Apostila de Linux

(Baseada em Ubuntu 14.04)

Autor: Jefferson Costa

Tem 37 anos e é docente há 19 anos.

Atua há mais de 20 anos na área de informática, é administrador de redes Linux, Ethical Hacker e perito em segurança forense computacional.

Site: www.jeffersoncosta.com.br

Fan page: www.facebook.com/jeffersoncosta.com.br

Canal no youtube: www.youtube.com/jcosta20

Twitter: @ProfJcosta

E-mail: jefferson@jeffersoncosta.com.br

São Paulo

2014

Sumário

O que é o Linux?

De maneira simples podemos dizer que o Linux é um sistema operacional multiusuário, multitarefa e multiprocessado, de livre distribuição baseado no sistema operacional UNIX.

O sistema foi desenvolvido pelo finlandês Linus Torvalds, que na época era um estudante de ciência da computação na Finlândia, ele teve a ajuda de vários programadores voluntários através da Usenet (uma espécie de sistema de listas de discussão existente desde os primórdios da Internet).

Linus começou o desenvolvimento do núcleo como um projeto particular, inspirado pelo seu interesse no Minix, um pequeno sistema UNIX desenvolvido por Andrew S. Tanenbaum.

Depois de algum tempo de trabalho no projeto, sozinho, enviou a seguinte mensagem para usuários do Minix:

"*Você suspira pelos bons tempos do Minix-1.1, quando os homens eram homens e escreviam seus próprios "device drivers"? Você está sem um bom projeto em mãos e deseja trabalhar num Sistema Operacional que possa modificar de acordo com as suas necessidades? Acha frustrante quando tudo funciona no Minix? Chega de noite ao computador para conseguir que os programas funcionem? Então esta mensagem pode ser exatamente para você. Como mencionei há um mês atrás, estou trabalhando numa versão independente de um S.O. similar ao Minix para computadores AT-386. Ele está, finalmente, próximo do estado em que poderá ser utilizado (embora possa não ser o que você espera), e eu estou disposto a disponibilizar o código-fonte para ampla distribuição. Ele está na versão 0.02... contudo eu tive sucesso ao executar bash, gcc, gnu-make, gnu-sed, compressão etc. "*

(Finlândia – 1991)

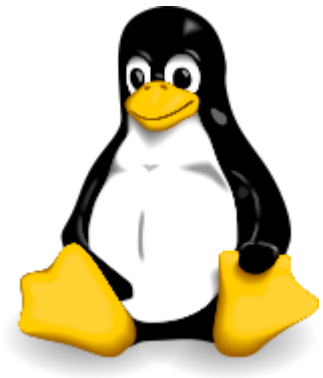
O nome Linux vem da junção do nome de seu criador, Linus Torvalds, com UNIX, o finlandês inicialmente tinha-o batizado como "Freax".

A *Free Software Foundation* afirma que tais sistemas operacionais são, na verdade, sistemas GNU, e o nome mais adequado para tais sistemas é *GNU/Linux*, uma vez que grande parte do código-fonte dos sistemas operacionais baseados em Linux são ferramentas do

projeto GNU. Linus Torvalds afirma que consideraria "justo" que tal nome fosse atribuído a uma distribuição do projeto GNU, mas que chamar os sistemas operacionais Linux como um todo de GNU/Linux seria "ridículo". Linus disse não se importar sobre qual o nome usado, considera a proposta da GNU como "válida", mas prefere usar o termo "Linux".

No dia 5 de outubro de 1991 Linus Torvalds anunciou a primeira versão "oficial" do núcleo Linux, versão 0.02. Desde então muitos programadores têm respondido ao seu chamado, e têm ajudado a fazer do Linux o sistema operacional que é hoje. No início era utilizado por programadores ou só por quem tinha conhecimentos, usavam linhas de comando. Hoje isso mudou, existem diversas empresas que criam os ambientes gráficos, as distribuições cada vez mais amigáveis de forma que uma pessoa com poucos conhecimentos consegue usar o Linux. Hoje o Linux é um sistema estável e consegue reconhecer muitos periféricos sem a necessidade de se instalar os drivers de som, vídeo, modem, rede, entre outros.

O símbolo



Em 1996, muitos integrantes da lista de discussão "Linux-Kernel" estavam discutindo sobre a criação de um logotipo ou de um mascote que representasse o Linux. Muitas das sugestões eram paródias ao logotipo de um sistema operacional concorrente e muito conhecido. Outros eram monstros ou animais agressivos. Linus Torvalds acabou entrando nesse debate ao afirmar em uma mensagem que gostava muito de pinguins. Isso foi o suficiente para dar fim à discussão.

Depois disso, várias tentativas foram feitas numa espécie de concurso para que a imagem de um pinguim servisse aos propósitos do Linux, até que alguém sugeriu a figura de um "pinguim sustentando o mundo". Em resposta, Linus Torvalds declarou que achava interessante que esse pinguim tivesse uma imagem simples: um pinguim "gordinho" e com expressão de satisfeito, como se tivesse acabado de comer uma porção de peixes. Torvalds também não achava atraente a ideia de algo agressivo, mas sim a ideia de um pinguim simpático, do tipo em que as crianças perguntam "mamãe, posso ter um desses também?". Ainda, Torvalds também frisou que trabalhando dessa forma, as pessoas poderiam criar várias modificações desse pinguim. Isso realmente acontece.

Quando questionado sobre o porquê de pinguins, Linus Torvalds respondeu que não havia uma razão em especial, mas os achava engraçados e até citou que foi mordido por um "pinguim assassino" na Austrália e ficou impressionado como a mordida de um animal aparentemente tão inofensivo podia ser tão dolorosa.

Principais características

- **Multiusuário:** Permite que vários usuários possam rodar o sistema operacional, e não possui restrições quanto à licença. Permite vários usuários simultâneos, utilizando integralmente os recursos de multitarefa. A vantagem disso é que o Linux pode ser distribuído como um servidor de aplicativos. Usuários podem acessar um servidor Linux através da rede local e executar aplicativos no próprio servidor.

- **Multiplataforma:** O Linux roda em diversos tipos de computadores, sejam eles RISC ou CISC.
- **Multitarefa:** Permite que diversos programas rodem ao mesmo tempo, ou seja, você pode estar imprimindo uma carta para sua vovó enquanto trabalha na planilha de vendas, por exemplo. Sem contar os inúmeros serviços disponibilizados pelo Sistema que estão rodando em background e você provavelmente nem sabe.
- **Multiprocessador:** Permite o uso de mais de um processador. Já é discutida, há muitos anos, a capacidade do Linux de poder reconhecer mais de um processador e inclusive trabalhar com SMP, clusters de máquinas, na qual uma máquina central controla os processadores das outras para formar uma só máquina.
- **Protocolos:** Pode trabalhar com diversos protocolos de rede (incluindo o TCP/IP que é nativo Unix).
- **Consoles virtuais:** Permite que o usuário tenha mais de um console para trabalhar, sendo que em cada console você pode ter diversas tarefas sendo executadas em background e mais em foreground (segundo plano e primeiro plano).

Linux como software Livre

O criador do movimento pelo software aberto e livre foi Richard Stallman. Em determinada ocasião, ele precisou corrigir o driver de uma impressora que não estava funcionando. Solicitou então, ao fabricante do driver o código fonte do programa para que pudesse realizar as correções necessárias. Para sua surpresa, o pedido foi negado. Daí ele iniciou então um esforço gigantesco para conceder versões abertas para todas as categorias de software existentes, comercializadas sem acesso ao código fonte.

Richard Stallman fundou a FSF – Free Software Foundation. A FSF criou os aplicativos utilizados por todos os sistemas semelhantes ao Unix, como Linux e FreeBSD, hoje tão populares.

Para evitar que alguém obtivesse o programa com o seu código fonte, fizesse alterações e se declarasse como dono do produto, ele estabeleceu a forma sob a qual esses programas poderiam ser distribuídos.

O projeto GNU, através da GPL (General Public License) que é a licença do Software Livre, especifica que o programa pode ser usado e modificado por quem quer que seja desde que as modificações efetuadas sejam também disponibilizadas em código fonte.

O Kernel do Linux também é distribuído sob a GNU (General Public License).

Seu Kernel associado a esses programas tornou possível a milhões de pessoas o acesso a um excelente ambiente computacional de trabalho e que melhora a cada dia.

Sistemas de Arquivos

É uma forma de armazenamento de arquivos em estruturas (na maneira hierárquica) de diretórios. Assim, o usuário não necessita conhecer detalhes técnicos do meio de armazenamento. Ele apenas precisa conhecer a estrutura (árvore) de diretórios para poder navegar dentro dela e acessar suas informações

Ele suporta diversos sistemas de arquivos, dentre eles:

- **EXT2 -** Foi durante muito tempo o sistema de arquivos padrão do Linux -- praticamente o único disponível depois que o Extended (ext) foi removido dos kernels 2.0 e 2.1.
- **EXT3 -** Sistema de arquivo bastante difundido. Além de funcionalidades usuais de sistemas de arquivos possui suporte de paginação e proteção contra o processo de reinicializar de maneira forçada.
- **EXT4** - Atualização do sistema EXT3, possui Journaling que faz indexação de arquivos, facilitando na recuperação de arquivos apagados. O tipo de fragmentação também é importante na escolha do sistema de arquivos, se ele fragmenta muito ou não, o NTFS fragmenta muito e o EXT não.
- **VFAT -** Usados para acessar sistemas de arquivos FAT 16/FAT 32.
- **NFS -** Usados para montagem de sistema de arquivos remotos, ou seja, no modo cliente servidor.
- **ISO 9660 -** Sistema de arquivo local usado para acessar CD-ROM.
- **REISERFS -** Sistema de arquivos com suporte, a características avançadas, como paginação e melhor suporte a diretórios muito grandes.
- **PROC -** Sistema de arquivo virtual que fornece, entre outras coisas informações gerais sobre sistemas (Sistema de arquivos do kernel).
- **HPFS -** Sistema de arquivo local usado para acessar partições HPFS do MAC OS X.

Distribuições Linux

Com a popularidade do Linux, a cada momento surgem diversas novas distribuições e versões diferentes, abaixo vou destacar as principais:

Slackware Linux

É o nome de uma das mais antigas e conhecidas distribuições do Linux, além de ser, junto com suas derivadas, a distribuição Linux mais UNIX-like existente

Criada em meados de 1993 e mantida por Patrick Volkerding, a Slackware (ou simplesmente "Slack") tem como objetivo manter-se fiel aos padrões UNIX, rejeitando também ferramentas de configuração que escondam do usuário o real funcionamento do sistema.

Por sua concepção UNIX-like, o Slackware e seus derivados fazem uma abordagem bastante diferente das outras distribuições populares como Red Hat, Fedora, Debian, Gentoo, SuSE, e Mandriva.

É considerado o mais poderoso dos Linux.

Red Hat Linux

É uma distribuição de Linux muito conhecida, líder do mercado nos EUA, criada e mantida pela Red Hat. Um grupo de programadores na Carolina do Norte decidiu tornar o Linux mais fácil para possibilitar às pessoas uma experiência mais tranquila com o mesmo.

Como muitos grupos, seu objetivo era empacotar todos os bits necessários numa distribuição coerente, facilitando aos inexperientes o contato com o novo sistema operacional.

Esta distribuição porém tinha uma característica distinta das demais. Em vez de ser uma cópia de um disco rígido que tivesse o Linux instalado, ou um conjunto de disquetes com partes diferentes do sistema operacional que podiam ser copiadas, esta distribuição foi baseada no conceito de pacotes.

Cada pacote fornece um pedaço diferente de software, configurado, completamente testado e pronto para rodar.

Debian

É o nome de uma distribuição não comercial livre (gratuita e de código fonte aberto) de GNU/Linux (amplamente utilizada) e de um grupo de voluntários que o mantêm à volta do mundo. Uma vez que o Debian se baseia fortemente no projecto GNU (a distribuição oficial do Projeto GNU é **Debian**), é usualmente chamado Debian GNU/Linux.

O Debian é especialmente conhecido pelo seu sistema de gestão de pacotes, chamado APT, que permite: atualizações relativamente fáceis a partir de versões realmente antigas; instalações quase sem esforço de novos pacotes e remoções limpas dos pacotes antigos.

O projecto Debian é mantido por doações através da organização sem fins lucrativos Software in the Public Interest (SPI).

O nome Debian vem dos nomes dos seus fundadores, Ian Murdock e de sua mulher, Debra. A palavra "Debian" é pronunciada em Português como Débian.

Várias distribuições comerciais baseiam-se (ou basearam-se) no Debian, incluindo: Linspire (antigo Lindows), Xandros, Kurumin, Debian-BR-CDD, Ubuntu Linux e Libranet.

OpenSUSE

É um sistema operacional baseado em GNU/Linux da comunidade mundial patrocinado pela Novell. O sistema operacional é uma alteração do SUSE Linux. Tem como base os pacotes *.rpm (Redhat Package Manager). Sua ultima versão é a 10.3.

O grande diferencial desta distribuição é possuir o Yast um aplicativo que traz a possibilidade de configurar de forma gráfica diversos elementos do sistema operacional que normalmente são configurados através de linha de comando.

Ubuntu

É um sistema operacional Linux baseado na distribuição Debian. É patrocinado pela Canonical Ltd (dirigida por Mark Shuttleworth) e o seu nome deriva do conceito sul africano Ubuntu, diretamente traduzido como "humanidade para com os outros". Diferencia-se do Debian por ser lançado semestralmente, por disponibilizar suporte técnico nos dezoito meses seguintes ao lançamento de cada versão (em inglês) e pela filosofia em torno de sua concepção.

Mandriva

É uma empresa franco-brasileira de software, dedicada à distribuição e suporte do sistema operacional Mandriva Linux. Tem sua sede administrativa em Paris e centro de desenvolvimento em Curitiba, além de possuir também um escritório em San Diego, nos Estados Unidos. O centro brasileiro recebe o nome de Mandriva Conectiva.

A empresa começou suas atividades em 7 de abril de 2005, ao juntar os ativos da empresa francesa Mandrake e a brasileira Conectiva. No mesmo ano, a Mandriva ainda adquiriu os ativos da Lycoris, responsável por outra distribuição Linux homônima, nos Estados Unidos, para usuários domésticos.

Outras Distribuições

Alinex – Arch - BigLinux - BrDesktop - BRLix - Caixa Mágica - DreamLinux - Famelix - Fedora - Fluxbuntu - Foresight - Fox - Freedows - Gentoo - gNewSense - gnuLinEx - Gobuntu - Indymix - Insigne - Kalango - Knoppix - Kubuntu - Libertas - Linspire - Lubuntu - Mint - Poseidon - Satux - SLS Linux - Super OS - TurboLinux - Ulteo - Vector - Xandros – Xubuntu

Específicos para Firewall – Coyote – BrazilFW

Pacotes Linux

Um pacote contém o código fonte pré-compilado e empacotado como um arquivo binário de instalação (executável). Nele podem estar ícones, bibliotecas, arquivos de configuração, binários, man pages, atalhos de desktop, headers, fontes etc. Além disso, um pacote pode conter metadados, como informações sobre versão, mantenedor do pacote, autor do software, informações de contato, licenciamento, alterações, READMEs e o site do projeto e do código fonte. Cada formato de pacote tem sua estrutura de arquivos para armazenar dados e é compactado. Quando o pacote é executado, seus dados são descompactados e copiados para o sistema de arquivos do sistema operacional, criando links simbólicos onde for necessário, atalhos no menu e no desktop e, às vezes, oferecendo opções de configuração ao usuário.

Há duas maneiras de se instalar um programa, a primeira é compilando o código fonte e a segunda é instalando um pacote.

Os pacotes são criados para uma versão específica de uma determinada distribuição, pois as dependências podem variar entre distribuições e versões de uma distribuição. Às vezes é possível baixar e instalar programas como no Windows, bastando clicar no pacote, desde que ele seja compatível com o sistema operacional.

Alguns tipos de pacote

pkgtool

O Slackware e seus derivados usam esse sistema. Os pacotes estão no formato tar.gz, com extensão .tgz. Ou seja, trata-se de um arquivo tar (tape archive) compactado com o gzip (gz). Na verdade, ele não é bem um gerenciador de pacotes, é só um formato de pacote e algumas ferramentas de linha de comando para criar, exibir, instalar, remover e atualizar.

O sistema de empacotamento permite que scripts de instalação sejam embutidos no arquivo. Esses scripts são a única diferença entre a instalação de uma tarball (um pacote tar compactado) e a instalação um pacote do Slackware.

Não há verificação de dependências, nem conexão automática a repositórios, atualizações automáticas do sistema ou verificação de checksums. Com o pkgtool, é possível acessar uma lista de pacotes instalados e removê-los, instalar pacotes ou executar scripts de instalação. Os pacotes são baixados manualmente de um repositório. Esse era o sistema que todos os gerenciadores de pacotes tentavam aperfeiçoar na época em que o Slackware Linux era a distribuição Linux dominante.

O Slackware permite navegar entre os pacotes pela internet, além de oferecer feeds RSS. Os dois recursos parecem ter sido adicionados recentemente. Há também ferramentas desenvolvidas por terceiros, como SWareT, slapt-get, slackpkg e o pkgsrc do NetBSD, que ajudam no gerenciamento de pacotes do Slackware e de seus derivados. Todas essas ferramentas oferecem resolução de dependências e algumas funções avançadas.

Advanced Packaging Tool (APT)

O APT é usado principalmente no Debian e em seus derivados. O APT é uma biblioteca de rotinas que agem como uma interface para o dpkg, que é um gerenciador de pacotes de baixo nível que permite instalar, desinstalar e atualizar pacotes .deb.

O APT traz funções avançadas ao dpkg, dentre as quais está a resolução de dependências. Todos os derivados do Debian usam o APT por padrão.

O desenvolvimento de novas capacidades tem se mantido num ritmo semelhante ao de outros gerenciadores de pacotes mais recentes. Não há como duvidar que o APT é um dos melhores e mais completos gerenciadores de pacotes.

Ele já foi portado para o OpenSolaris e o Mac OS X, e pode ser usado em distribuições baseadas em RPM por meio do apt-rpm.

Alguns comandos

- apt-get install [pacotes] - usado para instalar novos pacotes em sua distribuição. Podem ser instalados mais de um pacote ao mesmo tempo separando os nomes por espaços. Somente é preciso especificar o nome do pacote (sem a versão e revisão).
  O apt instalará automaticamente as dependências necessárias para o funcionamento correto do pacote. Quando pacotes além do solicitado pelo usuário são requeridos para a instalação, o apt mostrará o espaço total que será usado no disco e perguntará ao usuário se ele deseja continuar. Após a instalação, o pacote será automaticamente configurado pelo dpkg para ser executado corretamente em seu sistema.
- apt-get clean – usado para apagar qualquer arquivo baixado durante uma atualização ou instalação de arquivos com o apt. Os arquivos baixados residem em /var/cache/apt/archives (download completo) e /var/cache/apt/archives/partial (arquivos sendo baixados - parciais).
  Este local de armazenamento é especialmente usado com o método http e ftp para armazenamento de arquivos durante o download para instalação (todos os arquivos são primeiro copiados para serem instalados e configurados).
- apt-get remove [pacotes] – usado para remover completamente um pacote do sistema. Podem ser removidos mais de um pacote ao mesmo tempo separando os nomes dos pacotes com espaços. O apt-get remove remove completamente o pacote mas mantém os arquivos de configuração, exceto se for adicionada a opção --purge.
  É preciso especificar somente o nome do pacote.

RPM Package Manager (RPM)

O RPM é um formato e um gerenciador de pacotes que é tão popular quanto o APT.

Embora muitas funções de alto nível tenham sido implementadas diretamente no RPM desde o início, como a verificação de dependências (mas não sua resolução), parece que não é tão fácil adicionar ao padrão RPM os recursos encontrados em outros sistemas de gerenciamento de pacotes modernos. Isso levou ao surgimento de novas ferramentas de gerenciamento de pacotes, como o YUM, urpmi, YaST, up2date e o apt-rpm, que oferecem resolução de dependências e outros recursos avançados, deixando as rotinas de baixo nível por conta do RPM. Essas rotinas são por vezes chamadas de Meta Package Managers, ou metagerenciadores de pacotes, porque gerenciam o RPM que já é um gerenciador de pacotes.

O RPM foi portado para a arquitetura AIX da IBM e é o formato de pacotes padrão da LSB (a Linux Standard Base).

Alguns comandos:

- rpm -ivh - Instalação de pacotes;
- rpm -Uvh - Atualização de pacotes;
- rpm -qi - Informações sobre o pacote;
- rpm –ql – Lista os arquivos do pacote;
- rpm –e – Desinstala o pacote;
- rpm –qa – Lista os pacotes instalados

Urpmi

O Mandriva é a única distribuição a usar o urpmi, assim como só o openSUSE usa o ZYpp. O formato do pacote é o .rpm.

O urpmi é um dos primeiros (talvez o primeiro) gerenciador de pacotes RPM. Ele consiste de vários utilitários diferentes que realizam funções diversão: o urpme desinstala programas, o urpmq faz consultas ao banco de dados em busca de arquivos, o urpmi instala pacotes e por aí vai. Uma função interessante do urpmi é que ele adiciona metadados aos RPMs instalados a partir de diretórios locais.

Yellow Dog Updater Modified (YUM)

Nascido do Yellow Dog Updater (YUP), o YUM é o gerenciador de pacotes dos sistemas baseados no Red Hat e no Fedora, e usa pacotes RPM. Ele se tornou o gerenciador de pacotes padrão do Red Hat Enterprise Linux 5 e é usado por boa parte dos sistemas baseados no Red Hat e no Fedora. A modularidade é um dos principais recursos do YUM. Novos recursos podem ser adicionados ao YUM por meio de plugins e do pacote yum-utils. Há quem critique, afirmando que o YUM não é uma ferramenta suficientemente integrada, e que o desempenho e a maturidade dos módulos varia. O fato é que sua ampla adoção atesta a boa qualidade desse sistema de gerenciamento de pacotes. Há tempos a Red Hat oferece um serviço de assinatura, a RHN (Red Hat Networks), que fornece atualizações e patches . A assinatura é um serviço importante para o plano de negócios da Red Hat, e por isso ela não dedicou muito tempo ao desenvolvimento de um serviço de gerenciamento de pacotes para não assinantes. Assim, o YUM foi desenvolvido por terceiros, antes de ser adotado pela Red Hat. O RPM é o gerenciador de pacotes tradicional da Red Hat, sucedido pelo up2date e agora substituído pelo YUM.

Alguns comandos:

- yum list - Lista todos os pacotes disponíveis;
- yum check-update ou yum list updates - Verifica se há pacotes disponíveis para um update ;
- yum update - Faz o update de seu sistema;
- yum install <package(s)> - Instala um pacote específico e suas dependências;
- yum info <package> - Apresenta informações básicas de um determinado pacote

ZYpp

O openSUSE oferece um amplo cardápio de utilitários para o gerenciamento de pacotes. Entram-se os dados pelo rug (uma ferramenta de linha de comando) ou pelo zen-updater (uma interface gráfica), e eles são direcionados ao Zenworks Management Daemon (ZMD). O ZMD espera os comandos e os transmite aos assistentes ZMD do libzypp, que se comunicam com o banco de dados de software, processam os metadados e os transmitem ao libzypp. O libzypp resolve dependências, instala, remove e atualiza pacotes, usando o utilitário de gerenciamento de pacotes RPM. O YaST e o zipper (este último pela linha de comando) podem ser usados para comunicação direta com o libzypp. É o extremo oposto do pkgtool do Slackware. Três interfaces, dois sistemas de gerenciamento de pacotes e dois repositórios. O sistema zen-updater também acrescenta o daemon e uma camada de auxiliares (que nenhum outro sistema possui) até que o gerenciador de pacotes entre em ação para resolver as dependências e rotinas de instalação.

Compactação

O **Tar** e o **gzip** são duas ferramentas utilizadas em sistemas operacionais baseados no Unix, como o GNU/Linux, para o "empacotamento" e para a compressão de arquivos, respectivamente. Embora seja perfeitamente possível usar qualquer desses programas de forma individual, a utilização de ambos ao mesmo tempo é muito comum e útil.

Comando Tar

Backup (cópia de segurança) de arquivos é uma necessidade antiga. Há várias formas de se fazer isso, mas nos sistemas operacionais baseados no Unix, uma das maneiras mais tradicionais corresponde à utilização da ferramenta **Tar**, sigla de ***T**ape **Ar**chive*. O que o Tar faz é muito simples de entender: ele "empacota" vários arquivos em um só, isto é, faz com que um

único arquivo contenha vários outros. Assim, é possível, por exemplo, armazenar em único arquivo as cópias de documentos existentes na pasta de um usuário.

O arquivo resultante de um empacotamento feito com Tar tem, como é de se esperar, a extensão .tar (por exemplo, teste.tar), embora sua utilização não seja obrigatória (mas é recomendada para fins de organização). Quando for necessário extrair o conteúdo existente dentro de um arquivo .tar, naturalmente, basta acionar o programa Tar. Os procedimentos para empacotamento e extração de arquivos são executados através de comandos e parâmetros inseridos em terminais (shell). Quando um usuário domina essas instruções, consegue executar tais tarefas de forma ágil. Isso se deve principalmente ao fato do Tar manter as propriedades dos arquivos e a estruturas de diretórios originais, facilitando a localização e a utilização de cada item após a extração.

A sintaxe do Tar é a seguinte:

tar [parâmetros] [nome_do_arquivo_tar] [arquivos_de_origem]

Na linha acima, **tar** é o comando. Em **parâmetros**, é possível utilizar várias opções. Eis as principais:

-c - cria um novo arquivo tar;
-t - exibe o conteúdo de um arquivo tar;
-p - mantém as permissões originais do(s) arquivo(s);
-r - adiciona arquivos a um arquivo tar existente;
-f - permite especificar o arquivo tar a ser utilizado;
-v - exibe detalhes da operação;
-w - pede confirmação antes de cada ação no comando;
-x - extrai arquivos de um arquivo tar existente;
-z - comprime o arquivo tar resultante com o gzip (visto mais à frente);
-C - especifica o diretório dos arquivos a serem armazenados (note que, neste caso, a letra é maiúscula).

O campo **nome_do_arquivo_tar** especifica qual o nome que o arquivo .tar terá, e o campo **arquivos_de_origem** define o diretório ou os arquivos que se tornarão um .tar. Vamos ver alguns exemplos para facilitar a compreensão:

tar -cf documento.tar teste1.txt teste2.txt

O comando acima cria o arquivo **documento.tar**, que contém os arquivos **teste1.txt** e **teste2.txt**. Aqui, você deve ter reparado que é possível combinar parâmetros. Neste exemplo, isso ocorreu com **-c** e **-f**. No exemplo abaixo, o diretório **hardware** tem todo o seu conteúdo compactado no arquivo **teste.tar**, só que os detalhes são exibidos graças à opção **-v**:

tar -cvf teste.tar hardware

O exemplo a seguir lista o conteúdo do arquivo **teste.tar**:

tar -tf teste.tar

Por sua vez, o comando abaixo faz com que todos os arquivos de **teste.tar** sejam extraídos (neste ponto, você certamente já sabe as funções dos parâmetros **x**, **v** e **f** no comando):

tar -xvf teste.tar

Já no comando a seguir, apenas o arquivo teste1.txt é extraído:

tar -xvf documento.tar teste1.txt

Uma coisa interessante é que, se a opção **-v** for usada duas vezes, detalhes como permissões e data do(s) arquivo(s) aparecerão:

tar -xvvf documento.tar teste1.txt

Comando gzip

A ferramenta Tar, por si somente, serve apenas para juntar vários arquivos em um só. No entanto, o programa não é capaz de diminuir o tamanho do arquivo resultante, isto é, de compactá-lo. É neste ponto que entra em cena o **gzip** (*GNU zip*) ou outro compactador de sua preferência. Se utilizado isoladamente, o gzip faz uso da seguinte sintaxe:

gzip [parâmetros] [nome_do_arquivo]

Entre os parâmetros disponíveis, tem-se:
-c - extrai um arquivo para a saída padrão;
-d - descompacta um arquivo comprimido;
-l - lista o conteúdo de um arquivo compactado;
-v - exibe detalhes sobre o procedimento;
-r - compacta pastas;
-t testa a integridade de um arquivo compactado.

Ainda no que se refere às opções de parâmetros, é possível utilizar uma numeração de 1 a 9 para indicar o nível de compactação. Quanto maior o número, maior será a compactação do arquivo.

Eis alguns exemplos para facilitar a compreensão do comando gzip:

gzip importante.odt

O comando acima compacta o arquivo **importante.odt**. Note que os arquivos compactados com gzip recebem a extensão .gz.

gzip -d importante.odt.gz

O comando acima descompacta o arquivo **importante.odt.gz**.

gzip -1 importante2.ods

O procedimento acima faz com que o arquivo **importante2.ods** seja compactado considerando o nível mais baixo de compreensão.

Usando Tar e gzip

O uso conjunto dos comandos Tar e gzip é um belo exemplo de coerência da frase "a união faz a força". Muitas vezes, é necessário juntar arquivos e, ao mesmo, fazer com que o arquivo resultante, além de conter todos os outros, também seja compactado. É aí que entra em cena a capacidade de juntar arquivos do Tar com a capacidade de compactação do gzip. Para utilizar ambos ao mesmo tempo, o procedimento é muito simples: basta aplicar o comando **tar** com o parâmetro **-z**. O arquivo resultante desse procedimento receberá a extensão .tar.gz.

Neste ponto, vemos um comando bastante usado na instalação de programas e bibliotecas:

tar -zxvf nome_do_arquivo.tar.gz

Se você estiver lendo este artigo deste o começo, certamente já sabe o que o comando acima faz, mesmo assim, vamos explicar para não restar dúvidas: a letra **z** deve ser usada porque o arquivo foi compactado com gzip; a letra **x** indica que o comando deve extrair o arquivo (portanto, a referida instrução serve para extrair e descompactar o arquivo tar.gz); a letra **v** exibe os detalhes do procedimento; por fim, a letra **f** especifica qual arquivo será usado nesta atividade.

Suponha, agora, que você queira deixar em um único pacote os arquivos **camila.png**, **jefferson.txt** e **jhtec.odt**. O arquivo resultante terá o nome **informacoes.tar.gz**. Eis o comando que utilizaremos para este exemplo:

tar -zcvf informacoes.tar.gz camila.png jefferson.txt jhtec.odt

Note que o comando é muito parecido com o procedimento de descompactação do exemplo anterior, com a diferença de que o parâmetro **c** foi utilizado no lugar de **x**, pois o objetivo aqui é criar um arquivo novo, e não fazer a extração de um já existente. Para extrair o conteúdo desse arquivo, basta executar o comando abaixo (também exibido na figura acima):

tar -zxvf informacoes.tar.gz

Se você quiser extrair apenas um dos arquivos contidos no arquivo compactado, basta indicá-lo no final do comando. Por exemplo, suponha que você queira extrair o arquivo **camila.png** de **informacoes.tar.gz**. Eis o que você deve digitar:

tar -zxvf informacoes.tar.gz camila.png

Usando Tar e bzip2

A combinação Tar e gzip é muito utilizada, mas não é a única. Também é possível utilizar o algoritmo de compressão **bzip2**, cuja extensão é .bz2. Há quem prefira esta opção pela característica do bzip2 de gerar arquivos menores que o gzip, embora o programa o faça de maneira mais lenta que este último.

Para utilizar Tar com bzip2, basta utilizar o parâmetro **-j**. Por exemplo:

tar -jcvf frases.tar.bz2 texto1.html texto2.html

Para extrair o conteúdo arquivo, o comando é:

tar -jxvf frases.tar.bz2

Caso queira utilizar o bzip2 isoladamente, a sintaxe é:

bzip2 [parâmetros] [nome_do_arquivo]

Os parâmetros são praticamente os mesmos do gzip, por isso não serão mostrados aqui. Eis um comando de exemplo:

bzip2 -d texto3.htm.bz2

Esse comando descompacta o arquivo **texto3.htm**.

Alguns Comandos

login

É importante ressaltar que o Linux possui dois tipos de usuário o superusuário chamado **root** (Administrador do Sistema), ele tem total poder no sistema, é apelidado de “Dono da maquina" e os usuários com poderes limitados.

Este comando abre uma nova sessão para um usuário. Esta nova sessão assume o perfil do usuário com todas as características associadas a ele.

logout

Tem como função desconectar um usuário de uma determinada sessão.

exit

Seu objetivo é encerrar uma sessão de trabalho.

su ou sudo

Para identificarmos o root no ambiente texto em frente ao seu nome temos o símbolo # e os outros usuários **$.**

Exemplo de usuário Root:

[root@nomedocomputador /]#

Para promovermos temporariamente um usuário a root, digitamos o comando su e depois colocamos a senha do usuário root para permitir a mudança.

[usuario@nomedocomputador /]$ su

Passwd: *******

[usuario@nomedocomputador /]#

Podemos também utilizar o sudo, isso dependerá da distribuição que você estiver utilizando.

Para tirar a permissão de administrado do root, basta digitar **exit.**

touch

O comando touch é usado atualizar as informações sobre as datas de último acesso e última modificação de um arquivo.

Sintaxe:

$ touch [opções] [arquivo]

Se o arquivo não existir, ele é criado, por padrão. Isso faz o touch ser muito utilizado para criar arquivos vazios, através do comando touch [arquivo].

Opções:

-a: Modifica apenas a data do último acesso;

-c: Não cria arquivos, caso eles não existam;

-m: Modifica apenas a data de modificação;

-t: A data e hora a ser utilizada para o último acesso ou última modificação. O formato utilizado é MMDDhhmm (mês, dia, hora e minuto);

Exemplo:

Criar um arquivo chamado teste:

$ touch teste

mkdir

O comando mkdir, abreviatura de make directory (criar diretório), é usado para criar um novo diretório.

Sintaxe:

$ mkdir [opções] [novo diretório]

Opções:

-m: Especifica as permissões que do novo diretório terá;

-p: Cria todos os diretórios e subdiretórios necessários;

-v: Exibe uma mensagem para cada diretório criado.

Em [novo diretório] devemos colocar os diretórios que queremos criar.

Não é necessário digitar o caminho completo, caso queiramos criar um diretório dentro do diretório atual. Podemos criar vários diretórios com um único comando, bastando separá-los com espaços. Se quiser que o nome do novo diretório tenha espaços em branco, escreva-o entre aspas duplas, dessa forma:

Exemplo:

Criar uma pasta chamada pasta1:

$ mkdir pasta1

rm

O rm é utilizado para excluir arquivos.

Sintaxe:

$ rm [opções] [arquivo]

Opções:

-f: Modo forçado, não pede confirmação para realizar as operações;

-i: Pede confirmação antes de remover qualquer arquivo;

-R, -r: Exclui recursivamente todo o conteúdo do diretório e o próprio diretório. Quando quiser excluir um diretório que não está vazio, utilize esse parâmetro;

-v: Mostra os detalhes das exclusões.

Exemplo:

Excluir um arquivo chamado teste:

$ rm teste

rmdir

Esse comando é utilizado para apagar um diretório vazio.

Sintaxe:

$ rmdir [opções] [diretório]

Opções:

-p: Remove os diretórios-pai do diretório selecionado, se possível. Assim, o comando rmdir -p a/b/c/ vai apagar o diretório a/b/c/, depois o diretório a/b/ e por fim o diretório a/, se possível;
-v: Mostra os detalhes da remoção dos diretórios.

Podem ser especificados mais de um diretório por vez. O rmdir só remove diretório vazios.

Exemplo:

Excluir um diretório vazio chamado pasta1 :
$ rmdir pasta1

Excluir um diretório com arquivos chamado pasta2 :
$ rm -r pasta2
(você Poderá usar o –r, acompanhado do –i e/ou –v)

clear

Elimina todo o conteúdo visível, deixando a linha de comando no topo, como se o sistema acabasse de ter sido acessado;

cd

O comando cd, sigla de change directory (selecionar diretório), serve para acessar um determinado diretório.

.. : Diretório acima do atual. Se você estiver no diretório /home/aluno/ e quiser acessar o diretório /home/, digite cd ... Se quiser acessar o diretório /home/davidson/, digite cd ../davidson/;
~ : Diretório pessoal do usuário atual, ou seja, /home/[usuário]/;
- : Diretório anterior. Se você estava no diretório /etc/ e mudou para o diretório /home/, digite cd - para voltar ao diretório /etc/.

Se for usado sem parâmetro, ou seja, **apenas cd**, você será redirecionado para o diretório pessoal do usuário atual.

Sintaxe:

$ cd [diretório]

Exemplo:

Acessar o diretório chamado pasta3 (Gravado dentro de / que a raiz do sistema Linux):
$ cd /pasta3

pwd

O pwd, sigla de print working directory (exibir diretório de trabalho), exibe o diretório atual. É equivalente a echo $PWD.

Uso:

$ pwd

ls

O comando ls exibe arquivos ou o conteúdo de um ou vários diretórios, semelhante ao comando dir do MS DOS.

Sintaxe:

$ ls [opções] [diretório]

Opções:

-a: Exibe arquivos ocultos;

-A: Não exibe os diretórios

--author: Mostra o autor (criador) de cada arquivo;

-b: Exibe caracteres de escape octais no lugar dos caracteres que não podem ser vistos, como o espaço em branco;

--block-size = [tamanho]: Exibe o tamanho dos arquivos em múltiplos do número de bytes especificado nesse parâmetro;

-B: Não exibe arquivos de backup (terminados com ~);

-c: Lista os arquivos por ordem da data da última modificação;

-C: Exibe a listagem em colunas;

--color = [quando]: Controla quando as cores devem ser usadas para distinguir os tipos de arquivos. Os valores aceitos são:
never: Não usa cores pra nenhum tipo de arquivo;
always: Usar cores para todo tipo de arquivo;
auto: Seleciona quais arquivos serão exibidos em cores.
-d: Exibe o diretório especificado, e não o seu conteúdo;
-f: Ativa os parâmetros -a e -U e desabilita os parâmetros -l, -s e -t;
-F: Acrescenta um caracter gráfico ao final de cada arquivo para identificar o seu tipo;
-G: Não exibe informações dos grupos a que os arquivos pertencem;
-h: Exibe os tamanhos dos arquivos em uma forma legível (2K, 21M, 1G);
--si: Semelhante ao -h, mas usa múltiplos de 1000 bytes ao invés de 1024;
-H: Exibe os arquivos para os quais os links simbólicos apontam, ao invés de listar só o link;
-i: Exibe o número de índice (I-node) dos arquivos;
-I: Não exibe entradas que contiverem o padrão informado;
-k: Equivalente a --block-size=1k;
-l: Listagem detalhada, com diversas informações sobre os arquivos;
-L: Quando listar links simbólicos, lista o local para onde o link aponta, e não o link propriamente dito;
-m: Lista os arquivos em linhas, separando cada item com uma vírgula;
-n: O mesmo que o parâmetro -l, mas mostra as UID's e GID's ao invés dos nomes dos grupos;
-o: O mesmo que o parâmetro -l, mas não exibe as informações sobre o grupo;
-p: Adiciona um caracter para identificar o tipo do arquivo. O mesmo que **–F:** mas não utiliza o caracter * (asterisco);
-Q: Exibe os nomes das entradas entre " (aspas duplas);
-r: Organiza a lista na ordem inversa;
-R: Lista recursivamente o conteúdo dos diretórios e subdiretórios do diretório atual;
-s: Exibe o tamanho de cada arquivo, em múltiplos de blocos (especificados com o parâmetro --block-size=[tamanho]);
-S: Organiza a lista de acordo com o tamanho do arquivo;
-t: Lista pela data de modificação;
-u: Organiza a listagem pela data do último acesso;
-U: Não organiza a listagem, exibindo os arquivos na sequência em que estão gravadas no diretório;
-w: Ajusta o tamanho da tela para o número de colunas especificado;
-X: Organiza a listagem em ordem alfabética;
-1: Lista apenas um arquivo por linha;

Em [arquivo], devemos informar quais arquivos (arquivos, diretórios, dispositivos, links, etc.) devem ser listados. Se não for informado nada, será listado o conteúdo do diretório atual (.).

Pode-se também utilizar curingas para filtrar os arquivos que serão listados. Por exemplo, podemos usar ls *.sxw para listar somente os arquivos terminados em .sxw.

cp

O cp, abreviação de copy (copiar), é utilizado para copiar arquivos e diretórios de um local para outro, com o mesmo nome ou com nome diferente.

Sintaxe:

$ cp [opções] [origem] [destino]

Opções:

-b: Cria um arquivo dos arquivos de destino se eles estiverem para ser sobrescritos;

-P: Quando tratar de links simbólicos, copia o link, e não o local para onde o link aponta;

-f: Operação forçada. Se um dos arquivos de destino não puder ser aberto, apaga-o e repete a operação;

-i: Pede confirmação antes de sobrescrever um arquivo;

-L: Quando tratar de links simbólicos, copia o local para o onde o link aponta, e não o link;

-p: Preserva as propriedades do arquivo (permissões, dono e datas);

--preserve = [propriedade]: Escolhe quais propriedades preservar, separadas por vírgula. Podem ser:

mode: Preserva as permissões;

ownership: Preserva a informação de dono do arquivo;

timestamp: Preserva as datas de acesso e modificação.

--no-preserve = [propriedade]: Escolhe quais propriedades não devem ser preservadas. As opções são as mesmas que do parâmetro --preserve;

-R ou -r: Modo recursivo, copia todos os arquivos e subdiretórios do diretório especificado. Esse parâmetro deve ser usado para copiar diretórios inteiros;

--target-directory = [diretório]: Especifica para qual diretório devem ser copiados os arquivos/diretórios especificados;

-u: Copia apenas os arquivos novos. Se um arquivo que estiver sendo copiado já existir no diretório de destino, sua cópia será ignorada;

-v: Mostra os detalhes da cópia dos arquivos.

Exemplos de uso:

Para copiar o arquivo file.gz para o diretório /tmp/:

$ cp file.gz /tmp

Para fazer uma cópia do arquivo file.gz com o nome file-copia.gz:

$ cp file.gz file-copia.gz

Para copiar os arquivos file1, file2 e file3 para o diretório /home/davidson/doc/:

$ cp file1 file2 file3 /home/davidson/doc

Para copiar o diretório img/ para o diretório /tmp/upload/:

$ cp -r img /tmp/upload

Para copiar os arquivos file1, file2 e file3 e o diretório img/ para o diretório /tmp/upload/:

$ cp -r file1 file2 fil3 img /tmp/upload

mv

Utilizamos o mv mover ou renomear arquivos.

Sintaxe:

$ mv [opções] [destino]

Opções:

-b: Cria um backup dos arquivos de destino, se eles forem sobrescritos;

-f: Força as operações, sem fazer perguntas caso seja necessário sobrescrever arquivos e outros;

-i: Modo interativo, pede confirmação para sobrescrever arquivos;

--target-directory = [diretório]: especifica o diretório de destino para os arquivos;

-u: Só move os arquivos novos. Se o arquivo que está sendo movido já estiver presente no diretório de destino, ele é ignorado;

-v: Mostra os detalhes do processo de movimentação.

Exemplos de uso:

Para mover o arquivo file1 para o diretório /home/davidson/doc/:

$ mv file1 /home/davidson/doc

Para mover o diretório /home/davidson/doc/ para /tmp/upload/:

$ mv /home/davidson/doc /tmp/upload

Para renomear o arquivo package.tar.gz para pacote.tar.gz:

$ mv package.tar.gz pacote.tar.gz

Para mover o arquivo file1 e o diretório img/ para o diretório /tmp/upload/:

$ mv file1 img /tmp/upload

cat

O comando cat concatena arquivos e imprime na saída padrão (exibe na tela). Em arquivos, usamos o cat para listar seu conteúdo na tela. Com o uso de direcionadores, podemos usá-lo para unir diferentes arquivos em um só, dentre outra funções.

Sintaxe:

$ cat [opções] [arquivo]

Opções:

-b: Numera as linhas, com exceção das linhas em branco;

-E: Mostra um "$"? (cifrão) para indicar fim de linha;

-n: Numera todas as linhas, incluindo as em branco;

-s: Não mostra mais do que uma linha em branco. Se houver duas ou mais linhas em branco consecutivas, elas são truncadas e apenas uma é mostrada;

-T: Substitui tabulações pelos caracteres "^I"?;

-v: Substitui os caracteres não imprimíveis por símbolos, exceto tabulações e final de linha.

Exemplos de uso:

$ cat [arquivo1 arquivo2 arquivo3 ... arquivoN] > [arquivo]

Isso pode ser usado em qualquer tipo de arquivo, inclusive arquivos binários. É prática comum utilizar isso para juntar arquivos de vídeo grandes, como filmes, que muitas vezes são divididos em várias partes.

Veja no exemplo abaixo, como unir as 2 partes do filme Matrix em um único arquivo:

$ cat the-matrix_part-1.mpeg the-matrix_part-2.mpeg > the-matrix.mpeg

Vale lembrar que tal procedimento não é corretamente executado com arquivos AVI.

adduser ou useradd

Comando utilizado para criar usuários, dependendo da distribuição ou usamos adduser ou useradd.

Sintaxe:

 # adduser [nomedousuario]

userdel

Comando utilizado para excluir usuários

Sintaxe:

 # userdel [nomedousuario]

groupadd

Comando utilizado para criar grupos de usuários

Sintaxe:

 # groupadd [nomedogrupo]

chown

Muda o dono do arquivo para o novo dono

Sintaxe:

 # chown [novodono] [arquivo]

find

Procura no diretório por arquivos ou subdiretórios

Sintaxe:

 # find [local] -name [oqueestaprocurando]

lynx

Abre o navegador de internet de mesmo nome

Sintaxe:

$ lynx [site]

ps

Mostra os processos em execução.

shutdown

Desliga ou reinicia o computador

$ shutdown -r now reinicia o computador
$ shutdown -h now desliga o computador

reboot

Reinicia o sistema imediatamente (pouco recomendável, preferível shutdown -r now);

halt

Desliga o computador

Editor de textos

O Linux possui diversos editores de textos, dentre eles:

- mcedit
- Vi
- Vim

Vou comentar a respeito do mais popular, o vi.

“vi" é a sigla para "**V**isual **I**nterface". A origem desse nome se deve ao seguinte fato: quando o vi foi criado (começo da década de 80), não era comum existirem editores de textos como nos dias de hoje. Naquela época, você digitava um texto mas não podia vê-lo. Em 1992, foi criado o vim (Vi IMitator), um clone fiel ao vi, porém com muitas outras funcionalidades, que só foram sendo adicionadas. Algum tempo depois, o vim passou a ser chamado de `Vi IMproved' (vi melhorado).

Modo de inserção e de comandos

Para identificar o modo (estado) do vi, basta visualizar o rodapé da tela.

Agora, vamos à prática. Para executar o vi, utilize:

$ vi => Abre o vim vazio, sem nenhum arquivo e exibe a tela de apresentação.
$ vi arquivo => Abre o arquivo de nome "arquivo".
$ vi arquivo + => Abre o arquivo de nome "arquivo", com o cursor no final do mesmo**$ vi arquivo +10** => Abre o arquivo de nome "arquivo", com o cursor na linha 10.

Ao executar o vi, ele inicia diretamente em modo de comando. Para comprovar, é só olhar na última linha (rodapé) e não vai haver nada lá. Isso quer dizer que você não conseguirá escrever nada, pode digitar a vontade que só vai ouvir beeps. Para começar a escrever, **pressione "i" em seu teclado**. O vi entra em modo de inserção, que você comprova (como falado anteriormente) pelo rodapé da tela, onde fica a seguinte marcação:

- - -- INSERT --

Suponha que você já digitou o bastante, e quer salvar, por segurança.

Pressione a tecla ESC para voltar em modo de comandos. E veja os comandos para salvar/sair:

:w => Salva o arquivo que está sendo editado no momento.
:q => Sai.
:wq => Salva e sai.
:x => Idem.
ZZ => Idem.
:w! => Salva forçado.
:q! => Sai forçado.
:wq! => Salva e sai forçado.

Permissões de arquivos

As permissões são atributos dos arquivos que especificarão se ele pode ser lido, executado e/ou escrito. Estas permissões vão definir o que um usuário pode fazer ou não.

Visualizando e entendendo as permissões

Para saber se um programa é executável ou não, execute um ls -l e veja no lado esquerdo se o arquivo tem um x nos seus argumentos, como no exemplo abaixo:

 # ls -l

drwxr-xr-x	2 root	root	1024	Dec 23 15:22	bin
drwxr-xr-x	2 root	root	1024	Dec 31 05:48	boot
drwxr-xr-x	2 root	root	1024	Dec 6 15:51	cdrom
drwxr-xr-x	3 root	root	8192	Mar 11 10:17	dev
drwxrwxr-x	2 root	root	1024	Feb 27 13:52	dosa
dr-xr-xr-x	11 root	root	2048	Mar 11 10:19	etc
drwxr-xr-x	11 root	root	2048	Feb 23 19:08	home
drwxr-xr-x	3 root	root	1024	Feb 23 19:13	lib
drwxr-xr-x	2 root	root	12288	Nov 2 11:25	lost+found
-rwxr--r--	1 root	root	57	Mar 10 03:44	backup
-rw-rw-r--	1 jefferson	users	2342	Mar 10 03:12	teste.txt
-rw-rw-rw-	1 camila	visits	23412	Mar 09 22:22	teste2.txt

No exemplo acima todos os arquivos tem como dono o ***root*** e como grupo-dono também o ***root***, com exceção do teste.txt que o dono é ***jefferson*** e o grupo é ***users*** e também teste2.doc, no qual ***camila*** é o dono e o grupo ***visits*** também é dono. Como você pode ver do lado esquerdo de cada arquivo/diretório existe um série de letras r, w, x ou d! Vamos ver o que representa cada uma delas:

drwxrwxrwx
0111222333

No caso acima, a primeira coluna significa (número 0) se o nome listado é um diretório ou não, caso não seja um diretório ele será exibido da seguinte maneira:

-rwxr--r-- 1 root root 57 Mar 10 03:44 backup
\--------> Não contém a letra 'd', não é diretório e sim arquivo

O exemplo abaixo mostra o que seria um diretório:

drwxr--r-- 1 root root 1 Mar 10 01:12 bin
\----------> Contém a letra 'd' na primeira coluna, é um diretório

Também há casos em que no lugar do d, aparecem outras letras que indicam outros tipos de arquivos. A letra l significa que é um link simbólico, as letras c e b correspondem a dispositivos (/dev).

Continuando, na segunda coluna (números 1 de acordo com o exemplo mais acima) temos as definições para o dono do arquivo, como mostra o exemplo:

```
-rwxr--r--   1 jefferson   users        1231 Mar 09 12:12 teste.txt
 ||\--------> O dono do arquivo (jefferson) pode executar o arquivo, x=executable!
 |\---------> O dono do arquivo (jefferson) pode gravar no arquivo, w=writable!
 \----------> O dono do arquivo (jefferson) pode ler o arquivo, r=readable!
```

Seguindo, na terceira coluna (números 2 de acordo com o exemplo mais acima) temos as definições para o grupo que é dono do arquivo, como mostra o exemplo:

```
-r--rwxr--  1 joao visits        212 Mar 01 12:42 exemplo.sxi
    ||\-----> O grupo dono do arquivo (visits) pode executar o arquivo!
    |\------> O grupo dono do arquivo (visits) pode gravar no arquivo!
    \-------> O grupo dono do arquivo (visits) pode ler o arquivo!
```

Finalmente, temos a quarta coluna (composto pelos números 3), essa coluna se refere às permissões para ***todos os outros*** usuários do sistema, sem ser os donos e grupos-donos dos mesmos, exemplo:

```
-r--r--rwx   1 maria visits       1231 Mar 03 12:42 exemplo2.doc
       ||\--> Todos os usuários (exceto maria e usuários do grupo visits)
       ||                  tem permissão para acessar o arquivo!
       |\---> Todos os usuários (exceto maria e usuários do grupo visits)
       |                   tem permissão para gravar no arquivo!
       \----> Todos os usuários (exceto maria e usuários do grupo visits)
                           tem permissão para ler o arquivo!
```

Quando nos referimos ao diretório ao invés de arquivos, o flag x (executável) diz se o diretório é ou não acessível, já que não podemos executar diretórios... Exemplo:

```
drwxr--r-- 1 root       root         2134 Mar 01 12:54 exemplo3
|||||  \----> Todos os usuários podem ler o interior do diretório, mas não
|||||         podem usar o comando 'cd' para entrar nele, pois não existe
|||||         o FLAG 'x' para a quarta coluna!
||||\-------> Usuários do grupo 'root' podem ler o interior do diretório,
||||         mas também não podem usar 'cd' para entrar no diretório!
|||\--------> O usuário 'root' pode usar 'cd' para entrar no diretório!
||\---------> O usuário 'root' pode gravar arquivos nesse diretório!
|\----------> O usuário 'root' pode ler o interior desse diretório!
\-----------> Indica que o nome listado é um diretório!
```

Mudando as permissões com o chmod

O comando chmod pode ser usado para mudar os flags rwx dos arquivos e/ou diretórios, a sintaxe básica do comando é:

chmod [ugoa]{-+}[rwx] <nome_do_arquivo_ou_diretório>

Se eu quero mudar a permissão para o dono do arquivo (***u=user***) poder ler e gravar (rw) no arquivo1.txt, faço o seguinte:

$ chmod u+rw arquivo1.txt

Caso você queira desfazer o comando, você faria: chmod u-rw arquivo1.txt. Como se vê, o + ou - define se as flags serão ativadas ou desativadas! Outros exemplos:

$ chmod a+r arquivo2.txt

Acima, todos os usuários (***a=all***) podem ler o arquivo2.txt. Outro exemplo:

$ chmod o+w arquivo3.txt

Neste caso outros usuários (***o=others***) sem ser o dono e o grupo dono do arquivo, podem gravar no arquivo3.txt. Outro exemplo:

$ chmod g+x netscape

Acima, o grupo-dono do arquivo (***g=group***) pode executar o arquivo netscape.

O comando chmod também pode ser usado com números, em vez das flags. Este método é chamado de ***octal***, veja o exemplo abaixo:

Número	Significado
0	Nenhuma permissão
1	Permissão para executar
2	Permissão para gravar
3	Permissão para gravar e executar
4	Permissão para ler
5	Permissão para ler e executar
6	Permissão para ler e gravar
7	Permissão para ler, gravar e executar

$ chmod 664 arquivo.txt

No exemplo o comando informou que o arquivo.txt pode ser lido e gravado pelo seu dono (número 6 na primeira coluna), informou que pode também ser lido e gravado pelos usuários que compõem o grupo-dono (número 6 na segunda coluna) e informou que pode ser lido por todos os outros usuários do sistema (número 4 na última coluna).

Mudando o dono dos arquivos

O comando chown é simples e pode ser usado para mudar o dono e o grupo dono de um arquivo/diretório. E é usado da seguinte maneira:

chown <usuario.grupo> <arquivo_ou_diretorio>

Como exemplo, vamos definir que um arquivo teste4.txt terá como dono ***jefferson*** e como grupo ***users***:

$ chown jefferson.users teste4.txt

Definindo IP Fixo

Digite no Shell:

1. **Passo – Editando o arquivo da rede (Endereçamento exemplo)**

```
#vi /etc/network/interfaces

auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 10.168.0.1
netmask 255.255.255.0
network 10.168.0.0
broadcast 10.168.0.255
gateway 10.168.0.100
```

2. **Passo – Iniciando a placa**

```
#ifdown eth0
#ifup eth0
```

Servidores Linux

Hoje os servidores Linux estão entre os mais poderosos do mundo.

Destaca-se por ter uma estrutura baseada em Unix, trazendo altos níveis de segurança e confiabilidade.

Guia básico de configuração de servidores

Servidor DHCP

Vem de *Dynamic Host Configuration Protocol*, é um protocolo usado para controlar informações das máquinas de uma rede de forma centralizada num único servidor, ou seja, é o servidor DHCP que controla que endereço IP, máscara de rede determinada máquina da rede terá. Segue o passo-a-passo:

1. **Passo – Instalar o pacote dhcp**

   #apt-get install isc-dhcp-server
   O pacote pode ter outro nome, como por exemplo, dhcp3-server

2. **Passo – Configurar endereçamento estático na placa de rede do servidor**

   #vi /etc/network/interfaces

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
        address 192.168.0.100
        netmask 255.255.255.0
        network 192.168.0.0
        broadcast 192.168.0.255
        gateway 192.168.0.1

#ifdown eth0
#ifup eth0
```

3. **Passo - Identificar a interface do servidor**

   #vi /etc/default/isc-dhcp-server

4. **Passo – Editar o arquivo dhcpd.conf**

   #vi /etc/dhcp/dhcpd.conf
   (Pode apagar o conteúdo e escrever o que segue abaixo)

```
default-lease-time 300;
max-lease-time 3600;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.0.255;
option routers 192.168.0.1;
option domain-name-servers 192.168.0.100 , 192.168.0.200;
option domain-name "dominio.com";

subnet 192.168.0.0 netmask 255.255.255.0 {
 range 192.168.0.140 192.168.0.180;
}
```

5. **Passo – testes e inicialização**

Verificar erros no arquivo
#dhcpd –d

6. **Passo (caso for necessário) -** Colocar na inicialização

   #apt-get install sysv-rc-conf
   #sysv-rc-conf

7. **Iniciar o serviço**

   #service isc-dhcp-server restart

8. **Passo – conexão ao servidor**

   “Subir” um cliente e “pegar” IP deste servidor

Explicando linha a linha:

default-lease-time 300;
Servidores DHCP fornecem endereços sob pedido por um tempo pré-configurado.No exemplo será fornecido o endereço IP por 300 segundos ou 5 minutos.

max-lease-time 3600;
Caso o cliente solicite um tempo maior, o tempo máximo permitido será de 3600 segundos (1 hora).

option subnet-mask 255.255.255.0;
Esta opção define a máscara de sub-rede a ser fornecida aos clientes.

option broadcast-address 255.255.255.255;
Esta opção define o endereço de envio para requisições de broadcast.

option routers 192.168.0.1;
O cliente, além do número IP, recebe também a informação do número do equipamento que é o gateway de sua rede.

option domain-name-servers 200.204.0.10, 200.204.0.138;
Esta opção lista os servidores de nomes (DNS) a serem utilizados para resolução de nomes, em meu caso, utilizei os DNSs da Telefônica.

option domain-name "exemplo.rede.br";
Esta máquina pertence ao domínio exemplo.rede.br, nosso exemplo.

```
subnet 192.168.0.0 netmask 255.255.255.0 {
  range 192.168.0.50 192.168.0.100;
}
```

Esta opção lista a sub-rede à qual o equipamento pertencerá e a máscara de rede a ser utilizada. Em seguida encontra-se a faixa de endereços IP que pode ser fornecida pelo servidor DHCP aos seus clientes. A linha indica que podem ser fornecidos endereços na faixa de 192.168.0.50 a 192.168.0.100.

Servidor Samba

Samba é uma ferramenta que simula um servidor Windows, permitindo que seja feito gerenciamento e compartilhamento de arquivos em uma rede Microsoft.

O samba também pode ser configurado como servidor de domínio.

Segue o passo-a-passo:

1. Passo – Instalação do pacote

#apt-get install samba

2. Passo - Observar as permissões das pastas

#chmod 777 <pasta>
Exemplo:
#chmod 777 comum
(que está em /home)

3. Passo - Editar o arquivo do samba

#vi /etc/samba/smb.conf
(pode apagar o arquivo existente e criar um novo)

```
[global]
server string = server samba
workgroup = workgroup
wins support = yes
security = user
username map = /etc/samba/smbpasswd

[comum]
path = /home/comum
```

valid users = %U

comment = este é um servidor de arquivos

read only = no

Importante: Após definir o nível de segurança, lembre-se:

- Se for Share, não haverá autenticação de usuários
- Se for user, para adicionar usuário:

#smbpasswd –a <nomeusuario>

4. Passo - Testar e executar o samba

#testparm

Servidor de Web

Uma das ferramentas usadas para configurar o servidor de WEB em Linux é o Apache. Segue o passo-a-passo:

1. Passo – Instalar pacote

#apt-get install apache2

2. Passo - Localizar o arquivo index.htm ou index.html

#cd /var/www/html

3. Passo – Edite o arquivo hosts

IPdaMaq	<site>	<dominiosite>	<site>
127.0.0.1	<site>	<dominiosite>	<site>

Exemplo

192.168.0.1	teste	www.teste.com.br	teste
127.0.0.1	teste	www.teste.com.br	teste

4. Passo – Iniciar o apache

#/etc/init.d/apache2 reload

5. **Passo – Teste no Browser**

   http://localhost
   ou
   http://127.0.0.1
   ou
   http://<ipdamaquina>

Configurando o Apache com vários domínios

1. Para configurar vários domínios em uma única máquina, você devera seguir o exemplo abaixo:
   Passo – Domínios para teste

   Vários domínios em uma única máquina.

   Exemplo dois endereços:

   www.teste1.com.br

   www.teste2.com.br

2. **Passo – Criar um site em cada domínio**

   Em /var/www/html crie dois diretórios teste1 e teste2
   Em cada diretório coloque um site.

3. **Passo – Adicionando o nome do domínio**

   Abra o arquivo */etc/hosts* e acrescente as seguintes linhas:

   127.0.0.1 teste1 www.teste1.com.br teste1
   127.0.0.1 teste2 www.teste2.com.br teste2

4. **Passo – Configurando os domínios**

   Em /etc/apache2 crie o arquivo *httpd.conf* e edite da seguinte forma:

   NameVirtualHost *:80

   <VirtualHost *:80>
   DocumentRoot "/var/www/html/teste1"
   ServerName www.teste1.com.br

</virtualhost>

<VirtualHost *:80>
DocumentRoot "/var/www/html/teste2"
ServerName www.teste2.com.br
</virtualhost>

Servidor de Internet (Firewall)

Um firewall é uma barreira inteligente entre duas redes, através do qual só passa tráfego autorizado.

Este tráfego é examinado pelo firewall em tempo real e a seleção é feita de acordo com a política de segurança estabelecida.

O iptables é o Firewall do Linux, ele é estável, eficiente e rápido, acompanha versões de Linux com o kernel igual ou superior ao 2.4.x.

Para versões anteriores existia o ipchains.

Sua configuração mais adequada é feita através de scripts, que são adicionados a inicialização do sistema.

Sintaxe

Iptables [-t tabela] [comando] [chain] [parâmetro] [ação]

Tabela

São os locais usados para armazenar as chains.

Quando não menciono a tabela, estou usando a **tabela padrão**, que é a **filter**.

- **Filter** - tabela de filtros de pacotes.
- **NAT (network address translation)** - as maquinas com IP frio serão mascaradas pelo IP quente existente na rede.
- **Mangle** - altera o conteúdo dos pacotes.

Comando

Os Comandos podem, por exemplo, adicionar ou excluir uma regra.

É representado sempre por letra maiúscula, tendo algumas exceções (-h).

- **-N** - cria uma user chain

- **-X** - Exclui uma regra por seu nome
- **-P** - muda a política default de uma chain
- **-L** - lista as regras de uma chain
- **-F** - apaga todas as regras de um chain
- **-Z** - limpa todos os contadores de bytes e pacote de uma chain
- **-A** - acrescenta uma regra a uma chain
- **-I** - insere regra numa posição da chain
- **-R** - troca posição de regra na chain
- **-D** - apaga regra de uma chain
- **-h** - Ajuda do comando
- **-C** - Verifica as regras básicas do firewall

Chain

Podemos especificar a situação do tratamento do pacote.

- **INPUT -** consultado para dados que chegam ao PC.
- **OUTPUT** - consultado para dados que saem do PC.
- **FORWARD** - consultado para dados que são redirecionados para outra máquina ou outra interface de rede.

Parâmetros

- **-p!** (protocolo) - define qual protocolo TCP/IP. (TCP,UDP e ICMP).
- **-s!** (origem) - define qual o endereço de origem
- **-d!** (destino) - define qual o endereço destino
- **-i!** (interface) - define o nome da interface da rede onde trafegarão os pacotes de entrada e saída do firewall. Utilizado em mascaramento com NAT
- **-j!** (ir para) - redireciona uma ação desde que as regras sejam similares.
- **-f!** (fragmentos) - trata datagramas fragmentados.

Ação

Indicam se os pacotes serão ou não aceitos.

- **ACCEPT** – aceita o pacote
- **DROP** – rejeita o pacote
- **REJECT** – rejeita o pacote e envia um aviso

Você pode utilizar o ambiente Shell para configurar o servidor iptables, mas no caso de reiniciar o servidor todas as configurações serão perdidas. Por isso utilizamos os scripts.

Configurando um Script Básico

Primeiro você deve criar um arquivo dentro de /etc/init.d

Dentro do arquivo digite:

```
# Ativando o roteamento

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

# Limpando as regras do firewall

iptables -F

# Bloqueando acessos suspeitos
iptables -A INPUT -d www.uol.com.br -j REJECT
iptables -A OUTPUT -d www.uol.com.br -j REJECT
iptables -A FORWARD -d www.uol.com.br -j REJECT
iptables -A INPUT -d www.jeffersoncosta.com.br -j REJECT
iptables -A OUTPUT -d www.jeffersoncosta.com.br -j REJECT
iptables -A FORWARD -d www.jeffersoncosta.com.br -j REJECT
```

Salve o arquivo e saia.

Agora transforme em executável:

#chmod +x <nomedoarquivo>

Transforme em “inicializavel”:

#update-rc.d <nomedoarquivo> defaults

Observação: Caso queira remover o arquivo da inicialização, digite:

#update-rc.d -f <nomedoarquivo> remove

Para finalizar, “Starte” o arquivo

#./<nomedoarquivo>

Servidor de Internet (Proxy)

Entendendo como o Squid lê as ACLs

O proxy do Linux é o Squid, ele é considerado o mais rápido e seguro do mundo.

A cada caractere inserido no *squid.conf*, lembre-se que o *Squid* lê as ACLs de cima para baixo e quando encontra alguma que se aplique, ele para. Parece meio complicado, mas funciona assim:

Vou criar três ACLs: acesso_total, acesso_restrito e bloqueado. Estes arquivos terão os seguintes conteúdos:

acesso_total - IPs dos clientes que terão acesso total à internet. Não passarão por nenhuma restrição do Squid.

acesso_restrito - IPs dos clientes que passarão pelo bloqueio de sites estabelecido na acl seguinte.

Bloqueado - Lista de palavras que o Squid bloqueará se forem encontradas na URL.

De posse destas três ACLs, vamos declará-las no squid.conf desta maneira:

acl acesso_total src "/etc/squid/acesso_total"
acl acesso_restrito src "/etc/squid/acesso_restrito"
acl bloqueado url_regex -i "/etc/squid/bloqueado"

OBS: A opção -i encontrada na linha que declara o bloqueio serve para NÃO fazer distinção entre maiúsculas e minúsculas.

Agora que as regras foram declaradas, vamos ativá-las dessa maneira:

http_access allow acesso_total
http_access deny bloqueado
http_access allow acesso_restrito
http_access deny all

A primeira regra que o Squid lê (http_access allow acesso_total) diz que será LIBERADO acesso a quem estiver com o IP cadastrado no arquivo "/etc/squid/acesso_total". Então, quem ele encontra aqui já é liberado e não passa mais pelas outras ACLs seguintes. Por isso o acesso é direto e total.

A segunda regra que ele encontra (http_access deny bloqueado) diz que será NEGADO o acesso às URLs que coincidirem com as palavras que estão no arquivo "/etc/squid/bloqueado". Se, por exemplo, neste arquivo tiver a palavra sexo, qualquer site que tenha esta palavra na sua URL não será acessado, como em www.uol.com.br/sexo, www.sexomais.com.br, etc. Mas atenção neste detalhe. O Squid vem lendo o arquivo de cima para baixo e só chegará a segunda regra quem não cair na primeira, ou seja quem não tiver o IP cadastrado no arquivo de acesso total.

A terceira regra que o Squid lê (http_access allow acesso_restrito) diz que será LIBERADO acesso a quem tiver com o IP cadastrado no arquivo "/etc/squid/acesso_restrito". Como na terceira regra só chega quem não caiu na regra anterior, o acesso pode ser liberado tranquilamente.

A quarta e última regra (http_access deny all) nega o acesso a qualquer IP de qualquer máscara (0.0.0.0/0.0.0.0), pois ela já vem declarada no início das ACLs (acl all src 0.0.0.0/0.0.0.0).

Se ele lê a primeira (acesso_total) e ninguém se aplica a ela, então passará para a segunda. Se nesta também ninguém se aplica, ele passará para a terceira. É óbvio concluir que se o cliente não está cadastrado no acesso_total, nem está no acesso_restrito, então ele não faz parte da rede e deve ser bloqueado. Só chegará a última quem não caiu em nenhuma das anteriores. Por isso, divida sua rede em quem pode ter acesso total e restrito e cadastre os clientes em seus respectivos arquivos.

Referências bibliográficas

- Infowester - http://www.infowester.com/

- Viva o Linux – http://www.vivaolinux.com.br

- Professor Jefferson Costa – Educação e tecnologia – http://www.jeffersoncosta.com.br]

- Guia do Hardware - http://www.guiadohardware.net/

- Guia Foca Linux - http://focalinux.cipsga.org.br/