Administração de Redes com LINUX: Fundamentos e Práticas

Book · January 2010

CITATIONS
0

READS
5,402

1 author:

Odilson Tadeu Valle
Federal Institute of Santa Catarina
19 PUBLICATIONS **144** CITATIONS

SEE PROFILE

Administração de Redes com Linux

Fundamentos e Práticas

Odilson Tadeu Valle

Administração de Redes com LINUX

Fundamentos e Práticas

Odilson Tadeu Valle

Florianópolis
2010

PUBLICAÇÃO do IF-SC

V181a Valle, Odilson Tadeu.

Administração de redes com linux: fundamentos e práticas/ Odilson Tadeu Valle. – Florianópolis: Publicações do IF-SC, 2010.

302 p. : il. ; 15x21 cm.

ISBN: 978-85-64426

1. Redes de computadores. 2. Linux. 3. Título.

CDD: 004.6

Catalogado por: Coordenadoria de Bibliotecas IF- SC

Kênia Raupp Coutinho CRB14/951

Este livro é dedicado às pessoas que dão sentido à minha vida:

Maria,
Bruno e
Cláudia.

Agradecimentos

Quero agradecer especialmente ao Emerson pelos ensinamentos e ajustes no LaTeX, a Cláudia pela revisão do texto e ao Cassiano pela linda capa. Quero agradecer também aos colegas Alexandre, Eraldo, Evandro, Marcos, Mário, Saul e todos os demais professores da Área de Telecomunicações do IF-SC, pelo apoio e companheirismo ao longo destes anos de trabalho. Ao IF-SC pelo suporte e publicação deste livro.

Agradeço também a minha família, por todo histórico de vida e que é responsável por ser eu quem sou.

Prefácio

Este livro é fruto da compilação de roteiros didáticos, de anos de trabalho com disciplinas de Administração de Redes no IF-SC. Muitos destes roteiros são adaptações de materiais encontrados na Internet. Ele tem por objetivo ser base para o ensino de Administração de Redes com uso do sistema Operacional Linux. A abordagem dada nos capítulos sempre inicia com uma pequena introdução teórica sobre determinado assunto ou serviço, seguido por um roteiro prático para experimentação do mesmo. Do mesmo modo que a abordagem teórica é sucinta e breve, a experimentação também o é. São abordados dezenas de tópicos desde a parte introdutória até a parte mais avançada na administração de serviços de rede usando o Linux. Por isto não se tem a pretensão de uma abordagem profunda sobre todos os temas abordados.

O diagrama da Figura 1 apresenta os capítulos integrantes do livro agrupados em 6 blocos. O objetivo é ilustrar a categorização de assuntos e indicar a sequência de leitura, indicada pelas setas entre blocos, que pode ser utilizada para um melhor aprendizado. O bloco base pretende dar um embasamento mínimo ao leitor para poder acompanhar o desenvolvimento do texto como um todo. Em seguida existe o bloco que trata das configurações mínimas necessárias a serem feitas num servidor de produção. A programação do *Shell* pretende dar suporte para agilizar tarefas repetitivas em todo o sistema. Os próximos três blocos são relativos aos servidores de rede que são o principal foco de estudo do livro. Dependendo do nível de conhecimento preliminar do leitor, etapas podem ser puladas, ou a

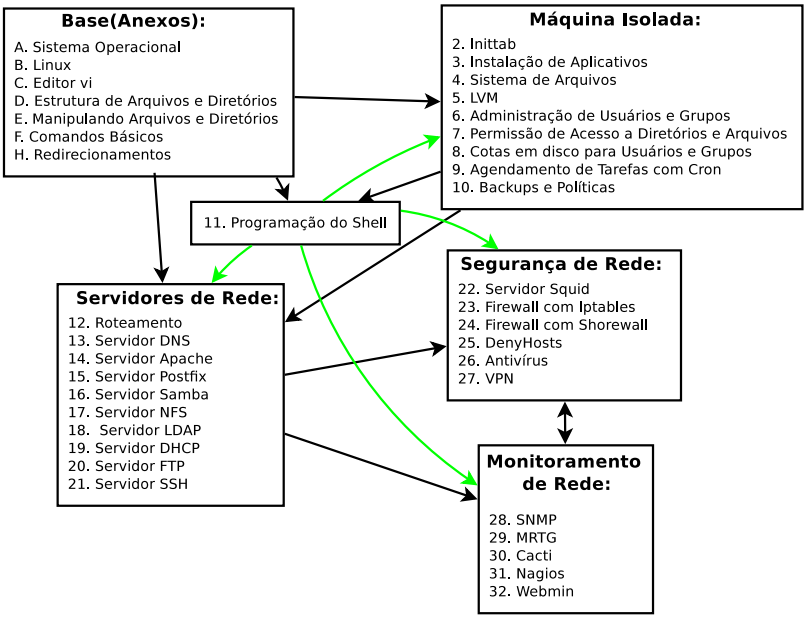

Figura 1: Diagrama esquemático para leitura dos capítulos

sequência de leitura pode ser modificada.

Todos os roteiros experimentais são baseados na distribuição Mandriva Linux, portanto, dependendo da distribuição Linux adotada, podem ser necessários ajustes.

Odilson Tadeu Valle

Sumário

Capítulo 1

Introdução

Como princípio o gerenciamento de redes consiste em controlar os dispositivos que compõe a rede local e prover serviços de rede aos usuários da maneira transparente e fácil aos mesmos, sem esquecer a segurança da mesma.

Usuários de rede acessam serviços como: correio eletrônico, navegação na internet, servidor para hospedar as páginas de seus projetos, um lugar seguro para guardar seus dados e documentos, impressoras etc. A ótica dos mesmo é que os serviços devem estar sempre disponíveis, na máxima velocidade e sem impecílios em seu uso.

A direção da instituição normalmente enxerga a rede como um recurso para melhorar a produtividade mas com um custo muitas vezes elevado. Sendo assim estabelece limites para gastos, seja com equipamentos, seja com contratação de enlaces com a Internet ou ponto a ponto.

Por outro lado, do ponto de vista do administrador da rede, para prover estes serviços de maneira estável e segura são necessários uma série de outros serviços e equipamentos que, a princípio, não interessa diretamente ao usuário ou à direção, sendo que muitas vezes os mesmos não percebem a existência destes. Ao administrador da rede cabe a tarefa de manter a rede ativa e funcional, sob todas as perspectivas, seja do usuário em busca de serviços, seja da direção da instituição

com as políticas de permissionamento de acessos, seja do ponto de vista de estabilidade, disponibilidade e confiabilidade que todos esperam da mesma.

Na Figura 1.1 é apresentado um diagrama com exemplos de serviços básicos de rede que um administrador de sistemas deve ofertar. Deve-se observar que os serviços podem ou não estar agrupados na mesma máquina.

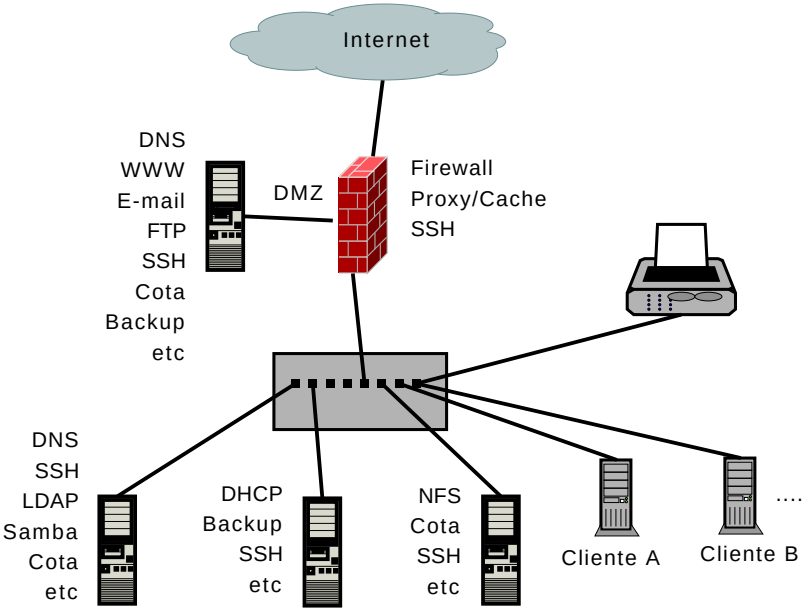

Figura 1.1: Serviços de Rede

A seguir são apresentados dezenas de tópicos, separados por capítulos. Inicia-se pela configuração básica do servidor, Capítulos 2 à 10, onde são abordados temas relativos à configuração e manutenção do servidor propriamente dito. Normalmente estas tarefas devem ser executadas no início da vida útil do servidor, para que o mesmo tenha longevidade.

Em seguida, como um capítulo a parte, é abordado rapidamente conceitos da programação *shell*, Capítulo 11, que tem por objetivo

auxiliar o administrador nas tarefas repetitivas de manutenção dos servidores e da rede.

Chega-se então ao foco do livro: os serviços de rede. Entre os Capítulos 12 à 32 são rapidamente abordados vários exemplos de serviços que devem fazer parte de uma rede. Iniciando-se pelos serviços propriamente dito, Capítulos 12 à 21, passando-se pelos serviços relacionados à segurança da rede, Capítulos 22 à 27 e, finalmente chegando aos serviços que permitem o monitoramento da mesma, Capítulos 28 à 32.

O texto tem por objetivo prover uma configuração básica dos vários serviços, com um conhecimento básico teórico/prático dos mesmos. Em caso de necessidades de configurações "avançadas" nos serviços, deve-se consultar bibliografias especializadas.

Todos os serviços e configurações realizadas ao longo do texto tem como base o sistema operacional Linux, ver Apêndices A e B. Mais especificamente foi utilizada a distribuição Mandriva©. O Linux foi escolhido principalmente por motivos didáticos, já que a configuração de parâmetros agrupados em arquivos texto, normalmente acrecidos de comentários, levam a um melhor entendimento dos processos envolvidos.

Capítulo 2

Inittab

2.1 Introdução

O arquivo `inittab` descreve que processos são iniciados no *boot* e durante a operação normal. O *init* distingue múltiplos `runlevels`, cada um dos quais pode ter o seu próprio conjunto de processos que serão iniciados. `Runlevels` válidos são 0-6.

2.2 Configuração

Antes que qualquer *script* de inicialização tenha sido executado pelo sistema operacional, ver Apêndice A, 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 (`runlevel`), 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.

Capítulo 3

Instalação de Aplicativos com RPM

3.1 Introdução

RPM, é um acrônimo de *Red Hat Package Manager* que 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 Novell e AIX da IBM.

3.2 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 de dados armazena todos os arquivos que são criados ou modificados quando um usuário instala um programa e facilita a remoção destes mesmos arquivos, quando da remoção do programa. Se o banco de dados é corrompido,

[0] http://pt.wikipedia.org/wiki/RPM

as ligações duplas garantem que eles possa ser reconstruído sem nenhum problema. Nos computadores com o sistema operacional RedHat e derivados instalado, este banco de dados se encontra em `/var/lib/rpm`.

3.3 Rótulo dos Pacotes

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

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

- 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 etc.).

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

3.4 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:

- É comum haver mudanças no formato de pacote incompatíveis com versões anteriores.
- Documentação incompleta e desatualizada.
- Pouca aprendizagem sobre os pacotes.

3.5 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.

3.6 Instalação/remoção 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 parecido com o apresentado abaixo.

---

nenhum nome de pacote digi

Os seguintes pacotes contem digi:

acroread-plugins-digitalsignature

digicamerge

digikam

digikamimageplugins

digitemp

libdigidoc2

libdigidoc2-devel

libdigikam0

libdigikam0-devel

rmedigicontrol

vdr-plugin-digicam

x11-driver-input-digitaledge

Como houve problemas digitamos então o nome completo do pacote, conforme uma das sugestões apresentadas.

urpmi digikam

3.6.1 Remoção de aplicativos

Para desinstalar (extrair) basta digitarmos:

urpme nome_do_pacote

3.7 Mídias do URPMI

Mídia é o local onde tem-se 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 sítios da internet, permitindo assim o administrador manter o sistema sempre atualizado. Existe um excelente sítio para cadastro de mídias que é o `http:\slash\slasheasyurpmi.zarb.org\slash`. Neste sítio configuramos as mídias de acordo com a versão e necessidades, basta seguir o roteiro do sítio.

Capítulo 4

Sistema de Arquivos

4.1 Introdução

Um dispositivo de armazenamento é um dispositivo físico construído para armazenar informações na forma binária, zeros e uns. São exemplos: discos rígidos, pen-drives, disquetes, cartões de memória etc.

Um dispositivo de armazenamento pode ser dividido em várias partes, cada uma destas partes é conhecida como partição que são meramente divisões lógicas. Ao particionar um dispositivo obtêm-se algumas vantagens, as principais são:

- A possibilidade de se instalar vários sistemas operacionais. Cada partição, ou conjunto de partições, pode conter um sistema operacional distinto. A escolha do sistema a ser executado se dará na inicialização da máquina, através de um gerenciador de inicialização. No Linux os principais são o Grub - *Grand Unified Bootloader* - e o Lilo - *Linux Loader*. Este está caindo em desuso.
- Maior aproveitamento do espaço, já que para cada sistema de arquivos existe uma granulometria mínima, ou seja, o espaço físico mínimo que um arquivo pode ocupar. Este espaço pode ser maior que o tamanho do arquivo.

- Maior segurança nos dados. Se possui uma única partição, ao ter-se problema com o sistema de arquivos perde-se todos os dados. Se possui-se várias partições provavelmente o problema se restringirá à uma partição, portanto perde-se somente os dados relativos à mesma.

O MBR (*Master Boot Record*) é uma parte especial do dispositivo. É um setor de boot de tamanho de 512 bytes, que é o primeiro setor de um dispositivo de armazenamento particionado. Ele pode ser utilizado para uma ou mais das seguintes funções: armazenar a tabela de partições, inicializar o sistema operacional após a BIOS (*Basic Input Output System*) repassar a execução para as instruções de código de máquina contidas no MBR e identificação única dos dispositivos de armazenamento.

Sistema de arquivos[1] é o modo como os dados são armazenados numa partição do dispositivo de armazenamento. Cada partição pode ter seu sistema de arquivos próprio, mas nunca mais de um. Já um disco com várias partições pode ter vários sistemas de arquivos. Para armazenar dados deve-se organizá-los e armazenar conjuntamente várias informações que permitam identificar estes dados, suas relações com os demais, a quem pertencem etc.

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 destes. Ainda, certamente, organiza os livros segundo suas características (assunto, autor 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 (Apêndice A) seria o bibliotecário da "biblioteca de dados" do computador: o dispositivo de armazenamento. Exatamente igual à organização de uma biblioteca, o `Sistema Operacional/Sistema de Arquivos` (ver Capítulo 4) guarda os dados nos espaços vazios do disco, rotulando-os com um FCB (*File*

[1] http://pt.wikipedia.org/wiki/Sistema_de_ficheiros

Control Block) e ainda criando uma lista com a posição deste dado, chamada de MFT (*Master File Table*).

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.

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 dispositivo de armazenamento. 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.

Quando um sistema de armazenamento e particionado com uma única partição e é formatado com um sistema de arquivos do Linux, o mesmo é dividido em 4 partes, Figura 4.1.

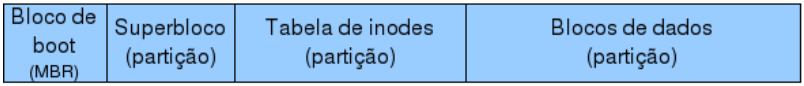

Figura 4.1: Estrutura de um sistema de arquivos genérico no LINUX

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 suporte 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, ext3 e ext4), 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. O ext4, lançado no final de 2008, amplia os limites de armazenamento para 64 bits, aproximadamente 18 Exa Bytes, além de melhorias de desempenho etc.
- **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.
- **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 nativamente 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.

4.2 Particionando e formatando discos

O Linux trata os discos, diferentemente do Windows, por nomes `hda`, `hdb`, `hdc` e `hdd` para discos 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 os nomes das partições seriam `hda1`, `hda5`, `hda6` etc.

Para particionar-se discos pode-se usar as ferramentas `fdisk`, `cfdisk` ou, no caso específico do Mandriva, o `diskdrake`.

Exemplo:

Para criar uma partição no espaço livre em disco da máquina executa-se a seguinte sequência de comandos:

1. `cfdisk /dev/sda` - abrirá uma janela conforme Listagem 4.1.

```
                cfdisk (util-linux-ng 2.14.2)

                        Disco: /dev/sda
          Tamanho: 160041885696 bytes, 160.0 GB
Cabecas: 255 Set. por Trilha: 63 Cilindros: 19457

 Nome Opcoes Tipo          Tipo SA          Tamanho (MB)
                Particao   [Rotulo]
 ------------------------------------------------------
 sda1 Inic.    Primaria    Linux ext3          41940,71
 sda5          Logica      Linux ext3          20003,89
 sda6          Logica      Linux swap/Solaris   1998,75
 sda7          Logica      Linux ext3          10001,95
```

```
                Pri/log   Espaco livre        29652,14
 sda3           Primaria  Linux ext3          56441,88

[ Iniciali .] [ Excluir ] [  Ajuda ] [ Maximize ]
[ Mostre ] [ Sair ] [ Tipo  ] [Unidades] [ Gravar ]

Alterna a opcao da particao atual como inicializavel
```

Listing 4.1: Tela exemplo do cfdisk

2. Selecionar, com as setas, `Espaço livre`.
3. Selecionar, no menu inferior, `Nova`.
4. Selecionar, no menu inferior, conforme desejado uma partição do tipo `Primária` ou `Lógica`.
5. Informar o tamanho desejado para a partição a ser criada.
6. Selecionar, no menu inferior, conforme desejado que a nova partição ocupe o `Início` ou `Fim` do espaço livre no disco.

Em seguida deve-se formatar a partição com o comando:

mkfs.ext3 /dev/hdaX

Onde X é o número da partição recém criada. Muito cuidado, pois se informado o número errado "detona-se" uma partição indevida. Se não lembrar do número use o `cfdisk` para ter certeza.

4.3 Montando partições

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

mkdir /dados

Tabela 4.1: Exemplo do arquivo /etc/fstab

Device	Moint point	File System	Options	Dump	Fsck
/dev/hda4	/	ext3	relatime	1	1
/dev/hda5	/home	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	/media/win	ntfs	noexec	0	0

mount /dev/hdaX /dados

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

O arquivo `/etc/fstab`[2] (*File System Table*) guarda as informações de montagens das partições desejadas. Este arquivo é lido em toda inicialização do *kernel* e define as partições a serem montadas automaticamente. O `fstab` também serve para montar outras partições com outros sistemas de arquivos, por exemplo sua partição Windows©, toda vez que ligar o computador. Também é usado para montagens dos dispositivos como cd-rom, dvd, *floppy-disk*, *pen-drive* etc.

Na Tabela 4.1 tem-se um exemplo de um arquivo `/etc/fstab`, que pode ser editado com o vi (Apêndice C).

O modelo de entrada no arquivo `fstab` é assim:

[dispositivo] [ponto de montagem] [sistema de

[2] http://www.linuxbsd.com.br/phpLinuxBSD/modules/artigos _tecnicos/fstab.htm

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: servidor.de.arquivos:/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 tem-se a pasta raiz do Gnu/linux, já na segunda linha a pasta /home que será montado no sistema.

sistema de arquivos – Nesse campo descreve-se qual o sistema de arquivo. Consulte `/proc/filesystems` para saber quais sistemas de arquivos são suportados pelo seu *kernel*.

opções – Abaixo tem-se algumas das opções disponíveis no `fstab` e seus significados:

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 montá-los mesmo que não existissem discos inseridos.

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

noexec – Essa opção é muito útil quando monta-se partições Windows, pois ele não gerencia os arquivos com permissões como no Gnu/Linux, com isso os arquivos ficam todos como executáveis, se 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 faz-se com que isso não ocorra.

uid – Essa opção é útil quando se monta partições FAT, pois elas não trabalham com permissões de arquivo, assim todas as partições que forem montadas com esta opção terão como dono dos arquivos o `root`. Assim um usuário normal não poderia ter total controle desse diretório.

gid – Com essa opção pode-se redefinir o grupo do diretório, na verdade é a mesma função da opção acima, só que faz isso com o grupo.

jean:x:522:700::/home/jean:/bin/false

No exemplo o gid do grupo é 700.

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 de cópias de segurança. 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 deve-se colocar a ordem em que os sistemas de arquivos serão verificados pelo `fsck` - *File*

System Check - 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 se quer 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.

4.4 A estrutura de diretórios

O GNU/Linux segue o *Filesystem Hierarchy Standard* para nomeação de arquivos e diretórios, ver detalhes nos Apêndices D e E. 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 os diretórios listados abaixo como padrão.

bin – Binários (executáveis) de comandos essenciais.

boot – Arquivos estáticos, gerenciador de inicialização e imagem do *kernel*.

dev – Arquivos associados a ponteiros para dispositivos.

etc – Arquivos de configuração dos sistemas e servidores instalados na máquina.

home – Diretórios de usuários.

lib – Bibliotecas essenciais compartilhadas e módulos do *kernel*.

media – Diretório para montagem de sistemas de arquivos temporários.

proc – Diretório virtual de informações de processos e *hardware* do sistema.

root – Diretório `home` do usuário `root`.

sbin – Binários essenciais do sistema do usuário `root`.

tmp – Arquivos temporários.

usr – *Unix System Resources* - A maioria dos aplicativos estão neste diretório.

var – Dados variáveis.

opt – Aplicativos adicionais e pacotes de softwares.

Capítulo 5

LVM - *Logical Volume Manager*

5.1 Introdução

O LVM tem por objetivo facilitar a administração do espaço em disco. Permite a ampliação ou redução da capacidade de armazenamento de uma determinada partição virtual, através do aumento ou diminuição do espaço ocupado pela mesma, dentro de um disco virtual.

Por outro lado o disco virtual pode ser expandido pela inserção de novas partições no mesmo, permitindo assim uma ampliação "ilimitada" do espaço de armazenamento. Esta inserção é feita com alguns comandos, é transparente aos usuários e são executadas com o sistema de arquivos completamente operacional, sem precisar de nenhuma interrupção de acesso aos mesmos. Cabe ressaltar que para inserção de um novo disco físico no sistema deve-se ter um *hardware* que comporte *hot-swap*, ou seja, os dispositivos podem ser conectados ou desconectados com o sistema ligado.

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

Figura 5.1.

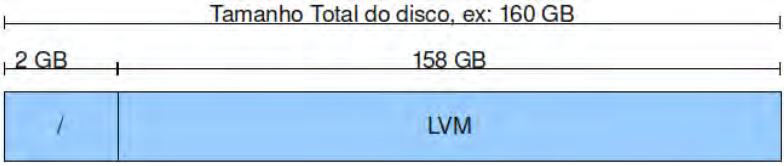

Figura 5.1: Modelo de particionamento com LVM

5.2 O que é o 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 o 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.

A camada LVM é composta pelas seguintes partes:

- **PV (*Physical Volume*)** - Os volumes físicos são as partições de discos alocadas para o LVM.
- **VG (*Volume Group*)** - É o grande disco virtual. Um conjunto (1 ou mais) de PVs forma um VG. Um conjunto de PVs podem ser necessários para criar *filesystems* maiores que a limitação física de um disco rígido. Pode-se ter mais de um VG num único sistema.

- **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).

- **LV (*Logical Volume*)** - É a partição virtual. Esse elemento é formado por um conjunto de LEs, na qual criamos o *filesystem*. Ao criar-se um volume lógico, recebe-se um *device* para referenciar, ao criar ou manipular, o sistema de arquivos. O nome do dispositivo é do tipo /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/.

Na Figura 5.2 pode-se observar um exemplo de uso do LVM, onde tem-se 3 discos rígidos. O primeiro e o terceiro discos contém partes utilizadas pelo o LVM e partições normais. O segundo disco é completamente ocupado pelo LVM. Supondo que o tamanho da PV_1, PV_2, PV_3 e PV_4 sejam respectivamente 158 GB, 120 GB, 60 GB e 40 GB, tem-se assim um VG totalizando 378 GB. Este será o "grande" disco rígido visto pelo sistema operacional. Este disco poderá ser particionado e formatado, com os procedimentos já vistos no Capítulo 4. Durante o uso do mesmo pode-se redimensionar estas partições, já que as mesmas encontram-se alocadas nesta unidade virtual de armazenamento.

Num sistema pode-se ter vários VGs independentes. Cada um deles será interpretado com um disco físico pelo sistema operacional e, portanto, seus limites não poderão ser ultrapassados.

A qualquer momento pode-se adicionar novos PVs a um determinado VG. Isto é muito interessante para o caso de um sistema estar chegando ao limite de sua capacidade de armazenamento já que, com o LVM, pode-se acrescentar um novo disco rígido, criar uma ou mais PVS e incrementar o VG. Assim sendo o VG, que é o disco virtual,

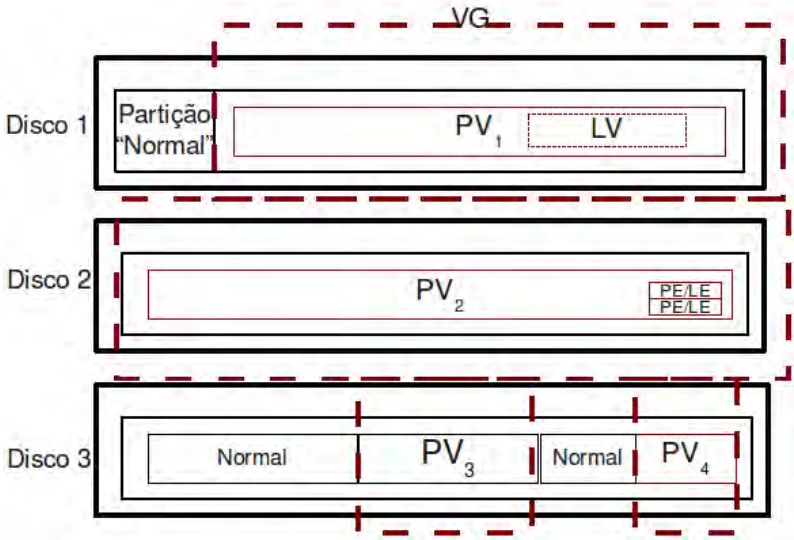

Figura 5.2: Como é a camada LVM

terá sua capacidade de armazenamento aumentada e pode-se ampliar as capacidades de armazenamento de uma ou mais LV, que são as partições virtuais, sem a necessidade de parada do sistema.

5.3 Implantando LVM

Se o LVM não foi instalado juntamente com o sistema pode-se instalar/configurar o mesmo posteriormente. Como primeira ação deve-se instalar os pacotes necessários para gerenciamento e criação de partições LVM. Para isto usa-se o comando:

urpmi lvm2

No exemplo abaixo cria-se um grupo de volumes de nome `vgteste`, numa partição já existente na máquina. Primeiro "inicializa-se"o LVM com o comando:

vgscan

Cria-se um volume físico - PV - para cada partição que deseja-se alocar ao LVM. O comando abaixo deve ser executado um para cada partição. No caso da Figura 5.2 teria-se 4 PVs: PV1, PV2, PV3 e PV4.

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

Inclui-se todos os volumes físicos – PV – no grupo de volumes – VG – com o comando:

vgcreate vgteste /dev/hdaX /dev/hdbX ...

Atualiza-se o LVM com o comando:

vgscan

Ativa-se o grupo de volumes – VG – com o comando:

vgchange -a y

Verifica-se a criação com o comando:

vgdisplay -v vgteste

Dentro do grupo de volumes – VG – cria-se, por exemplo, dois volumes lógicos `lv1` de 300 MB e `lv2` de 500 MB com os comandos:

lvcreate -L 300M -n lv1 vgteste

lvcreate -L 500M -n lv2 vgteste

Ative os volumes lógicos com os comandos:

lvchange -a y /dev/vgteste/lv1

lvchange -a y /dev/vgteste/lv2

Formatam-se as partições lógicas criadas com os comandos abaixo, ou de acordo com o sistema de arquivos usado: `mkfs.xfs`, `mkfs.ext3` e `mkfs.reiserfs`.

mkfs.ext4 /dev/vgteste/lv1

mkfs.ext4 /dev/vgteste/lv2

Criam-se os diretórios, onde serão montadas as partições, com o comando:

mkdir /dados /backup

Montam-se as partições com os comandos:

mount /dev/vgteste/lv1 /dados

mount /dev/vgteste/lv2 /backup

Verificam-se as partições montadas com o comando:

df -h

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

5.4 Modificando o tamanho de partições LVM

Para aumentar o tamanho de uma partição é preciso primeiro verificar a disponibilidade de espaço no `volume group` - VG. O espaço disponível pode ser visto com o comando:

vgdisplay

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

lvdisplay

No exemplo abaixo aumenta-se o volume lógico lv1 em 100 MB:

lvextend -L +100M /dev/vgteste/lv1

Procuram-se e reparam-se erros do volume lógico:

e2fsck -f /dev/vgteste/lv1

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

Sistemas de arquivos **XFS**:

xfs_growfs /dados

Sistemas de arquivos **ReiserFS**:

resize_reiserfs -f /dev/vgteste/lv1

Sistemas de arquivos **EXT3** ou **EXT4**:

resize2fs /dev/vgteste/lv1

A redução de tamanho de um volume lógico – LV – deve ser feita com muito cuidado, já que pode-se incorrer no erro de diminuir para um tamanho aquém do já ocupado pelos dados gravados. Pode-se ter problemas de o "simples" não funcionamento ou, o que é pior, de perda de dados. Neste caso deve-se ter em mente que o caminho a ser percorrido é o inverso do caminho para aumento. Esta operação deve ser feita com o sistema de arquivos desmontado, portanto, para o exemplo do caso `/dados`:

umount /dados

Em seguida procuram-se e reparam-se eventuais erros do volume lógico:

e2fsck -f /dev/vgteste/lv1

Diminui-se o tamanho do sistema de arquivos. No caso do sistema de arquivos `ReiserFS` é possível a especificação da quantidade a diminuir, como no exemplo abaixo com a *flag* `-s`, ou pode-se especificar o novo tamnho desejado. Já para o caso do `EXT3` e `EXT4`, deve-se especificar o novo tamanho total desejado. Já o `XFS` não permite diminuir o tamanho.

resize_reiserfs -s -50M /dev/vgteste/lv1 50M

resize2fs /dev/vgteste/lv1 200M

E por último reduz-se o tamnho do volume lógico – LV. Este tamanho deve ser compatível (o mesmo) com o especificado no sistema de arquivos. Para o exemplo abaixo está-se reduzindo o tamanho total em 50 MB.

lvreduce -L -50M /dev/vgteste/lv1

Agora já é possível a montagem e uso da partição com seu novo tamanho:

mount /dev/vgteste/lv1 /dados

df -h

Capítulo 6

Administração de Usuários e Grupos

6.1 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 ao usuários e o que ele pode fazer depois de obter a permissão de acesso.

Cada pessoa que queira acessar o sistema deverá ter uma conta de usuário que terá permissões exclusivas a determinados recursos.

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.

Normalmente os grupos são um conjunto de usuários com alguma afinidade, por exemplo que trabalham no mesmo setor ou departamento. Os grupos compartilham recursos, como por exemplo: pastas, impressoras etc. Na criação de grupos o administrador deverá ter em mente a estrutura organizacional da empresa para criá-los de maneira pertinente.

Um usuário deve pertencer obrigatoriamente a pelo menos um grupo e este é seu grupo primário. Outros grupos podem ser associados ao usuário e serão seus grupos secundários. Ao criar um arquivo este será associado a um usuário, o criador, e a um grupo, o grupo primário do usuário criador. No Apêndice F existe uma relação de alguns comandos que podem ser úteis.

6.2 Criação de contas de usuários e grupos

Para criar um grupo usa-se o comando:

groupadd nome_do_grupo

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

adduser login

Pode-se ainda "sofisticar" a criação de contas com algumas flags, as principais são apresentadas a seguir.

-d – Pode-se informar qual será o diretório *home* do usuário.

-g – Pode-se 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 – Define o(s) grupo(s) suplementar(s) ao(s) qual(is) o usuário pertencerá.

-c – Normalmente o nome completo do usuário, endereço etc. Campos separados por vírgulas.

-s – Esta opção é interessante se desejado, por exemplo, que um usuário não acesse a máquina com sua conta. Para isto informa-se `/bin/false`.

Em seguida define-se a senha com o comando:

passwd login

6.3 Parâmetros das contas

As contas de usuários e grupos ficam armazenadas nos seguintes arquivos `/etc/passwd`, `/etc/group` e `/etc/shadow`.

O arquivo `/etc/passwd` armazena informações de todos os usuários cadastrados no sistema. Por padrão contas com UID e GID abaixo de 500 são contas do sistema, normalmente pre-definidas, ou criadas no momento de instalação de algum serviço. Este arquivo deve ter permissão de leitura para todos os usuários. A separação dos campos é feita pelo caractere ':' segundo o modelo:

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

A descrição dos respectivos campos são:

login – O login do usuário.

x – Este campo continha a senha. Nos sistemas modernos esta passou para o arquivo `shadow`, descrito abaixo, por questões de segurança, já que este arquivo é de acesso geral.

503 – UID, *User Identification*, número único de identificação do usuário.

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

comentário – Pode conter nome completo do usuário, endereço etc.entre aspas simples e campos separados por vírgulas.

home/login – Define o diretório `home` do usuário.

/bin/bash – O `shell` do usuário.

O arquivo `/etc/group` tem uma relação dos grupos do sistema. Este arquivo deve ter permissão de leitura para todos os usuários. Segue o modelo:

nome_do_grupo:**senha**:**500**:**lista_de_usuários**

A descrição dos respectivos campos são:

nome_do_grupo – O nome do grupo.

senha – A senha (criptografada) do grupo. Se este campo estiver vazio, nenhuma senha está atribuída ao grupo. Para atribuir uma senha a um grupo usamos o comando `gpasswd`.

500 – O identificador numérico do grupo.

lista_de_usuários – Login de todos os membros suplementares do grupo, separados por vírgulas. Usuários primários não são visíveis neste arquivo.

O arquivo `/etc/shadow` contém as senhas dos usuários. Este arquivo é de uso esclusivo do usuário `root` e não é acessível a usuários comuns. Sendo assim, nenhum usuário consegue copiar este arquivo para uma possível tentativa de quebra de senha. Segue o modelo:

login:**1/**
clCUNfk$9UE29qRp5oJi0:**13581**:**0**:**99999**:**7**:**2**::

A descrição dos respectivos campos são:

login – nome de usuário.

1/clCUNfk$9UE29qRp5oJi0 – Senha criptografada.

13581 – Número de dias, desde 01/01/1970, em que a senha sofreu a última alteração.

0 – O prazo, em dias, em que a senha não pode ser alterada.

99999 – O prazo, em dias, em que a senha deverá ser alterada.

7 – A partir de quantos dias antes da expiração da senha o usuário receberá aviso para alterá-la.

2 – Quantos dias após a expiração da senha a conta será desabilitada.

em branco – Número de dias, desde 01/01/1970, que a conta está desabilitada. Se este campo estiver vazio significa que a conta está ativa.

em branco – Campo reservado.

6.4 Arquivos de definições para contas

O arquivo `/etc/login.defs` contém uma série de diretivas e padrões que serão utilizados na criação das contas de usuários. Vale ressaltar que as alterações neste arquivo terão efeito a partir das próximas contas criadas, ou seja, não tem valor sobre contas já existentes. Seu principal conteúdo é apresentado na Tabela 6.1.

O arquivo `/etc/default/useradd` também contém padrões para criação de contas. Seu principal conteúdo é apresentado na Tabela 6.2.

6.5 Alterando parâmetros das contas

Para modificar uma conta já existente pode-se usar o comando

usermod opções login

Onde as principais opções são apresentadas a seguir.

-c – Comentário: nome completo, endereço, telefone etc.

-d – Diretório *home*.

-e – Data de expiração. Data, na forma AAAA-MM-DD (ano-mês-dia), que a conta será desativada.

Tabela 6.1: Principais diretivas do arquivo `/etc/login.defs`

Diretiva	**Valor padrão**	**Comentário**
MAIL_DIR	/var/spool/mail	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

-g – Grupo primário.

-G – Grupo(s) suplementar(es).

-l – Novo login.

-L – Trava a senha e portanto a conta de usuário.

-s – `Shell` do usuário.

6.6 Removendo contas

Para remover um grupo usamos o comando:

groupdel nome_do_grupo

Para remover contas de usuário usamos o comando:

Tabela 6.2: Principais diretivas do arquivo `/etc/default/useradd`

Diretiva=	**Valor padrão**	**Comentário**
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. Ano/mes/dia
SHEL=	/bin/bash	Shell atribuído ao usuário
SKEL=	/etc/skel	Arquivos e diretórios padrão para os novos usuários

userdel login

A principal opção é:
-r apaga o diretório `home` do usuário e todo seu conteúdo.

Capítulo 7

Permissão de Acesso à Arquivos

7.1 Introdução

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.

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.

7.2 Permissões de acesso

O conjunto de permissões é dividido em três classes: proprietário, grupo e usuários. Assim, pode-se dar permissões de acesso diferentes para cada uma destas três classes.

Quando executa-se *ls -l* em um diretório qualquer, os arquivos são exibidos de maneira semelhante ao seguinte:

1	2	3	4	5	6	7	8	9
drwxr-xr-x	4	mar	adm	4096	Abr	2	14:48	BrOffice/
-rw-r- -r- -	1	ze	adm	11188	Out	31	21:28	bro.bz2
-rw-r- -r- -	1	root	root	112614	Dez	27	14:47	bro.rpm
-rw-r- -r- -	1	ze	prof	900186	Out	31	22:04	BrOo.exe
-rw-r- -r- -	1	root	root	917615	Jan	5	21:27	BrOo2.exe

As colunas que aparecem na listagem são:

1. Esquema de permissões (`-rw-r--r--`).
2. Número de ligações do arquivo ou quantos outros arquivos estão associados a este.
3. Nome do usuário dono do arquivo.
4. Nome do grupo dono do arquivo.
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 cadastrados no sistema (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 pode-se atribuir um conjunto diferente de permissões de acesso. Por exemplo, atribuir permissão de acesso irrestrito (de leitura, escrita e execução) para o dono do arquivo, apenas de leitura para usuários que estão no mesmo grupo do dono, e nenhum acesso aos demais usuários.

Deve-se aplicar a permissão de execução somente a arquivos que podem ser executados, como programas já compilados ou *script shell*, caso contrário ocorrerão erros de execução por parte do *shell*.

Para caso de diretórios a permissão de somente leitura implica na possibilidade de um usuário, que se enquadra nesta permissão, visualizar o conteúdo do mesmo através de um comando do tipo `ls`, mas não permite que este usuário "entre" no mesmo. Já a permissão de somente execução, permite que usuário "entre" no diretório, mas sequer consiga listar seu conteúdo, mesmo estando "dentro" do mesmo.

Os valores válidos mais comuns para cada uma das colunas são os seguintes:

Coluna	**valor**	**atributo**	**valor**	**atributo**
1	d	se o arquivo for um diretório	-	se for um arquivo comum
2,5,8	r	se existe permissão de leitura	-	se **não** existe permissão de leitura
3,6,9	w	se existe permissão de alteração	-	se **não** existe permissão de alteração
4,7,10	x	se existe permissão de execução	-	se **não** existe permissão de execução

O conjunto de permissões pode ser encarado, em resumo, como as permissões para leitura, escrita ou execução de seu conteúdo. O conteúdo de um arquivo comum é, por exemplo, seu texto e, portanto, algum usuário que tem permissão de escrita sobre este arquivo, pode

inclusive apagar todo o seu conteúdo mas talvez não possa apagar o próprio arquivo, já que este pertence a um diretório (tipo especial de arquivo). Portanto as permissões de um diretório podem afetar a disposição final das permissões de um arquivo. Outro 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 que vai remove-lo.

7.3 Alterando a permissão de acesso

Para alterar a permissão de acesso a um determinado arquivo usa-se o comando:

chmod `modo-de-permissão` *arquivo*

O `modo-de-permissão` na linha de comando pode ser informada 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 classe de usuários. 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`, muda-se o modo de permissão do arquivo `arq1` para que o proprietário tenha acesso total e todos os outros usuários (grupo e outros) tenham apenas permissão de leitura e execução:

chmod 755 arq1 7=rwx (4+2+1); 5=r-x (4+1)

ls -l arq1

`-rwxr-xr-x` 1 guest users 1475 May 20 11:02 arq1

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 dono do arquivo.
  g ⇒ Grupo dono do arquivo.
  o ⇒ Outros ou demais usuários do sistema.

- Ação significa como serão alteradas as permissões em relação ao estado atual.

  \+ ⇒ Acrescenta permissão(ões).
  \- ⇒ Remove permissão(ões).
  = ⇒ Atribui a permissão explicitamente. O resultado será exatamente o indicado, não levando em consideração o estado passado.

- Tipo de permissão

  r ⇒ Leitura (***r****ead*).
  w ⇒ Gravação (***w****rite*).
  x ⇒ Execução (*e****x****ecute*).

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

Exemplos :

Tira-se 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

Mudam-se 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

7.3.3 Mudando as permissões padrão

Para modificar o conjunto de permissões que serão aplicadas a todos os novos arquivos a serem criados, deve-se usar o comando:

umask [permissão]

Modifica a permissão padrão para os novos arquivos a serem criados. No comando, número é um número octal de três dígitos, como visto no comando *chmod*. Com o comando *umask* ao se especificar número igual a 777, se estará negando acesso a todas as classes em qualquer modo, seria equivalente a aplicar um *chmod 000*. 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 para diretórios a permissão padrão é 777 (`rwxrwx rwx`), ao atribuir-se um valor de umask 023 e criar-se novos diretórios tem-se as permissões 754 (`rwxr-xr--`). Já para arquivos a permissão padrão é 666 (`rw-rw-rw-`), ao atribuir-se um valor de umask 022 e criar-se novos arquivos tem-se as permissões 644 (`rw-r--r--`).

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

Se deseja-se modificar em definitivo o valor de *umask* deve-se editar o arquivo ∼`.profile` e redefinir o campo apropriado.

7.3.4 Modificando o grupo dono de um arquivo

chgrp grupo arquivo

O comando *chgrp* muda a identificação do grupo dono de um arquivo. Pode ser utilizado para conceder permissão de leitura e escrita para outro grupo que não o do seu usuário, sem ter que conceder as mesmas permissões para todos os demais usuários. Pode-se mudar somente grupos de arquivos sendo o usuário proprietário do mesmo ou sendo o usuário `root`.

No exemplo a seguir muda-se o grupo dono do arquivo `arq1` para `users2`.

ls -l arq1

`-rw-r--r--` 1 guest users 984 May 12 11:02 arq1

chgrp users2 arq1

ls -l arq1

`-rw-r--r--` 1 guest users2 984 May 12 11:02 arq1

7.3.5 Dono de um arquivo

chown usuário arquivo

Usado para mudar a identificação de proprietário associada a um arquivo. Pode-se mudar somente o dono de arquivos sendo o usuário proprietário do mesmo ou sendo o usuário `root`.

No exemplo a seguir muda-se o proprietário do arquivo `arq1` para `guest2`.

ls -l arq1

`-rw-r-xr--` 1 guest users 1765 May 17 13:34 arq1

chown guest2 arq1

ls-l arq1

`-rw-r-xr--` 1 guest2 users 1765 May 17 13:34 arq1

Capítulo 8

Cotas em disco

8.1 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 sendo equânime com os usuários e grupos, antecipando, por exemplo, travamentos por partições lotadas.

A implementação do sistema de cotas no Linux se dá por partições, ou seja, em todas as partições onde deseja-se um efetivo controle deve-se configurar o sistema de cotas. O controle se dá dentro de determinada partição, onde todos os arquivos pertencentes a determinado usuário/grupo serão contabilizados em sua cota, independente do diretório onde está contido.

As cotas são determinadas por usuário e/ou grupo, sendo que pode-se 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.

Tabela 8.1: Arquivo `/etc/fstab`, sem e com sistema de cotas

Arquivo original					
/dev/hda5	/	ext3	relatime	1	1
/dev/hda7	/home	ext3	relatime	1	2
/dev/hda9	/dados	ext3	relatime	1	2
/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/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

8.2 Instalando o Sistema de Cotas

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

urpmi quota

Em seguida deve-se informar ao sistema de arquivos em qual(is) partição(ões) pretende-se implantar o sistema de cotas. Para isto deve-se editar o arquivo `/etc/fstab` e inserir ao final da quarta coluna, separado por vírgula, a diretiva `usrquota` e/ou `grpquota`, dependendo se deseja-se quotas para usuários, grupos ou ambas. No exemplo da Tabela 8.1 habilitam-se cotas para usuários na partição `/home` e para usuários e grupos na partição `/dados`. Na primeira parte tem-se o arquivo original e na segunda parte o mesmo arquivo com a inserção do sistema de cotas em disco.

Após isto deve-se desmontar e montar a(s) partição(ões) onde se está implantando as cotas, para que o gerenciador de sistema de arquivos do *kernel* releia o arquivo `fstab` e remonte segundo os novos

parâmetros. Para isto deve-se: ou reinicializar a máquina ou usar a sequência de comandos abaixo.

Ao utilizar um ambiente gráfico deve-se sair do mesmo, já que neste ambiente existem vários arquivos temporários abertos e o Linux não permite o desmonte de alguma partição com algum arquivo aberto. Pode-se sair do modo gráfico e entrar num modo texto puro, por exemplo com o comando:

init 3

Loga-se como usuário `root`. E então desmonta-se e monta-se as partições onde se está implantando cotas. Aqui cabe a observação que ao estabelecer cotas na partição / (raiz) o desmonte/monte somente será possível com a reinicialização da máquina, já que não é possível o desmonte desta partição com o sistema em uso.

umount /home /dados

mount /home /dados

Inicializa-se o sistema de cotas com o comando:

quotacheck -augv

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 usados pelos mesmos.

Em seguida deve-se ativar o sistema de cotas com o comando:

quotaon -augv

Poder-se-ia desativar, caso desejado, o sistema de cotas com o comando:

quotaoff -augv

Se desejado pode-se retornar ao modo gráfico a partir deste ponto.

8.3 Manipulando Cotas

8.3.1 Cotas para Usuários

Para o estabelecimento de cotas para usuários ou grupos utiliza-se o comando:

edquota user

Este comando abrirá um editor de texto vi, ver Apêndice C, habilitando a edição de cotas para o usuário. O formato do arquivo é apresentado a seguir. Na primeira linha tem-se a informação do usuário, seu `login` e seu `UID` (*User Identification*). As demais linhas são divididas em colunas com os significados: `Filesystem` informa qual partição tem o sistema de cotas habilitada; `blocks` informa o espaço em disco já em uso pelo referido usuário; `soft` a cota do espaço em disco; `hard` o limite máximo de espaço em disco ocupado pelo usuário; `inodes` informa a quantidade de arquivos e diretórios em nome do usuário; `soft` e `hard` informa as cotas para a quantidade de arquivos e diretórios em nome do usuário.

O limite entre `soft` e `hard` poderá ser usado por até uma semana, após a qual o usuário não conseguirá salvar ou modificar mais nenhum arquivo, só conseguirá apagar arquivos até que fique abaixo do valor de `soft`. 0 (zero) significa que não há limites.

8.3.2 Cotas para Grupos

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

Disk quotas for user `user` (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

8.3.3 Múltiplas Cotas Simultâneamente

Estabelecendo cotas para vários usuários e/ou grupos

No caso que tenhamos que implantar cotas para vários usuários e/ou grupos não é viável ficar 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 user

8.3.4 Verificando Cotas

Pode-se verificar a cota individual de algum usuário ou grupo [-g] com o comando:

quota [-g] nome_do_grupo

Para verificar a cota de todos os usuários ou grupos [g] usa-se o comando:

repquota -a[g]

Capítulo 9

Crontab

9.1 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 numa data exata. Um uso bem comum do `cron` é o agendamento de tarefas administrativas de manutenção do sistema, como por exemplo, análise de segurança do sistema, *backup*, atualizações etc. Estas tarefas são programadas para serem automaticamente executadas através da `crontab` e um `script shell` comum, todo dia, toda semana ou todo mês. 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 restrito apenas ao que o usuário tem permissão de executar (e não tudo, como é o caso do `root`).

Tabela 9.1: Campos editáveis do Crontab

Minuto	Hora	Dia do mês	Mês	Dia da semana	Programa a ser executado
0-59	0-23	1-31	1-12	0-6 (0 é o domingo)	Nome do programa

9.2 Uso do Crontab

Primeiramente deve-se 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 a relação de possibilidades.

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 o desejado é verificar os arquivos `crontab` dos usuários, deve-se ser `root`. O `crontab` armazena os arquivos dos usuários no diretório: `/var/spool/cron/nome_de_usuário`. Onde `nome_de_usuário` corresponde ao usuário dono do arquivo `crontab`. Por outro lado, se o desejado é editar o `crontab` global, este fica no arquivo `/etc/crontab`, e só pode ser manipulado pelo `root`.

Cada tarefa agendada é configurada numa linha. O formato da linha do `crontab` é que vai dizer o que executar e quando. A linha é dividida em 6 campos separados por `tabs` ou `espaço`. Na Tabela 9.1 são apresentados os campos e suas respectivas faixas de valores aceitos.

A especificação dos campos numéricos podem ser dadas por `valor absoluto`, por `intevalo`, por `peridiocidade` ou pela

combinação destes. No valor absoluto explicita-se o valor exato do campo, se desejado pode-se explicitar mais de um valor separando-os por ‘,’. No intervalo fornece-se uma faixa dentre os possíveis valores, deve-se separar o valor inicial e o final por ‘–’, neste caso os eventos ocorrerão em todos os valores ‘cheio’. Na periodicidade deve-se especificar o valor absoluto ou intervalo seguido do período, separado pelo caractere ‘/’. O caractere especial ‘*’ significa todos os valores do respectivo campo.

Alguns exemplos:

Executar o comando *ls -l*, salvando seu conteúdo em `/tmp/lixo.txt` (ver redirecionamentos em G), toda hora (*), todo dia (*), nos minutos 1, 21 e 41

```
1,21,41 * * * * ls -l »/tmp/lixo.txt
```

Apagar o arquivo `/tmp/lixo.txt` toda segunda-feira, de hora em hora aos 30 min, no intervalo das 4 às 10h da manhã.

```
30 4-10 * * 1 rm -f /tmp/lixo.txt
```

Executar o comando *backup* com periodicidade de 5 em 5 dias às 19h45min.

```
45 19 */5 * * /usr/local/bin/backup
```

Executar o *script* entre as 19 e 23 de 2 em duas horas.

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

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 onde executa-se o *mrtg* como usuário `root` de 5 em 5 minutos:

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

Capítulo 10

Backups e Políticas

10.1 Introdução

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

A cópia de segurança ou *backup* consiste de uma ou várias cópias dos dados, que permite sua recuperação em caso de acidente. Objetiva assegurar a integridade dos dados contra possíveis problemas com os discos de armazenamento ou apagamento, seja acidental ou intencional.

Existem várias formas de se garantir a disponibilidade da informação, a mais importante sem dúvidas é a cópia dos dados em local seguro, pois traz flexibilidade à instituição de, a qualquer momento, recuperar dados de uma data qualquer passada.

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 se pensa que o local seguro possa ser a torre do prédio ao lado, o que nem sempre é verdade. Para dados com necessidade extrema de recuperabilidade deve-se providenciar várias cópias e estas devem ser armazenadas em prédios e/ou cidades distintas. Nos dias atuais, com as facilidades de enlaces confiáveis e velozes, pode-se providenciar estas cópias por meio da rede, permitindo assim um

espalhamento físico dos dados de uma maneira rápida e segura.

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 se tem absoluta necessidade do *backup* desses dados. Isso leva a questão de política de *backup* e formas de armazenamento, com questão de custo X segurança. Neste caso tem-se num extremo a confiabilidade e no outro o baixo custo. A escolha de uma boa política aplicada a uma forma de armazenamento suficientemente adequada a situação pode trazer à empresa um custo compatível com o valor da informação que ela deseja salva-guardar.

10.2 Tipos de *backup*

O tipo de *backup* a ser utilizado varia de acordo com a necessidade de 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.

10.2.1 *Backups* totais

Um *backup* total captura todos os dados, incluindo arquivos de todas as unidades de disco rígido. Uma fita, ou conjunto de fitas, atualizada de backup total pode ser usada para restaurar completamente um servidor em um determinado momento.

Como vantagens deste método tem-se a cópia total dos dados, significando que existe uma cópia completa de todos os dados num mesmo dispositivo. O rápido acesso aos dados de *backup*, já que não é necessário a pesquisa em várias fitas para localizar o arquivo que deseja-se restaurar, já que os *backups* totais incluem todos os dados contidos nos discos rígidos em um determinado momento.

As desvantagens são os dados redundantes, já que vários *backups* totais vão manter a cópia de todos os dados, alterados ou não, em fitas ou conjuntos distintos. *Backups* totais podem ser muito demorados. Se a quantidade de dados a ser salva é muito grande. Há um desperdício de fitas, devido a redundância dos dados.

10.2.2 *Backups* incrementais

Backup incremental captura todos os dados que foram alterados desde o último *backup* total ou incremental. Deve-se usar uma fita de *backup* total (não importa há quanto tempo ela tenha sido criada) e todos os conjuntos de backups incrementais subsequentes para restaurar um servidor.

Como vantagens pode-se citar o uso eficiente do tempo, já que o processo de *backup* incremental 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*.

Como desvantagens tem-se que a restauração completa fica mais complexa, já que pode-se precisar de um conjunto de várias fitas para obter uma restauração completa do sistema. Restaurações parciais demoradas, já que pode-se ter que pesquisar em várias fitas para localizar os dados necessários para uma restauração parcial.

10.2.3 *Backups* diferenciais

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

Como vantagem pode-se citar a rápida restauração, já que há menos fitas envolvidas. Uma restauração completa exige no máximo duas fitas ou conjuntos: a fita do último *backup* total mais a do último *backup* diferencial.

Como desvantagens tem-se *backups* mais demorados e maiores, em relação aos backups incrementais, porque quanto mais tempo tiver se passado desde o último *backup* total, possivelmente mais dados terão sido modificados e portanto mais dados devem ser copiados para a fita diferencial.

10.3 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.

10.3.1 *Backups online*

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*.

A vantagem é que o servidor vai estar sempre disponível, podendo ser realizado o *backup* durante o expediente normal de trabalho.

A desvantagem está em relação ao desempenho do servidor que é prejudicado durante a realização do *backup*, já que serão consumidos recursos do sistema para fazê-lo.

10.3.2 *Backups offline*

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

Como vantagem tem-se a rapidez do *backup*, já que o servidor estará fazendo apenas, possivelmente, o *backup* dos dados.

10.4 Armazenamento

Unidades de armazenamento - fitas - e seus dispositivos controladores guardam uma relação direta entre a qualidade e tamanho com seu custo, trazendo muitas vezes a solução para o nível considerado bom e não ótimo. Quando fala-se em unidades de armazenamento também é necessário pensar em local para armazená-las.

É de extrema importância que, seja qual for a tecnologia utilizada, esteja-se seguro de que suas fitas estão sendo corretamente armazenadas. As fitas não podem ser simplesmente guardadas em um caixa de papelão localizada sobre o monitor do servidor ou sequer 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 tem-se dispositivos para armazenamento em fitas magnéticas, discos rígidos e discos ópticos. Dentro de cada categoria existem prós e contras.

10.4.1 Discos Rígidos

Os discos rígidos são uma ótima unidade no ponto de vista do custo por quantidade de informação armazenada. Esta solução tem sido utilizada para realização do '*backup* incremental para sempre' (RAID - *Redundant Array of Independent Drives*). Possui alta capacidade de armazenamento e boa velocidade de acesso. Esta solução não é confiável como única forma de *backup*, devido a sua grande sensibilidade a choques mecânicos, baixa durabilidade e dificuldade na troca de dispositivos em caso de estouro de capacidade.

10.4.2 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. Atualmente encontram-se modelos com capacidade de até 1,6 TB para gravação de dados compactados.

10.4.3 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.

10.5 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, 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, por quanto tempo e onde será armazenado.

Agora se está entrando no maior ponto de problemas relacionado a *backup*, o que salvar e o que não salvar. Quando entra-se nesta discussão fica claro que o melhor, por garantia, é aquela 'vamos salvar tudo', mas o que é 'tudo'? Pode-se salvar 'tudo'? Ótimo! Cabe a uma política definir o 'tudo', informando caminhos completos destes arquivos, por quanto tempo e onde serão armazenados, só então entra o administrador, que é o responsável pela operacionalização dos procedimentos descritos. É importante que haja completa descrição para que toda a empresa possa saber onde deixar seus arquivos, seja na rede ou numa estação específica, tendo a certeza de que neste local as informações estão seguras contra perdas ou apagamentos.

Devido a limitações técnicas, é impossível que se faça a cópia de segurança dos dados a cada segundo. O mesmo fala-se com relação à restauração. E ainda, seja por falta de espaço físico ou limitação financeira, nem todas estas cópias poderão ser mantidas *ad aeternum*.

A tarefa, quando falamos em política de *backup*, é definir os seguintes pontos:

Quando esta cópia será feita – Neste ponto, precisa-se 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 do que é tecnicamente possível realizar ou não.

Quanto tempo demora esta cópia (*Backup Window*) – É necessário saber quanto tempo leva para o *backup* ser concluído, para que se possa estimar com precisão a periodicidade máxima de *backup*.

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 para efetuar a restauração dos dados do *backup*, já que isto pode interferir no tipo de tecnologia adotada.

Por quanto tempo uma cópia estará disponível – Seja qual for o local para armazenar as fitas do *backup*, sabe-se que haverá limitações, seja de espaço físico, lógico ou realmente de quantidade de unidades de armazenamento. Isto 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 tem-se 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. Salienta-se 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 podem não ser o de outra, e principalmente, o tempo de retenção da cópia de dados.

10.6 O sistema Amanda

O `Amanda`[1] (*Advanced Maryland Automatic Network Disk Archiver*) é um sistema de *backup* cliente/servidor. Ele permite a programação das políticas de backup de uma maneira simples, econômica e segura.

Um servidor `Amanda` irá realizar numa única unidade controladora de fita, o *backup* de qualquer número de computadores que tenham o cliente `Amanda` e uma conexão em rede com o servidor.

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' que são um grupo de fitas utilizadas sobre o tempo para criar os *backups* completos de todos os sistemas de arquivos (diretórios) 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.

O arquivo de configurações provê um controle total da realização dos *backups* e do tráfego de rede que o `Amanda` gera. O `Amanda`

[1] http://www.amanda.org/

utilizará qualquer programa de *backup* para gravar os dados nas fitas, por exemplo *tar*.

O `Amanda` faz controle total das fitas rotulando-as. Sendo assim ele não permite a sobrescrição de fitas indevidamente e controla a exata sequência dos *backups* de acordo com a política adotada.

O `Amanda` está disponível como pacote, porém ele não é instalado por padrão.

O `Amanda` permite políticas de *backup* bem complexas. Isto é conseguido criando-se vários esquemas de configurações, cada um contendo uma parte da política que comporá o todo. Ao ser instalado o `Amanda` cria um conjunto diário, que é definido dentro do diretório `/etc/amanda/DailySet1/`. Caso deseje-se mais esquemas deve-se fazer cópias, de todo o conteúdo deste diretório, com os nomes desejados. Por exemplo semanal, mensal, anual etc. Dentro de cada um dos diretórios pode-se configurar o que, de onde, e por quanto tempo salvar.

Para restaurar um sistema de arquivos é necessário o *backup* completo mais recente e os incrementais. Este controle é feito pelo próprio `Amanda` que, na necessidade de um restauro, requisitará ao administrador as fitas necessárias e sua ordem.

10.6.1 Instalação e configuração

Para instalar o `Amanda` basta digitarmos o comando:

urpmi -a amanda

Após a instalação o diretório padrão `/etc/amanda/DailySet1/`, estará criado.

Como exemplo vai-se simplesmente configurar um *backup* diário mas, como já dito, pode-se ter vários esquemas estabelecidos, em diretórios distintos. Neste diretório estão contidos os arquivos de configuração do respectivo esquema. Abaixo tem-se os principais parâmetros de cada arquivo.

No `amanda.conf` os principais parâmetros a serem configurados são o *org*, onde deve-se informar um nome sugestivo para o esquema. O *mailto* que define o endereço eletrônico do responsável pelo *backup*. O *tapecycle* que define a quantidade de fitas no *backup*. O *tapedev* que define o dispositivo de fita. O *tapetype* que define o tipo de fita usado, o `Amanda` tem suporte nativo para vários dispositivos de fita, caso necessário pode-se buscar no sítio do `Amanda` atualizações para novos modelos de dispositivos de fita. O *labelstr* é o rótulo que será dado à cada fita, no exemplo: diario01 à diario99. O *dumpcycle* define o número de dias entre os *backups* totais. O contêiner *holdingdisk* traz as informações do disco local auxiliar, onde serão feitas as cópias temporárias. No exemplo isto é feito no disco *hd1*, no diretório */amanda* com tamanho máximo de 21 GB. Recomenda-se que o tamanho deste diretório deve ser maior ou igual a capacidade de armazenamento da fita em uso.

---

org “diario”

mailto “administrador@domínio.com.br”

tapecycle 21 tapes

tapedev “/dev/nst0”

tapetype DDS4

labelstr “diario[0-9][0-9]*”

dumpcycle 4 weeks

holdingdisk hd1 {

comment “main holding disk”

directory “/amanda”

use 21 Gb

}

O arquivo `disklist` contém a listagem de todos os diretórios que devem ser mantidos em *backups*, seja do computador local ou

computadores remotos. Evidentemente estes deverão ser acessíveis na rede pelo nome indicado e ter um cliente `Amanda`, ver seção 10.6.2, devidamente configurado.

---

serv1.domínio.com.br /scripts comp-root-tar

serv1.domínio.com.br /etc comp-root-tar

[....]

serv2.domínio.com.br /etc comp-root-tar

serv2.domínio.com.br /root comp-root-tar

[....]

serv3.domínio.com.br /var/www comp-root-tar

serv3.domínio.com.br /home comp-root-tar

[....]

Cada linha deve conter o nome da máquina, por exemplo serv1.domínio.com.br, seguido do diretório a ser 'backupeado' e o método de *backup*. Este pode assumir diversos valores pré-definidos, conforme apresentados na Tabela 10.1. Pode-se também personalizar o método de backup, para tanto é necessário criar uma nova definição no arquivo `amanda.conf`.

Em seguida deve-se rotular todas as fitas que pertencerão ao esquema. No exemplo abaixo rotula-se uma nova fita, pertencente ao esquema `diario`, com o rótulo `diario04`.

amlabel -f diario diario04

Durantes a rotulação das fitas o arquivo `tapelist` será criado e conterá a sequência de fitas a serem usadas no respectivo esquema pelo `Amanda`.

Tabela 10.1: Métodos de *backup* no `Amanda`

Método de *backup*	**Aplicação**
always-full	*Backup* completo para todas as cópias
root-tar	Partições `root` copiadas com o *tar*
user-tar	Partições de usuários copiadas com o *tar*
high-tar	Partições copiadas com o *tar*
comp-root-tar	Partições `root` copiadas com compressão
comp-root-tar-low	Partições `root` copiadas com compressão e baixa prioridade
comp-user-tar	Partições de usuários copiadas com compressão
holding-disk	Salva diretamente na fita, não armazenando no disco auxiliar
comp-user	Partições não `root` em máquinas razoavelmente rápidas
nocomp-user	Partições não `root` em máquinas lentas
comp-root	Partições `root` com compressão
nocomp-root	Partições `root` sem compressão
comp-high	Partições muito importantes em máquinas muito rápidas
nocomp-high	Partições muito importantes em máquinas lentas
nocomp-test	Testa a cópia sem compressão
comp-test	Testa a cópia com compressão

10.6.2 Configurando o cliente

Em qualquer máquina que irá pertencer a um 'domínio' `Amanda`, deve ser instalado o cliente `Amanda`. Esta instalação deve ser feita inclusive no próprio servidor `Amanda`, caso deseje-se fazer *backup* de algum diretório deste. A instalação dos pacotes é feita com o seguinte comando:

urpmi -a libam amanda-client

Após isto deve-se editar o arquivo `/etc/xinetd.d/amanda` e modificar a diretiva `disable` para que assuma o valor `no`. E então reinicia-se o serviço xinetd, que habilitará respostas ao servidor `Amanda`, com o comando:

service xinetd restart

Por último, deve-se editar o arquivo `/var/lib/amanda/.amandahosts` e incluir o nome do servidor, ou servidores, que terão permissão de contatar o cliente em busca de dados a "backupear". Este arquivo deve conter o nome da máquina (FQDN) e o usuário a ser invocado, normalmente o `amanda`. Um exemplo de uma linha deste arquivo: `hendrix.sj.ifsc.edu.br amanda`.

10.6.3 Uso do Amanda

Para testar as configurações, no servidor, digita-se o comando a seguir. Este faz a verificação de todo o estado do `Amanda`, se o dispositivo contém a fita correta (esperada), se há espaço livre suficiente no disco auxiliar e se as máquinas clientes estão adequadamente configuradas.

/usr/sbin/amcheck diario

Obtém-se 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 diario04 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)

Backups com o Amanda

Para *backup* dos diretórios descritos no `disklist` deve-se usar o *amdump* com a sintaxe:

amdump diario

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

Restaurando *backups* com o Amanda

Para restaurar os *backups* é utilizado o *amrecover*. Na máquina servidora, como `root`, cria-se um diretório de restauro e entra-se no mesmo.

mkdir restauro

cd restauro

Em seguida chame-se o programa de recuperação:

amrecover diario

Deste modo entra-se num `shell` específico do `Amanda`, que irá habilitar uma série de comandos próprios. Dentro deste ambiente pode-se listar os arquivos armazenados, por data, por servidor etc. A seguir mostra-se um exemplo de recuperação de alguns arquivos e alguns comandos úteis.

Pode-se ter uma visão geral dos diretórios e clientes `Amanda` com o comando:

listdisk

Como primeira etapa determina-se a data em que foi armazenado o arquivo que pretende-se restaurar. Evidentemente esta é uma informação que deve ser passada pelo usuário que está requisitando a recuperação de algum arquivo. O formato a ser utilizado é AAAA-MM-DD, onde AAAA é o ano, com quatro dígitos, MM é o mês e DD é o dia. No exemplo abaixo ajusta-se a data para 05 de setembro de 2009.

setdate 2009-09-05

O próximo passo é determinar de qual cliente `Amanda` se quer restaurar o *backup*. No exemplo abaixo seleciona-se o `localhost`, mas pode-se optar por qualquer cliente definido na seção 10.6.2.

sethost localhost

A seguir determina-se de qual diretório do respectivo cliente `Amanda` se restaurará o *backup*. Abaixo, um exemplo, de como extrair somente um arquivo do diretório `/etc`.

setdisk /etc

Agora pode-se navegar pelos sub-diretórios do diretório escolhido, ou simplesmente listar seu conteúdo.

cd rc.d

ls

O próximo passo é adicionar todos os arquivos a serem restaurados. Lembre-se que pode-se usar coringas, como o '*' para adicionar todos os arquivos, ou pode adicionar um diretório completo bastando informar seu nome.

add rc.local

Continua-se adicionando diretórios e arquivos, conforme o necessário.

Depois disso resta extrair os arquivos. Os arquivos e diretórios marcados para extração serão salvos no diretório de entrada, ou seja, no diretório a partir de onde se lançou o comando *amrecover*, no exemplo `restauro`.

extract

Comandos Extras do Amanda

Não pretende-se abordar todas as facilidades do Amanda, somente as mais usuais.

Como já informado o `Amanda` faz cópias temporárias dos arquivos num disco auxiliar, ver Seção 10.6.1. Caso o `Amanda` não encontre a fita correta ou a mesma não esteja na unidade, o sistema não fará a cópia para fita e portanto os dados permanecerão neste disco. Para descarregar o conteúdo do disco auxiliar para a fita, deve-se fazer uso do comando *amflush*.

Caso tenha ocorrido alguma interrupção em qualquer fase de *backup* ou descarrego do `Amanda`, podem ocorrer problemas com arquivos corrompidos ou incompletos. Para limpeza geral nos arquivos do Amanda executa-se o *amcleanup*.

Para tarefas administrativas *amadmin*. Aqui define-se permissão de usuários, arquivos de configuração etc.

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*.

Exemplo de Crontab do Amanda

A seguir mostra-se um exemplo de configuração do *Crontab* do usuário `Amanda`, que é o usuário padrão de controle do serviço `Amanda`. Neste exemplo coloca-se um agendamento diário, para o esquema diário. Ao meio dia faz-se uma verificação do sistema `Amanda`, enviando os resultados por correio eletrônico (*-m*) ao administrador. De posse desta mensagem o administrador será lembrado de trocar a fita, caso seja necessário, ou verificará qualquer problema que possa estar acontecendo. À meia-noite faz-se o *backup* e ao seu término a fita será ejetada. Estes *backups* ocorrerão todos os dias, de segunda a sexta-feira.

*0 12 * * 1-5 /usr/sbin/amcheck -m diario*

*0 12 * * 1-5 /usr/sbin/amdump diario && eject /dev/nst0*

Recomenda-se ainda fazer os respectivos agendamentos para os demais esquemas: semanais, mensais, anuais etc. Seguindo o padrão já apresentado. Abaixo tem um exemplo de agendamento para esquema semanal, que deverá ser devidamente configurado conforme o apresentado na seção 10.6.1. Neste caso estes *backups* ocorrerão todos os sábados.

*0 12 * * 6 /usr/sbin/amcheck -m semanal*

*0 12 * * 6 /usr/sbin/amdump semanal && eject /dev/nst0*

Capítulo 11

Programação do *Shell*

11.1 Introdução

Programação *Shell* permite que administradores criem pequenos programas para automatizar a administração do sistema, configurar melhor o sistema e entender vários recursos do próprio sistema operacional `Linux`, por exemplo, toda a configuração da sequência de *boot* do `UNIX` envolve *scripts shell*.

A programação *shell* permite a criação de um arquivo de texto contendo comandos do `UNIX` e comandos especiais do *shell* para definir e utilizar: variáveis, fazer testes no sistema, laços de repetição, funções, comentários etc.

Existe uma relação direta entre conhecimento dos comandos do *shell* e a complexidade dos *scripts* que podem ser escritos. Este livro parte do pressuposto do conhecimento preliminar dos mesmos e não tem a intenção de se aprofundar nestes comandos, no Apêndice F são citados alguns poucos e seus usos.

Um *script shell*[1] é interpretado pelo respectivo *shell* e, portanto, não é um programa a ser compilado. Isto pode tornar um *script shell* lento, já que cada comando que o *script* chama dispara um pro-

[1] http://walfredo.dsc.ufpb.br/cursos/suporte20012/progsh/index.htm

cesso. Caso se deseje um melhor desempenho deve-se partir para uma linguagem de programação propriamente dita.

Este livro não pretende passar conceitos de lógica de programação. O intuito é tão somente passar a sintaxe dos principais comandos que darão base para composição de bons *scripts*.

11.2 Comandos Básicos

Recomenda-se que um *script shell* inicie com a definição de qual será o *shell* a interpretá-lo. Como todo programa recomenda-se também escrever comentários para facilitar a compreensão do mesmo, isto pode ser feito em qualquer parte do programa bastando iniciar o comentário com o caractere especial '#'. Todo texto que estiver após o caractere '#' e até o final da linha será considerado um comentário e, portanto, não será interpretado. A exceção é a definição do *shell*. Na linha 1 da Listagem 11.1 tem-se a definição do *shell*, na linha 2 um comentário sobre o programa, este tipo de comentário pode ocorrer em qualquer posição do programa, e na linha 4 o comando que irá imprimir no *shell* corrente uma frase.

```
1 #!/bin/bash
2 # Este programa diz alo
3
4 echo "Alo $LOGNAME, tenha um bom dia!"
```

Listing 11.1: Script que imprime uma mensagem de boas vindas

Para executar o programa deve-se mudar suas permissões para execução, veja na seção 7.3, e em seguida executá-lo com o comando:

./alo.sh

11.2.1 Variáveis e Parâmetros

O *shell* possui várias variáveis pré-definidas, como por exemplo: *$LOGNAME*, *$HOSTNAME*, *$TERM* etc. Para obter uma listagem

completa da variáveis basta executar o comando *printenv*. Pode-se criar variáveis com qualquer nome e conteúdo dentro de um *script shell*, para tanto não é necessário uma pré-definição de seu tipo, basta usá-la no momento adequado. Por exemplo, na linha 4 na Listagem 11.2 tem-se uma variável 'string' e na linha 5 uma variável 'numérica'. O `Shell` não distingue as variáveis por seus tipos, esta distinção aparecerá somente com o uso das mesmas, por exemplo, caso tente-se fazer uma operação matemática com uma string haverá um erro. Ambas as variáveis são utilizadas na linha 6 para imprimir seus conteúdos no *shell*. O resultado da execução de tal *script* será: `Esta e uma variável do tipo string 5`.

```
1 #!/bin/bash
2 # Script mostrando o uso de variveis
3
4  var_string ="Esta e uma varivel do tipo string "
5 var_numerica=5
6 echo $var_string " " $var_numerica
```

Listing 11.2: Script com variáveis

Outro conjunto interessante de variáveis é o que pode ser passado diretamente do *shell* ao *script*. Isto é feito "imediatamente" com o uso do caractere especial '$' ao longo do *script*. O '$0' retorna ao *script* seu próprio nome, o '$1' à '$9' retorna respectivamente os campos 1 à 9 passados junto com o nome do *script* em sua execução, o '$#' retorna o número de parâmetros passados e o $* retorna todos os parâmetros do programa, se em alguma linha do *script* tem-se o conteúdo *echo $**, todos os parâmetros passados serão impressos. No exemplo da Listagem 11.3, tem-se um teste simples para verificar a entrada de parâmetros num *script*. Neste *script*, entre as linhas 5 e 8, faz-se o teste buscando a presença de 1 (um) parâmetro passado ao *script*, caso isto não aconteça o *script* retorna uma mensagem de erro, informando como o mesmo deve ser executado e termina sua execução. O comando *exit* encerra o *shell* e retorna um *status* para o processo pai.

```
1 #!/bin/bash
2 # Script mostrando a manipulacao de variveis
```

```
# Verificando os parametros passados
if [ $# -ne 1 ]; then
    echo "Falta parametro - Rode o script com o comando ./$0 variavel "
    exit
fi
. . .
```

Listing 11.3: Script com manipulação de variáveis

Para tornar um *script* menos suscetível a erros pode-se definir um valor padrão para determinada variável. Isto pode ser interessante para o caso de não termos certeza que um valor será atribuído à mesma. Isto é muito comum em *scripts* que requisitam valores durante sua execução. Para a definição de um valor padrão de variáveis faz-se uso do seguinte modelo:

```
${variavel:-valor_padrao}
```

ou

```
${variavel:?mensagem_desejada}
```

No primeiro caso a variável assumirá o `valor_padrao` caso nenhum valor seja inserido e no segundo caso será apresentada ao usuário uma mensagem de erro com conteúdo `mensagem_desejada`. No exemplo da Listagem 11.4 tem-se um *script* que irá somar 3 valores passados ao mesmo, caso um ou mais dos valores não sejam passados, os mesmos serão assumidos com o valor 0 (zero). Por exemplo, ao executar-se o *script* com valores 2, 4 e 7 tem-se como resposta `A soma e 13`, sem valores tem-se como resposta `A soma e 0`.

```
#!/bin/bash
# Script mostrando a manipulacao de variveis

valor=`expr ${1:-0} + ${2:-0} + ${3:-0}`
echo A soma e $valor
```

Listing 11.4: Script com manipulação de variáveis padrão

Parâmetros compostaos podem ser utilizados desde que estejam agrupados dentro de aspas duplas - ‘ ˝’. Por exemplo o comando *echo a b c d* apresenta 4 parâmetros, já o *echo ˝a b˝c d* apresenta 3, já que ‘a b’ serão encarados como um único parâmetro.

Num *script*, como em qualquer outra linguagem de programação, pode-se ter entrada e saída de parâmetros. Já foi apresentada a possibilidade de saída (*print*) de parâmetros com o comando echo e entrada na chamada do *script*. Para uma leitura de parâmetros em qualquer posição do *script* utiliza-se o comando *read*. No exemplo apresentado na Listagem 11.5 tem-se na quarta linha uma requisição, ao usuário que executou o *script*, de inserção de seu nome, o ‘-n’ serve para a não inserção de nova linha após a mensagem. Na quinta linha faz-se a leitura propriamente dita da variável, no caso `nome`, e na última linha utiliza-se esta variável para imprimir uma mensagem ao usuário. Caso o usuário não insira caracteres será impresso: `Seu nome e sem_nome`.

```
1 #!/bin/bash
2 # Script modelo para interacao com usuario. Entrada e saida de
      parametros
3
4 echo -n "Por favor informe seu nome: "
5 read nome
6 echo "Seu nome e ${nome:-sem_nome}"
```

Listing 11.5: Script para interação com usuário

11.3 Comandos de testes no shell

11.3.1 A Estrutura *if*

Como em qualquer linguagem de programação um `script shell` pode executar testes com variáveis. Abaixo tem-se as principais formações da estrutura ‘se’ (*if*).

```
if comando
```

```
then
  execução se "comando" retornar status "ok" (=0)
else
  execução se "comando" retornar status "não ok"
fi

if comando1
then
  execução se "comando1" retornar 0
elif comando2
  execução se "comando2" retornar 0
else
  execução se não entrar nos "if" acima
fi

if comando1 && comando2
then
  execução se "comando1" E "comando2" retornarem 0
fi

if comando1 || comando2
then
  execução se "comando1" OU "comando2" retornarem 0
fi
```

Os `comandos`, da estrutura 'se' podem ser substituídos por variáveis, neste caso devem ser inseridas entre colchetes - [ ]. Na Listagem 11.6 tem-se algumas possibilidades de uso desta estrutura, inclusive com variáveis. Entre as linhas 4 e 7 o 'if' testa o número de parâmetros passados, se for igual a zero aborta a execução. Entre as linhas 9 e 15 o 'if' executa o comando *ls* em dois arquivos. Se os dois existirem o *script* envia uma mensagem que os arquivos foram encontrados. Caso pelo menos um deles não exista o retorno do comando sera 'não ok' e o *script* envia uma mensagem de erro e aborta a execução.

```
#!/bin/bash
#  Script  com exemplos da  estrutura   if

#  Teste  de  passagem de parametros.  Caso o  numero de parametros  passado
      seja  igual  a  zero  o  script  e  abortado
if  [  $# -eq 0 ];  then
  echo "Falta  parametro  - Rode o  script   ./  estruturas_if  .sh  parametro1  [
      parametro2   ...] "
  exit
fi

# No teste  abaixo  o  script   verifica   a  existencia   ou nao do  arquivos:
     arq1  e  arq2.
if  ls  arq1  && ls arq2
then
  echo "Arquivos  arq1  e  arq2  encontrados!"
else
  echo "Pelo  menos um dos arquivos  nao  encontrados  - caindo  fora"
  exit  1
fi
```

Listing 11.6: Script com estruturas 'se'

Existem vários testes padronizados no `shell` os principais são mostrados nas tabelas a seguir. Na Tabela 11.1 mostra-se testes especificos para tratamento de *strings*, na 11.2 testes para manipulação numérica, na 11.3 testes para verificação de propriedades de arquivos e na 11.4 outros operadores.

Tabela 11.1: Testes de strings

Expressão	**Verdeiro se**
-z string	o tamanho da `string` é 0 (zero)
-n string	o tamanho da `string` NÃO é 0 (zero)
string1 = string2	`string1` é igual a `string2`
string1 != string2	`string1` NÃO é igual a `string2`
string	`string` NÃO é nulo

Tabela 11.2: Testes numéricos

Expressão	**Verdeiro se**
int1 -eq int2	`int1` é igual a `int2`
int1 -ne int2	`int1` NÃO é igual a `int2`
int1 -gt int2	`int1` é maior que `int2`
int1 -ge int2	`int1` é maior ou igual a `int2`
int1 -lt int2	`int1` é menor que `int2`
int1 -le int2	`int1` é menor ou igual a `int2`

Tabela 11.3: Testes de arquivos

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

Tabela 11.4: Operadores adicionais

Expressão	**Propósito**
!	inverte expressão lógica
-a	operador AND
-o	operador OR
(expr)	agrupar expressões

11.3.2 A Estrutura *case*

O `case` pode ser considerado um *if* múltiplo. Na Listagem 11.7 pode-se ver um exemplo de sua aplicação. Na linha 7 faz-se uma leitura de

uma variável, na linha 8 inicia-se o tratamento da mesma. Entre as linhas 9 e 11 explicita-se o tratamento para o caso de a variável ter valor 's' ou 'S', no lugar da linha 10 pode-se colocar várias ações a serem executadas neste caso. Entre as linhas 12 e 14 repete-se o caso anterior, já entre as linhas 15 e 17 dá-se o tratamento para exceções, ou seja, caso a variável não tenha se enquadrado em nenhuma da opções anteriores o tratamento dado será este.

```
1  #!/bin/bash
2  #  Script  com exemplo de uso da  estrutura  case.
3
4   ...
5
6  echo -n "Sua reposta: "
7  read  var
8  case  "$var"  in
9  S* | s*)
10    resp="sim"
11    ;;
12 N* | n*)
13    resp="nao"
14    ;;
15 *)
16    resp="talvez"
17    ;;
18 esac
19 echo $var
```

Listing 11.7: Script com estrutura 'case'

11.4 Laços de Repetição

11.4.1 A Estrutura *while*

A estrutura while é utilizada para a repetição de comandos até que algum evento de término ocorra. A estrutura básica de seu funcionamento pode ser vista a seguir.

```
while comando
```

```
do
    execução enquanto "comando" retornar 0
done
```

No exemplo da Listagem 11.8 se observa seu emprego. Na linha 4 o argumento passado como parâmetro ao *script* é associado à `nomeArq`. Na linha 5 inicializa-se a variável `numLinha`, para contagem de linhas do arquivo. Em seguida, entre as linhas 6 e 10, é lido sequencialmente uma a uma as linhas do arquivo `nomeArq`. Entre as linhas 8 e 9 é feito o respectivo tratamento dos dados, para cada uma das linhas do arquivo. O *script* é encerrado quando terminam as linhas do `nomeArq`, linha 6. O *script* como um todo, simplesmente lista o arquivo passado como parâmetro numerando suas linhas.

```
1  #!/bin/bash
2  # Exemplo de uso da  estrutura  while
3
4  nomeArq=$1
5  numLinha=1
6  while  read  linha
7  do
8    echo  "$numLinha $linha"
9    numLinha=‘expr $numLinha + 1‘
10 done < nomeArq
```

Listing 11.8: Script com estrutura ‘while’

11.4.2 A Estrutura *for*

Outra estrutura de repetição é a *for*. A estrutura básica de seu funcionamento pode ser vista a seguir.

```
for var in lista
do
    execução para cada componente ($var) da "lista".
done
```

No exemplo da Listagem 11.9 tem-se um exemplo de *script* que gera um arquivo com mensagem de boas vindas para todos os

usuários contidos no arquivos `lista_de_usuarios.txt`. Na linha 4 o *for* faz a leitura de cada uma das linhas do arquivo

```
/caminho/do/diretorio/lista_de_usuarios.txt
```

e a atribui à variável `user`. Na linha 6 é gerado o arquivo.

```
1  #!/bin/bash
2  # Exemplo de uso da estrutura for.
3
4  for user in `cat \caminho\do\ diretorio \ lista_de_usuarios .txt`
5  do
6      echo "Ola $user. Seja bem-vindo a instituicao " > ~$user/boas_vindas.
           txt
7  done
```

Listing 11.9: Script com estrutura 'for'

11.5 Funções

Apesar das limitações na programação do *shell* é possível criar funções. Na Listagem 11.10 é apresentado um exemplo de seu uso. Entre as linhas 7 e 15 encontra-se a estrutura da função e entre a 17 e 25 o programa principal. Neste caso a `função` simplesmente verifica a existência ou não do arquivo passado como parâmetro e informa ao `Programa principal` através do código `0` - existe ou código `1` - não existe.

```
1  #!/bin/bash
2  #
3  # Exemplo de uso das Funcoes
4
5  nome_do_arquivo=$1
6
7  # Funcao para verificacao da existencia ou nao de determinado arquivo
8  verificaArquivos () {
9    if [ -r $* ]
10   then
11     return 0
```

```
12      else
13          return 1
14      fi
15  }
16
17  # Programa principal
18
19  if verificaArquivos $nome_do_arquivo
20  then
21      echo "Arquivo $nome_do_arquivo existe."
22  else
23      echo "Arquivo nao encontrado."
24      exit 1
25  fi
```

Listing 11.10: Script com exemplo do uso de função

Capítulo 12

Rede e Roteamento

12.1 Introdução

Neste capítulo não se tem a intenção de abordar a arquitetura das redes e nem seus protocolos. O objetivo é tão somente mostrar como configurar as interfaces de rede de uma máquina com sistema operacional Linux, bem como configurá-la como roteador ou roteador de borda.

Parte-se do prévio conhecimento das camadas do protocolo TCP/IP – *Transmission Control Protocol /Internet Protocol*, do endereçamento IP e dos protocolos de roteamento.

12.2 Configuração de Interface de Rede

Ao adicionar-se uma máquina à rede é obrigatória a configuração de no mínimo dois parâmetros, o endereço IP e a máscara de rede. Com estes dois parâmetros a máquina já está apta a se comunicar com outras máquinas da mesma rede local.

Para uma configuração completa é necessário configurar-se ainda o nome de máquina, o roteador padrão (*default gateway*) e o servidor de nomes, neste caso podem haver vários.

Todos estes parâmetros podem ser configurados estaticamente ou por meio de um servidor DHCP - *Dynamic Host Configuration Protocol*. A configuração estática é recomendada para servidores de rede, já a dinâmica é a clássica para estações de uso comum.

Para configurar estes parâmetros deve-se editar o arquivo `/etc/sysconfig/network-scripts/ifcfg-ethN`, onde N é o número da interface. No `Mandriva` tem-se um arquivo por interface, iniciando-se por `ifcfg-eth0`. Na Tabela 12.1 tem-se um exemplo onde aparecem a configuração estática e a dinâmica (dhcp).

Tabela 12.1: Arquivo de configuração de Interface de Rede

Configuração dhcp	**Configuração estática**
DEVICE=eth0	DEVICE=eth0
BOOTPROTO=dhcp	BOOTPROTO=static
ONBOOT=yes	ONBOOT=yes
	IPADDR=192.168.2.1
	NETMASK=255.255.255.0
	BROADCAST=192.168.2.255
	GATEWAY=192.168.2.1
	MS_DNS1=172.18.0.1
	MS_DNS2=200.135.37.65

Além do arquivo acima deve-se configurar do arquivo `/etc/sysconfig/network`, que define basicamente o nome da máquina (*hostname*).

```
HOSTNAME=nome_de_maquina.nome.do.dominio
```

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 setados serão dados por estes arquivos.

O comando `ifconfig` serve, entre outras coisas, para modificar dinamicamente alguns destes parâmetros. Para um teste, por exemplo, pode-se 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

Por exemplo, no comando abaixo estamos mudando a configuração da interface de rede `eth0` para assumir o endereço IP 192.168.2.23 com máscara de rede 255.255.255.0. Isto passará a valer imediatamente mas ao reiniciarmos a interface, ou máquina, os arquivos de configuração serão lidos e a interface voltará a ter os parâmetros pré-ajustados.

ifconfig eth0 192.168.2.23/24 up

12.2.1 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 etc. Para fazer-se isto basta criar interfaces virtuais com nomes ethN:0, ethN:1 etc. Por exemplo:

ifconfig eth0:0 10.10.1.23/24 up

Ou criando um arquivo `/etc/sysconfig/network-scripts/ifcfg-eth0:0` com somente o conteúdo abaixo e reiniciarmos o serviço de rede.

```
IPADDR=192.168.2.1X
NETMASK=255.255.255.0
```

12.2.2 Conferindo e Testando

Para conferir o perfeito funcionamento da máquina em rede deve-se primeiramente certificar-se de que todas as conexões físicas estão em perfeita ordem, em seguida inicia-se o processo de conferência lógico. O primeiro passo pode ser pode ser o `ping` para uma máquina da mesma rede local, sabidamente em pleno funcionamento.

ping 192.168.2.1

Se a resposta for do tipo:

```
PING 192.168.2.1 (192.168.2.1) 56(84) bytes of data.
64 bytes from 192.168.2.1: icmp_seq=1 ttl=64 time=1.44 ms
64 bytes from 192.168.2.1: icmp_seq=2 ttl=64 time=0.513 ms
64 bytes from 192.168.2.1: icmp_seq=3 ttl=64 time=0.516 ms
--- 192.168.2.1 ping statistics ---
3 packets transmitted, 3 received, 0% loss, time 2000ms
rtt min/avg/max/mdev = 0.513/0.824/1.445/0.439 ms
```

significa que houve êxito e a interface está operacional em relação aos parâmetros `IP` e `máscara de rede`. Se a resposta for de erro deve-se conferir os parâmetros `IP` e `máscara de rede` e a conexão física do equipamento.

Continuando deve-se testar o `default gateway`, "pingando" para um IP conhecido fora da rede local. Por exemplo:

ping 200.135.37.65

Se a resposta também for positiva isto significará que o `default gateway` também está devidamente configurado. Se for negativa deve-se conferir este parâmetro com o comando:

route -n

Na Tabela 12.2 tem-se um exemplo da saída deste comando. Da terceira linha conclui-se que pacotes destinados a

rede `192.168.2.0/255.255.255.0` sairá diretamente pela interface `eth0`, sem passar por roteadores, já que neste campo tem-se como conteúdo `0.0.0.0`. Na quinta linha, por outro lado, conclui-se que para qualquer destino `0.0.0.0` deve-se usar o roteador `192.168.2.1` que está acessível via interface `eth0`, ou seja na mesma rede local. Devido a sequência de regras, todos os pacotes destinados a rede `192.168.2.0/255.255.255.0` sairão diretamente pela interface `eth0`, já que esta regra aparece em primeiro lugar, todos os pacotes destinados a rede `127.0.0.0/255.0.0.0` sairão diretamente pela interface `lo` *loopback* e para os demais destinos através do roteador `192.168.2.1`.

Tabela 12.2: Tabela de Roteamento

Tabela de Roteamento IP do Kernel

Destino	Roteador	MáscaraGen.	Opções	Métrica	Ref	Uso	Iface
192.168.2.0	0.0.0.0	255.255.255.0	U	1	0	0	eth0
127.0.0.0	0.0.0.0	255.0.0.0	U	0	0	0	lo
0.0.0.0	192.168.2.1	0.0.0.0	UG	0	0	0	eth0

Caso algum dos parâmetros apontados esteja incorreto deve-se corrigí-lo e reiniciar o serviço de rede.

Por último confere-se o parâmetro de servidor DNS. Para isto pode-se "pingar" para um nome de máquina, por exemplo:

ping www.sj.ifsc.edu.br

Se a resposta for do tipo:

```
PING www.sj.ifsc.edu.br (200.135.37.65) 56(84) bytes of data.
64 bytes from hendrix (200.135.37.65): icmp_seq=1 ttl=58 time=8.03 ms
64 bytes from hendrix (200.135.37.65): icmp_seq=2 ttl=58 time=6.66 ms
--- www.sj.ifsc.edu.br ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 6.668/7.353/8.039/0.690 ms
```

significa que está tudo em ordem, o nome foi resolvido para um endereço IP (200.135.37.65). Se houver problemas deve-se consultar

o arquivo `/etc/resolv.conf` e verificar os respectivos parâmetros. Este arquivo contém a listagem dos servidores de nomes setados na máquina, por exemplo:

```
nameserver 172.18.0.1
nameserver 200.135.37.65
```

Para alterá-los deve-se redefinir os devidos parâmetros na seção 12.2.

12.3 Roteadores e sub-redes

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 utilizadas para conexão da rede local com a Internet.

Existem também os chamados `modem/router` que além de roteadores são modens, comumente usados para conexão ADSL.

Simplificadamente, em um roteador os dois principais componentes lógicos são: os protocolos de roteamento e as tabelas de encaminhamento. Neste texto não detalham-se os protocolos de roteamento, atem-se tão somente a análise de tabelas de encaminhamento. Estas é que definem a interface de saída de todo e qualquer pacote que chega ao roteador.

Uma máquina Linux, com duas ou mais interfaces de rede, também funciona como um roteador. Esta pode ser uma opção interessante se desejarmos criar sub-redes na instituição e obrigatória na implementação de um *firewall* transparente.

As tabelas estáticas de roteamento podem ser uma opção para pequenas redes. Na análise destas tabelas um roteador toma as decisões de encaminhamento, trabalhando com a leitura linha-a-linha,

de tal modo que quando for encontrada uma regra que atenda a "demanda" o sistema para imediatamente. Analisando a tabela de roteamento apresentada na Tabela 12.2 pode-se fazer a seguinte interpretação: se um pacote for destinado à 192.168.2.0/24 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 `192.168.2.1` e este "que se vire". Observe que o endereço 192.168.2.1 é um endereço "atingível" pela interface `eth0`, ou seja, encontra-se em sua rede local.

12.3.1 Montando Tabelas Estáticas de Roteamento

Na Figura 12.1, observam-se várias sub-redes interligadas por máquinas Linux, configuradas como roteadores, e uma delas conectada à Internet. Por questões de simplicidade supõe-se que todas as máscaras de rede são 255.255.255.0.

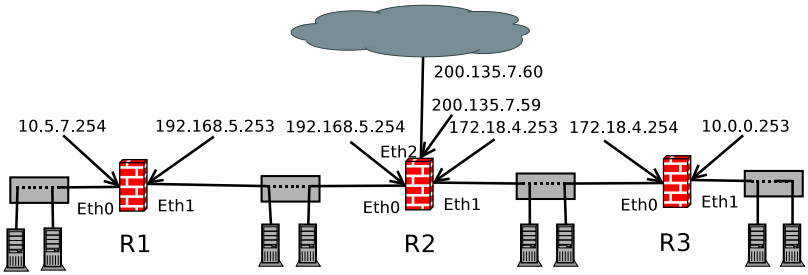

Figura 12.1: Exemplo de Rede Local

Baseado na Figura 12.1 montam-se as tabelas estáticas de roteamento para cada uma das máquinas roteadoras, R1, R2 e R3, analisando-se a rede como um todo e construindo uma rota para cada uma das demais redes e uma rota final que será a rota padrão para a Internet. Nas Tabelas 12.3, 12.4 e 12.5 observa-se esta construção.

Tabela 12.3: Tabela Estática de Roteamento de R1

Rede	Roteador
10.5.7.0	eth0
192.168.5.0	eth1
~~172.18.4.0~~	~~192.168.5.254~~
~~10.0.0.0~~	~~192.168.5.254~~
~~200.135.7.0~~	~~192.168.5.254~~
0.0.0.0	192.168.5.254

Tabela 12.4: Tabela Estática de Roteamento de R2

Rede	Roteador
192.168.5.0	eth0
172.18.4.0	eth1
200.135.7.0	eth2
10.5.7.0	192.168.5.253
10.0.0.0	172.18.4.254
0.0.0.0	200.135.7.60

Tabela 12.5: Tabela Estática de Roteamento de R3

Rede	Roteador
172.18.4.0	eth0
10.0.0.0	eth1
~~10.5.7.0~~	~~172.18.4.253~~
~~192.168.5.0~~	~~172.18.4.253~~
~~200.135.7.0~~	~~172.18.4.253~~
172.18.4.0	172.18.4.253

Observe que nas tabelas de roteamento existem algumas rotas que podem eliminadas, já que são desnecessárias pois direcionam rotas para redes específicas passando pelo roteador padrão (0.0.0.0). Portanto, para a rede proposta na Figura 12.1 as tabelas estáticas de roteamento simplificadas seriam as apresentadas na Tabela 12.6.

Pode-se observar nas tabelas que todos os caminhos são contemplados e todos os roteadores passam a conhecer as demais redes e

Tabela 12.6: Tabelas de Roteamento Simplificadas

Roteador 1			**Roteador 3**	
Rede	Roteador		Rede	Roteador
10.5.7.0	eth0		172.18.4.0	eth0
192.168.5.0	eth1		10.0.0.0	eth1
0.0.0.0	192.168.5.254		172.18.4.0	172.18.4.253

a Internet, muitas vezes usando o roteador padrão (*default* = 0.0.0.0) para isto. Isto possibilita que qualquer pacote de qualquer cliente pode chegar ao destino e pode receber sua resposta.

Para definir-se as rotas em uma máquina Linux usa-se o comando `route`, com a seguinte sintaxe:

```
route add -net rede/máscara gw ip_roteador
```

Para a rota padrão, seria:

```
route add default gw ip_do_roteador
```

12.3.2 Configurando o roteador

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

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

```
sysctl -w net.ipv4.ip_forward=1
```

Assim, a máquina imediatamente 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.

Caso deseja-se deixar permanentemente o roteamento habilitado deve-se editar o arquivo `/etc/sysctl.conf` e acrescentar ao final no mesmo o seguinte conteúdo:

```
# To enable routing
net.ipv4.ip_forward = 1
```

12.3.3 Configurando sub-redes

Para efeitos didáticos analizar-se-á a estrutura de sub-redes mostrada no diagrama esquemático na Figura 12.2. Neste diagrama existem 4 sub-redes cada uma composta de 1 roteador e 1 cliente para testes. A máquina "R1", que também é um roteador, interligará estas sub-redes à rede local e à Internet. Esta máquina será o roteador padrão de cada um dos 4 roteadores "secundários".

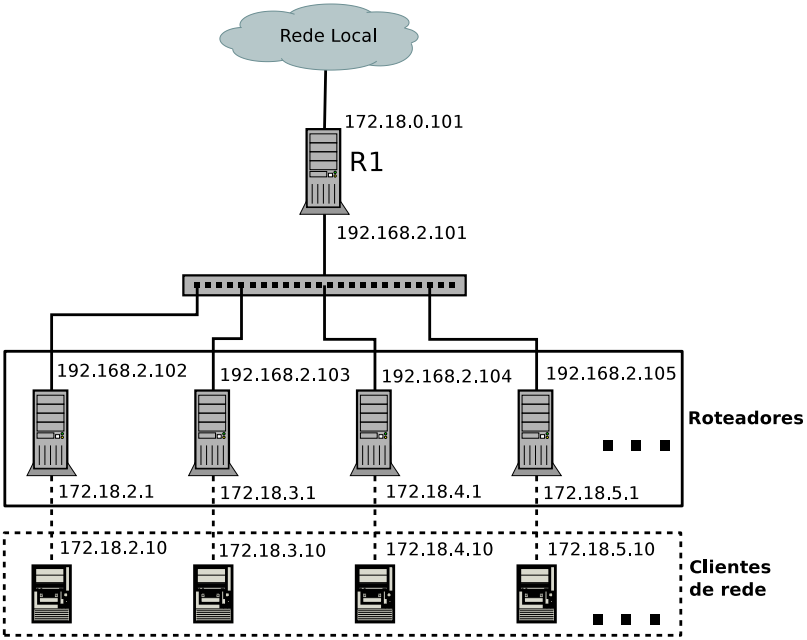

Figura 12.2: Diagrama com sub-redes

Configuração dos Roteadores

No roteador deve-se, em primeiro lugar, configurar as interfaces de rede, conforme apresentado na seção 12.2. Por questões de simplicidade, pode-se usar `ip aliases`, ou seja, uma única interface de rede do roteador responderá pelos dois IP's, mesmo sendo de classes diferentes. Isto permitirá que não seja necessário reestruturar o cabeamento.

Deve-se adotar a numeração do diagrama esquemático. Por exemplo para o roteador mais a esquerda:

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

```
IPADDR=192.168.X.X
NETMASK=255.255.255.0
```

Reinicia-se o serviço de rede com o comando:

service network restart

Adicionam-se as rotas, para as 3 demais sub-redes, com a sequência de comandos abaixo:

route add -net 172.18.3.0/24 gw 192.168.2.103

route add -net 172.18.4.0/24 gw 192.168.2.104

route add -net 172.18.5.0/24 gw 192.168.2.105

Adiciona-se também a rota padrão:

route add default gw 192.168.2.101

Habilita-se o roteamento com o comando:

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

Configuração dos Clientes

Para habilitar os clientes basta configurar a interface de rede, conforme seção 12.2, e o roteador padrão como sendo o IP da interface a qual está conectado ao roteador, 172.18.2.1 para o cliente mais a esquerda.

Testes

Para testar todas as rotas estabelecidas deve-se adotar, por exemplo, a seguinte sequência de testes:

1. A partir do cliente "pingar" a interface mais próxima do roteador. Se este `ping` não funcionar deve-se revisar a configuração física e lógica entre este e o roteador.

2. 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.

3. A partir do roteador "pingar" para a interface interna de um outro roteador, por exemplo 192.168.4.1. Se houve problemas os motivos podem ser dois: não foi escrita uma rota adequada para tal rede, verifica-se com o comando `route -n`, e/ou porque o roteador "pingado" está mal configurado. Lembre-se que os pacotes devem ter rota para ida e volta.

4. 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".

5. A partir do cliente, usar o `traceroute` para outro cliente, verificando a rota utilizada.

Capítulo 13

Servidor DNS com Bind

13.1 Introdução

Um serviço de nomes, como o DNS – *Domain Name System*, armazena o conjunto de um ou mais contextos de atribuição de nomes – conjuntos de vínculos entre nomes textuais e atributos de objetos, como usuários, computadores, serviços e objetos remotos.

O DNS é um projeto de `serviço de nomes` cujo banco de dados de atribuição de nomes é usado na Internet. Foi planejado por Mockapetris, no RFC 1034, para substituir um arquivo central que era carregado por download, não escalável e de administração centralizada. É projetado para ser usado em várias implementações. Na prática, somente é amplamente utilizado para atribuição de nomes da Internet. Os objetos nomeados são principalmente os computadores com, basicamente, seu endereço IP como atributo.

O *Domain Name System* é um banco de dados "global" com estrutura hierárquica com separador ponto – '.'. Sendo que os servidores DNS não reconhecem nomes relativos já que todos os nomes se referem à raiz global.

13.2 Domínios Hierárquicos

O espaço de nomes de domínio são hierárquicos, ou seja, cada parte do nome pode ser resolvida por um servidor independente. Por exemplo, para o nome `www.sj.ifsc.edu.br.` ter-se-ia a seguinte estrutura hierárquica para resolução do mesmo:

- *Root Level Domain* – '.'.
- *Top Level Domain* – '.edu.br'.
- *Second Level Domain* – '.sj.ifsc'.

Os *Root Level Domain* são localizados majoritariamente nos Estados Unidos, estes servidores tem conhecimento de todos os *Top Level Domain*. O *Top Level Domain* para o caso brasileiro é de responsabilidade do `Registro.br`, que conheçe todos os *Second Level Domain* do Brasil. Neste caso o *Second Level Domain* pertence ao IF-SC e ele conhece a máquina `www` que pertence ao domínio `.sj.ifsc.edu.br.`

Deve-se salientar que apesar do sufixo de caráter geográfico `br`, um domínio como `sj.ifsc.edu.br.` poderia ter dados (servidores) localizados em qualquer parte do mundo.

Cada domínio, ou `zona`, deve ter pelo menos um servidor DNS, mas cada servidor DNS pode ser responsável por mais de um domínio. Cada `zona` deve conter pelo menos os seguintes dados:

- Dados de atributos de nomes em um domínio, menos os subdomínios a ele subordinados.
- Nomes e endereços de pelo menos 2 servidores que possuem autoridade sobre dados da zona.
- Nome de servidores que contém autoridade sobre dados de subdomínios delegados.
- Parâmetros de gerenciamento da zona, como por exemplo: uso do *cache*, replicação de dados, TTL etc.

Os administradores inserem dados de uma `zona` em um arquivo mestre num `servidor principal` ou `mestre` e os `servidores secundários` fazem *donwload* destes dados. Os principais registros de recursos de um arquivo de zona são mostrados na Tabela 13.1.

Tabela 13.1: Principais registros DNS

Tipo de registro	**Significado**	**Conteúdo principal**
A	Endereço de computador	Número IP
NS	Servidor de nome com autoridade	Nome de domínio do servidor
CNAME	Nome canônico de um *alias*	Nome de domínio do *alias*
SOA	Marca o início dos dados de uma zona	Parâmetros que governam a zona
MX	Responsável pelo correio eletrônico	Lista de pares <preferência,host >
PTR	*Pointer Record* (resolução reversa)	Nome de domínio
TXT	String de texto	Texto arbitrário
WKS	Descrição de um serviço conhecido	Lista de nomes de serviços e protocolos
HINFO	Informações de host	Arquitetura de máquina e SO

13.3 Resolução de Nomes

Ao se realizar uma consulta DNS pode-se obter as seguintes informações principais: resolução de nomes de máquinas, ou seja, seu número IP; localização de servidores de correio eletrônico e a resolução reversa, ou seja, dado um número IP obtém-se o nome da máquina. Além disto, mas muito pouco utilizado na prática, pode-se obter informações sobre máquinas e serviços conhecidos.

O processo recursivo de resolução de nomes, que é o mais usual, segue o esquema da Figura 13.1. Este processo de resolução pode ser descrito do seguinte modo (setas em destaque):

1. O cliente, máquina `A`, deseja acessar algum serviço baseado em um nome qualquer. Este "olha" em sua `tabela DNS cache` e verifica se tem ou não o endereço equivalente, caso não tenha solicita esta informação ao DNS local, seu `nameserver`.

2. O DNS local (`nameserver`) 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 em questão informa então o IP da máquina procurada ao DNS Local.

8. O DNS local armazena na `tabela DNS cahe` a correspondência entre IP e nome, prevendo futuras consultas, e entrega a informação ao cliente – máquina `A` – que também guarda em seu *cache*.

Sempre que se acessa uma máquina por meio de seu nome, por exemplo, faz-se uma consulta as tabelas DNS. Um mecanismos

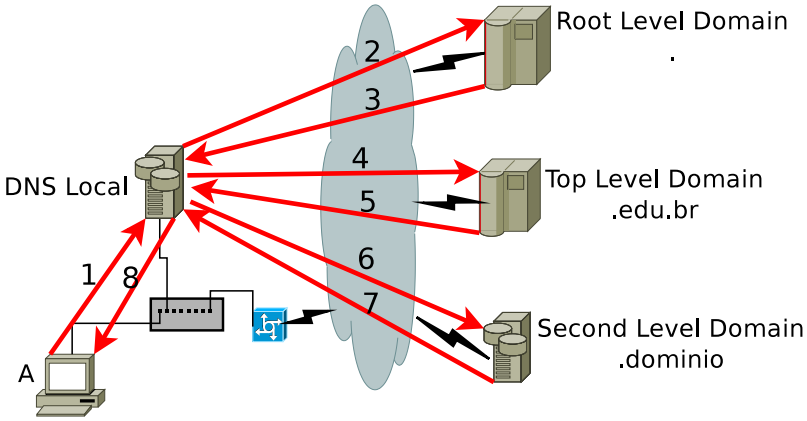

Figura 13.1: Resolução Recurviva de Nomes

bastante utilizado do mesmo são os apelidos ou *aliases* que servem para atribuir mais de um nome a mesma máquina, por exemplo: `www.sj.ifsc.edu.br` ⇒ `hendrix.sj.ifsc.edu.br`.

Pode-se também fazer consultas explícitas às bases, para isto existem várias ferramentas, entre elas o `nslookup` e o `dig`. Por exemplo a Listagem 13.1 mostra uma consulta recursiva ao nome `www.sj.ifsc.edu.br.`, que segue o modelo apresentado na Figura 13.1, diferenciando-se pelo fato de neste caso termos um `domínio delegado`. Entre as linhas 3 e 15 são listados todos os *Root Level Domain* disponíveis para consulta. Na linha 16 é explicitado qual foi o servidor DNS local consultado, neste caso `172.18.0.1` que é o servidor DNS da máquina usada no teste. Entre as linhas 18 e 23 são listados os *Top Level Domain* disponíveis para consulta, estes dados foram obtidos do `B.ROOT-SERVERS.NET`, linha 24. Nas linhas 26 e 27 são mostrados os *Second Level Domain* responsáveis pelo domínio `.ifsc.edu.br.`, informação obtida de `B.DNS.br`, linha 28. A máquina `artemis.ifsc.edu.br`, linha 33, repassa a consulta à máquina `hendrix.sj.ifsc.edu.br`, linha 30, que é delegada pelo subdomínio `.sj.ifsc.edu.br`. Esta finalmente, responde à consulta, informando o IP equivalente da máquina `www.sj.ifsc.edu.br.`, linhas 35 à 39.

```
; <<>> DiG 9.6.1-P1 <<>> +trace www.sj.ifsc.edu.br
;; global options: +cmd
.           430735   IN NS B.ROOT-SERVERS.NET.
.           430735   IN NS D.ROOT-SERVERS.NET.
.           430735   IN NS F.ROOT-SERVERS.NET.
.           430735   IN NS M.ROOT-SERVERS.NET.
.           430735   IN NS I.ROOT-SERVERS.NET.
.           430735   IN NS L.ROOT-SERVERS.NET.
.           430735   IN NS G.ROOT-SERVERS.NET.
.           430735   IN NS H.ROOT-SERVERS.NET.
.           430735   IN NS K.ROOT-SERVERS.NET.
.           430735   IN NS E.ROOT-SERVERS.NET.
.           430735   IN NS C.ROOT-SERVERS.NET.
.           430735   IN NS J.ROOT-SERVERS.NET.
.           430735   IN NS A.ROOT-SERVERS.NET.
;; Received 440 bytes from 172.18.0.1#53(172.18.0.1) in 3 ms

br.         172800   IN NS E.DNS.br.
br.         172800   IN NS F.DNS.br.
br.         172800   IN NS A.DNS.br.
br.         172800   IN NS B.DNS.br.
br.         172800   IN NS C.DNS.br.
br.         172800   IN NS D.DNS.br.
;; Received 288 bytes from 192.228.79.201#53(B.ROOT-SERVERS.NET) in
    272 ms

ifsc.edu.br.        86400 IN NS artemis.ifsc.edu.br.
ifsc.edu.br.        86400 IN NS hermes.ifsc.edu.br.
;; Received 122 bytes from 200.189.40.10#53(B.DNS.br) in 24 ms

sj.ifsc.edu.br.         3600   IN NS hendrix.sj.ifsc.edu.br.
sj.ifsc.edu.br.         3600   IN NS ns.pop-ufsc.rct-sc.br.
sj.ifsc.edu.br.         3600   IN NS ns.pop-udesc.rct-sc.br.
;; Received 134 bytes from 200.18.10.1#53(artemis.ifsc.edu.br) in 9 ms

www.sj.ifsc.edu.br.   3600   IN A   200.135.37.65
sj.ifsc.edu.br.         3600   IN NS ns.pop-udesc.rct-sc.br.
sj.ifsc.edu.br.         3600   IN NS hendrix.sj.ifsc.edu.br.
sj.ifsc.edu.br.         3600   IN NS ns.pop-ufsc.rct-sc.br.
;; Received 182 bytes from 200.135.37.65#53(hendrix.sj.ifsc.edu.br) in 0
    ms
```

Listing 13.1: dig +trace www.sj.ifsc.edu.br

13.4 Instalação e configuração

Como primeiro passo deve-se instalar o pacote Bind[1] com o comando abaixo. Com isto todos os pacotes e arquivos padrão de configuração do `Bind` serão instalados.

urpmi bind

13.4.1 Caso de estudo

Para verificar o funcionamento do DNS propõem-se a estrutura lógica mostrada na Figura 13.2. Esta estrutura permite testes em vários servidores separados, podendo cada servidor ser administrado por uma pessoa diferente, e permite também verificar a troca de informações entre servidores. Para permitir testes simples a máquina `M1` terá uma cópia de todos os arquivos (*zone files*) de todos os domínios e será a única a ter domínio reverso de todas as máquinas. Como ela terá todos os dados esta deverá ser usada como servidor DNS de todos.

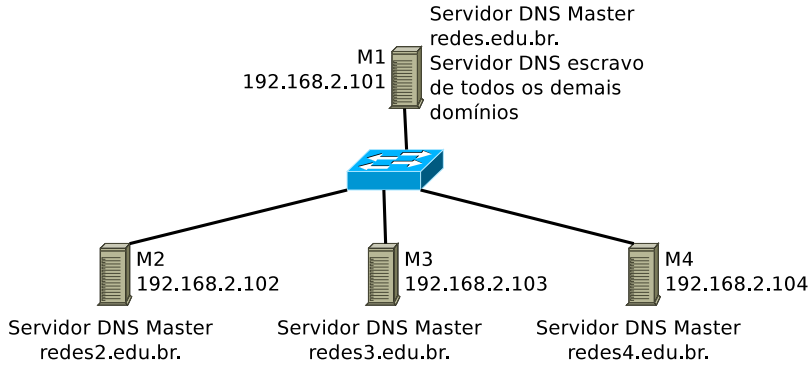

Figura 13.2: Diagrama do caso de estudo DNS

[1] https://www.isc.org/software/bind

Configuração do servidor principal

Para o servidor DNS na máquina `M1` deve-se configurar o arquivo da zona direta e reversa para o domínio `redes.edu.br` e também configurá-lo para ser escravo dos demais domínios. Para tanto inicia-se configurando o arquivo `/etc/named.conf`. Neste arquivo são configurados vários parâmetros, para uma configuração simples mantem-se os valores padrão para estes parâmetros e acrescentam-se as declarações das novas zonas que serão criadas. Para isto basta acrescentar ao final do arquivo as linhas da Listagem 13.2. Na linha 1 cria-se uma nova zona `redes.edu.br`, que será do tipo `master`, linha 2. O arquivo de especificação (*zone file*) desta zona será o `redes.zone`, que estará contido dentro do diretório `master` do diretório padrão dos "zone files" do `Bind`. Entre as linhas 6 e 9 declara-se a zona reversa, segue o mesmo padrão anterior com a diferença que o nome da zona é composto parcialmente pela faixa de IPs da rede e o restante segue o padrão da RFC. Entre as linhas 11 e 27 são declarados as zonas escravas com seus respectivos masters.

```
1  zone "redes.edu.br" IN {
2      type master;
3      file "master/redes.zone";
4  };
5
6  zone "2.168.192.in-addr.arpa" IN {
7      type master;
8      file "reverse/redes.rev";
9  };
10
11 zone "redes2.edu.br" IN {
12     type slave;
13     file "slaves/redes2.zone";
14     masters { 192.168.2.102; };
15 };
16
17 zone "redes3.edu.br" IN {
18     type slave;
19     file "slaves/redes3.zone";
20     masters { 192.168.2.103; };
21 };
22
```

```
23 zone "redes4.edu.br" IN {
24     type slave;
25     file "slaves/redes4.zone";
26     masters { 192.168.2.104; };
27 };
```

Listing 13.2: Anexos ao /etc/named.conf

Agora edita-se o arquivo `/var/lib/named/var/named/master/redes.zone` de acordo com o modelo da Listagem 13.3. Na linha 1 tem-se a definição do TTL (*Time To Live*) que neste caso significará por quanto tempo, em segundos, que a cópia permanecerá no *cache* do servidor que o copiou. Na linha 2 tem-se a demarcação do início da zona (*Start Of Authority*) seguida do endereço eletrônico do responsável pelo domínio. Na terceira limha tem-se o número serial do arquivo. Este número deve ser incrementado a cada alteração do arquivo, sob pena de inconsistência com as cópias *cache* espalhadas pela Internet. A recomendação que este número seja composto pela data, na ordem inversa, mais dois dígitos para o número serial do dia. As linhas 4 e 5 afetam principalmente o mantenedor da zona e os prestadores de serviços secundários e pode ser negociado entre eles. O campo *expiry*, linha 6, define a validade dos dados da zona, portanto os dados de *cache* podem ser utilizados mesmo que o servidor master esteja fora do ar. A linha 7 determina o valor padrão para todos os TTL dos registros sem valor explícito. Na linha 8 e 9 estão declarados quais são os servidores de nomes (*Name Servers*) responsáveis pelo domínio. Nas linhas 10 e 11 os servidores de correio eletrônico, sendo que o `mail1` é o principal, prioridade `0`, e o `mail2` o secundário, prioridade `10`. Na linha 12 é definido o endereço (*Address*) da máquina `localhost`. Na linha 13 o `$ORIGIN` define a extensão que valerá todos os endereços definidos na sequência. Da linha 14 em diante são definidos todos os endereços de máquinas do domínio, por exemplo a máquina `m1.redes.edu.br.` terá o endereço `192.168.2.101`. Cabe salientar que as declarações entre as linhas 18 e 21 servem para fazer um balanceamento de carga entre os servidores declarados. Neste caso, a cada consulta o DNS entregará um IP distinto, num esquema

de rodízio, para o solicitante, sendo assim, cada servidor receberá aproximadamente um quarto das consultas.

```
1  $TTL 86400
2  @       IN SOA   m1.redes.edu.br  root  (
3                   2009123100      ;  serial   (d.  adams)
4                   3H      ;  refresh
5                   15M       ;  retry
6                   1W      ;  expiry
7                   1D )        ;  minimum
8                IN NS       m1.redes.edu.br.
9                IN NS       m7.redes.edu.br.
10       IN MX 0      mail1.redes.edu.br.
11       IN MX 10 mail2.redes.edu.br.
12 localhost     IN A        127.0.0.1
13 $ORIGIN redes.edu.br.
14 m1 A  192.168.2.101
15 m7 A  192.168.2.177
16 mail1  A   192.168.2.201
17 mail2  A   192.168.2.202
18 www  A   192.168.2.155
19 www  A   192.168.2.156
20 www  A   192.168.2.157
21 www  A   192.168.2.158
```

Listing 13.3: Arquivo de declaração da zona direta

Por fim edita-se o arquivo `/var/lib/named/var/named/master/redes.rev` de acordo com o modelo apresentado na Listagem 13.4. Este arquivo segue o mesmo padrão de configuração do `redes.zone`. Neste caso objetiva-se a declaração da zona reversa, ou seja, dado um número IP qual o seu nome equivalente. Entre as linhas 10 e 13 estão declarados os números IPs de 101 à 104, da classe 192.168.2 definido no arquivo `/etc/named.conf`, com seus nomes equivalentes, por exemplo `m1.redes.edu.br.`.

```
1  $TTL 86400
2  @       IN       SOA     m1.redes.edu.br.  root.m1.redes.edu.br.   (
3                                          2009123100 ; Serial
4                                          28800       ; Refresh
5                                          14400       ; Retry
```

```
                                        3600000     ; Expire
                                        86400 )     ; Minimum
    IN      NS      m1.redes.edu.br.
        IN      NS      m7.redes.edu.br.
101     IN      PTR     m1.redes.edu.br.
102     IN      PTR     m2.redes2.edu.br.
103     IN      PTR     m3.redes3.edu.br.
104     IN      PTR     m4.redes4.edu.br.
```

Listing 13.4: Arquivo de declaração da zona reversa

Inicia-se o serviço de DNS com o comando:

service named start

Verifica-se o arquivo de *log* em busca de possíveis problemas, neste *log* serão indicados o arquivo e linhas dos mesmos. Usa-se o comando:

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

Configurando os demais servidores

Para os demais servidores basta a configuração do arquivo `/etc/named.conf` e o *zone file*. No arquivo `/etc/named.conf` deve-se acrescentar as linhas conforme o exemplo para a máquina M2, ver Listagem 13.5. O arquivo de declaração da zona direta deve ficar conforme o exemplo da Listagem 13.6. Lembre-se que para este caso de estudo, estes servidores não terão a zona reversa configurada.

```
zone "redes2.edu.br" IN {
    type master;
     file "master/redes2.zone";
};
```

Listing 13.5: Anexos ao /etc/named.conf para M2

```
$TTL 86400
@       IN SOA  m2.redes2.edu.br root (
                        2009123100      ; serial (d. adams)
                        3H      ; refresh
                        15M     ; retry
                        1W      ; expiry
                        1D )    ; minimum
                IN NS     m2.redes2.edu.br.
                IN NS     m15.redes2.edu.br.
        IN MX 0     mail.redes2.edu.br.
localhost     IN A      127.0.0.1
$ORIGIN redes2.edu.br.
m2 A  192.168.2.102
mail   A  192.168.2.102
www  A  192.168.2.102
```

Listing 13.6: Arquivo de declaração da zona direta para M2

Inicia-se o serviço de DNS com o comando:

service named start

Verifica-se o arquivo de *log* em busca de possíveis problemas, neste *log* serão indicados o arquivo e linha do mesmo que estão com problemas. Use o comando:

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

Testes de funcionamento

Como primeiro teste configura-se a máquina `M2`, por exemplo, para ser sua própria cliente, editando o `/etc/resolv.conf` e adicionando a diretiva `nameserver 192.168.2.102` no início do arquivo. Em seguida dá-se um `ping` para `m2.redes2.edu.br` (a própria máquina). Se "pingar" é sinal de que o próprio servidor está funcionando.

Para poder resolver os nomes das demais máquinas deve-se configurar as máquinas para serem cliente DNS da máquina `M1`, editando o arquivo `/etc/resolv.conf` e deixando-o com o conteúdo:

```
nameserver 192.168.2.101
```

Em seguida pode-se testar a resolução de nomes com qualquer cliente ou com as ferramentas específicas, como o `dig` ou `nslookup`. Por exemplo, para testes de resolução direta usa-se o `dig m2.redes2.edu.br` e para resolução reversa `dig -x 192.168.2.102`.

Capítulo 14

Servidor Apache

14.1 Introdução

Um servidor web é um programa de computador, ou o computador que executa este programa, responsável por responder a pedidos HTTP (*Hypertext Transfer Protocol*) feitos por clientes incluindo opcionalmente dados. As requisições geralmente são páginas web, tais como documentos HTML com objetos embutidos como imagens, arquivos etc.

O servidor Apache[1][2] (*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.

O servidor é compatível com o protocolo HTTP versão 1.1. Suas funcionalidades são mantidas através de uma estrutura de módulos, permitindo inclusive ao 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..

[1] http://pt.wikipedia.org/wiki/Servidor_Apache
[2] http://www.apache.org/

14.2 Instalação e configuração

Como primeiro passo deve-se instalar o pacote com o comando:

urpmi apache apache-doc

Em seguida inicia-se o servidor com o comando:

service httpd start

O servidor Apache já estará rodando numa configuração padrão que pode atender boa parte das demandas deste tipo de serviço. Testes podem ser feitos usando um navegador acessando o endereço `http://localhost/`. Pode ser acessado também o endereço `http://localhost/manual/`, que contém o manual (`apache-doc`) do servidor com uma série de textos explicativos sobre possíveis configurações e *links* importantes.

O principal arquivo de configuração do Apache é o `/etc/httpd/conf/httpd.conf` que está dividido em três seções: global, opções do servidor e máquinas virtuais. Além deste podem ser adicionados novos módulos, via `include` no arquivo principal, que permitem ampliar as funcionalidades do servidor.

As páginas publicadas ficam, por padrão, no diretório `/var/www/html`. Como um primeiro teste pode-se sobrescrever o conteúdo do arquivo `index.html`, deste diretório, pelo exposto abaixo. Fas-ze um novo acesso à `http://localhost/` e verifica-se o resultado.

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

A seguir serão apresentadas várias (re)configurações possíveis, seja implementadas no arquivo principal ou seja pela adição de novos módulos.

14.2.1 Direcionamento

A diretiva `Listen` diz ao servidor para aceitar requisições somente da porta ou da tupla <endereço, porta >. Por exemplo, para fazer o servidor aceitar conexões na porta 80 e 8080 em todas as interfaces basta:

```
Listem 80
Listem 8080
```

Para o servidor aceitar conexões na porta 80 numa interface e na porta 8080 em outra:

```
Listem 192.168.2.1:80
Listem 192.168.2.10:8080
```

14.2.2 Hospedeiros Virtuais

O termo `Hospedeiro Virtual` refere-se a prática de hospedar mais de um sítio web num mesmo servidor Apache. O Hospedeiro Virtual pode ser baseado em IP, ou seja, tem-se um número IP distinto para cada sítio web, ou baseado em nome, significando que tem-se múltiplos nomes para o mesmo IP. O fato de estar rodando no mesmo servidor Apache não será visível ao usuários que acessarem os diversos Hospedeiros.

Como um teste rápido cria-se um domínio virtual baseado em IP. Como primeiro passo deve-se criar um apelido de IP, conforme Seção 12.2.1, por exemplo 192.168.2.188. Em seguida cria-se o arquivo /etc/httpd/conf/vhosts.d/vhosts.conf com o seguinte conteúdo:

```
<VirtualHost 192.168.2.188>
ServerAdmin webmaster@redes.edu.br
DocumentRoot /var/www/docs/virtual
ServerName virtual.redes.edu.br
ErrorLog logs/virtual-error_log
```

```
TransferLog logs/virtual-access_log
<\slash VirtualHost>
```

<VirtualHost >e </VirtualHost >são usados para delimitar um conjunto de diretrizes que se aplicam apenas a um Hospedeiro Virtual específico. Qualquer diretiva que é permitido em um contexto de acolhimento virtual pode ser utilizada. As diretivas apresentadas no exemplo são auto-explicativas, sendo a principal a `DocumentRoot` que especifica o diretório que conterá os arquivos html e outros relativos ao Hospedeiro Virtual em questão. Quando o servidor recebe um pedido de um documento em um Hospedeiro Virtual específico, ele usa as diretivas de configuração incluído na seção <VirtualHost >.

Cria-se o diretório virtual com o comando:

mkdir -p /var/www/docs/virtual

Cria-se mais um arquivo `index.html` dentro deste diretório com o conteúdo:

```
<html><body><h1>Esta é minha página virtual.
Servidor 192.168.2.188</h1></body></html>
```

Reinicia-se o servidor, para que ele releia as configurações de domínios virtuais, com o comando:

service httpd restart

Agora testa-se e observa-se que possui-se duas páginas independentes acessando `http://192.168.2.1` e `http://192.168.2.188`.

14.2.3 Páginas de Usuários

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

urpmi apache-mod_userdir

Com isto, entre aoutras coisas, será criado um arquivo /etc/httpd/modules.d/67_mod_userdir.conf, que conterá as definições de "compartilhamento" dos diretórios pessoais. Este arquivo pode conter um contêiner com o seguinte formato:

```
<Directory \slash home\slash */public_html>
    AllowOverride All
    Allow from all
    Options Indexes FollowSymLinks MultiViews
    <IfModule mod_access.c>
      Order allow,deny
      Allow from all
    <\slash IfModule>
<\slash Directory>
```

O significado das diretivas é:

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 define o tipo de acesso.

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

Allow from all – Permite para todos.

Reinicia-se 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 e diretórios para permitir o *download* externo.

14.2.4 Restrição de acesso à páginas

O Apache permite que seja requisitado um `login` e `senha` para acesso a determinadas páginas. Para prover esta funcionalidade deve-se acrescentar algumas diretivas nos respectivos contêiners de definição das páginas ou acrescentar um arquivo chamado, por padrão, `.htaccess` na raiz de cada diretório das páginas a serem protegidas. Um exemplo do conteúdo deste arquivo:

```
require valid-user
AuthType Basic
AuthName "Aqui define-se o nome da janela que
          será aberta solicitando login e senha"
AuthUserfile /etc/httpd/conf/senhas.pw
```

O `login` e `senha` requisitados pelo Apache podem ser consultados em uma base LDAP, veja no Capítulo 18, ou podem ser formados por uma pequena base local do próprio Apache. Neste caso devem ser criados com o seguinte comando:

htpasswd -c /etc/hhtpd/conf/senhas.pw nome_do_usuario

Este comando diz ao Apache para criar (-c) o arquivo de senhas (/etc/hhtpd/conf/senhas.pw) para o usuário `nome_do_usuario`. Em seguida será requisitada a senha. Pode-se atribuir senhas a outros usuário repetindo-se o mesmo comando, simplesmente eliminando a *flag* `-c`.

14.2.5 Páginas em https

O módulo `mod_ssl` fornece uma interface para a biblioteca OpenSSL, que fornece criptografia forte usando os protocolos *Secure Sockets Layer* e *Transport Layer Security*. Para instalar este módulo, com um certificado auto-assinado, basta executar o seguinte comando:

urpmi apache-mod_ssl

Com isto o sistema já gera os certificados e permite que o acesso de **todas** as páginas, inclusive as dos usuários, sejam feitos com criptografia. Se desejado pode-se bloquear o acesso das páginas não certificadas através da diretiva `Listem` no arquivo `httpd.conf`. Juntamente com o módulo serão instalados os arquivos de configuração, iniciando pelo `/etc/httpd/ modules.d/40_mod_ssl.conf`, que define, entre outras coisas, a "escuta" no endereço/porta `0.0.0.0:443`.

Outro arquivo criado é o `/etc/httpd/conf/vhosts.d/01_default_ssl_vhost.conf` que cria um servidor virtual para as páginas `https`, que entre outras coisas tem um diretiva – `SSLCertificateKeyFile` – que aponta para a chave criptográfica que fica localizada no arquivo `/etc/pki/tls/certs/localhost.crt`. Esta chave é gerada no momento de instalação do módulo.

Capítulo 15

Servidor Postfix

15.1 Introdução

Um servidor de correio eletrônico[1] 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 pode-se citar o `Postfix`[2][3] , que é um dos candidatos a substituir o `SendMail`. O Postfix é hoje uma das melhores alternativas para todas as insti-

[1] http://pt.wikipedia.org/wiki/Sistema_de_correio_eletrônico
[2] http://www.postfix.org/postconf.5.html
[3] http://www.postfix.org/

tuições que desejam utilizar um servidor de e-mail 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 apresentar grande facilidade para sua configuração ele é um servidor de e-mail robusto e com vários recursos.

15.2 Funcionamento do Correio Eletrônico

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 (*Mail User Agent*). Ao terminar de redigir a sua mensagem o MUA enviará a mensagem a um MTA (*Mail Transfer Agent*), 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 (*Simple Mail Trasnfer Protocol*) ou Protocolo Simples de Transferência de Mensagens. O protocolo SMTP será utilizado também entre o MTA do remetente e o MTA do destinatário. Figura 15.1.

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 possui uma conta `shell` neste servidor, ele poderá ler os seus e-mails direto no servidor, caso contrário o usuário deverá ler sua mensagens 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 (*Post Office Protocol*), Protocolo de "Agência" de Correio, que recebe este

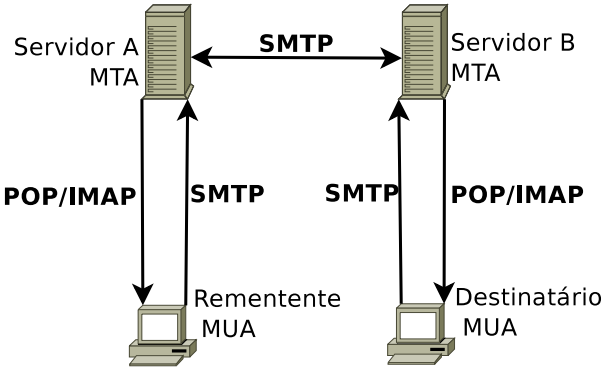

Figura 15.1: Protocolos para troca de mensagens de Correio Eletrônico

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 (*Internet Message Access Protocol*), 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.

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.

15.3 Instalação e configuração

Deve-se instalar os pacotes com o comando:

urpmi postfix

O servidor de correio eletrônico[4] exige o funcionamento em conjunto com um servidor DNS, ou melhor, é necessário que exista um servidor DNS apontando para a máquina que pretende-se instalar o `Postfix`. Para configurar o Servidor DNS corretamente veja o Capítulo 13. Para funcionamento na máquina local, trocando mensagens somente entre os usuários da própria máquina, o Postfix já vem pronto, bastando iniciá-lo. Para funcionar como um servidor de Intranet ou Internet deve-se configurar o "segundo bloco" do arquivo `/etc/postfix/main.cf`, acrescentando/mudando somente os parâmetros apresentados na Listagem 15.1

```
1  # User configurable parameters
2  myhostname = m1.redes.edu.br
3  mydomain = redes.edu.br
4  myorigin = $mydomain
5   inet_interfaces = all
6  mynetworks_style = subnet
7   mydestination = $myhostname, $mydomain
8  # Filtros ..
9    smtpd_recipient_restrictions =
10   permit_mynetworks,
11    reject_invalid_hostname ,
12    check_client_access hash:/etc/postfix/access,
13   reject_non_fqdn_hostname,
14    reject_non_fqdn_sender ,
15    reject_non_fqdn_recipient ,
16    reject_multi_recipient_bounce ,
17   reject_sender_login_mismatch ,
18    permit_sasl_authenticated ,
19    reject_unauth_destination ,
20   reject_unknown_client ,
21   reject_unknown_sender_domain,
22   reject_unknown_recipient_domain,
23     reject_rbl_client relays.ordb.org,
24     reject_rbl_client opm.blitzed.org,
25     reject_rbl_client list.dsbl.org,
26     reject_rbl_client sbl.spamhaus.org,
```

[4] http://www.conectiva.com/doc/livros/online/10.0/servidor/pt_BR/ch11.html

```
27    reject_rbl_client    cbl.abuseat.org,
28    reject_rbl_client    rbl. brasilrbl .com.br,
29    reject_rhsbl_client    rhsbl. brasilrbl .com.br,
30   reject_rhsbl_sender   dsn.rfc-ignorant.org
```

Listing 15.1: Principais diretivas do arquivo `/etc/postfix/main.cf`

Algumas diretivas tem função evidente e não serão comentadas. Na linha 4 define-se a extensão do e-mail, ou seja, o campo a ser colocado após o `@`. O `inet_interfaces`, linha 5, define em quais interfaces o servidor de correio vai atender. `mynetworks_style`, linha 6, define a rede de confiança do servidor, ou seja, quais os clientes poderão acessá-lo. Pode ser `class`, confia em toda a "classe" de rede onde o servidor está alocado, `subnet` confia somente na sub-rede local, *host* confia somente na própria máquina. Na linha 7 é definido os domínios de e-mail aos quais o servidor receberá correspondências.

A filtragem de *spans* sempre é um motivo de preocupação para os administradores do Postfix. Como primeira tentativa de evitá-los são propostas algumas regras, entre as linhas 9 e 31, novamente algumas são auto explicativas. Na linha 12 é definido um arquivo onde pode-se colocar manualmente endereços de email ou endereços de domínios que serão bloqueados, isto serve para evitar envios reincidentes de fontes maliciosas e que não são filtradas pelas outras regras. Na linha 16 define-se o código padrão de resposta ao endereço bloqueado. Na linha 17 são rejeitados os e-mails de usuários onde não há casamento entre o usuário logado e o nome de usuário que envia o e-mail, ou seja, quando um usuário tenta se passar por outro. O `reject_rbl_client`, linhas 23 à 28, consulta o sítio informado e verifica uma lista reversa de endereços de rede contendo possíveis *spammers*, caso o emitente esteja nesta lista seu e-mail será rejeitado. É o mesmo caso para `reject_rhsbl_client` a diferença é que neste caso a consulta é para endereços diretos. É possível a consulta a vários sítios. Deve-se tomar cuidado com este tipo de consulta já que as mesmas podem apresentar falso positivo, ou seja, conterem um endereço que efetivamente não é gerador de *spam*.

Para aprimorar a configuração deve-se adicionar regras para minimizar o tráfego de spams e correspondências infectadas por vírus. Uma boa ferramenta para tal é o `Greylinsting`, encontrado em `http://projects.puremagic.com/greylisting/`. Além disto, pode-se usar um anti-vírus como apresentado mais adiante no Capítulo 26.

Outra diretiva interessante a se acrescentar em um caso real é `home_mailbox = Maildir/`. Esta diretiva salvará os e-mails destinados a determinado usuário em seu diretório `home`, dentro de uma pasta chamada `Maildir`. Com as vantagens de poder contabilizar os arquivos em sua cota, caso exista, e deixar os mesmos fisicamente alocados na pasta do usuário, garantindo livre acesso e permissionamento.

O Postfix também permite que se crie grupos de usuários (listas ou refletores). Para isto basta editar o arquivo `/etc/postfix/alias` e criar diretivas do tipo `nome_do_grupo: usuário1, usuário2, ... , usuárion`. Executar o comando `newaliases` e recarregar o Servidor. Assim toda vez que chegar um e-mail destinado à nome_do_grupo@dominio.edu.br este será copiado para todos os usuários relacionados. Poder-se-ia também criar listas de discussão, onde também é possível a criação de grupos de email e ter-se um maior controle das trocas de mensagens. Uma boa ferramenta para isto é o `Mailman`[5].

Finalmente inicializa-se o serviço com:

service postfix start

15.4 Testes

Para realizar alguns testes no servidor deve-se ter uma ferramenta cliente de e-mail. Para isto pode-se usar uma ferramenta a nível de linha de comando, `mail` (se não existir deve-se instalar o pacote `mailx`).

[5] http://www.gnu.org/software/mailman/index.html

Para enviar uma mensagem procede-se do seguinte modo:

```
mail usuario@redes.edu.br <Enter>,
inserir o subjet <Enter>,
inserir a mensagem. <Enter>
Continuar a mensagem... <Enter>
<Ctrl>+<d>.
```

Como primeiro teste pode-se enviar uma mensagem, como exposto acima, para um usuário da própria máquina e monitorar a troca de mensagens ou eventuais problemas no arquivo de *log* com o comando:

tail -f /var/log/mail/info

Se aparecer algo do tipo apresentado na Listagem 15.2 é porque está tudo certo. O principal aviso é o `status=sent`, linha 4.

```
1 Mar 23 09:56:29 m1 postfix/pickup[15108]: 81A7C2C44C80: uid=1572 from
      =<usuario>
2 Mar 23 09:56:29 m1 postfix/cleanup[15113]: 81A7C2C44C80: message-id
      =<20070323125629.81A7C2C44C80@m1.redes.edu.br>
3 Mar 23 09:56:29 m1 postfix/qmgr[15109]: 81A7C2C44C80: from=<
      usuario@redes.edu.br>, size=454, nrcpt=1 (queue active)
4 Mar 23 09:56:29 m1 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)
5 Mar 23 09:56:29 m1 postfix/qmgr[15109]: 81A7C2C44C80: removed
```

Listing 15.2: Exemplo de *log* do Postfix

Uma vez que esteja funcionando localmente, pode-se enviar email para qualquer outro suário da Internet. Lembrando que os e-mails externos não chegarão/retornarão ao servidor, caso não se tenha um domínio válido, devidamente cadastrado no `Registro.br`.

Para ler mensagens basta digitar mail, aparecerá uma listagem de e-mails, e em seguida o número da mensagem que se pretende ler.

Capítulo 16

Servidor Samba

16.1 Introdução

O SAMBA[1] 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© pode-se navegar pelos computadores que estão disponíveis pela sub-rede e acessar seus recursos compartilhados. O servidor WINS, *Windows Internet Name Server*,

[1] http://www.samba.org/

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.

16.2 Instalação e configuração

Para instalar o samba basta executar o comando:

urpmi samba

O principal arquivo de configuração do samba é o `/etc/samba/smb.conf`. Este arquivo é dividido em duas seções, `global` e referente aos compartilhamentos. Deve-se editá-lo e modificar/acrescentar as diretivas segundo apresentado na Listagem 16.1.

```
1  [global]
2      workgroup = redes
3      netbios name = m1
4      security = user
5      encrypt passwords = yes
6      smb passwd file = /etc/samba/smbpasswd
7      passwd program = /usr/sbin/smbldap-passwd -u "%u"
8      passwd chat = "Mudando senha para*\nNew password*" %n\n "*Redigite a
           nova senha" %n\n"
9      ldap suffix = dc=redes,dc=edu,dc=br
10     ldap admin dn = "cn=Manager,dc=redes,dc=edu,dc=br"
11     passdb backend = ldapsam:ldap ://127.0.0.1/
12     ldap user suffix = ou=People
13     ldap group suffix = ou=Group
14     ldap machine suffix = ou=People
15     add user script = /usr/sbin/smbldap-useradd -m "%u"
```

```
16      ldap delete dn = No
17      add machine script = /usr/sbin/smbldap-useradd -w "%u"
18      add group script = /usr/sbin/smbldap-groupadd -p "%g"
19      add user to group script = /usr/sbin/smbldap-groupmod -m "%u" "%g
            "
20      delete user from group script = /usr/sbin/smbldap-groupmod -x "%u"
            "%g"
21      load printers = yes
22      printcap name = cups
23  [homes]
24      comment = Home Directories
25      browseable = no
26  [software]
27      comment = Softwares
28      path = /dados/software
29      guest ok = no
30      public = no
31      writable = yes
32      browseable = yes
33      force create mode = 0555
34      force directory mode = 0555
35      veto files = /*.mp3/
36      valid users = user1,user2
37      force group = admin
```

Listing 16.1: Principais diretivas do arquivo `smb.conf`

Na linha 2, da Listagem 16.1, é definido o grupo de trabalho ou Domínio Windows© à qual o servidor pertencerá. Na linha 3 define-se o nome da máquina que aparecerá nos compartilhamentos Windows©. Entre as linhas 9 e 21 são definidos parâmetros para integração com a base LDAP, veja detalhes no Capítulo 18. Entre as linhas 25 e 26 é definido o compartilhamento de todos os diretórios homes dos usuários cadastrados no servidor. Entre as linhas 28 e 38 é definido o compartilhamento do diretório `/dados/software`, linha 29, com nome `software`, linha 27, que pode ser escrito, linha 32, navegável, linha 33, onde novos arquivos e diretórios serão criados com o permissionamento 555 (ver Capítulo 7), linhas 34 e 35. Arquivos com a extensão `mp3` não serão aceitos, linha 36. Somente os usuários `user1` e `user2` poderão acessar este compartilhamento, linha 237. Neste caso poderia-se também usar `@nome_do_grupo`,

para definir o grupo que teria permissão de acesso, evidentemente, desde que tenha acesso ao diretório propriamente dito. Finalmente, na linha 38 é definido o grupo dono dos novos arquivos e diretórios a serem criados. Aqui cabe uma observação, quando da criação de um arquivo ou diretório o Linux atribui, por padrão o grupo dono do mesmo como sendo o grupo primário do usuário que está executando a operação, com este artifício do Samba é possível mudar este comportamento.

Como o padrão de senhas do SMB é diferente do Linux deve-se criar as senhas "SMB" para os usuários da máquina. Isto é feito com o comando:

smbpasswd -a nome_de_usuario

Após a configuração deve-se iniciar ou reiniciar o serviço com o comando:

service smb start

16.3 Testes

A primeira verificação é feita pelo comando:

testparm

que verifica a integridade e coerência do arquivo `/etc/samba/smb.conf`. Caso exista alguma inconsistência no arquivo, a mesma será apontada neste momento.

Pode-se também testar os compartilhamentos disponíveis com o comando abaixo, que listará todos os compartilhamentos ativos no respectivo servidor.

smbclient -L redes -U nome_de_usuário

Para testes de compartilhamento propriamente dito, deve-se acessar uma máquina Windows©, logar com um usuário e senha cadastrado no servidor, acessar o `Windows Explorer` e fazer um mapeamento de rede apontando para `//redes/nome_de_usuario` ou `//192.168.2.1/nome_de_usuario`, ou seja, pelo nome do grupo de trabalho ou pelo número IP.

Também é possível o mapeamento com máquinas Linux, neste caso usa-se o comando `mount.cifs`, conforme o modelo:

mount.cifs //192.168.2.1/nome_de_usuário ./diretorio_local_de_montagem -o username=nome_de_usuário

Após isto pode-se fazer testes criando/copiando/removendo arquivos e diretórios, seja via máquina Windows© ou Linux.

Capítulo 17

Servidor NFS

17.1 Introdução

NFS – *Network File System* – é o sistema de compartilhamento de arquivos em rede local nativo Linux (Unix). 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.

17.2 Instalação e configuração

Para instalar o servidor NFS usa-se o comando:

urpmi nfs-utils

A configuração dos diretórios a serem compartilhados é feita por meio do arquivo `/etc/exports`, que por padrão não existe e de-

verá 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 é informado simplesmente o diretório, todas as máquinas terão permissão de escrita e leitura no dito diretório. Em `cliente` deve pode ser informado: o nome da máquina cliente ou seu IP; coringas de domínio como por exemplo *sj.ifsc.edu.br, que representa o conjunto de todas as máquinas do domínio `sj.ifsc.edu.br`; pares de endereço IP/máscara, por exemplo 192.168.2.0/24; *, qualquer máquina. As `opções` pode ser `ro` (*read only*), `rw` (*read and write*) e `no_root_squash`. Nesta última o usuário `root` do cliente passa a ter as mesmas permissões de `root` no servidor. Não recomenda-se o uso desta opção a não ser entre servidores e com muito cuidado. Deve-se observar que compartilhamentos de diretório com opção `rw` não garante que o usuário que fizer o mapeamento terá permissão de escrita, deve-se atentar para as permissões em si do diretório a ser compartilhado. Como regra geral o que vale é o conjunto de permissões mais restritivas.

Abaixo tem-se alguns exemplos. No primeiro, os computadores da rede 192.168.2.0 terão acesso para escrita e leitura ao diretório `/home`. No segundo, a máquina 192.168.2.7 poderá ler e a máquina 192.168.1.1 ler e escrever sendo que seu usuário `root` será "replicado", tendo permissões de `root` no servidor. No terceiro, a máquina `www.sj.ifsc.edu.br` terá acesso de leitura e escrita ao dito diretório.

```
\slash home 192.168.2.0/24(rw)
\slash usr 192.168.2.7(ro)  192.168.2.1(rw,no_root_squash)
\slash var/www/html www.sj.ifsc.edu.br(rw)
```

Após definir-se os compartilhamentos deve-se iniciar o serviço com a sequência de comandos:

service portmap restart

service nfs-server start

exportfs -a

17.3 Testes

Para verificar os compartilhamentos atuais em um determinado servidor usa-se o comando:

showmount –exports ip_do_servidor

Pode-se fazer isto a partir da própria máquina. O comando retornará a listagem de todos os diretórios compartilhados.

Agora pode-se montar o diretório compartilhado no cliente. Para fins de testes o cliente pode ser a própria máquina. Sendo assim, cria-se um diretório local, onde ocorrerá a montagem, com o comando:

mkdir nfs

e monta-se o compartilhamento neste diretório com o comando:

mount 192.168.2.X:/usr nfs

Pode-se conferir listando o conteúdo do diretório `nfs`, copiando arquivos de/para o dito diretório, com o comando `df` etc.

Capítulo 18

Servidor LDAP

18.1 Introdução

O LDAP – *Lightweight Directory Access Protocol* – 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. Possuem um forte padronização tanto na nomenclatura quando no tipo de dado.

Em redes locais, Figura 18.1, 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.

O schema é um conjunto de definições que regem o armazena-

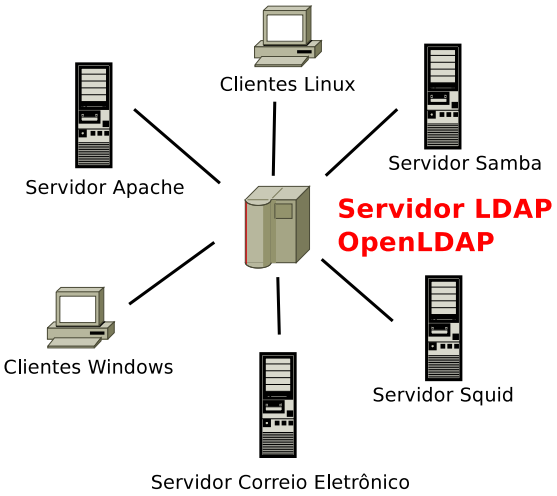

Figura 18.1: Interligação LDAP e outros serviços

mento de informações no serviço de diretório. Os `schemas` definem as classes de objetos, atributos, regras de ordenação e índices para atributos e heranças de classes de objetos.

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

18.2 Instalação e configuração

Para instalar o OpenLDAP[1] deve-se usar o comando:

urpmi openldap-server

Editar o arquivo `/etc/openldap/slapd.conf` trocar todas as ocorrências de `dc=example,dc=com` por `dc=redes,dc=edu,dc=br`

[1] http://www.openldap.org/

e modificar o parâmetro `rootpw SSHARUHmz+yqoZrZzfNCN5ewu/Ol77`, onde a senha criptografada (segundo campo) é obtida do resultado da execução do comando shell `slappasswd -c crypt`, que será o equivalente criptografado da senha fornecida.

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

```
BASE    dc=redes, dc=edu, dc=br
HOST    localhost
URI     ldap://localhost
```

Após as configurações iniciais é necessário definir os esquemas (*schemas*) e em seguida "popular" o banco de dados. Para estas tarefas recomenda-se o uso das ferramentas do pacote `smbldap-tools`, que, além do `schema` pré-definido, permite integração total com o Samba, que deve estar previamente instalado e configurado, ver Capítulo 16. Inicialmente deve-se instalar o pacote com o comando:

urpmi smbldap-tools

Iniciam-se os serviços com os comandos:

service smb (re)start

service ldap (re)start

Após a instalação deve-se gerar os `schemas` por meio de um *script* já desenvolvido que é o `configure.pl`. Esta ferramenta deve ser executada com o Samba e OpenLdap 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`, para a configuração padrão. Na

Listagem 18.1 é apresentada a sequência de perguntas que este *script* faz para gerar os arquivos específicos. As linhas iniciais da execução deste script foram propositalmente suprimidas, já que apresentam somente informações sobre o mesmo. Entre as linhas 3 e 17 são questionadas as configurações do Samba, veja que o próprio *script* faz uma leitura do arquivo `/etc/samba/smb.conf` e retira automaticamente os parâmetros do mesmo. Na linha 18 é requisitado o sufixo da base LDAP, deve-se fornecer de acordo com o pré-configurado nos arquivos, para o exemplo: `dc=redes,dc=edu,dc=br`. Nas linhas 29, 30, 34 e 35 é requisitado o nome do usuário administrador da base e sua senha, devendo esta ser a mesma inserida no arquivo `slapd.conf`. Nas linhas 52 e 53 é informado que os arquivos foram escritos e os mesmos podem ser futuramente consultados ou alterados.

```
1  Let's start configuring the smbldap-tools scripts ...
2
3  . workgroup name: name of the domain Samba act as a PDC
4    workgroup name [redes] >
5  . netbios name: netbios name of the samba controler
6    netbios name [m1] >
7  . logon drive: local path to which the home directory will be connected
       (for NT Workstations). Ex: 'H:'
8    logon drive [] >
9  . logon home: home directory location (for Win95/98 or NT Workstation).
10   (use %U as username) Ex:'\\redes\%U'
11   logon home (press the "." character if you don't want homeDirectory)
         [\\redes\%U] >
12 . logon path: directory where roaming profiles are stored. Ex:'\\redes\
       profiles\%U'
13   logon path (press the "." character if you don't want roaming profile )
         [\\redes\profiles\%U] >
14 . home directory prefix (use %U as username) [/home/%U] >
15 . default users' homeDirectory mode [700] >
16 . default user netlogon script (use %U as username) [] >
17   default password validation time (time in days) [45] >
18 . ldap suffix [] > dc=redes,dc=edu,dc=br
19 . ldap group suffix [] > ou=Group
20 . ldap user suffix [] > ou=People
21 . ldap machine suffix [] > ou=Computer
22 . Idmap suffix [ou=Idmap] >
23 . sambaUnixIdPooldn: object where you want to store the next uidNumber
```

```
24    and gidNumber available for new users and groups
25    sambaUnixIdPooldn object ( relative to ${ suffix }) [sambaDomainName=
          redes] >
26  . ldap master server: IP adress or DNS name of the master ( writable )
        ldap server
27    ldap master server [127.0.0.1] >
28  . ldap master port [389] >
29  . ldap master bind dn [] > cn=Manager,dc=exemplo,dc=com,dc=br
30  . ldap master bind password [] > insira_a_senha
31  . ldap slave server: IP adress or DNS name of the slave ldap server: can
          also be the master one
32    ldap slave server [127.0.0.1] >
33  . ldap slave port [389] >
34  . ldap slave bind dn [] > cn=Manager,dc=exemplo,dc=com,dc=br
35  . ldap slave bind password [] > insira_a_senha
36  . ldap tls support (1/0) [0] >
37  . SID for domain redes: SID of the domain (can be obtained with 'net
          getlocalsid redes')
38    SID for domain redes [S
          -1-5-21-1066659121-185135820-1519059970] >
39  . unix password encryption: encryption used for unix passwords
40    unix password encryption (CRYPT, MD5, SMD5, SSHA, SHA) [SSHA] >
41  . default user gidNumber [513] >
42  . default computer gidNumber [515] >
43  . default login shell [/bin/bash] >
44  . default skeleton directory [/etc/skel] >
45  . default domain name to append to mail adress [] >
46  -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
47  Use of uninitialized value in concatenation (.) or string at /usr/share/
        doc/smbldap-tools-0.9.2/configure.pl line 314, <STDIN> line 33.
48  backup old configuration files:
49    /etc/smbldap-tools/smbldap.conf->/etc/smbldap-tools/smbldap.conf.old
50    /etc/smbldap-tools/smbldap_bind.conf->/etc/smbldap-tools/smbldap_bind.
          conf.old
51  writing new configuration file:
52    /etc/smbldap-tools/smbldap.conf done.
53    /etc/smbldap-tools/smbldap_bind.conf done.
```

Listing 18.1: Script para geração de *schemas* – `configure.pl`

Em seguida deve-se "popular" o banco de dados, baseado nos *schemas* gerados com a execução do *script* anterior. Isto pode ser feito com o comando:

smbldap-populate

Este comando gerará uma saída informando todas as entradas (escritas) que serão feitas no banco de dados. Ao final será requisitada a senha para a escrita no mesmo.

A Listagem 18.2 representa parte do diretório gerado. A representação é feita por meio do formato texto chamado LDIF – *LDAP Data Interchange Format*. O conteúdo do diretório pode ser consultado a qualquer momento via o comando `slapcat`.

```
dn: dc=redes,dc=edu,dc=br
objectClass: dcObject
objectClass: organization
o: redes
dc: redes
structuralObjectClass: organization
entryUUID: b42dbf23-5002-4e97-bd3c-6b328f5c5243
creatorsName: cn=Manager,dc=redes,dc=edu,dc=br
createTimestamp: 20091127130738Z
entryCSN: 20091127130738.161251Z#000000#000#000000
modifiersName: cn=Manager,dc=redes,dc=edu,dc=br
modifyTimestamp: 20091127130738Z

dn: ou=People,dc=redes,dc=edu,dc=br
objectClass: top
objectClass: organizationalUnit
ou: People
structuralObjectClass: organizationalUnit
entryUUID: 33590f41-42f2-4fd3-ad41-6d6875066645
creatorsName: cn=Manager,dc=redes,dc=edu,dc=br
createTimestamp: 20091127130738Z
entryCSN: 20091127130738.188550Z#000000#000#000000
modifiersName: cn=Manager,dc=redes,dc=edu,dc=br
modifyTimestamp: 20091127130738Z

dn: uid=usuario,ou=People,dc=redes,dc=edu,dc=br
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: shadowAccount
```

```
33 uid: usuario
34 uidNumber: 1000
35 gidNumber: 513
36 homeDirectory: /home/usuario
37 loginShell : /bin/bash
38 gecos: System User
39  structuralObjectClass : inetOrgPerson
40 entryUUID: 0edf9c10-7149-4565-9e9b-77d488b20acd
41 creatorsName: cn=Manager,dc=redes,dc=edu,dc=br
42 createTimestamp: 20091127131021Z
43 userPassword:: e1NTF9T1NsTU11aXVqdEVyKzRoV25Cd2VqUTE=
44 shadowLastChange: 14575
45 shadowMax: 45
46 cn:: VXN1w4PCoXJpbyBUZXN0ZSBMREFQ
47 sn: LDAP
48 givenName:: VXN1w4PCoXJpbyBUZXN0ZQ==
49 entryCSN: 20091127131441.310265Z#000000#000#000000
50 modifiersName: cn=Manager,dc=redes,dc=edu,dc=br
51 modifyTimestamp: 20091127131441Z
```

Listing 18.2: Exemplo de um diretório

Agora pode-se 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, ver Capítulo 6. Entre as linhas 26 e 51, da Listagem 18.2, está representado um usuário que foi criado por meio destes comandos.

Além do já exposto o OpenLDAP, juntamente com o protocolo LDAP, algumas ferramentas para consulta e manutenção:

slapadd – Para acrescentar novas entradas nos `schemas`.

ldapmodify – Para modificar entradas já existentes nos `schemas`.

ldapsearch – Para buscas pelo diretório.

O Backend BDB possui algumas ferramentas para a manutenção do diretório propriamente dito, as principais são:

db_archive – Permite a manutenção dos arquivos de registros de transações do diretório.

db_recover – Permite a recuperação da base de dados no caso da ocorrência de algum problema. A recuperação se dá por meio dos arquivos de registros, normalmente localizados em /var/lib/ldap.

db_stat – Fornece estatísticas sobre o uso da base de dados que podem levar, por exemplo, a otimizações.

Para gerenciamento da base de dados, inserção e modificação de usuários, grupos etc. existem várias ferramentas, uma delas é a já citada `smbldap-tools`, mas existem também as ferramentas com suporte gráfico. Uma delas é a `phpLDAPadmin`[2], que funciona por meio de um navegador qualquer, na Figura 18.2 tem-se um exemplo desta interface, onde é possível se observar as opções de esquemas, importação, exportação, inserção etc. Outra é `LDAP Admin` para Windows©, um aplicativo de deve ser instalado neste tipo de sistema operacional.

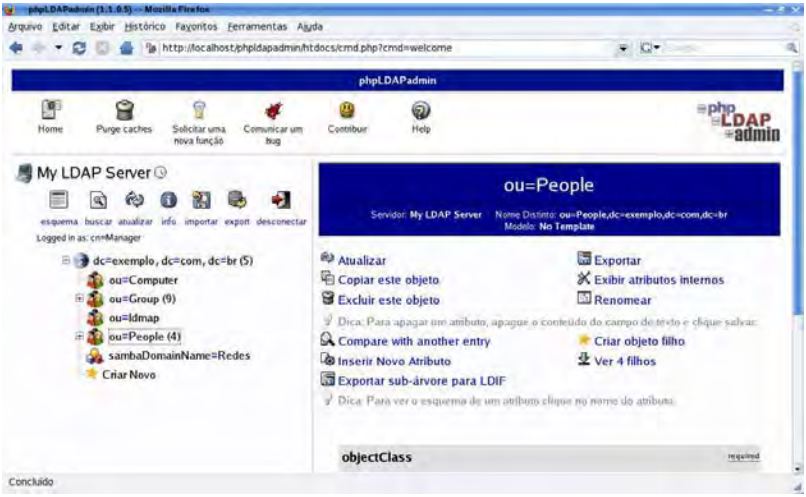

Figura 18.2: Exemplo de uso do phpLDAPadmin

Agora a base está pronta para consultas. Pode-se então configurar as máquinas, sejam clientes ou outros servidores de rede, para

[2] http://phpldapadmin.sourceforge.net/

fazer consultas à mesma.

18.2.1 Para configurar um cliente Linux

Para o Linux a configuração para consultas à base é feita primeiramente instalando os pacotes necessários para o cliente LDAP, com o seguinte comando:

> *urpmi nss_ldap openldap-client pam_ldap autofs pam_ccreds nss_updatedb*

Para o caso específico do `Mandriva` existe a opção de configuração com o programa `drakauth` (outras distribuições tem soluções similares), seguindo os passos recomendados pelo mesmo. As Figuras 18.3 e 18.4 mostram o procedimento necessário para configurar um cliente. Este programa configura devidamente os arquivos `/etc/openldap/ldap.conf`, `/etc/ldap.conf`, `/etc/nsswitch.conf`, `/etc/ldap.secret` e `/etc/pam.d/system-auth` para que seja possível a autenticação de usuários e consultas a base remota.

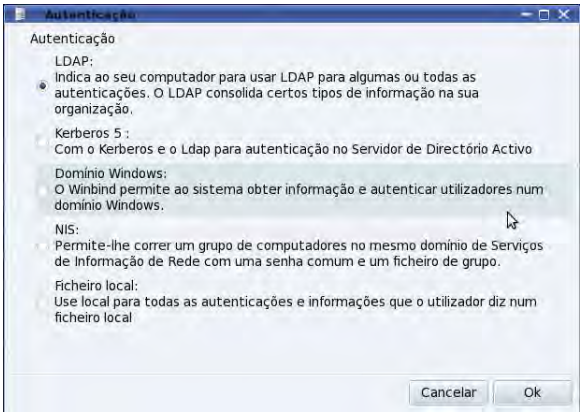

Figura 18.3: Janela 1 de configuração do DrakAuth

Figura 18.4: Janela 2 de configuração do DrakAuth

18.3 Testes

O primeiro teste é fazer uma consulta ao banco de usuários. Após a configuração do cliente tem-se todos os usuários locais mais os do servidor LDAP. Para a consulta use o comando:

getent passwd

Em seguida pode-se logar com um usuário devidamente cadastrado na base remota. No primeiro login deste usuário, será criado seu diretório pessoal (*home*), no restante o comportamento será exatamente o mesmo de um usuário local.

Capítulo 19

Servidor DHCP

19.1 Introdução

Conforme visto no Capítulo 12, para funcionamento em rede toda máquina necessita de alguns parâmetros mínimos em sua interface de rede. Em pequenas redes este trabalho pode ser feito manualmente, máquina à máquina, mas em uma rede grande, com centenas de máquinas, esta tarefa de configuração torna-se trabalhosa.

Para facilitar a vida do administrador foi criado um mecanismo de configuração automática das interfaces de rede de máquinas em uma rede TCP/IP: o DHCP (*Dynamic Host Configuration Protocol*).

Dentre os parâmetros que podem ser passados à máquina cliente por DHCP estão: endereço IP, máscara de sub-rede, roteador padrão, servidor(es) DNS, nome de host e/ou de domínio, servidores e domínio NIS, servidores WINS, servidores NTP, imagens de boot para terminais "burros" etc.

19.2 O protocolo DHCP

O protocolo DHCP[1] permite a configuração das interfaces de rede seguindo uma sequência de negociação, com trocas de mensagens com parâmetros da camada 2 e da camada 3, da pilha de protocolos TCP/IP. A seguir estão detalhadas estas trocas de mensagens.

DHCP Discover – Quando uma máquina é ligada, ela deve ter um serviço (*daemon*) cliente do DHCP configurado, para localizar o(s) servidor(s) DHCP. Este cliente DHCP envia um datagrama UDP com destino à porta 67 chamado `DHCP Discover`. Este datagrama *broadcast* tem o endereço IP de destino **255.255.255.255** e mac address de destino **ff:ff:ff:ff:ff:ff**, ver Figura 19.1. Todas as máquinas da sub-rede local receberão tal datagrama e somente os servidores DHCP responderão ao mesmo.

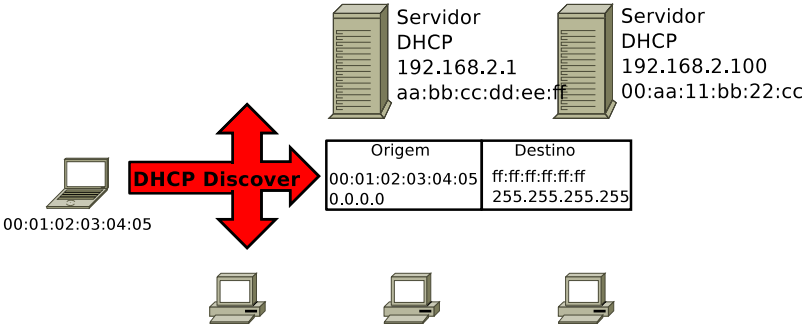

Figura 19.1: DHCP Discover

DHCP Offer – Os servidores, ao receber o referido datagrama em sua interface de rede, responderão ao solicitante com um datagrama destinado ao endereço mac deste, com endereço IP 255.255.255.255 (broadcast). O datagrama de resposta chama-se `DHCP Offer`. Este datagrama conterá todos os parâmetros

[1] http://sites.google.com/site/marciokatan/dhcp-mandriva

necessários para a configuração da interface de rede do solicitante, ver Figura 19.2. O servidor armazena (reserva) o número IP passado para evitar possíveis conflitos.

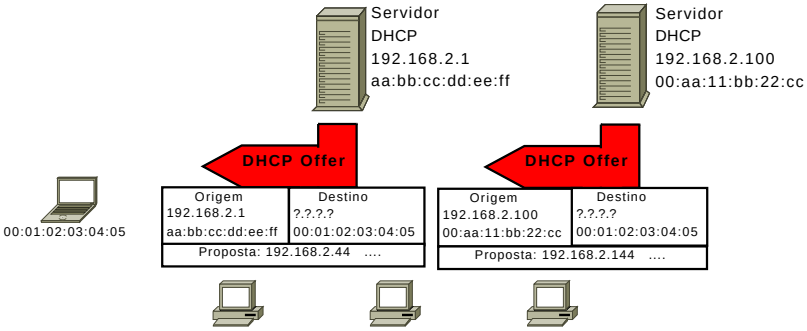

Figura 19.2: DHCP Offer

DHCP Request – O cliente ao receber o datagrama do servidor, decide se aceita ou não a configuração oferecida pois pode receber mais de uma oferta, como é o caso representado nas figuras. Em caso positivo, retorna um datagrama, já com seu número IP, comunicando ao servidor que aceitou sua oferta. Este datagrama chama-se `DHCP Request`, ver Figura 19.3.

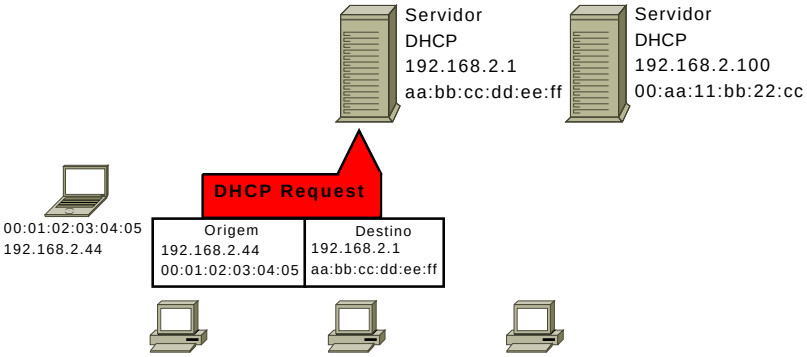

Figura 19.3: DHCP Request

DHCP Ack – Para finalizar a "conversação" entre cliente e

servidor DHCP, este finaliza (efetiva) o aluguel (*lease*) do endereço do cliente enviando àquele, um datagrama `DHCP Ack`, ver Figura 19.4. Armazena em seu arquivo de *cache* que o número IP passado está alugado, com parâmetros de tempo de aluguel etc. O servidor que não recebeu o `DHCP Request`, terá seu temporizador com tempo expirado e liberará o endereço previamente disposto.

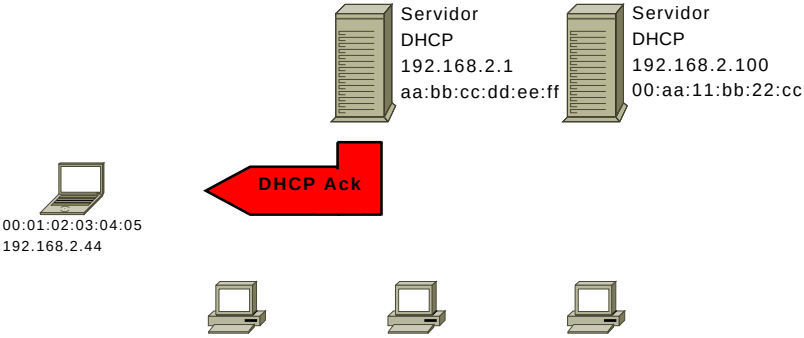

Figura 19.4: DHCP Ack

19.3 Instalação e configuração

Para instalar o servidor DHCP deve-se executar o seguinte comando:

urpmi dhcp-server

Edita-se o arquivo `/etc/dhcpd.conf` de acordo com o modelo apresentado na Listagem 19.1. Boa parte das opções são auto explicativas. Os parâmetros das linhas 1 e 2 são parâmetros globais e servem tanto para a primeira como para a segunda sub-rede declarada. Se um determinado parâmetro for colocado dentro do escopo de uma sub-rede e também como parâmetro global o que valerá é o parâmetro interno, o global só valerá para os escopos onde os mesmos não estão explicitamente definidos. Deve-se atentar para o fato que o servidor

DHCP atende somente rede(s) à(s) qual(is) pertence, portanto a sub-rede declarada deve ser compatível, pertencer à mesma rede, da qual sua(s) interface(s) pertença(m).

```
1  ddns-update-style none;
2  option domain-name-servers m1.redes.edu.br;
3  subnet 192.168.2.0 netmask 255.255.255.0 {
4            option routers 192.168.2.1;
5            option subnet-mask 255.255.255.0;
6            option domain-name "redes.edu.br";
7            range dynamic-bootp 192.168.2.30 192.168.2.60;
8            range dynamic-bootp 192.168.2.91 192.168.2.99;
9            default-lease-time 21600;
10           max-lease-time 43200;
11           host novo {
12                     hardware ethernet 12:34:56:78:AB:CD;
13                     fixed-address 192.168.2.88;
14           }
15 }
16 subnet 192.168.3.0 netmask 255.255.255.0 {
17           option routers 192.168.3.1;
18           option subnet-mask 255.255.255.0;
19           option domain-name "redes3.edu.br";
20           range dynamic-bootp 192.168.3.122 192.168.3.144;
21           default-lease-time 21600;
22           max-lease-time 43200;
23 }
```

Listing 19.1: Arquivo de Configuração do DHCP

A opção da linha 1 serve para indicar se o servidor DNS será atualizado quando um aluguel de ip for solicitado. Neste caso deve-se ter tanto o servidor DNS como o DHCP "conversando" para atualizar dinamicamente os arquivos do DNS. As linhas 7 e 8 definem a faixa de IPs que serão alugados para a respectiva sub-rede. A linha 9 define o tempo padrão de aluguel. Após este tempo o cliente tenta alugar novamente o mesmo ou um novo endereço. A linha 10 define o tempo máximo de aluguel. Quando este tempo for excedido a interface do cliente será desconfigurada e o mesmo não poderá mais acessar a rede. Entre as linhas 11 e 14 está definido uma máquina que receberá o nome `novo`, e sempre terá seu número IP igual a `192.168.2.88`, já que isto está diretamente atrelado ao seu *mac adress*. Entre as linhas

16 e 23 estão definidos os parâmetros que serão repassados a outra sub-rede (192.168.3.0).

Após a configuração do arquivo (re)inicia-se o servidor com o comando:

service dhcpd restart

19.4 Testes

Para fazer um teste pode-se usar o comando `dhclient eth0` numa máquina qualquer. Se o servidor estiver corretamente configurado este alugará algum IP e demais parâmetros de rede para esta máquina. Todas as trocas de mensagem para negociação do aluguel de IP podem ser observadas no arquivo de log `/var/log/messages`, ou capturadas com algum software de captura de pacotes. Pode-se também conferir os endereços alugados no arquivo `/var/lib/dhcp/dhcpd.leases`.

Capítulo 20

Servidor FTP

20.1 Introdução

FTP – File Transfer Protocol – é uma forma bastante rápida e versátil para transferência de arquivos entre máquinas remotas, sendo uma das mais usadas na Internet. É o padrão da pilha TCP/IP para transferir arquivos, é um protocolo genérico independente de *hardware* e do sistema operacional (A), levando em conta restrições de acesso e propriedades dos arquivos.

O serviço FTP é dos poucos que abre duas conexões TCP entre o cliente e o servidor: uma para controle, porta 21, e outra para transferência de dados, porta 20. Isto é conhecido como "controle fora de banda".

20.2 Instalação e configuração

Para instalar o servidor FTP deve-se primeiramente escolher um dos servidores disponíveis na distribuição dentre `Proftpd`, `Pure-ftpd`[1], `Vsftpd`[2] e `Wu-ftpd`[3]. Um dos mais usados na distribuição Mandriva

[1] http://www.proftpd.org/
[2] http://vsftpd.beasts.org/
[3] http://www.wu-ftpd.org/

é o Proftpd. Cabe salientar que todos são similares na funcionalidade e configuração. Para instalá-lo basta executar o comando:

urpmi proftpd

O Proftpd vem absolutamente pronto para uso, não sendo necessária nenhuma configuração preliminar. Mas ao desejar-se alguma configuração especial deve-se editar o arquivo /etc/proftpd.conf. Um dos parâmetros editáveis permite uma dose extra de segurança que 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 deve-se descomentar a opção DefaultRoot ∼.

Agora deve-se iniciar o serviço com o comando:

service proftpd start

20.3 Testes

Para testar pode-se usar a própria máquina como cliente. Para isto basta executar 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 deve-se informar o usuário. Em seguida será requisitada a senha e após esta conecta-se ao servidor remoto e pode-se usar praticamente todos os comandos normais do `shell` para visualizar, criar, modificar ou apagar arquivos e diretórios. Cabe salientar 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 nome_do_arquivo`, para enviar da máquina local ao servidor, e `get nome_do_arquivo`, no caso contrário. Pode-se a qualquer momento visualizar as opções de comandos disponíveis digitando-se `help`. Para sair da aplicação digita-se o comando `bye`.

Capítulo 21

Servidor SSH

21.1 Introdução

Em informática, o *Secure Shell*[1] ou SSH é, simultaneamente, um programa de computador e um protocolo de rede que permite a conexão segura 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.

21.2 Instalação e configuração

Para instalar o servidor OpenSSH[2], uma das implementações do serviço SSH, deve-se executar o seguinte comando:

[1] http://pt.wikipedia.org/wiki/SSH
[2] http://www.openssh.com/

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 pretender fazer alguns ajustes deve-se editar o arquivo `/etc/ssh/sshd_config`. A recomendação para este arquivo é descomentar somente o que pretende-se mudar do padrão. Se for descomentada uma linha e deixada no valor padrão podem ocorrem instabilidades no serviço. Os principais parâmetros que podem ser modificados ou acrescentados são:

PermitRootLogin no – O usuário `root` não poderá abrir um conexão `ssh` diretamente. Por questões de segurança reco-menda-se deixar `no`, logar como usuário normal e em seguida dar o comando `su`.

X11Forwarding yes – 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.

AllowUsers root user1 user2 – Se esta diretiva existir somente os usuários listados, `root`, `user1` e `user2`, no caso, poderão abrir conexão `ssh`.

AllowGroups grupo – Idem anterior para grupos.

DenyUsers root user1 user2 – O posto do `AllowUsers`.

DenyGroups grupo – Idem anterior para grupos.

UsePAM yes – Habilita, por exemplo, o *login* em bases LDAP.

Feitas as configurações pode-se (re)iniciar o serviço com o comando:

service sshd (re)start

21.3 Testes

Para testar pode-se usar a própria máquina como cliente. Para isto basta executar o comando:

ssh usuario@192.168.2.X

Com isto abre-se uma conexão segura com o servidor, podendo-se executar todos os comandos, como se logados localmente.

Pode-se também fazer cópias de arquivos com o `scp`. Por exemplo:

scp -r usuario@192.168.2.1:diretorio ./

Com este comando copia-se recursivamente o diretório `diretorio` do servidor 192.168.2.1 para o diretório corrente local.

Capítulo 22

Servidor Proxy/Cache

22.1 Introdução

O servidor Proxy/Cache[1] é como uma espécie de "*cache* comunitário", onde toda página web que um usuário acessar, ou arquivo baixado, ficará armazenada na *cache* do servidor, Figura 22.1. Quando um usuário acessa determinado conteúdo, seu navegador vai fazer tal solicitação ao servidor Proxy/Cache, este consulta sua base e verifica a existência ou não de tal conteúdo em sua *cache*, caso esteja disponível a entrega imediatamente, caso não, faz a consulta à Internet, requisita o conteúdo, armazena em sua *cache* e então entrega ao requisitante. No caso de uma nova consulta sobre o mesmo objeto, o servidor acessará de sua base e a entregará imediatamente ao requisitante.

Além disto o servidor pode fazer o controle de conteúdo acessado, barrando ou liberando o acesso a certos sítios, por exemplo. Este controle é feito por meio de `listas de controle de acesso` – ACLs – que, em conjunto com diretivas de liberação ou bloqueio, permitem criar arranjos eficientes no controle de acesso das máquinas da rede local.

[1] http://www.fundao.wiki.br/articles.asp?cod=199

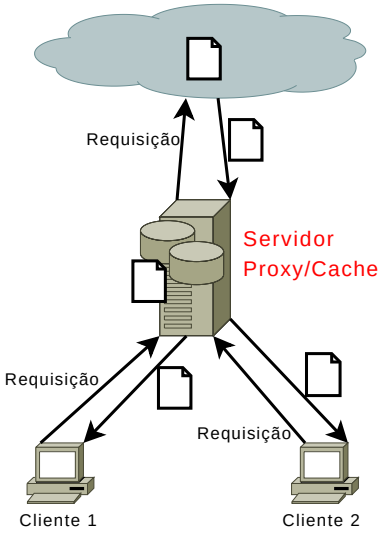

Figura 22.1: Funcionamento do Servidor Proxy/Cache

Um dos servidores Proxy/Cache mais conhecidos e mais utilizado é o Squid[2].

22.2 Instalação e configuração

Para instalar o Squid basta digitar o comando:

urpmi squid

Em seguida iniciar o serviço com o comando:

service squid start

O Servidor Squid já vem com uma configuração padrão e pronto para ser utilizado. Caso deseje-se alguma configuração

[2] http://www.squid-cache.org/

especial deve-se editar o arquivo `/etc/squid/squid.conf`, onde as principais diretivas são:

http_port 3128 – Porta à qual o Squid atenderá. Aqui cabe uma observação, como os servidores Web atendem na porta 80, deve-se fazer uso de um *firewall*, ver Capítulos 23 e 24, para redirecionar as requisições desta porta, para a porta de atendimento do Squid.

cache_dir ufs /var/spool/squid 100 16 256 – Define o diretório de cache do Squid e seus parâmetros. O "sistema de arquivos" usado pelo Squid será do tipo ufs, com armazenamento em `/var/spool/squid`, ocupando no máximo 100 MB, com 16 diretórios e cada um deles com 256 subdiretórios. Obs.: recomenda-se aumentar o tamanho total para entender as necessidades do sistema em uso e mais nenhum outro parâmetro.

cache_mem 8 MB – Tamanho da cache em RAM. Se a máquina for exclusiva para o Squid configura-se para ocupar a metade da RAM real.

maximum_object_size 4096 KB – Tamanho máximo de um único objeto a ser armazenado na *cache*. Recomenda-se 16384 (16 MB). Isto é interessante quando faz-se *downloads* de arquivos etc. Assim objetos maiores também serão armazenados.

visible_hostname m1.redes.edu.br – Nome de máquina do servidor.

As mensagens de erro que o Squid envia aos usuários, por exemplo, informando que determinado sítio tem acesso proibido, podem ser personalizadas. Existem várias, em várias línguas, já predefinidas, que estão localizadas no diretório `/etc/squid/errors/` .

22.3 Testes

Para testes simples configura-se o navegador para usar o próprio servidor Squid. No Firefox, por exemplo, clique em `Editar`, `Preferências`, `Avançado`, aba `Rede`, `Configurações`, diminua a *cache* do navegador para 0 MB, ajuste para `Configuração manual de proxy` e acrescente em `HTTP`: localhost, Porta: 3128 e clique em `Usar este proxy para todos os protocolos`, Figura 22.2.

Figura 22.2: Configurando o Firefox para acessar o Squid

Em seguida acessar um sítio qualquer baixar algum arquivo com tamanho menor que o configurado na diretiva `maximum_object_size`. Medir o tempo. Apagar o arquivo e baixar novamente, se tudo estiver correto no segundo acesso o tempo será praticamente nulo.

Os arquivos e páginas acessados são armazenados no diretório `/var/spool/squid`, num formato legível somente ao Squid, mas pode-se encontrar os objetos, pelo menos no início da formação da *cache*, pelo seu tamanho aproximado.

22.4 Listas de controle de acesso

Com o Squid é possível o bloqueio ou liberação de acesso a determinados sítios, redes, em determinados dias e horários etc.

Para operacionalizar esta função primeiramente deve-se criar as chamadas ACLs (*Access Control List*) que são simplesmente listas de sítios, números IPs ou classes de rede.

Após a criação destas listas deve-se criar as regras de acesso ou bloqueio que referenciarão as listas criadas.

A análise das regras, por parte do Squid, é sequencial, ou seja, o Squid vai lendo as regras uma a uma e, assim que encontrar uma regra que enquadre a mensagem, para imediatamente a análise, e libera ou bloqueia o acesso, dependendo da regra. Portanto, nas regras pode-se ter duas políticas diversas: libera o acesso a alguns conteúdos e proíbe todo o resto ou proíbe o acesso a alguns conteúdos e libera o restante. A adoção de uma ou outra depende da política pretendida.

A regra geral de formação de uma ACL é a seguinte:

```
acl nome tipo (string|"nome_do_arquivo")
```

onde o parâmetro `nome` pode ser inventado a vontade. O `tipo` define origem, destino, padrão de texto, tempo etc. O `string` define números de IPs ou classes de rede. O `nome_do_arquivo` deve conter o caminho e o nome do arquivo que definirá uma coleção de nomes de máquinas ou números IPs, palavras etc.

22.4.1 Exemplos de ACLs e usos

Liberando o acesso à internet à uma sub-rede:

```
acl rede_local src 192.168.5.0/24
http_access allow rede_local
```

Bloqueando o acesso à internet, a um único computador ou à uma rede:

```
acl rede src 192.168.2.7/32 192.168.9.0/24
http_access deny rede
```

Bloqueando acesso a sítios indesejados. Neste caso, deve-se criar o arquivo `/etc/squid/listas/proibidos` e inserir os sites indesejados, um por linha.

```
acl lista_sites dstdomain "/etc/squid/listas/proibidos"
http_access deny lista_sites
```

Bloqueando ou liberando o acesso a sítios com palavras chaves. Neste caso deve-se criar o arquivo `/etc/squid/listas/palavras` e inserir as palavras indesejadas, uma por linha. Devido a *flag* `-i` a busca será insensível à capitalização.

```
acl proibir_pal url_regex -i "/etc/squid/listas/palavras"
http_access deny proibir_pal
```

Restringindo o horário de acesso. Neste caso a rede local somente poderá navegar nos dias úteis da semana das 8 às 18h. Obs.: S=domingo, M=segunda, T=terça, W=quarta, H=quinta, F=sexta e A=sábado.

```
acl horario time MTWHF 08:00-18:00
http_access allow rede_local horario
http_access deny rede_local
```

Para um caso um pouco mais completo pode-se montar a sequência apresentada na Listagem 22.1. Boa parte das ACLs apresentadas já foram descritas. Entre as linhas 9 e 19 são definidas as portas caracterizadas como seguras, isto já vem pré-configurado no Squid. Entre as linhas 21 e 22 é definido que pode-se gerenciar o Squid somente da máquina local. Nas linhas 23 e 24 é definido que o acesso será negado exceto para portas seguras (`Safe_ports` e `SSL_ports`).

Na linha 25 é liberado o acesso para a máquina local. Entre as linhas 26 e 29 tem-se o seguinte, caso o acesso seja `lista_sitios` e `palavras` o mesmo será bloqueado, e será liberado somente nos horários previsto em `horario`. Por último, linha 29, é bloqueado qualquer outro acesso.

```
1  acl all src 0.0.0.0/0.0.0.0
2  acl manager proto cache_object
3  acl rede_local src 192.168.2.0/24
4  acl lista_sitios dstdomain "/etc/squid/listas/sitios"
5  acl palavras url_regex -i "/etc/squid/listas/palavras"
6  acl horario time MTWHF 08:00-18:00
7  acl localhost src 127.0.0.1/255.255.255.255
8  acl to_localhost dst 127.0.0.0/8
9  acl SSL_ports port 443 563
10 acl Safe_ports port 80        # http
11 acl Safe_ports port 21        # ftp
12 acl Safe_ports port 443 563   # https, snews
13 acl Safe_ports port 70        # gopher
14 acl Safe_ports port 210       # wais
15 acl Safe_ports port 1025-65535 # unregistered ports
16 acl Safe_ports port 280       # http-mgmt
17 acl Safe_ports port 488       # gss-http
18 acl Safe_ports port 591       # filemaker
19 acl Safe_ports port 777       # multiling http
20 acl CONNECT method CONNECT
21 http_access allow manager localhost
22 http_access deny manager
23 http_access deny !Safe_ports
24 http_access deny CONNECT !SSL_ports
25 http_access allow localhost
26 http_access deny lista_sitios
27 http_access deny palavras
28 http_access allow rede_local horario
29 http_access deny all
```

Listing 22.1: Exemplo de ACLs no Squid

Capítulo 23

Firewall com iptables

23.1 Introdução

O *Firewall* é um programa que tem como objetivo proteger a máquina ou rede local contra acessos e tráfegos indesejados, proteger serviços que estejam rodando na máquina e bloquear a passagem de pacotes que não se deseja receber. No *kernel* do Linux 2.4, foi introduzido o *firewall* `iptables` (também chamado de `netfilter`[1]) que substitui o `ipchains` dos *kernels* da série 2.2 e anteriores. Este novo *firewall* tem como vantagem ser muito estável, confiável, permitir muita flexibilidade na programação de regras, mais opções disponíveis para controle de tráfego etc.

O `iptables` é um *firewall* a nível de pacotes e funciona baseado no endereço/porta de origem/destino, prioridade etc. Resumidamente pode-se dizer que o mesmo pode ler e manipular todos os campos do cabeçalho IP – camada de rede. Com esta manipulação é possível liberar ou bloquear a passagem de cada datagrama individualmente, após a respectiva análise. 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

[1] http://www.netfilter.org/

pacotes que chegam/saem do seu sistema, contagem de bytes, fazer balanceamento de carga entre máquinas, criar proteções `anti-spoofing`, contra `syn flood`, `DoS` etc.

O `iptables` ainda tem a vantagem de ser modularizável, funções podem ser adicionadas ao *firewall* ampliando as possibilidades oferecidas.

Cabe salientar que um *firewall* não funciona de forma automática, é necessário pelo menos conhecimentos básicos de redes TCP/IP, roteamento e portas para criar as regras que farão a segurança de sistema.

Principais características do *firewall* `iptables`

- Especificação de endereço/porta de origem/destino.
- Suporte a protocolos TCP/UDP/ICMP (incluindo tipos de mensagens icmp).
- Suporte a interfaces de origem/destino dos pacotes.
- Manipula serviços de proxy na rede, ver Capítulo 22.
- 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 especificação de tipo de serviço para priorizar o tráfego de determinados tipos de pacotes.
- Permite especificar exceções às regras.
- Suporte a detecção de fragmentos.

- Permite enviar alertas personalizados ao serviço de logs – `syslog` – sobre o tráfego aceito/bloqueado.
- Permite o redirecionamento de portas.
- Permite o mascaramento de pacotes.
- Suporte completo ao NAT – *Network Address Translation*.
- Contagem de pacotes que atravessaram uma interface/regra.
- Limitação de passagem de pacotes no tempo.

23.2 Princípio de funcionamento do *firewall*

O filtro de pacotes do Linux funciona mediante regras estabelecidas. No `iptables` existem 3 tabelas, que devem ser usadas dependendo do objetivo. As `chains` (correntes) são uma espécie de roteamento interno do kernel. Quando um pacote entra no *kernel*, este verifica o destino do pacote e decide qual `chain` irá tratar do pacote. Os tipos de `chains` irão depender da tabela em uso no momento.

A tabela `filter` refere-se às atividades de filtragem propriamente dita no tráfego de dados, sem a ocorrência de NAT. Admite as `chains` INPUT, OUTPUT e FORWARD. É a tabela padrão, quando não especificada a tabela, a `filter` será utilizada.

A tabela `nat` é utilizada quando é necessário a troca de alguma campo do cabeçalho IP. Exemplo: passagem de dados de uma rede privada para a Internet. Admite as `chains` PREROUTING, OUTPUT e POSTROUTING.

A tabela `mangle` (despedaçar) é usada para marcar pacotes permitindo, por exemplo, o controle de fluxo nas interfaces de entrada e/ou saída.

23.2.1 Regras iptables

Para cada tabela do iptables, `filter`, `nat` e `mangle` o processamento da regras é sequencial, ou seja, caso o `iptables` encontre uma regra onde o pacote se enquadre, ele para imediatamente e encaminha ou descarta o pacote de acordo com a regra. A única exceção são os casos de log, onde o que se enquadra à regra é armazenado e o processamento das demais regras continua.

A composição das regras (*rules*) de *firewall* é feita pelo encadeamento de comandos `iptables` seguindo a sintaxe:

```
iptables [-t tabela] [opção] [chain] [dados] -j [ação]
```

Por exemplo, `iptables -A FORWARD -d 192.168.2.1 -j DROP`, neste caso a tabela é a `filter`, pois é a padrão e não precisa ser explicitada; a opção é `-A`, que significa adicionar; a *chain* é `FORWARD`; os dados são `-d 192.168.2.1`, que significa destinado à 192.168.2.1 e a ação é `DROP`, descarte.

23.2.2 Instalando e configurando

Para instalarmos o *firewall* `iptables` basta digitarmos o comando:

```
urpmi iptables
```

23.2.3 Salvando e recuperando as regras

As regras `iptables` poderão ser salvas com o comando:

```
iptables-save > /nome/do/diretorio/nome_do_arquivo
```

A recuperação poderá ser feita pelo comando:

```
iptables-restore < /nome/do/diretorio/nome_do_arquivo
```

23.3 Tabela Filter

Com a tabela `filter` pode-se construir regras para permissão ou bloqueio de pacotes oriundos de, destinados a, ou de passagem pelo *firewall*. As regras devem ser escritas pensando no resultado final a ser obtido já que sua sequência define o exato comportamento da filtragem ou não dos pacotes.

A seguir detalha-se cada um dos campos do comando `iptables`, relativo a tabela `filter`.

23.3.1 Opções

-P – *Policy*. Altera a política da chain. A política padrão de cada chain é ACCEPT. Isso faz com que o *firewall* inicialmente aceite qualquer INPUT, OUTPUT ou FORWARD. Para uma boa segurança, recomenda-se que todas as *chains* sejam ajustadas para a política DROP e em seguida, libera-se o que for necessário com o `-A`. O `-P` não aceita REJECT ou LOG. Exemplos:

```
iptables -P FORWARD DROP
iptables -P INPUT ACCEPT
```

-A – `Append`. Adiciona uma nova regra à *chain*. 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.dominio.com.br -j DROP
```

-D – *Delete*. Apaga uma regra. Para apagar deve-se escrever a regra com a mesma sintaxe usada na adição. 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.dominio.com.br -j DROP
```

Também é possível apagar a regra pelo seu número de ordem. Pode-se utilizar o `-L -line-numbers` para verificar o número de ordem. Verificado esse número, basta citar a chain e o número de ordem. No exemplo abaixo é apagada a regra de número 4:

```
iptables -D FORWARD 4
```

-L – *List*. Lista as regras existentes. Exemplos:

```
iptables -L
iptables -L FORWARD
```

-F – *Flush*. Remove todas as regras existentes. No entanto, não altera a política. Exemplos:

```
iptables -F
iptables -F FORWARD
```

23.3.2 Chains

Existem 3 `chains` pré-definidas para a tabela `filter`: **INPUT** – utilizada quando o destino final é a própria máquina firewall; **OUTPUT** – pacotes gerado pela máquina *firewall* com qualquer destino; **FORWARD** – qualquer pacote que atravessa o *firewall*, oriundo de uma máquina e direcionado a outra. Figura 23.1. Outras `chains` podem ser criadas com o comando `iptables -N nome_da_chain`.

23.3.3 Dados

Os elementos mais comuns para se gerar dados são os listados as seguir.

`-s` – *Source*. Estabelece a origem do pacote. Geralmente é uma combinação do endereço IP com a máscara de rede separados por uma barra, um único IP, um nome de máquina etc. Exemplos:

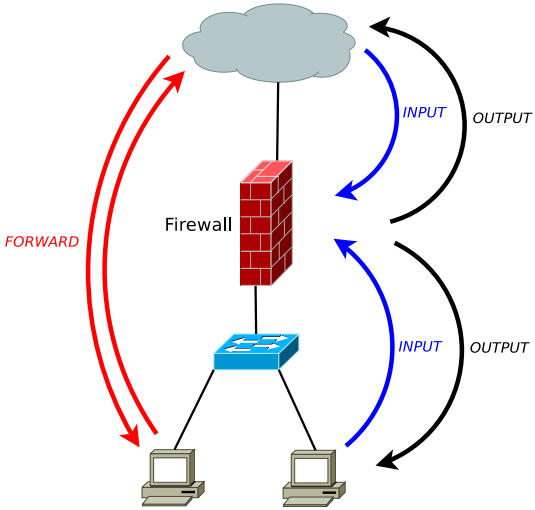

Figura 23.1: Cadeias da tabela `filter`

```
-s 172.20.0.0/255.255.0.0
-s 172.20.5.10/32
-s 172.20.5.10
-s www.dominio.com.br
-s 0/0
```

-d – *Destination*. Estabelece o destino do pacote. A sintaxe é a mesma do `-s`.

-p – *Protocol*. Especifica o protocolo a ser filtrado. O protocolo IP pode ser especificado pelo seu número (vide `/etc/protocols`) ou pelo nome. Exemplo:

```
-p icmp
```

-i – *Input interface*. 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. O sinal "+" pode ser utilizado para simbolizar várias interfaces. Exemplos:

```
-i ppp0
-i eth+
```

-o – *Output interface*. Especifica a interface de saída. A sintaxe é a mesma do `-i`. O `-o` não pode ser utilizado com a chain INPUT.

! – Exclusão. Utilizado com `-s`, `-d`, `-p`, `-i`, `-o` e outros, para excluir o argumento. Exemplos:

```
-s ! 10.0.0.1 ==> Refere-se a qualquer
     endereço de entrada, exceto o 10.0.0.1.
-p ! tcp ==> Todos os protocolos, exceto o TCP.
```

–sport – *Source Port*. Especifica a porta de origem. Só funciona com as opções `-p udp` e `-p tcp`. Exemplos:

```
-p tcp --sport 80
```

–dport – `Destination Port`. Especifica a porta de destino. Só funciona com as opções `-p udp` e `-p tcp`. Similar a `-sport`.

23.3.4 Ações

As principais ações são:

ACCEPT – Permite a passagem do pacote.

DROP – Descarta o pacote impedindo sua passagem. 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`. Deve-se usar antes de outras ações, já que esta ação é a única que permite a continuidade de análise da regras, por parte do `iptables`.

23.3.5 Exemplos comentados de regras

Os pacotes oriundos da sub-rede 10.0.0.0/8 e destinados à máquina www.dominio.com.br deverão ser descartados.

```
iptables -A FORWARD -s 10.0.0.0/8
   -d www.dominio.com.br -j DROP
```

Os pacotes oriundos da sub-rede 10.0.0.0/8 e destinados à máquina www.dominio.com.br deverão ser descartados. Deverá ser enviado um ICMP avisando à origem.

```
iptables -A FORWARD -s 10.0.0.0/8
   -d www.dominio.com.br -j REJECT
```

Os pacotes destinados à máquina www.dominio.com.br deverão ser descartados.

```
iptables -A FORWARD -d www.dominio.com.br -j DROP
```

Os pacotes destinados à sub-rede 10.0.0.0 (máscara 255.0.0.0) e oriundos da máquina www.dominio.com.br deverão ser descartados.

```
iptables -A FORWARD -d 10.0.0.0/8
   -s www.dominio.com.br -j DROP
```

Os pacotes oriundos da máquina www.dominio.com.br deverão ser descartados.

```
iptables -A FORWARD -s www.dominio.com.br -j DROP
```

Os pacotes oriundos da sub-rede 200.221.20.0 (máscara 255.255.255.0) deverão ser descartados.

```
iptables -A FORWARD -s 200.221.20.0/24 -j DROP
```

Os pacotes icmp oriundos da máquina 10.0.0.5 deverão ser descartados.

```
iptables -A FORWARD -s 10.0.0.5 -p icmp -j DROP
```

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 -i ! eth0 -j ACCEPT
```

O tráfego de pacotes TCP oriundos da porta 80 da máquina 10.0.0.5 deverá ser gravado em log. No caso, `/var/log/messages`.

```
iptables -A FORWARD -s 10.0.0.5 -p tcp --sport 80
   -j LOG
```

Os pacotes TCP destinados à porta 25 de qualquer máquina deverão ser aceitos.

```
iptables -A FORWARD -p tcp --dport 25 -j ACCEPT
```

23.3.6 Impasses

Deve-se tomar cuidado ao montar regras com a política DROP. Neste caso inicialmente proíbe-se tudo, então, ao fazer-se uma regra com um nome de máquina por exemplo, deve-se antes liberar o acesso ao servidor de nomes. Também não pode-se esquecer que deve-se liberar o "caminho" de ida e retorno de determinada requisição. No exemplo abaixo tem-se uma sequência de regras onde, após estabelecida a política DROP, libera-se ao acesso de ida e volta ao servidor de nomes e em seguida o acesso ao sítio propriamente dito.

```
iptables -P FORWARD DROP
iptables -A FORWARD -s 10.0.0.0/8 -d 200.135.37.65 -j ACCEPT
iptables -A FORWARD -s 200.135.37.65 -d 10.0.0.0/8 -j ACCEPT
iptables -A FORWARD -s 10.0.0.0/8 -d www.dominio.com.br -j ACCEPT
iptables -A FORWARD -s www.dominio.com.br -d 10.0.0.0/8 -j ACCEPT
```

23.3.7 Extensões

As extensões permitem filtragens especiais, principalmente contra ataques bem conhecidos. Quando necessárias, devem ser as primeiras regras do *firewall*. Alguns exemplos:

Inibe o recebimento e, portanto, respostas ao `ping`.

```
iptables -A FORWARD -p icmp --icmp-type
   echo-request -j DROP
```

Limita o recebimento de requisições de `ping` a no máximo 1 por segundo, ou seja, coíbe o ataque do tipo `ping` da morte (*Ping of Death*), onde é feita uma inundação de requisições `ping`. Neste caso a política deve ser DROP.

```
iptables -A FORWARD -p icmp --icmp-type
   echo-request -m limit --limit 1/s -j ACCEPT
```

Coibe ataques do tipo *Syn-flood*, ou seja inundações de abertura de conexão TCP, que tornam os servidores indisponíveis para as demais requisições.

```
iptables -A FORWARD -p tcp -m limit --limit 1/s
   -j ACCEPT
```

Coíbe a varredura de portas ou *port scanners*, com ferramentas do tipo `nmap`.

```
iptables -A FORWARD -p tcp --tcp-flags
   SYN,ACK,FIN,RST -m limit --limit 1/s -j ACCEPT
```

23.3.8 Exemplo Prático

Na Listagem 23.1 tem-se um exemplo de um *firewall* construído com `iptables`. O exemplo é construído para proteger a rede do diagrama apresentado na Figura 23.2. Entre as linhas 1 e 3 estão definidas as

políticas de bloqueio. Entre as linhas 4 e 7 são liberados os tráfegos desejados. Entre as linhas 8 e 16 é mostrada uma tentativa de bloqueio de *trojans*, que são os lendários cavalos de Tróia, são aplicativos instalados nas máquinas que liberam portas para futuras conexões externas. A linha 17 vai no mesmo sentido de tentar o bloqueio de *worms*. Nas linhas 17 e 18 são os bloqueios aos *Syn-Flood* e *Ping of Death*. Na linha 20 o bloqueio de varredura de portas. Entre as linhas 21 e 33 é feito o `log` de várias tentativas de acesso à portas conhecidas do *firewall*. E nas duas últimas linhas é liberado o tráfego SMTP (porta 25) para a máquina 200.200.10.40.

```
1  iptables -P INPUT DROP
2  iptables -P OUTPUT DROP
3  iptables -P FORWARD DROP
4  iptables -A INPUT -i ! eth1 -j ACCEPT
5  iptables -A INPUT -m state --state ESTABLISHED,RELATED -j
       ACCEPT
6  iptables -A OUTPUT -m state --state ESTABLISHED,RELATED,NEW -
       j ACCEPT
7  iptables -A FORWARD -m state --state ESTABLISHED,RELATED,NEW
       -j ACCEPT
8  iptables -N TROJAN
9  iptables -A TROJAN -m limit --limit 15/m -j LOG --log-level 6 --
       log-prefix "FW:trojan: "
10 iptables -A TROJAN -j DROP
11 iptables -A INPUT -p TCP -i eth1 --dport 666 -j TROJAN
12 iptables -A INPUT -p TCP -i eth1 --dport 666 -j TROJAN
13 iptables -A INPUT -p TCP -i eth1 --dport 4000 -j TROJAN
14 iptables -A INPUT -p TCP -i eth1 --dport 6000 -j TROJAN
15 iptables -A INPUT -p TCP -i eth1 --dport 6006 -j TROJAN
16 iptables -A INPUT -p TCP -i eth1 --dport 16660 -j TROJAN
17 iptables -A FORWARD -p tcp --dport 135 -i eth0 -j REJECT
18 iptables -A FORWARD -p tcp --syn -m limit --limit 2/s -j ACCEPT
19 iptables -A FORWARD -p icmp --icmp-type echo-request -m limit --
       limit 1/s -j ACCEPT
20 iptables -A FORWARD -p tcp --tcp-flags SYN,ACK,FIN,RST -m limit
       --limit 1/s -j ACCEPT
21 iptables -A INPUT -p tcp --dport 21 -i eth1 -j LOG --log-level 6 --
       log-prefix "FW:ftp: "
22 iptables -A INPUT -p tcp --dport 23 -i eth1 -j LOG --log-level 6 --
       log-prefix "FW:telnet: "
23 iptables -A INPUT -p tcp --dport 25 -i eth1 -j LOG --log-level 6 --
```

```
      log-prefix "FW:smtp: "
24  iptables  -A INPUT -p tcp --dport 80 -i eth1 -j LOG --log-level 6 --
      log-prefix "FW:http: "
25  iptables  -A INPUT -p tcp --dport 110 -i eth1 -j LOG --log-level 6
      --log-prefix "FW:pop3: "
26  iptables  -A INPUT -p udp --dport 111 -i eth1 -j LOG --log-level 6
      --log-prefix "FW:rpc: "
27  iptables  -A INPUT -p tcp --dport 113 -i eth1 -j LOG --log-level 6
      --log-prefix "FW:identd: "
28  iptables  -A INPUT -p tcp --dport 137:139 -i eth1 -j LOG --log-level
      6 --log-prefix "FW:samba: "
29  iptables  -A INPUT -p udp --dport 137:139 -i eth1 -j LOG --log-level
      6 --log-prefix "FW:samba: "
30  iptables  -A INPUT -p tcp --dport 161:162 -i eth1 -j LOG --log-level
      6 --log-prefix "FW:snmp: "
31  iptables  -A INPUT -p tcp --dport 6667:6668 -i eth1 -j LOG --log-
      level 6 --log-prefix "FW:irc: "
32  iptables  -A INPUT -p tcp --dport 3128 -i eth1 -j LOG --log-level 6
      --log-prefix "FW:squid: "
33  iptables  -A INPUT -p tcp --dport 22 -i eth1 -j ACCEPT
34  iptables  -A FORWARD -p tcp -d ! 200.200.10.40 --dport 25 -j LOG --
      log-level 6 --log-prefix "FW:SMTP proibido: "
35  iptables  -A FORWARD -p tcp -d ! 200.200.10.40 --dport 25 -j REJECT
```

Listing 23.1: Exemplo de sequências de regras para um *firewall*

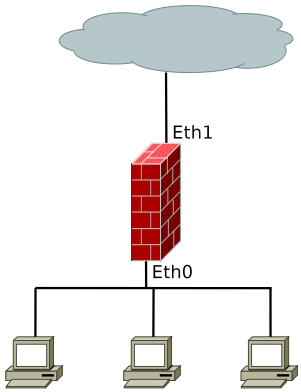

Figura 23.2: Diagrama de rede controlada pelo *firewall*

23.4 Tabela nat – Network Address Translation

O NAT é empregado em vários recursos da rede. O mascaramento (*masquerading*) permite que vários computadores da rede local naveguem na Internet com apenas um IP público do *firewall*, ao mesmo tempo impedindo um acesso direto da Internet a estes computadores. O redirecionamento de portas (*port forwarding*) que permite que algum tráfego especial, de alguma porta específica, seja reencaminhada para outra máquina. O redirecionamento de servidores (*forwarding*), todos os pacotes destinados a um determinado número IP de um servidor podem ser reencaminhados a outro. Proxy transparente (*transparent proxy*), na verdade é um redirecionamento de portas, todo o tráfego destinado à porta 80 será reencaminhado ao servidor Proxy/Cache, ver Capítulo 22.

O NAT pode ser subdividido em SNAT – *Source NAT* – quando somente o endereço de origem é alterado e DNAT – *Destination NAT* – quando se altera o endereço de destino.

A composição dos comandos `iptables` para a tabela `nat` é muito similar aos da tabela `filter`, mudando somente nos casos (exemplos) apresentados a seguir.

23.4.1 Chains

Para a tabela `nat` existem as `chains` listadas abaixo e sua representação na Figura 23.3.

PREROUTING – Utilizada para analisar pacotes que estão entrando no *kernel* para sofrerem NAT. O PREROUTING permite a troca do endereço de destino do datagrama, isso é conhecido como DNAT (*Destination NAT*);

POSTROUTING – Utilizada para analisar pacotes que estão saindo do kernel, após sofrerem NAT. O POSTROUTING permite a troca do endereço de origem do datagrama, 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 permite a troca do endereço de destino do datagrama, também é DNAT.

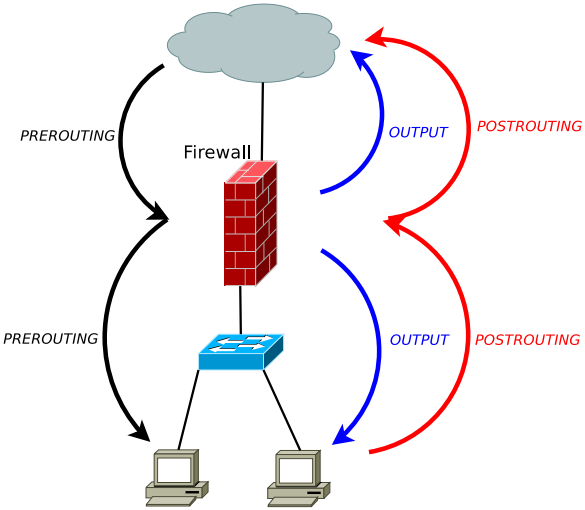

Figura 23.3: Cadeias da tabela `nat`

23.4.2 Dados

A maioria dos dados básicos apresentados para a tabela `filter` continuam valendo. Por exemplo, `-p` servirá para definir um protocolo de rede e `-d` define uma máquina de destino.

–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). Exemplos:

```
-j DNAT --to 10.0.0.2
-j DNAT --to 10.0.0.2:80
-j SNAT --to 172.20.0.2
```

–**dport** – Define a 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** – Define uma porta de origem. Deve ser utilizado antes de uma ação (-j ação).

–**to-port** – Define uma porta de destino, após um REDIRECT.

23.4.3 Ações

SNAT – Utilizado com POSTROUTING para fazer ações de mascaramento da origem, ou seja, troca do número IP/porta de 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 – Também faz mascaramento da origem, diferente do SNAT por ser indicado para máquinas que tenham interface externa configuradas com IP dinâmico.

REDIRECT – Redireciona uma requisição para uma porta local do próprio *firewall*.

23.4.4 Exemplos comentados de regras

Todos os pacotes que saírem pela interface ppp0 (modem) serão mascarados.

```
iptables -t nat -A POSTROUTING -o ppp0
    -j MASQUERADE
```

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 PREROUTING -t nat -p tcp
   -d 10.0.0.2 --dport 80 -j DNAT --to 172.20.0.1
```

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 OUTPUT -p tcp -d 10.0.0.10
   -j DNAT --to 10.0.0.1
```

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 POSTROUTING -o eth0 -j SNAT
   --to 200.20.0.1
```

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
```

Esta regra é para o balanceamento de carga entre servidores. 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 -i eth0 -j DNAT
   --to 172.20.0.1-172.20.0.3
```

Todos os pacotes TCP que vierem da rede 10.0.0.0, com máscara 255.0.0.0, destinados à porta 80 de qualquer máquina, não sairão; serão redirecionados para a porta 3128 do *firewall*. Isso é o passo necessário para fazer um proxy transparente. No exemplo, o Squid, deverá estar instalado na máquina *firewall*, servindo na porta 3128.

```
iptables -t nat -A PREROUTING -s 10.0.0.0/8 -p tcp
   --dport 80 -j REDIRECT --to-port 3128
```

Todos os pacotes que saírem da rede 192.168.1.0 terão seu endereço de origem transformados em 200.20.5.20.

```
iptables -t nat -A POSTROUTING -s 192.168.1.0/24
   -o eth1 -j SNAT --to 200.20.5.20
```

Capítulo 24

Firewall com Shorewall

24.1 Introdução

O `Shoreline Firewall`[1], comumente conhecido como `Shorewall`, é uma ferramenta de "alto nível" para configurar o `iptables`, facilitando a vida dos administradores mais inexperientes. Sua configuração é simples já que descreve-se as necessidades de *firewall* usando entradas/diretivas num conjunto de arquivos de configuração, o `Shorewall` lê estes arquivos e monta as regras do `iptables` para atender as demandas.

`Shorewall` não é um *daemon*, apesar de ter aparência de um. Uma vez que o `Shorewall` tenha configurado o `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 *firewall* `iptables`.

As principais características do `Shorewall` são: flexibilidade, ampla documentação e facilidade de uso.

Algumas alternativas ao `Shorewall`, para configuração facilitada do `iptables`: `Firestarter`[2], `Firewall Builder`[3],

[1] http://www.grulic.org.ar/eventos/charlas/shorewall-2005-09.html
[2] http://www.fs-security.com
[3] http://www.fwbuilder.org

Arno's IPTABLES Firewall Script[4], Linux Firewall do Webmin[5] e FireHOL[6].

24.2 Zonas

O Shorewall trabalha com conceito de zonas. O Shorewall não assume nenhum papel específico a cada zona, característica sobre a qual recaem a maior parte de sua flexibilidade. Uma zona é um conceito abstrato que identifica a origem e/ou destino de um pacote. As mesmas são utilizadas para a definição de regras de aceitação ou recusa de pacotes em função da zona de procedência ou destino. As zonas podem ou não estar associadas a interfaces de rede, sub-redes ou conjuntos de IPs.

A única zona explicitamente associada a um equipamento/IP é a zona fw, a qual contém o *firewall* que se está configurando.

24.3 Instalação e configuração

Para instalar o Shorewall[7] digita-se o comando:

```
urpmi shorewall
```

Todos os arquivos de configuração do Shorewall são documentados e contém uma série de exemplos. Abaixo são apresentados alguns parâmetros dos principais arquivos de configuração do mesmo.

24.3.1 shorewall.conf

É o principal arquivo de configuração do Shorewall. Algumas das diretivas importantes a se configurar neste arquivo são:

[4] http://rocky.eld.leidenuniv.nl/joomla/
[5] http://doxfer.com/Webmin/LinuxFirewall
[6] http://firehol.sourceforge.net
[7] http://www.shorewall.net/

STARTUP_ENABLED – Recomenda-se configurar para *Yes*, caso contrário o mesmo não será iniciado junto ao *boot* da máquina.

ADMINISABSENTMINDED – Se estiver ajustada para *No*, somente o tráfego listado em `/etc/shorewall/routestopped` serão aceitos quando o `Shorewall` for parado ou reiniciado. Caso esteja em *Yes*, todas as conexões abertas continuarão a ser aceitas, isto é muito útil durante o processo de manutenção do *firewall*, já que, ao fazer uma regra que bloquearia o acesso do próprio administrador, na reinicialização do mesmo não ocorreria o bloqueio da máquina em uso, somente após sua desconexão.

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, sem a necessidade de fazer-se manualmente.

24.3.2 zones

Neste arquivo define-se as `zonas`, que são abstrações que definem uma máquina, ou um conjunto delas, que tem características em comum quanto ao tráfego gerado e/ou recebido. No exemplo abaixo redefine-se a `zona` *firewall*, e cria-se a `zona net`, `loc`, e `dmz`, que dizem respeito respectivamente à Internet, à rede local e a `zona` desmilitarizada.

```
#ZONE   TYPE
fw      firewall
net     ipv4
loc     ipv4
dmz     ipv4
```

24.3.3 interfaces

Neste arquivo se definem as interfaces de rede presentes no *firewall* 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. Exemplo:

```
ZONE    INTERFACE    BROADCAST    OPTIONS
dmz     eth1         detect
net     eth1         detect       tcpflags,routefilter,nosmurfs
loc     eth0         detect       tcpflags,detectnets,logmartians
```

24.3.4 policy

Neste arquivo se definem as políticas para os pacotes que trafegam entre uma `zona` e outra. As possíveis políticas e interpretações são praticamente as mesmas do *iptables*, ou seja, ACCEPT, DROP, REJECT. Pode ainda especificar a taxa máxima de conexões TCP e o limite das rajadas. Exemplo:

```
#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
```

24.3.5 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. As principais ações são: ACCEPT, DROP, REJECT, SNAT, DNAT, REDIRECT, CONTINUE e LOG. No exemplo abaixo apareçe na primeira linha a estrutura (foram omitidos intencionalmente os dois últimos campos: RATE LIMIT USER/GROUP), na segunda o redirecionamento de porta para o servidor Proxy/Cache com exceção para consultas à máquina 200.135.233.1. Na terceira é liberado o acesso da rede local à porta 25 (SMTP) da Internet. Na quarta está a liberação para a máquina da Internet 200.135.233.1, acessar a rede local. Na sexta linha tem-se o redirecionamento das portas, 20,21,22,53 e 80 do IP 200.135.233.9 para as mesmas portas do IP 192.168.1.9 da rede local. A sétima linha tem um redirecionamento da 200.135.233.9/2222 para 192.168.1.7/22.

##ACTION	SOURCE	DEST	PROTO	DEST PORT	SOURCE PORT	ORIGINAL DEST
REDIRECT	loc	3128	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

24.3.6 masq

Este é o arquivo utilizado para definir o mascaramento e SNAT. É essencial para as redes locais que desejam se conectar à internet via *firewall*. Exemplo:

INTERFACE	SUBNET	ADDRESS	PROTO	PORT(S)	IPSEC
eth0	eth1				

24.3.7 Outros arquivos de configuração

Existem ainda alguns outros arquivos de configuração do `Shorewall`. Alguns exemplos úteis são o `hosts` que é utilizado para associar grupos de máquinas a uma `zona`. Em especial para definir múltiplas `zonas` sobre uma interface. O `tunnels` que é utilizado para configurar automaticamente regras para tipos distintos de túneis (IPSEC,

OpenVPN etc.). O conjunto de arquivos iniciados por **tc** que são utilizados para carregar regras de `tc` – *traffic shaping* do `kernel` – do *firewall*, muito útil para controlar a banda por tipo de tráfego num enlace "saturado" com a Internet.

24.4 Alguns exemplos práticos com `Shorewall`

24.4.1 Firewall em uma 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 assume-se que a rede interna está conectada à uma interface `ethernet eth0` e a Internet à `eth1`, ver Figura 23.2. Assume-se também que se deseja fazer *masquerading* da rede interna e que o servidor possui um domínio DHCP para a auto configuração das máquinas na rede local. A seguir estão listados os arquivos de configuração necessários, e seus respectivos conteúdos, para atender esta demanda.

```
     /etc/shorewall/shorewall.conf

STARTUP_ENABLED=Yes

     /etc/shorewall/zones

loc Local Local networks
net Net Internet

     /etc/shorewall/interfaces

net eth1 - norfc1918,nobogons,routefilter,
   logmartians,blacklist,tcpflags,nosmurfs
loc eth0 detect dhcp

     /etc/shorewall/policy
```

```
loc net ACCEPT
loc fw ACCEPT
net all DROP info
all all REJECT info
```

```
     /etc/shorewall/masq
```

```
eth1 eth0
```

```
     /etc/shorewall/rules
```

```
AllowWeb fw all
AllowWeb all fw
AllowSSH fw all
AllowSSH loc fw # Servidor ssh para rede interna
ACCEPT:info net fw tcp 22000 # Idem para Internet
AllowSMB loc fw # Servidor samba
AllowSMB fw loc # Servidor samba
ACCEPT loc fw tcp 3128
ACCEPT fw all tcp 6667:6669
```

24.4.2 Múltiplas zonas sobre uma interface

Neste exemplo supõe-se 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 deseja-se conectar à internet e uma interface eth1 que conecta a máquina a dita rede. A diferença é que agora discrimina-se entre duas classes de máquinas que podem conectar-se à Internet: uma com acesso a todos os serviços de rede disponíveis e outra que pode utilizar somente `http` sobre a Internet e `smtp` no *firewall* onde está instalado um MTA (*Mail Transfer Agent*). Também se deseja que as máquinas da rede interna sejam configurados via DHCP.

```
     /etc/shorewall/shorewall.conf
```

```
STARTUP_ENABLED=Yes
```

```
     /etc/shorewall/zones

loc2 Local Local networks
loc1 Local Local networks
net Net Internet

     /etc/shorewall/interfaces

net eth1 - norfc1918,nobogons,routefilter,
   logmartians,blacklist,tcpflags,nosmurfs
- eth0 detect dhcp

     /etc/shorewall/host

loc1 eth0:192.168.0.0/25
loc2 eth0:192.168.0.128/25
net eth0:0.0.0.0/0

     /etc/shorewall/policy

loc1 net ACCEPT
loc2 net REJECT
loc1 loc2 ACCEPT
loc2 loc1 ACCEPT
net all DROP info
all all REJECT info

     /etc/shorewall/rules

AllowWeb loc2 net
AllowSMTP loc2 fw
AllowWeb fw all
AllowWeb loc1 fw # Servidor http,https
AllowSSH fw all
AllowSSH loc1 fw # Servidor ssh para rede interna
ACCEPT:info net fw tcp 22000 # Idem para Internet
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
```

Capítulo 25

DenyHosts

25.1 Introdução

`DenyHosts`[1] é um *script* escrito por Phil Schwartz para ajudar administradores de sistemas bloquear ataques de força bruta em seus servidores SSH. Ele monitora o arquivo de `log` do sistema, `/var/log/auth.log`, e quando um ataque é detectado adiciona o IP do atacante no `/etc/hosts.deny`, sendo assim o próprio sistema se encarregará de bloquear tentativas de acesso posteriores.

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 facilmente legível e categorizados por tipo de tentativa.

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 sistema serão aceitas, pode enviar correspondências com relatórios etc.

[1] http://www.drsolutions.com.br/exemplos/protegersshd.pdf

25.2 Instalação e configuração

O DenyHosts[2] é dependente do Python v2.3 ou superior. Exe-cuta-se o seguinte comando para saber qual é a versão Python instalada:

```
rpm -q python
```

Caso o Python não esteja instalado, faça-o com o comando:

```
urpmi python
```

Agora instala-se o DenyHosts propriamente dito. Faz-se *download* da última versão do DenyHosts na página oficial, http:\slash\slashdenyhosts.sourceforge.net\slash, existe um rpm noarch (para qualquer arquitetura que admita rpm) e um tar.gz que também é independente de plataforma e pode ser utilizado em qualquer distribuição, por este motivo os teste serão feitos com este último.

Desempacota-se o arquivo do DenyHosts baixado com o comando:

```
tar -zxvf DenyHosts-2.6.tar.gz
```

Para manter o sistema organizado, move-se o diretório DenyHosts criado para , por exemplo, o diretório /sbin.

```
mv DenyHosts-2.6 /sbin/DenyHosts
```

O DenyHosts tem um arquivo de configuração exemplo, pode-se utilizá-lo como base para o arquivo de configuração propriamente dito, com os comandos:

```
cd /sbin/DenyHosts
cp denyhosts.cfg-dist denyhosts.cfg
```

[2] http://denyhosts.sourceforge.net/

Agora edita-se o arquivo de configuração. Na Listagem 25.1 tem-se um exemplo deste arquivo com a principais diretivas. Na linha 1 informa-se ao `DenyHosts` qual é o arquivo de `log` que ele deve monitorar, caso não seja informado corretamente, de acordo com o sistema instalado, o `DenyHosts` simplesmente não funcionará. Na linha 2 determina-se o tempo de bloqueio dos IPs, no exemplo está em duas semanas, caso este campo seja deixado em branco o dito IP nunca sairá da lista de bloqueados. Na linha 3 é definido o(s) serviço(s) a ser(em) bloqueado(s), pode ser informado `ALL`. Na linha 4 é definido o número de tentativas que uma determinada máquina poderá fazer com usuários inválidos, ou seja, não cadastrados na máquina do `DenyHosts`, no exemplo, após 5 tentativas de conexão com usuários distintos a mesma será bloqueada. Na linha 5 tem-se o bloqueio após `n` tentativas com usuários válidos mas com a senha incorreta. Na linha 6 é o mesmo caso anterior mas específico para conta de `root`. Entre as linhas 8 e 14 são definidos os parâmetros para envio de correspondências reportando as tentativas de ataque. Na linha 16 é definido a periodicidade em que o `DenyHosts` varrerá o arquivo de `log` em busca de ataques.

```
1  SECURE_LOG = /var/log/auth.log
2  PURGE_DENY = 2w
3  BLOCK_SERVICE = sshd
4  DENY_THRESHOLD_INVALID = 5
5  DENY_THRESHOLD_VALID = 10
6  DENY_THRESHOLD_ROOT = 1
7  WORK_DIR = /usr/share/denyhosts/data
8  ADMIN_EMAIL = user@redes.edu.br
9  SMTP_HOST = smtp.redes.edu.br
10 SMTP_PORT = 25
11 SMTP_FROM = DenyHosts <user@redes.edu.br>
12 SMTP_SUBJECT = Relatorios do DenyHosts
13 SMTP_USERNAME = user
14 SMTP_PASSWORD = senha.aqui
15 DAEMON_LOG = /var/log/denyhosts
16 DAEMON_SLEEP = 30s
```

Listing 25.1: Arquivo de configuração do `DenyHosts`

Após a devida configuração roda-se o DenyHosts, como ele é somente um *script*, pode-se colocá-lo em modo daemon com o comando:

```
/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. Faz-se isto com o comando:

```
touch /var/log/secure
```

No diretório /usr/share/denyhosts/data serão criados os arquivos: hosts, hosts-restricted, hosts-root, offset, supicious-login, user-hosts, user-invalid e users-valid. Os arquivos iniciados com hosts trazem uma relação das máquinas que tentaram ou se conectaram ao sistema. O offset tem uma informação de controle para o DenyHosts da posição da última busca no arquivo de log. Os arquivos iniciados com user informam o nome de usuários que tentaram ou se conectaram ao sistema, com os respectivos IPs de origem.

Lembre-se que este é somente um *script* então pode ser interessante adicionar o comando para sua inicialização ao final do arquivo /etc/rc.local, para que o DenyHosts rode sempre que a máquina for reiniciada.

25.3 Testes

Agora pode-se fazer alguns testes com tentativas de conexão SSH, informando usuários inexistentes, usuários válidos mas com senha incorreta etc. Em seguida observa-se os arquivos criados/modificados no diretório /usr/share/denyhosts/data. Por exemplo o arquivo hosts-restricted conterá algo parecido com:

```
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
```

Capítulo 26

Antivírus

26.1 Introdução

Um vírus de computador[1], da mesma forma que o biológico, precisa de uma taxa de reprodução maior do que a taxa de erradicação (morte), para se proliferar. E, se a taxa de reprodução cai abaixo do nível de morte, o vírus está condenado à extinção.

Na plataforma Linux, vários obstáculos reduzem significativamente a taxa de reprodução. E, portanto, praticamente não existem vírus que o ataquem. O motivo é a sobreposição de alguns fatores listados a seguir.

Para que um vírus infecte um programa executável num sistema com *kernel* Linux 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.

Quanto mais inexperiente for o usuário, menos provável que tenha ele mesmo feito a instalação do executável, e portanto, que seja

[1] http://www.cic.unb.br/docentes/pedro/trabs/virus_no_linux.html

o dono 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.

Prosseguindo, ainda que um vírus consiga infectar um programa executável, sua missão de proliferar-se esbarra em dificuldades das quais os limites de permissões são apenas o começo. As dificuldades continuam nos programas para conectividade, por serem estes construídos conservadoramente no Linux sem, por exemplo, recursos de macros de alto nível.

Aplicativos e sistemas baseados em Linux são quase todos de código fonte 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 sistema viral, se considera-se 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, quando essas barreiras atuam em conjunto é que a vida do vírus se complica.

A importância de antivírus rodando no Linux é para a proteção das máquinas clientes, já que o Linux normalmente assume o papel de servidor de rede. 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 e-mail ou arquivos, permitindo que sejam pesquisadas todas as mensagens que forem recebidas, por exemplo.

26.2 Instalação e configuração

O `Clan AntiVirus`[2] é um pacote de ferramentas antivírus sobre licença GPL desenhado especificamente para análise de correio eletrônico em servidores de correio, mas pode ser também utilizado para outros fins. Possui 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. Em seguida são listadas as suas principais funcionalidades:

- Analizador de linha de comandos.
- Serviço rápido, com paralelismo e 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 e FreeBSD).
- Multiplas atualizações diárias da Base de Dados de vírus.
- 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 arquivos executáveis portáveis comprimidos com UPX, FSG, Petite, NsPack, wwpack32, MEW, Upack e ofuscados com SUE, Y0da Cryptor e outros.

[2] http://www.clamav.net/

- Suporte incorporado para formatos de documentos comuns nomeadamente arquivos MS Office© e MacOffice©, HTML, RTF e PDF.

Para instalar o CLAMAV basta executar o comando:

```
urpmi clamav
```

O CLAMAV vem pré-configurado e pronto para uso mas caso seja necessário alguma personalização deve-se editar o arquivo `/etc/clamav.conf`. Deste arquivo as principais diretivas a serem reconfiguradas são: `ArchiveMaxFileSize 10M` que diz o tamanho máximo da mensagem em anexo que será escaneada e `ClamukoIncludePath /var/spool` discrimina os diretórios que serão escaneados, podem ser incluídas várias diretivas com os vários diretórios.

Depois de configurar o `clamav.conf`, pode-se configurar o `/etc/freshclam.conf`, que é o responsável pelas atualizações do banco de dados de vírus do CLAMAV. A principal diretiva a ser reconfigurada neste arquivo é `DatabaseMirror` que contém a lista com endereços de espelhos para atualizações, ver em `http:\slash\slashwww.clamav.net\slashmirrors.html`. Em seguida faz-se as atualizações com o comando:

```
freshclam
```

26.2.1 Integrando o CLAMAV ao Postfix

Para que o Postfix "varra" todos os e-mails que entram ou saem do servidor deve-se configurar o `AMaViS` – *A Mail Virus Scanner*, que irá usar o CLAMAV para escanear as mensagens. O `AMaViS` é uma interface entre o MTA (*Message Tranfer Agent*) e um ou mais sistemas de antivírus. Primeiramente deve-se instalar o `AMaViS` com o comando:

```
urpmi amavis
```

O serviço que já vem previamente configurado, caso se deseje alterar algum parâmetro deve-se acessar o arquivo `/etc/amavisd/`

`amavisd.conf`. Em seguida deve-se inicializar o mesmo com o comando:

```
service amavisd start
```

Supondo que o Postfix já esteja instalado, ver Capítulo 15, e funcionando, não são necessários mais ajustes, pois os pacotes configuram todos os serviços e arquivos necessários para a integração entre `Postfix`, `AMaViS` e CLAMAV.

26.2.2 Integrando o CLAMAV ao Samba

Para configurar o samba para escanear todos os arquivos que são compartilhados por ele deve-se primeiramente instalar o pacote `samba-vscan-clamav` com o comando:

```
urpmi samba-vscan-clamav
```

Em seguida cria-se um diretório `.recycle` em cada um dos diretórios compartilhados pelo Samba, este deve ter permissão 777, ou seja acesso livre a todos. E por último deve-se descomentar (ou acrescentar) as seguintes linhas no arquivo `/etc/samba/smb.conf`, ver Capítulo 16:

```
vfs objects = vscan-clamav recycle
vscan-clamav: config-file = /etc/samba/vscan-
    clamav.conf
```

26.2.3 Escanear diretórios em busca de vírus

Como segurança adicional pode-se programar na `cron`, que é uma atividade diária de escaneamento de arquivos em busca de vírus. Na Listagem 26.1 tem-se um pequeno *script* que faz esta tarefa e, caso encontre vírus, envia uma correspondência para as pessoas responsáveis avisando sobre o ocorrido.

```
1  #!/bin/bash
2  #-----------------------------------------------
3  # Script que procura por virus, move para /var/infectados, caso encontre
       , e envia e-mail.
4  #-----------------------------------------------
5  #
6  # Escaneia as pastas /home e /dados. Caso encontre virus move os
       arquivos infectados para /var/infectados e cria uma lista dos
       mesmos em /root/ lista_virus
7  clamscan -r -i /home/ /dados/ --move=/var/infectados/ --recursive > /tmp
       / lista_virus 2>&1
8  # Verifica se o arquivo /tmp/ lista_virus esta vazio, se nao estiver
       manda um e-mail de alerta
9  cat /tmp/ lista_virus |grep FOUND > /tmp/virusencontrado.txt 2>&1
10 tam_arq=`ls -l /tmp/ virusencontrado.txt |awk '{ print $5}'`
11 if [ $tam_arq -gt 5 ]
12 then
13          date >> /tmp/ alerta.txt
14          echo " " >> /tmp/ alerta.txt
15          echo "Atencao Administrador!" >> /tmp/ alerta.txt
16          echo " " >> /tmp/ alerta.txt
17          echo "Foram encontrados virus no servidor." >> /tmp/ alerta.txt
18 echo "Os arquivos infectados foram movidos para o diretorio /var/
       infectados." >> /tmp/ alerta.txt
19          echo "Segue a listagem dos mesmos." >> /tmp/ alerta.txt
20          echo " " >> /tmp/ alerta.txt
21          cat /tmp/ lista_virus >> /tmp/ alerta.txt
22 /bin/mail -s "Virus encontrado. Listagem em anexo" -a /tmp/ alerta.txt
       admin1@redes.edu.br
23 /bin/mail -s "Virus encontrado. Listagem em anexo" -a /tmp/ alerta.txt
       admin2@redes.edu.br
24 fi
25 exit
```

Listing 26.1: Script para escaneamento em busca de vírus

Capítulo 27

VPN

27.1 Introdução

O uso de Redes Privadas Virtuais[1] representa uma alternativa interessante na racionalização dos custos de redes corporativas oferecendo confidencialidade e integridade no transporte de informações através de redes públicas.

A ideia de utilizar uma rede pública como a Internet em vez de linhas privativas para implementar redes corporativas é denominada de 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 criptografadas que podem ser muito úteis para usuários móveis ou remotos, bem como filiais distantes de uma empresa.

[1] http://www.rnp.br/newsgen/9811/vpn.html

Uma das grandes vantagens decorrentes do uso das VPNs é a redução de custos com comunicações corporativas, pois elimina a necessidade de enlaces dedicados de longa distância que podem ser substituídos pelo ponto de acesso à 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. A maior desvantagem eu seu uso é o *overhead* imposto, já que para a criptografia, estabelecimento de túneis etc. tem-se um gasto adicional de tempo e recursos, quando comparado a contratação de um enlace dedicado.

27.2 Aplicações para redes privadas virtuais

Abaixo, são apresentadas as três aplicações ditas mais importantes para as VPNs.

27.2.1 Acesso remoto via Internet

O acesso remoto à redes corporativas através da Internet pode ser viabilizado com a VPN via ligação local a algum provedor de acesso. A estação remota se conecta 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, Figura 27.1.

27.2.2 Conexão de LANs via Internet

Normalmente todas as unidades das corporações já dispõe de uma conexão com a Internet. Uma solução para interligar as LANs destas unidades é a contratação de enlaces dedicados, o que garante confiabilidade e confidencialidade, o problema é o custo desta solução. Uma alternativa é o estabelecimento de VPNs entre as unidades, usando os enlaces já contratados, minimizando os custos, e também com bom grau de confiabilidade e confidencialidade. O software

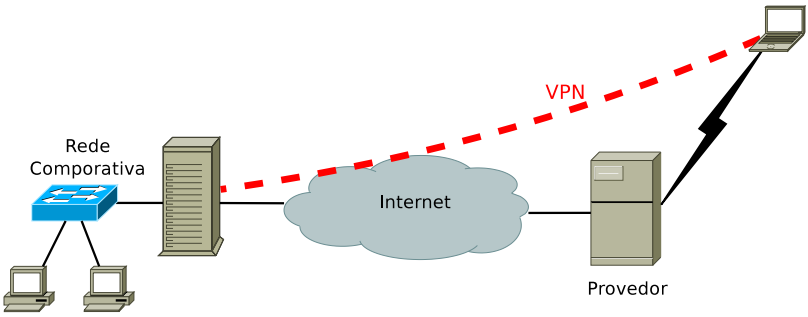

Figura 27.1: Estação acessa a rede comporativa através da VPN

de VPN assegura esta interconexão formando a WAN corporativa, Figura 27.2.

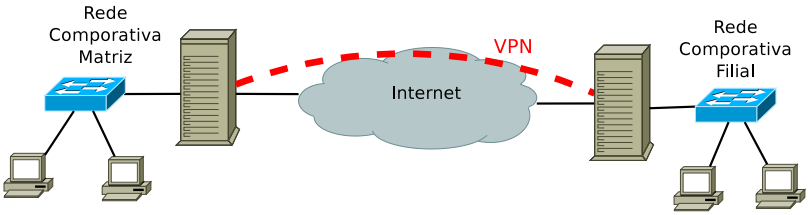

Figura 27.2: Duas redes comporativas interligadas via VPN

27.2.3 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. Outra possibilidade é a implantação de VLANs – *Virtual Local Area Network*, permitindo o isolamento parcial de partes da rede corporativa. Isto só é possível com equipamentos de rede modernos, que permitem a criação e manutenção destas VLANs.

Outra solução é com uso das VPNs que possibilitam a conexão entre redes locais, restringindo acessos indesejados através da inserção de um servidor VPN entre elas, Figura 27.3. 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 (invisível). 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 restrita.

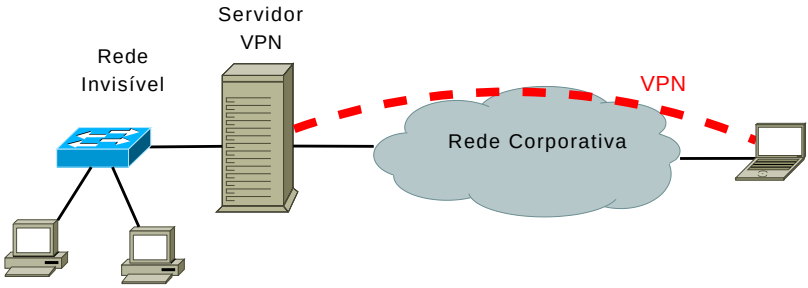

Figura 27.3: VPN protejendo acesso à parte da rede local

27.3 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.

27.3.1 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.

27.3.2 Gerenciamento de Endereço

O endereço do cliente na sua rede privada não deve ser divulgado, devendo-se adotar endereços fictícios para o tráfego externo.

27.3.3 Criptografia de Dados

Os dados devem trafegar na rede pública ou privada num formato cifrado e, caso sejam interceptados por usuários não autorizados, não poderão ser decodificados, garantindo a privacidade da informação. O reconhecimento do conteúdo das mensagens deve ser exclusivo dos usuários autorizados.

27.3.4 Gerenciamento de Chaves

O uso de chaves criptográficas que garantem a segurança das mensagens deve funcionar como um segredo compartilhado exclusivamente entre as partes envolvidas.

27.4 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 ou do mesmo. As VPNs, além do uso do tunelamento, incorporam um novo componente a esta técnica: antes de encapsular o pacote que será transportado, este é criptografado de forma a ficar ilegível caso seja inter-

ceptado durante o seu transporte. O pacote criptografado e encapsulado viaja através da Internet até alcançar seu destino onde é desencapsulado e descriptografado, retornando ao seu formato original.

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. 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.

27.4.1 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, na pilha de protocolos TCP/IP. Os mais famosos da camada 2 são o PPTP – *Point-to-Point Tunneling Protocol* – da Microsoft, o L2TP (*Layer 2 Tunneling Protocol*) da IETF – Internet Engineering Task Force e o L2F (*Layer 2 Forwarding*) da Cisco que é utilizada para VPNs discadas.

Nos tunelamentos em Nível 3 – camada rede – tem-se como exemplo o IPSec (*IP Security Tunnel Mode*) da IETF que permite que pacotes IP sejam criptografados e encapsulados com cabeçalho adicional deste mesmo protocolo para serem transportados numa rede IP pública ou privada.

27.5 Instalação e configuração

Como plataforma de testes propõe-se a estrutura da Figura 27.4.Com esta implementação qualquer cliente da rede local da Matriz “enxer-

gará" qualquer cliente da rede local da Filial e vice-versa. O usuário não necessariamente terá noção de que estará usando VPN.

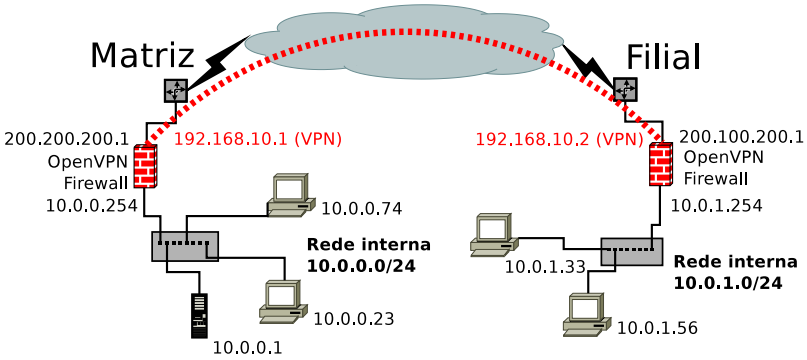

Figura 27.4: Exemplo de uso da VPN

Para implantarmos o modelo da Figura 27.4, primeiramente deve-se instalar o pacote `OpenVPN`[2], em ambos os *firewalls*, com o comando:

```
urpmi openvpn
```

Carregar o módulo `tun` com o comando:

```
modprobe tun
```

Instalar a biblioteca para compressão de dados, se necessário, com o comando:

```
urpmi liblzo
```

27.5.1 Configuração da Matriz

O `OpenVPN` pode operar em 3 modos: nenhuma criptografia (apenas o túnel), criptografia com chaves estáticas e no modo TLS. No último modo as chaves são trocadas periodicamente. Como exemplo,

[2] http://www.openvpn.net/

propõe-se a criptografia com chaves estáticas, que é o padrão proposto pelo pacote. O diretório `/etc/openvpn` é onde se encontram todos os arquivos de configuração.

Como primeiro passo cria-se a chave estática com o comando:

```
openvpn --genkey --secret /etc/openvpn/chave
```

Pode-se visualizar o conteúdo da chave com o comando:

```
cat /etc/openvpn/chave
```

Em seguida cria-se o arquivo `/etc/openvpn/matriz.conf`, de acordo com o exposto na Listagem 27.1. Na linha 1 é estabelecido o nome do dispositivo que será criado, por exemplo tun0, que será uma interface de rede virtual, veja Capítulo 12. Na linha 3 são definidos os números IP que formarão o túnel, respectivamente da Matriz e Filial. A linha 5 referencia a chave criptográfica a ser utilizada. A linha 7 define a porta a ser usada na conexão. Na linha 13 define-se a verbosidade do `log` e na linhas 14 e 15, seus arquivos. Caso deseje-se criar mais tuneis, com outras filiais ou clientes, deve-se repetir o conteúdo do arquivo com outro nome, outra porta e outro conjunto de endereços IP.

```
1  dev tun
2   persist -tun
3   ifconfig  192.168.10.1  192.168.10.2
4  cd /etc/openvpn
5   secret  chave
6   persist -key
7  port  5000
8  user  openvpn
9  group openvpn
10 comp-lzo
11 ping-timer-rem
12 comp-lzo
13 verb  3
14  status  /var/log/openvpn-status.log
15 log-append /var/log/openvpn.log
```

Listing 27.1: Configuração OpenVPN na Matriz

Em seguida, inicia-se o serviço com o seguinte comando:

```
service openvpn start
```

Pode-se conferir se está tudo em ordem, verificando se foi criada uma interface tun0, com o comando:

```
ifconfig
```

Se for mostrado algo como abaixo, o túnel na Matriz já está pronto a espera da conexão da filial.

```
tun0  Link encap:Não Especificado  Endereço de HW 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
      ....
```

27.5.2 Configuração da filial

A exemplo da Matriz, deve-se instalar o pacote OpenVPN e todas as dependências já citadas no início da seção. Em seguida, deve-se copiar a chave gerada na matriz para dentro do diretório `/etc/openvpn/`, para isto pode-se usar o `scp` ou outro mecanismo qualquer.

Cria-se o arquivo `/etc/openvpn/filial.conf`, com o conteúdo mostrado na Listagem 27.2. Perceba que as únicas mudanças para o arquivo da Matriz, Listagem 27.1, ocorrem na linha 3, com a inversão da ordem nos números IP e o acréscimo da linha 4, que deve conter o IP acessível da máquina com a `OpenVPN` da Matriz.

```
1 dev tun
2  persist -tun
3  ifconfig  192.168.10.2  192.168.10.1
4 remote 200.200.200.1
5 cd /etc/openvpn
6  secret  chave
7  persist -key
8 port 5000
9 user openvpn
```

```
10 group openvpn
11 comp-lzo
12 ping-timer-rem
13 comp-lzo
14 verb 3
15  status  /var/log/openvpn-status.log
16 log-append /var/log/openvpn.log
```

Listing 27.2: Configuração OpenVPN na Filial

Em seguida, inicia-se o serviço com o seguinte comando:

```
service openvpn start
```

Pode-se conferir se está tudo em ordem, verificando se foi criada uma interface `tun0`, com o comando:

```
ifconfig
```

27.5.3 Configurações nos *firewalls*

O estabelecimento das VPNs necessitará da abertura das portas que serão utilizadas. Caso esteja usando o `iptables` puro deve-se inserir as seguintes regras, para o 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
```

Caso se esteja utilizando o `Shorewall` editam-se os arquivos abaixo, acrescentando as linhas indicadas. `/etc/shorewall/police`

```
$FW vpn ACCEPT
policy:net vpn DROP
policy:vpn      $FW       ACCEPT
policy:vpn      net      ACCEPT
policy:vpn      loc      ACCEPT
policy:vpn      all      ACCEPT
```

```
     /etc/shorewall/rules

ACCEPT loc:172.18.0.0/16 vpn

     /etc/shorewall/interfaces

vpn     tun0    detect

     /etc/shorewall/zones

vpn     ipv4
```

Capítulo 28

SNMP

28.1 Introdução

O protocolo SNMP[1] – *Simple Network Management Protocol* – é um protocolo para gerência de redes TCP/IP, da camada de aplicação. Facilita o intercâmbio de informação entre os servidores de coleta e os dispositivos de rede que geram dados, segundo este protocolo. O SNMP possibilita aos administradores de rede gerenciar o desempenho da rede, encontrar e resolver problemas de rede, e planejar o crescimento desta.

28.2 Componentes Básicos do SNMP

Uma rede gerenciada pelo protocolo SNMP é formada por três componentes chaves:

1. Dispositivos Gerenciados – É um nó de rede que possui um agente SNMP instalado e se encontra em uma rede gerenciada. Este dispositivo coleta e armazena informações de gerenciamento e mantém estas informações disponíveis para sistemas

[1] http://pt.wikipedia.org/wiki/Simple_Network_Management_Protocol

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.

2. Agentes – 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.

3. Sistemas de Gerenciamento de Redes (NMS – *Network Management Systems*) – É responsável pelas aplicações que monitoram e controlam os dispositivos gerenciados. Normalmente é instalado em um (ou mais de um) servidor de rede dedicado, ou não, a estas operações de gerenciamento, que solicita informações (pacotes SNMP) de todos os dispositivos gerenciados daquela rede.

28.3 Arquitetura

O *framework* SNMP consiste de: Agentes Mestres (*Master Agents*), Sub-agentes (*Subagents*) e Estações de Gerenciamento (*Management Stations*).

28.3.1 *Master Agent*

O *Master Agent* em uma rede gerenciada é 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. Os *subagentes* são os responsáveis por passarem informações específicas para o *Masters Agent*.

28.3.2 *Subagent*

Os *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 enlace `ethernet` em um roteador, ou a quantidade de espaço livre em um disco de um servidor.

28.3.3 *Management Station*

O *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. Requisições são feitas via comandos específicos e podem ou não ser temporizadas. 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.

28.4 Nomes de objetos e MIB

Todos os objetos acessados pelo SNMP devem ter nomes únicos definidos e atribuídos. O conjunto de todos os objetos SNMP é coletivamente conhecido como MIB (*Management Information Base*). O padrão SNMP não define o MIB, mas apenas o formato e o tipo de codificação das mensagens.

A definição dos objetos do MIB é feita com o esquema de nomes do ASN.1 (*Abstract Syntax Notation*.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.

Uma comunidade define um método para autenticar acesso (senha), visibilidade da MIB, privilégios de acesso à MIB. Cada dispositivo implementa uma ou mais comunidades sendo que existe uma comunidade padrão chamada `public`.

28.5 Instalação e Configuração

Como primeira etapa para um cenário ilustrativo instala-se o agente SNMP em uma máquina Linux, usando o comando:

```
urpmi net-snmp
```

Depois de instalado configura-se o SNMP editando o arquivo `/etc/snmp/snmpd.conf`. Este arquivo é auto-explicativo e vem pronto para gerar relatórios relativamente completos mas pode-se personalizar. Por exemplo na Listagem 28.1 tem-se algumas diretivas que recomenda-se que sejam personalizadas. Na linha 1 e 2 definem-se as partições de disco a serem monitoradas sendo que o nome que as sucede representa a quantidade mínima de espaço livre, em MB, que deve existir na respectiva partição, antes de serem gerados alertas. Entre as linhas 3 e 10 são definidos vários processos que estão sendo explicitamente monitorados e os números subsequentes representam o intervalo, mínimo e máximo, de número de processos rodando que não gerarão alertas, fora deste intervalo o SNMP apontará, por meio de MIBS, que houve extrapolamento de limites. Este arquivo permite também controle de acesso às requisições SNMP por meio de sua(s) comunidade(s).

```
1 disk / 10000
2 disk /home 1000
3 proc portmap 100 0
```

```
4  proc syslogd 100 0
5  proc /usr/sbin/sshd 100 0
6  proc ntpd 100 0
7  proc /usr/bin/freshclam 100 0
8  proc crond 100 0
9  proc xinetd 100 0
10 proc /usr/bin/nxserver 100 0
```

Listing 28.1: Parte do arquivo de configuração do SNMP

Após as configurações inicia-se o serviço com:

```
service snmpd start
```

28.6 Testes

Para testar se o SNMP está ativo na máquina, pode-se fazer uma auto consulta. As consultas podem conter filtros por MIB etc. Para uma consulta geral usa-se o comando:

```
snmpwalk -v2c -c public 192.168.2.1
```

Onde `-v2c` significa a versão do SNMP, `-c public` a comunidade a acessar e o último campo trata-se do número IP do dispositivo consultado. Pode-se fazer várias consultas á vários dispositivos na rede, desde que os mesmos implementem o SNMP. Como resultado das consultas obtém-se um relação de todas as MIB implementadas no dispositivo e seu estado atual. Algumas são bastante intuitivas, outras nem tanto, veja uma pequena parte de um exemplo de coleta na Listagem 28.2. A coleta original possuía 2005 MIBs.

```
1 IP-MIB::ipNetToPhysicalPhysAddress.2.ipv4."172.18.0.5" 0:8:54:b0:8c:c3
2 IP-MIB::ipNetToPhysicalPhysAddress.2.ipv4."172.18.0.254" 0:e0:4c:de:21:
      e1
3 TCP-MIB::tcpListenerProcess.ipv4."0.0.0.0".22 0
4 TCP-MIB::tcpListenerProcess.ipv4."0.0.0.0".199 0
5 TCP-MIB::tcpListenerProcess.ipv4."0.0.0.0".389 0
6 TCP-MIB::tcpListenerProcess.ipv4."0.0.0.0".443 0
```

```
7  TCP-MIB::tcpListenerProcess.ipv4." 0.0.0.0 ".631  0
8  TCP-MIB::tcpListenerProcess.ipv4." 0.0.0.0 ".636  0
9  TCP-MIB::tcpListenerProcess.ipv4." 0.0.0.0 ".2273  0
10 TCP-MIB::tcpListenerProcess.ipv4." 0.0.0.0 ".5555  0
11 HOST-RESOURCES-MIB::hrStorageDescr.31 /
12 HOST-RESOURCES-MIB::hrStorageDescr.32 /home
13 HOST-RESOURCES-MIB::hrSWRunName.2130 "atd"
14 HOST-RESOURCES-MIB::hrSWRunName.2178 "crond"
15 HOST-RESOURCES-MIB::hrSWRunName.2457 "ifplugd"
16 HOST-RESOURCES-MIB::hrSWRunName.2771 "startkde"
17 HOST-RESOURCES-MIB::hrSWRunName.2856 "dhclient"
18 HOST-RESOURCES-MIB::hrDeviceDescr.768 GenuineIntel: Intel(R) Core(
       TM)2 Duo CPU E6750 @ 2.66GHz
```

Listing 28.2: Parte da consulta via `snmpwalk`

Fica evidente que a análise deste tipo de dados é muito difícil de ser feita manualmente. Para facilitar a administração da rede existe uma série de serviços que coletam e organizam estas informações de maneira bastante eficiente, personalizáveis e com excelente apresentação. Nos próximos capítulos serão apresentados algumas destas soluções, o `MRTG`, o `Cacti` e o `Nagios`. Outro exemplo é o Zabbix `http:\slash\slashwww.zabbix.com\slash` que não será abordado neste livro.

Capítulo 29

MRTG

29.1 Introdução

O MRTG[1] – *The Multi Router Traffic Grapher* – monitora os dispositivos de rede SNMP e desenha gráficos mostrando o tráfego passante de cada interface de rede dos equipamentos. Seus gráficos são apresentados por meio de navegadores web, num formato universal, que pode ser acessado de qualquer máquina.

MRTG é escrito em perl e funciona em Unix/Linux, bem como em Windows©.

29.2 Instalação e configuração

Para o funcionamento do MRTG é necessário estar com o Apache instalado e rodando, ver Capítulo 14. Em seguida instala-se o próprio MRTG com o comando:

```
urpmi mrtg
```

Cria-se em seguida um arquivo de configuração, com o `cfgmaker`, para cada dispositivo a ser monitorado via SNMP. Por

[1] http://oss.oetiker.ch/mrtg/

exemplo, para monitorar a comunidade `pública` da máquina local usa-se o comando:

```
cfgmaker --global 'WorkDir: /var/lib/mrtg' -output
   /etc/mrtg/mrtg.local.cfg public@localhost
```

Para monitorar um outro equipamento qualquer, com suporte ao SNMP:

```
cfgmaker --global 'WorkDir: /var/lib/mrtg' -output
   /etc/mrtg/mrtg.router.cfg public@172.18.0.254
```

Edite os arquivos criados `/etc/mrtg/mrtg.local.cfg` e `/etc/mrtg/mrtg.router.cfg` e acrescente as seguintes linhas, de acordo com o parâmetros desejados:

WorkDir: /var/lib/mrtg – Define qual será o diretório de trabalho do MRTG, ou seja, o diretório onde serão salvos os arquivos gerados pelo MRTG (logs, arquivos html, png etc..

Options [_]: growright, bits – São duas opções em uma: 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.

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*, ou seja, o MRTG ficará carregado e vai buscar os dados da

máquina conforme o parâmetro `Interval`. Se não for feito assim deve-se configurar atualizações dos gráficos via `crontab`.

Em seguida, cria-se a página `index` do MRTG, a partir dos 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, roda-se o MRTG:

```
env LANG=C /usr/bin/mrtg /etc/mrtg/mrtg.local.cfg
env LANG=C /usr/bin/mrtg /etc/mrtg/mrtg.router.cfg
```

29.3 Testes

Para visualizar os gráficos gerados pelo MRTG acessa-se a página `http:\slash\slashlocalhost\slashmrtg`, o gráfico gerado deve ser algo similar à Figura 29.1. Pode-se personalizar os cabeçalhos dos gráficos da página editando os arquivos html: `/etc/mrtg/mrtg.local.cfg` e `/etc/mrtg/mrtg.router.cfg`.

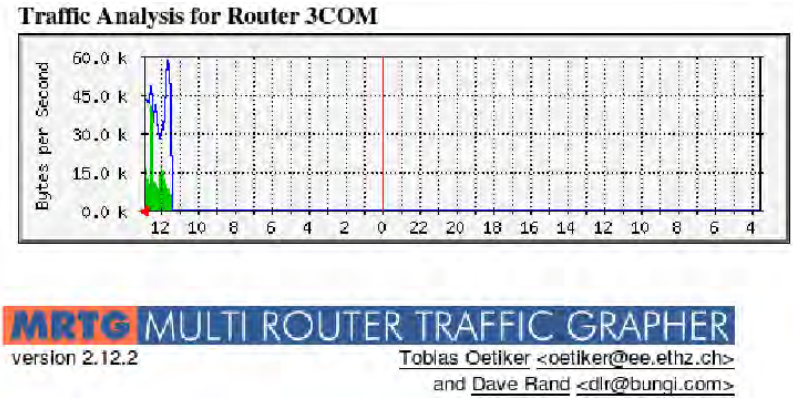

Figura 29.1: Exemplo de gráfico gerado pelo MRTG

Capítulo 30

Cacti

30.1 Introdução

Cacti[1] 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.

[1] http://pt.wikipedia.org/wiki/Cacti

30.2 Instalação e configuração

Para instalar o Cacti[2] é necessário o Apache, ver Capítulo 14, e um banco de dados funcionando no sistema. Todos os dados coletados são armazenados neste banco de dados, por exemplo o `mysql`. O exemplo a seguir considera a instalação do `mysql` e sua configuração simplificada, sem preocupações com segurança e permissionamento de acesso aos dados. Como primeiro passo instala-se o Cacti e o `mysql`:

```
urpmi cacti mysql
service mysqld start
```

Em seguida cria-se a base de dados a partir de um modelo que acompanha o Cacti, com os seguintes comandos:

```
mysql -u root -p
<Enter> (senha em branco)
CREATE DATABASE cacti;
use cacti
source /usr/share/cacti/cacti.sql
exit
```

Edita-se o arquivo `/etc/cacti.conf` para configurar o usuário e senha que terão acesso a base de dados. Isto é feito nos seguintes campos:

```
$database_username = "root";
$database_password = "";
```

Observe que esta não é a configuração mais segura. Num caso real deve-se criar um usuário e senha no banco `mysql` e dar permissões ao mesmo de acesso à base `cacti`.

[2] http://www.cacti.net/

Uma vez configurada a base, o Cacti se configurará praticamente de maneira automática, para isto basta acessar, via navegador, a página `http:\slash\slashlocalhost\slashcacti\slash` e seguir as orientações da mesma. Caso não haja nenhum problema basta aceitar as configurações padrão. Ao final será requisitado login e senha, entre com **admin** X **admin**, em seguida o Cacti forcará a troca de senha do usuário `admin`. O Cacti então terminará toda sua configuração, já preparado para coleta de vários tipos de dados da máquina local.

Após isto pode-se clicar na aba `graphs` e pode-se observar algo parecido com o da Figura 30.1.Pode-se ainda configurar mais monitores, para isto basta clicar na aba `console`, na opção `New Graph` e/ou `Create New Host` e selecionar as opções desejadas. Existem uma série de outras opções que podem ser acessadas nos menus disponíveis.

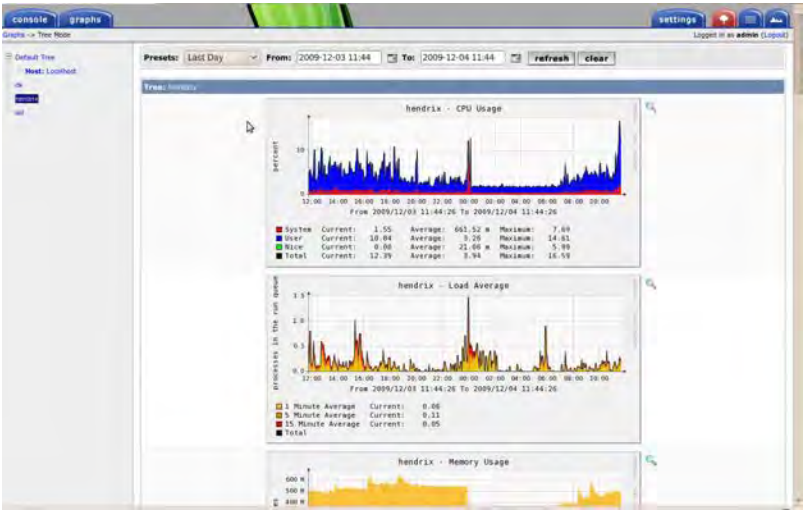

Figura 30.1: Exemplo de gráfico gerado pelo Cacti

Capítulo 31

Nagios

31.1 Introdução

O Nagios[1] é um aplicativo de monitoramento de sistemas e de redes. Ele checa clientes e serviços, especificados em seus arquivos, gerando alertas de problemas ou na iminência de.

O Nagios inclui várias ferramentas, as principais são:

- 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.

[1] http://www.nagios.org/

- Notificação de contatos quando problemas em serviços e clientes ocorrerem ou forem resolvidos, via e-mail, 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.

31.2 Instalação e configuração

Para instalar o Nagios deve-se ter previamente instalado um servidor Apache e o SNMP, ver Capítulos 14 e 28. Em seguida instala-se o Nagios propriamente dito com o comando:

```
urpmi nagios-www
```

Com este pacote 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 destaca-se os principais arquivos de configuração e suas aplicações, localizados no diretório `/etc/nagios/`:

nagios.cfg – Arquivo principal de configuração do Nagios.

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.

Para as definições de monitoramento de equipamentos e máquinas deve-se acessar o sub-diretório `/etc/nagios/objetcs` onde existem vários arquivos de configuração:

commands.cfg – Definições dos comandos a serem executados pelo Nagios.

contacts.cfg – Indivíduos que, possivelmente, deverão ser notificados no caso de problemas detectados pelo Nagios.

localhost.cfg – Definições de monitoramento da máquina local.

printer.cfg – Definições do monitoramento de impressoras.

switch.cfg – Definições do monitoramento de *switches* com suporte ao SNMP.

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. Existem vários *plugins*, dependendo da distribuição pode ser necessário a instalação em separado. No sítio `http:\slash\slashnagiosplugins.org\slashnode\slash2` existe a lista completa de todos os *plugins* disponíveis.

Além disto o Nagios 3 já possui outros diretórios para organizar o monitoramento agrupando as máquinas por seu tipo. Por exemplo: roteadores, servidores, *switches* etc.

Pode-se fazer uma conferência inicial da configuração dos arquivos com o comando:

```
nagios -v /etc/ nagios/nagios.cfg
```

Inicia-se o Nagios com o comando:

```
service nagios start
```

Após reinciar o servidor Apache já é possível acessar a página do Nagios, com o endereço `http:\slash\slashlocalhost\slashnagios\slash`.
Será requisitado um usuário, e sua senha, que devem ser o mesmo criado anteriormente. Observe que esta página estará acessível somente via localhost, caso deseje-se que a mesma seja acessível de outras máquinas deve-se editar o arquivo `/etc/httpd/conf/webapps.d/nagios.conf` e mudar as diretivas `allow from` para o(s) número(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 local. Se desejado, pode-se acrescentar alguns monitoramentos de algumas máquinas remotas.

31.2.1 Monitorando outras máquinas

Como exemplo edita-se o arquivo `/etc/nagios/conf.d/sample.cfg`. Este arquivo é dividido em

seções e recomenda-se a manutenção da ordem do mesmo. É completamente comentado e traz um série de exemplos.

A Listagem 31.1 mostra um exemplo de configuração destes arquivo, com as respectivas seções utilizadas. No caso é mostrado a inclusão de uma nova máquina chamada `teste` e vários serviços da mesma a serem monitorados. A maior parte dos campos são intuitivos. Na linha 18 é habilitada a notificação para o serviço HTTP, para desabilitar basta trocar o valor do campo para 0. Na linha 24 são definidos os parâmetros de consulta ao `ping`, basicamente para saber se a máquina está ou não ativa. Se o 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 de **500** ms e a perda de pacotes for **60**% será gerado um alarme crítico. A linha 30 gera um alarme de aviso quando o espaço livre da partição raiz (/) for menor que **20**% e um alarme crítico quando menor que **10**%. A linha 36 monitora a quantidade de usuários logados: aviso **20** ou mais, crítico **50**. A linha 42 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 pode-se omitir as diretirvas `!RSZDT` e assim serão monitorados todos os processos em todos os estados. E assim por diante.

```
1  ## HOST DEFINITION
2  define  host{
3          use                     linux-server
4          host_name  teste
5          alias          teste
6          address          172.18.0.1
7          }
8  ## HOST GROUP DEFINITION
9      members             localhost , teste
10 ## SERVICE DEFINITIONS
11 define  service{
12         use                                  generic-service
13         host_name                            dk
14          service_description                 HTTP
15         check_command                        check_http
16          notifications_enabled               1
17         }
18 define  service{
```

```
19          use                             generic-service
20          host_name                       dk
21           service_description            PING
22          check_command                   check_ping!100.0,20%!500.0,60%
23          }
24  define  service {
25          use                             generic-service
26          host_name                       dk
27           service_description            Root  Partition
28          check_command                   check_local_disk!20%!10%!/
29          }
30  define  service {
31          use                             generic-service
32          host_name                       dk
33           service_description            Current  Users
34          check_command                   check_local_users!20!50
35          }
36  define  service {
37          use                             generic-service
38          host_name                       dk
39           service_description            Total  Processes
40          check_command                   check_local_procs!250!400!
                RSZDT
41          }
42  define  service {
43          use                             generic-service
44          host_name                       dk
45           service_description            Current  Load
46          check_command                   check_local_load
                !5.0,4.0,3.0!10.0,6.0,4.0
47          }
48  define  service {
49          use                             generic-service
50          host_name                       dk
51           service_description            SSH
52          check_command                   check_ssh
53           notifications_enabled          1
54          }
```

Listing 31.1: Parte do arquivo de configuração do Nagios

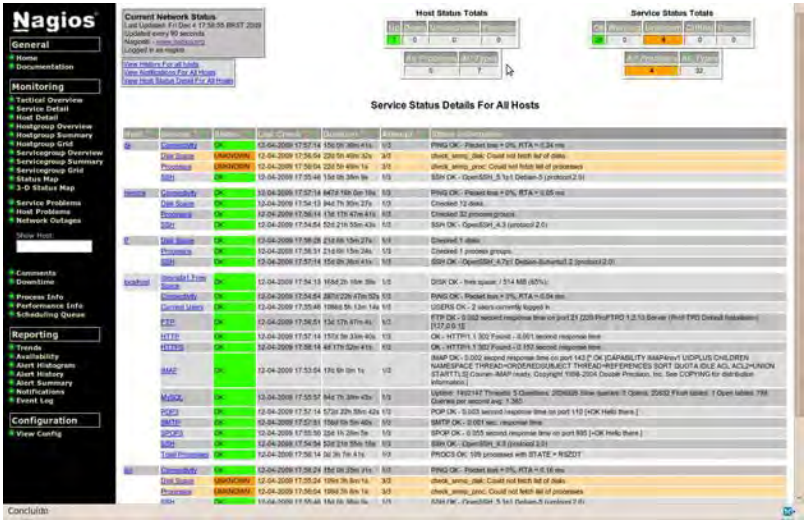

Figura 31.1: Exemplo de tela do Nagios

31.3 Testes

Pode-se fazer um teste rápido da integridade da configuração com o comando:

```
nagios -v /etc/nagios/nagios.cfg
```

Agora deve-se (re)iniciar o Nagios com o comando:

```
service nagios (re)start
```

E pode-se monitorar os novos serviços na página `http:\slash\slashlocalhost\slashnagios\slash`, ver Figura 31.1. 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 para correspondências eletrônicas do(s) administrador(es) da rede, isto deve ser

feito também no arquivo `/etc/nagios/conf.d/sample.cfg` mudando o campo `email` na seção `CONTACTS`. Com isto, qualquer aviso ou alarme gerado pelo Nagios gerará uma correspondência enviada ao(s) endereço(s) informado(s).

Para os administradores de rede existe uma ferramenta muito interessante 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.

Capítulo 32

Webmin

32.1 Introdução

O Webmin[1] é um gerenciador de sistema baseado numa interface web. Com este utilitário pode-se administrar a máquina remotamente através de um navegador comum. Ele é bem completo e tem vários módulos de configuração, desde *hardware* da máquina até serviços que rodam na mesma. Pode auxiliar muito na administração da rede por meio de uma interface intuitiva e de fácil configuração.

As principais tarefas possíveis de serem executadas por meio do Webmin[2] são:

- Mudar senhas, configurar o `crontab`, configurar *scripts* de inicialização, *backup*, configuração do `pam`, quotas, gerência de processos, pacotes, usuários e grupos.
- Configurar e administrar servidores `majordomo`, `cvs`, `sendmail`, `qmail`, `postfix`, `fetchmail`, `jabber`, `samba`, `postgresql`, `proftpd`, `ssh`, `squid`, `wu-ftpd`, `apache`, `dhcp`, `dns bind`, `MySQL`.

[1] http://www.devin.com.br/webmin/
[2] http://www.webmin.com/

- Configurar rede, exportações NFS, NIS, PPP, túneis SSL.
- Administrar impressoras, gerenciar `boot`, `cd-roms`, `raid`, partições, LVM, *clustering*.
- `shell` via web, gerenciador de arquivos, módulos perl etc.

32.2 Instalação e configuração

Para instalar e rodar o Webmin deve-se 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:\slash\slashlocalhost:10000\slash`. Ou seja uma conexão segura, `https`, na porta 10000 do servidor. Será requisitado um usuário e senha, que são as mesmas cadastradas na máquina. Se deseja-se fazer manutenção nos serviços o ideal é usar a própria conta de `root`. Após a troca de linguagem de apresentação para portugues_BR tem-se uma tela do tipo da Figura 32.1. Em seguida pode-se fazer instalação e configurações de serviços, *hardware* etc.

Figura 32.1: Tela de apresentação do Webmin

Apêndice A

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 executa-se outros programas, grava-se ou lê-se informações em disquetes, visualiza-se textos em vídeo ou impressora etc. Sem o Sistema Operacional não consegue-se realizar estas tarefas. Ou seja, simplesmente não se poderia utilizar o computador.

Existem inúmeros Sistemas Operacionais, tais como: UNIX, OS/2©, Windows©, Linux etc. Cada um deles possuem características próprias e são executados em máquinas diferentes. Assim, não pode-se executar um programa em Sistemas Operacionais distintos, a não ser que o fabricante do programa 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 (*Read Only Access Memory*). Da mesma forma, o Sistema Operacional deve ser carregado, ou seja, copiado do disco rígido ou disco flexível para a memória RAM. Denomina-se este processo de BOOT. Toda vez que liga-se 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.

Monousuário – Permitem apenas um usuário.

Multiusuário – Permitem vários usuários.

Monotarefa – Apenas um programa pode ser executado de cada vez.

Multitarefa – Vários programas podem ser executados ao mesmo tempo.

Em geral Sistemas Operacionais que são multiusuários são também multitarefa, como o UNIX, Linux e atuais versões do Windows©, onde pode-se 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:

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 e em servidores de rede.

Interface gráfica para usuários (GUI) – Quando os comandos são executados em um ambiente gráfico auxiliado pelo uso do mouse. Voltada principalmente para o usuário final.

A.1 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 ver-se uma "fotografia" dos processos rodando na máquina pode-se usar o comando `ps aux`, sendo que `ax` mostra todos os processos e `u` informa os usuários donos dos processos.

Pode-se usar também o `top`. Neste caso haverá atualização periódica da tela, fazendo uma amostra *on-line* de processos ativos. Mostra ainda outras informações da máquina, como uso de memória, tempo de atividade, uso de cpu etc. Para navegar entre janelas do top usam-se as teclas <>.

Para "matar-se" um processo em execução usa-se 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. Pode-se usar também o `killall` seguido do nome do processo (comando). Neste caso mata-se todos os processos com mesmo nome.

Apêndice B

Linux

B.1 Histórico

O Sistema Operacional UNIX foi desenvolvido nos laboratórios da ATeT 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 ATeT licenciou seu mais novo projeto para as Universidades, mostrando uma enorme visão e capacidade ino-

vadora, 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 à diversas versões além da original proveniente dos laboratórios da ATeT. 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.

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.

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.

B.2 Uma visão geral do Linux

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, multitarefa, estrutura hierárquica de arquivos, ferramentas, utilitários, comunicação com outros sistemas**.

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). 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.

B.3 Kernel/Shell

`Kernel` é o núcleo do Sistema Operacional Linux, que permanece residente na memória. É 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`.

Apêndice C

Editor vi

C.1 Introdução

O editor `vi` é bastante simples e muito utilizado por ser encontrado em todas as distribuições Linux. Pode-se optar por um editor um pouco mais avançado, mas com o inconveniente de encontrar uma distribuição/instalação onde não dispõem-se 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 se entenda o funcionamento do editor e consiga editar arquivos simples. Um administrador de redes deve constantemente editar os arquivos de configuração dos servidores com rapidez e segurança e isto é perfeitamente factível com o `vi`.

C.2 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) e 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 pode-se retroceder sobre o texto já digitado para corrigir eventuais erros. Cada caractere 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` digita-se 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 ver-se-á mais adiante. Os comandos no modo `last line` devem ser seguidos por `enter`, contrariamente ao que acontece no modo `escape`.

C.3 O *Buffer* de edição

Quando edita-se um arquivo com o `vi`, na verdade não se 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 quiser-se que as alterações fiquem permanentemente aplicadas ao arquivo, será 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.

C.4 Criação e edição de arquivos

Neste livro não pretende-se aprofundar o uso do editor `vi`. O objetivo é somente atender as necessidades básicas de um administrador de rede.

Para criar-se um arquivo simplesmente digita-se *vi* seguido do nome do arquivo. Por exemplo:

vi primeiro.arquivo

Após isto será aberto o editor com conteúdo vazio, no modo comando. Para editar-se qualquer coisa deve-se entrar no modo inserção, para isto basta teclar-se a letra '*i*' (aparecerá `- INSERT -` na base da janela). Em seguida digita-se o texto propriamente dito, usando o teclado normalmente.

Para salvar-se o texto deve-se teclar '`Esc`' mudando para o modo comando, e em seguida entra-se no modo `last line` e salva-se com: '`:w`'. Assim tem-se o texto salvo.

C.4.1 Alguns comandos úteis

Copiando algumas linhas Posiciona-se o cursor na primeira linha do texto a ser copiado e, no modo de comando, tecla-se '*nyy*', onde 'n' é número de linhas que deseja-se copiar. Por exemplo ao digitar-se '*5yy*' copia-se 5 linhas para o `buffer`.

Colando o conteúdo do buffer Posiciona-se o cursor na linha onde pretende-se inserir o texto e, no modo de comando, tecla-se '*p*' (*paste*) para inserir-se o texto abaixo da linha do cursor e '*P*' para inserir-se o texto acima da linha do cursor.

Excluindo algumas linhas Posiciona-se o cursor no início do texto a ser excluído e, no modo de comando, tecla-se '*ndd*'

, onde 'n' é número de linhas que deseja-se excluir. Por exemplo ao digitar-se '*3dd*' apaga-se 3 linhas do texto. Observe que o texto removido será armazenado no `buffer` e poderá ser inserido em qualquer outra parte do documento com o comando '`p`' ou '`P`'.

Procurar palavras No modo de comando, tecla-se '/palavra'. O `vi` mostrará a primeira ocorrência da mesma. Para ir para a próxima ocorrência tecla-se '*n*' (next).

Substituindo uma palavra por outra No modo de comando, tecla-se '*:s/palavra/outra*'. Por exemplo para subs-tituir-se `velha` por `nova`: '*:s/velha/nova*', assim tem-se a troca da primeira ocorrência de `velha` por `nova`. Para substituição de todas as ocorrências no texto deve-se acrescentar o caractere % no início do comando. Por exemplo: '*:%s/velha/nova*'

Inserção de um texto externo No modo de comando, tecla-se '*:r/caminho/e/nome/do/arquivo*' e o texto do arquivo mencionado será inserido a partir da posição do cursor.

Salvando com outro nome No modo de comando, tecla-se '*:w/caminho/e/nome/do/arquivo*'.

Apêndice D

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 armazenado dentro do diretório. Assim sendo, não há limite para o tamanho de um diretório. Teoricamente pode-se colocar no seu diretório tantos arquivos quanto quisesse, até o ponto de estourar a capacidade do 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 ou tabela de *inodes*, ver Capítulo 4. 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 se usa um arquivo, o sistema operacional consulta o diretório para verificar se existe no disco um com o nome que se 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 chamam-se 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.

D.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 e os

usuários possam organizar seus arquivos separando-os por temas, assuntos etc.

Ao entrar pela primeira vez numa conta, já se está em um subdiretório do sistema Linux, chamado seu diretório de entrada (*home directory*). A menos que se crie alguns subdiretórios em sua conta, todos os seus arquivos serão armazenados no diretório de entrada. Teoricamente, pode-se fazer isso, mas a manutenção dos arquivos será mais eficiente ao 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.

D.1.1 Diretório de Entrada

O diretório de entrada é aquele em que o usuário é colocado quando abre uma sessão em um sistema Linux. Esse diretório tem o mesmo nome que o nome de `login`. Pode-se pensar na conta como uma versão em miniatura do sistema de arquivos do Linux. No alto do sistema pessoal de arquivos, em vez do diretório raiz, está o diretório de entrada. Abaixo dele estarão os subdiretórios criados, 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 da conta, o diretório de entrada tem um status especial. Sempre que se entra no sistema, o Linux define uma variável chamada HOME que identifica o 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 pode-se acessar e também para orientar-se para onde levar o usuário ao mudar de diretório corrente.

D.1.2 Diretórios Corrente

O diretório corrente, ou de trabalho (*working directory*), é o diretório em que se está em um determinado momento. Por exemplo, quando se entra no sistema, o diretório corrente é sempre o diretório de entrada. Ao passar para um de seus subdiretórios, este passará a ser o diretório corrente.

Durante toda a sessão, o Linux mantém o controle do diretório corrente. Todos os comandos são executados sobre o diretório corrente, a menos que se especifique outro. Por exemplo, qualquer arquivo ou subdiretório que se criar será em princípio criado no diretório corrente. Sempre que se digitar `ls`, será apresentada 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 se quiser referir-se ao diretório corrente, pode-se 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, pode-se usar dois pontos '..' em vez do nome do diretório.

Quando se digita um comando que opera sobre um arquivo ou diretório, precisa-se 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.

Só é possível 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 se 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, também pode-se 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, pode-se começar o caminho com $HOME ou com um ∼ (til), que é um sinônimo para $HOME. Por exemplo, se o diretório de entrada é maria, a variável HOME terá o valor `/.../maria`, onde as reticências representam os diretórios entre o diretório raiz (/) e o diretório maria. Sempre que se 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, pode-se 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 o início se dará pelo o diretório corrente.

D.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.

- Asterisco (*)

  O `*` substitui qualquer conjunto de caracteres.

- Ponto de interrogação (?)

  O caracter `?` substitui qualquer caracter.

- 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. Lista todos os arquivos com extensão .new:

   *ls *.new*

   File.new arquivo.new

2. Lista todos os arquivos cujo nome termine com um número entre 1 e 5:

   *ls *[1-5]*

   file1 arquivo3 dir5

3. Lista todos os arquivos cujo nome tem três caracteres e começam com f:

   ls f??

   fig fin

D.3 Marcação do Caracter 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.

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 (\).

Apóstrofe ' – Apóstrofe é mais restritivo. Todos os caracteres especiais entre apóstrofes são ignorados.

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.

Apêndice E

Manipulando Arquivos e Diretórios

E.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.

O objetivo deste capítulo é aprender 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 serão vistas neste capítulo.

Os comandos aqui apresentados não são a totalidade dos comandos disponíveis, mas certamente são suficientes para executar funções típicas e usuais de um programador ou de um usuário de aplicativos em ambiente Linux.

E.2 Comandos para Arquivos e Diretórios

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*). Pode-se utilizá-lo para situar-se 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.

mkdir

Nos diretórios pode-se 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 – 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. Criar 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. Criar 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/

ls

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
```

cd

```
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 não for especificado 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 pode-se informar o nome relativo ou absoluto do diretório para o qual se deseja mudar.

touch

```
touch <arquivo>
```

Cria um arquivo vazio de nome `arquivo`.

Exemplo:

Criar um arquivo vazio de nome teste.arquivo

```
touch teste.arquivo
```

echo

```
echo "texto a ser inserido»> <arquivo>
```

Insere o "texto a ser inserido" ao final do arquivo. O comando `echo` também pode ser utilizado para mostrar (ecoar) valores de variáveis e/ou textos em *shell scripts*.

Exemplo:

```
echo "teste texto»> teste.arquivo
```

cat

```
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.

Exemplo: Mostrar o conteúdo do arquivo teste.arquivo:

```
cat teste.arquivo curso
teste texto
```

cp

```
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. Pode-se usar esse comando para criar cópias de segurança de arquivos importantes ou para copiar arquivos que se queira modificar. Se há algum arquivo que se queira ter em mais de um diretório, pode-se usar o comando `cp` para copiá-lo para outros diretórios.

Na linha de comando, arquivo-origem é o nome do arquivo que se quer copiar e `arquivo-destino` é o nome que será dado à cópia. Lembre-se: se fizer uma cópia de um arquivo no mesmo diretório, ela não poderá ter o mesmo nome do `arquivo-origem`. Com este

comando pode-se 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: Copiar o arquivo arquivo.teste para teste:

```
cp arquivo.teste teste
```

mv

```
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 utilizados.

Na linha de comando, arquivo-origem é o nome do arquivo cujo nome se 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, tem-se duas opções:

- Examinar o conteúdo do diretório antes de renomear um arquivo, para verificar se o novo nome a 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 objetiva-se mover um arquivo para um novo diretório, o arquivo terá o mesmo nome de arquivo-origem, a menos que se especifique o novo nome também, dando o nome do caminho (relativo ou absoluto) antes do nome do arquivo.

Exemplo: Mover o arquivo arquivo.teste para o diretório teste interativamente:

```
mv -i arquivo.teste teste
```

ln

```
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, ver Capítulo 4. 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 distintos, 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.

Opção:

`-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:

Criar uma ligação simbólica para o arquivo teste chamado slink:

```
ln -s arquivo.teste slink
ls -l
```

Notas: Quando se liga um arquivo a outro, não está criando um novo 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.

rm

```
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.

find

```
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. Aqui serão explicitados somente os principais.

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` pode-se 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. Encontrar o arquivo teste.arquivo a partir do seu diretório base, imprimir os caminhos:

```
find   -iname teste.arquivo
./curso/aula1/teste.arquivo
./teste.arquivo
```

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

```
find  /curso -user aluno
/home/guest/curso/aula1
...
```

3. Encontrar e apagar todos os seus arquivos teste.arquivo:

```
find ~-iname teste.arquivo -exec rm {}
```

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.

grep

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

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

Se mais de um arquivo for indicado como argumento do comando, o `grep` antecede cada linha de saída que contém a sequência

de caracteres com o nome do arquivo e dois pontos. O nome do arquivo é mostrado para cada ocorrência da sequência 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 sequência de caracteres.

Exemplos:

1. Procurar a sequência 'root' no arquivo `/etc/passwd`:

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

2. Procurar a sequência 'root' em todos os arquivos do diretório /etc, independente de maiúsculas ou minúsculas:

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

more/less

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

Exemplo: `more /etc/passwd`

head e tail

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

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
```

gzip e gunzip

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

Exemplos:

```
gzip teste.arquivo.2
ls -l
gunzip teste.arquivo.2.gz
ls -l
```

tar

`tar` (abreviatura de *Tape ARchive*), 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`.

`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.

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...].
```

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, mostram-se exemplos de utilização do tar. Em alguns parâmetros o uso de '-' (hífen) não é necessário.

```
tar -cvf arq.tar arq1 arq2
tar -czvf arq.tgz *
```

```
tar -rf arq.tar arq*
tar -tzf arq.tar
tar -xv -f arq.tar
```

Apêndice F

Comandos Básicos

F.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:

passwd – Modifica a senha.

ftp – Permite transferência de arquivos.

F.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, ele 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;

F.3 Comandos

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 pode-se executar os comandos do Linux.

Quando o terminal estiver ligado, provavelmente será apresentado uma mensagem 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 uma nova mensagem:

`Password:` Esta pede que seja digitada a senha de usuário. Nota-se que a medida que forem digitados os caracteres, eles não aparecerão no vídeo por medidas de segurança.

Se algo der errado (sendo novamente apresentado o sinal de *login*), tenta-se 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 pode-se ter mais de um comando em seqüência para serem 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, pode-se corrigí-lo utilizando as teclas de direção e para posicionar-se 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.

Logout

Este comando permite sair da seção `shell`, ou seja, desconectar o usuário do sistema.

Reboot

Este comando é equivalente à `init 6` e com ele pode-se reiniciar o sistema, sem desligamento do hardware.

Halt

Este comando é equivalente à `shutdown -r now` e permite desligar o sistema.

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>
```

Exemplo:

```
man ls
```

Apêndice G

Redirecionamentos

G.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.

O objetivo deste capítulo é o aprendizado da manipulação de entradas e saídas, para poder criar e ler arquivos durante o uso de alguns comandos, e também aprender a encadear comandos, fazendo com que um comando utilize como entrada a saída de outro.

G.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 se

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 se 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 for digitado. Desta forma, pode-se certificar de ter digitado o 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, também é possível 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.

G.3 Redirecionamento de Entrada e Saída

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.

G.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.

G.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.

G.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 se 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.

Exemplos :

1. Guardar no arquivo `data.de.hoje` a saída do comando `date`:

   ```
   date > data.de.hoje
   ```

   ```
   cat data.de.hoje
   ```

2. Acrescentar ao arquivo `data.de.hoje` a saída do comando `who`:

   ```
   who » data.de.hoje
   ```

   ```
   cat data.de.hoje
   ```

G.3.4 Pipes

Os símbolos de redirecionamento permitem realizar mais de uma operação em um mesmo arquivo. Somente com esses símbolos já se tem condições de realizar tudo o que quiser sobre um arquivo. Suponha, entretanto, que se 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. Pode-se 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 sequência 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. Contar o número de arquivos que começam com a sub-string 'arq' no diretório corrente:

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

   ```
   3
   ```

2. Contar o número de usuários que estão presentes no sistema neste momento:

   ```
   who | wc -l
   ```

   ```
   2
   ```

G.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. Contar o número de arquivos que começam com a *string* 'arq' no diretório corrente e guardar os arquivos encontrados no arquivo `nomes`:

```
ls | grep arq | tee nomes | wc -l
3
cat nomes arq
arq2
arquivo
```

2. Contar 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
```

G.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 ambiguidade.

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 ambiguidade.

Exemplo :

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

Referências Bibliográficas

[1] Ken O. Burtch. Scripts de Shell Linux com Bash. Editora Ciência Moderna, Rio de Janeiro, Brasil, 2004.

[2] Felipe Costa. Ambiente de Rede Monitorado com Nagios e Cacti. Editora Ciência Moderna, Rio de Janeiro, Brasil, 2008.

[3] George Coulouris, Jean Dollimore, and Tim Kindberg. Sistemas Distribuídos, Conceitos e projeto. Bookman, quarta edição edition, 2007.

[4] Rubem E. Ferreira. Linux – Guia do Administrador do Sistema. Novatec Editora Ltda., São Paulo, Brasil, 2003.

[5] Craig Hunt. Linux – Servidores de rede. Editora Ciência Moderna, Rio de Janeiro, Brasil, 2004.

[6] James F. Kurose and Keith W. Ross. Redes de computadores e a Internet – Uma aobordagem top-down. Pearson Addison Wesley, terceira edição edition, 2006.

[7] Evi Nemeth, Garth Snyder, Scott Seebass, and Trent R. Hein. Manual De Administração do Sistema Unix. Artmed, 3 edition, 2001.

[8] James Stanger, Patrick T. Lane, and Edgar Danielyan. Rede Segura Linux. Editora Alta Books, Rio de Janeiro, Brasil, 2002.

[9] Chuck V. Tibet. Linux – Administração e Suporte. Novatec Editora Ltda., São Paulo, Brasil, 2001.

[10] Odilson Tadeu Valle. Linux – básico, gerência, segurança e administração de redes. 2008.

[11] Marco Álvarez, Cláudia Nasu, Alfredo Lanari, and Luciene Marin. Curso de introdução ao linux.