk.info02.com.br/dicas.html
AURELIO.NET - http://aurelio.net/
iMasters - Por uma Internet mais criativa e dinÃ¢mica - http://www.imasters.com.br/
Comunidade de profissionais, estudantes e mestres em tecnlogias e ferramentas voltadas para o desenvolvimento web.
Guia Foca GNU/Linux - http://focalinux.cipsga.org.br/guia/avancado/index.htm#contents
Linux Operating System and Linux Distributions - http://linux.about.com/About_Focus_on_Linux.htm
This site is your Internet destination for information and resources for the Linux operating system and various Linux distributions.
Linux-Tip.net - Remote access Mandriva 2007 Free using FreeNX - http://www.linux-tip.net/cms/content/view/254/26/
Linux-Tip.net, Linux-Tip.eu, Linux-Tip.com Linux Tips and Tricks, workshops, news and articles, Linux Security, NoMachine NX is a Terminal Server and remote access solution based on a comprising set of enterprise class open source technologies. NX makes it possible to run any graphical application on any operating system across any network connection at incredible speed.
Linux in Brazil: Novo artigo: AutenticaÃ§Ã£o com o PAM-LDAP - http://br-linux.org/news2/006653.html
PrÃiticas de SeguranÃ§a para Administradores de Redes Internet - http://www.cert.br/docs/seg-adm-redes/
PrÃiticas de SeguranÃ§a para Administradores de Redes Internet
PÃigina principal - Nagios-BR - http://nagios-br.sourceforge.net/wiki/index.php/PÃigina_principal
RelatÃ³rios em Java - http://www.dsc.ufcg.edu.br/~jacques/cursos/daca/html/documentviews/relatorios.htm
Easy Urpmi - http://easyurpmi.zarb.org/
Jarbas Teixeira - Usando o URPMI - Parte 01 - http://www.imasters.com.br/artigo/5127/linux/usando_o_urpmi_-_parte_01/
Um pacote trÃis tambÃ©m informaÃ§Ãµes sobre prÃ©-requisitos para ser

instalado ou as chamadas dependÃªncias.
Jarbas Teixeira - URPMI - Parte 02
http://www.imasters.com.br/artigo/5127/linux/usando_o_urpmi_-_parte_02/
Vamos estudar mais detalhes de como usar o URPMI e configurar mÃdias via http.
Comunidade LinuxBSD - Montando partiÃ§Ãµes no fstab - http://www.linuxbsd.com.br/phpLinuxBSD/modules/artigos_tecnicos/fstab.htm
Aldeia NumaBoa - BIND (DNS) na jaula - http://www.numaboa.com/content/view/424/167/1/0/
Aldeia NumaBoa - um portal diferente em PortuguÃªs do Brasil
Shorewall http://www.grulic.org.ar/eventos/charlas/shorewall-2005-09.html

44.4 Sites com dicas do Amanda

ConsideraÃ§Ãµes bÃisicas sobre backup - http://www.openit.com.br/freebsd-hb/backup-basics.html
Backup em HD com Amanda â€" Projeto CyberShark.net - http://www.cybershark.net/tutoriais/amanda_hd
AMANDA, The Advanced Maryland Automatic Network Disk Archiver - http://www.amanda.org/
[Dicas-L] Amanda: Backup DistribuÃdo - http://www.dicas-l.com.br/dicas-l/20000711.php
Chapter 13. How to use the Amanda file-driver - http://www.amanda.org/docs/howto-filedriver.html
Amanda Faq-O-Matic: How to configure for tapeless operation? - http://amanda.sourceforge.net/fom-serve/cache/191.html

44.5 Sites com dicas do Postfix e listas de discussão

Postfix : INTEGRANDO O POSTFIX COM O CLAMAV - http://www.unitednerds.org/thefallen/docs/index.php?area=Postfix&tuto=Clamav-gsoares
Filtros para o Postfix | LinuxMan - http://www.linuxman.pro.br/node/4
The Postfix Home Page - http://www.postfix.org/
Integrating amavisd-new Into Postfix For Spam- And Virus-Scanning | HowtoForge - Linux Howtos and Tutorials - http://www.howtoforge.com/amavisd_postfix_debian_ubuntu
Linux: GNU Mailman email list installation and configuration - http://www.yolinux.com/TUTORIALS/LinuxTutorialMailman.html
Mailman Tutorial. The YoLinux portal covers topics from desktop to servers and from developers to users
GNU/Mailman Administrator's Manual - http://radonio.iq.usp.br/outros/mailman/manual_mailman.html
Tutoriais/FreeBSD/Mailman - UnderLinux Wiki - http://under-linux.org/wiki/index.php/Tutoriais/FreeBSD/Mailman
Sistema de correio eletrÃ´nico - WikipÃ©dia -

http://pt.wikipedia.org/wiki/Sistema_de_correio_eletrÃ´nico
Linux: Como configurar o servidor de correio eletrÃ´nico Postfix [Artigo] - http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=189
Viva o Linux - Porque nÃ³s amamos a liberdade! Seja livre, use Linux. A maior e melhor comunidade para se aprender Linux do Brasil, artigos, dicas, forum e muito mais.
BOM - Correio EletrÃ´nico - http://www.conectiva.com/doc/livros/online/10.0/servidor/pt_BR/ch11.html