4LINUX

OPEN SOFTWARE SPECIALISTS

Linux Fundamentals

Sumário

1

Introdução ao Linux

Sobre o material

Existem basicamente dois tipos de materiais de estudo: os materiais de referência e os tutoriais. O material de referência é orientado para a consulta por quem já conhece um assunto e precisa apenas relembrar ou necessita apenas de detalhes mais específicos sobre um tópico, como por exemplo, a sintaxe de um comando. Já o material do tipo tutorial visa ensinar um conceito ou assunto. Ele tenta ensinar a quem não sabe como fazer algo, como usar os recursos e as ferramentas disponíveis para atingir um determinado resultado.

Ambos são muito importantes e úteis tanto para o estudante quanto para o profissional. Ambos servem para consulta futura, pois enquanto o material de referência será acessado com mais frequência depois de acabada a etapa inicial de aprendizado, existem inúmeras maneiras diferentes de se apresentar um assunto, refletindo as variações existentes no mundo real. Então, mesmo quem já domina determinado assunto irá se beneficiar por estudar um novo tutorial de vez em quando. Fora o fato de que podemos "esquecer" determinadas coisas se ficarmos algum tempo sem utilizá-las, então talvez seja necessário voltar aos tutoriais para relembrar.

As apostilas deste curso são tutoriais. Não iremos apresentar todas as variações de sintaxe de um comando nem todas as diretivas de um arquivo de configuração. Isto é informação de referência, que é melhor atendida pelas suas respectivas documentações oficiais.

Acreditamos que o aluno não tem necessidade de "mais um" material de referência para Linux. Já existem materiais muito bons, disponíveis gratuitamente, mas acreditamos que podemos fazer a diferença com uma forma diferente de fazer tutoriais. Tentaremos seguir

uma abordagem de "aprender fazendo", privilegiando exemplos de exercícios completos, que o aluno pode executar imediatamente.

Esperamos que o aluno aprenda pela leitura e análise destes exemplos. Ainda assim, é importante complementar o material nestas apostilas com as referências apresentadas no nosso ambiente virtual de ensino.

Também não temos a pretensão de escrever o "melhor tutorial do mundo". Cada pessoa tem sua própria forma de ver o mundo e de aprender sobre os assuntos. Por isso indicaremos, no material complementar, alguns outros tutoriais que serão úteis durante ou depois do curso para solidificar o seu entendimento do Linux.

Mercado Linux

Nesta seção, você será introduzido no mundo Open Source, veremos a importância do Linux no mundo atual e quais são as principais profissões do mercado que utilizam Linux atualmente. Logo, vamos responder a perguntas como:

- Onde podemos encontrar o Linux?
- Por onde posso começar usando o Linux?
- Quais as principais profissões envolvendo Linux?

> "Se o Windows domina no Desktop... o Linux domina o mundo!" Steve Ranger – ZDnet

Onde podemos encontrar o Linux?

Atualmente, muitos nichos no mundo de tecnologia utilizam o Linux, começando pelo mercado de dispositivos móveis. Hoje o Android é o sistema operacional para dispositivos móveis mais utilizado no mundo. Não podemos deixar de falar sobre o mundo de segurança, testes de invasão ou técnicas forenses sem conhecer Linux, pois muitas das ferramentas utilizadas nesse ramo foram desenvolvidas para as distribuições Linux.

Hoje a computação em nuvem vem transformando muitos dos recursos tecnológicos das empresas em serviços prestados por provedores, nesses casos podendo fornecer, por exemplo, infraestrutura sob demanda e a maior parte dos recursos que estão hoje na computação em nuvem são executados em ambientes Linux. Grandes provedores de serviços em nuvem, como Amazon Web Services (AWS), Google Cloud Platform (GCP), Oracle Cloud (OCI) e Digital Ocean oferecem diferentes distribuições Linux e até mesmo a Microsoft oferece máquinas virtuais baseadas em Linux em sua nuvem, chamada Azure.

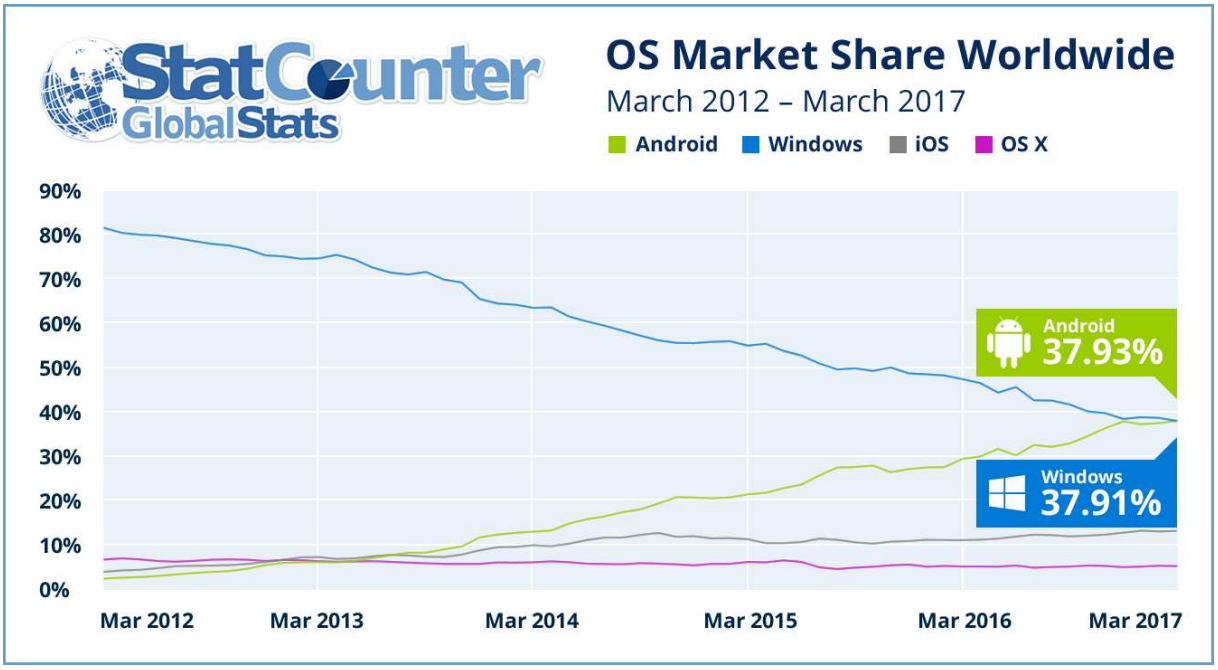
O mundo da Big Data também vem crescendo e atualmente as ferramentas para integração

e tratamento de dados são executados em Linux.

A DevOps é uma cultura aplicada para se obter principalmente a união entre os times de desenvolvimento (dev) e operações (ops) e assim surgiram diversas operações e deveres de ambas as áreas, muito disso funcionando dentro de ambientes Linux.

Market Share – Sistemas Operacionais pelo mundo

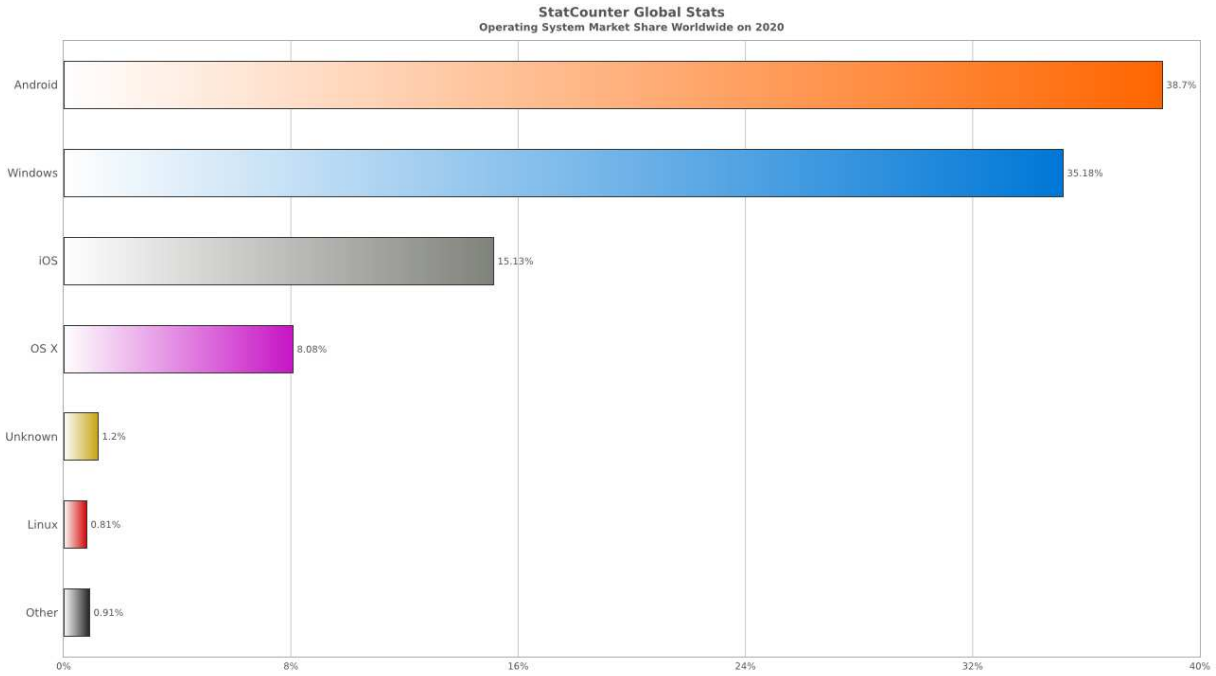
Em Março de 2017, a StatCounter anunciou o novo sistema operacional mais utilizado no mundo (levando em consideração apenas sistemas conectados à Internet).



O Android é um sistema operacional baseado no kernel Linux, licenciado pelo GPLv2.

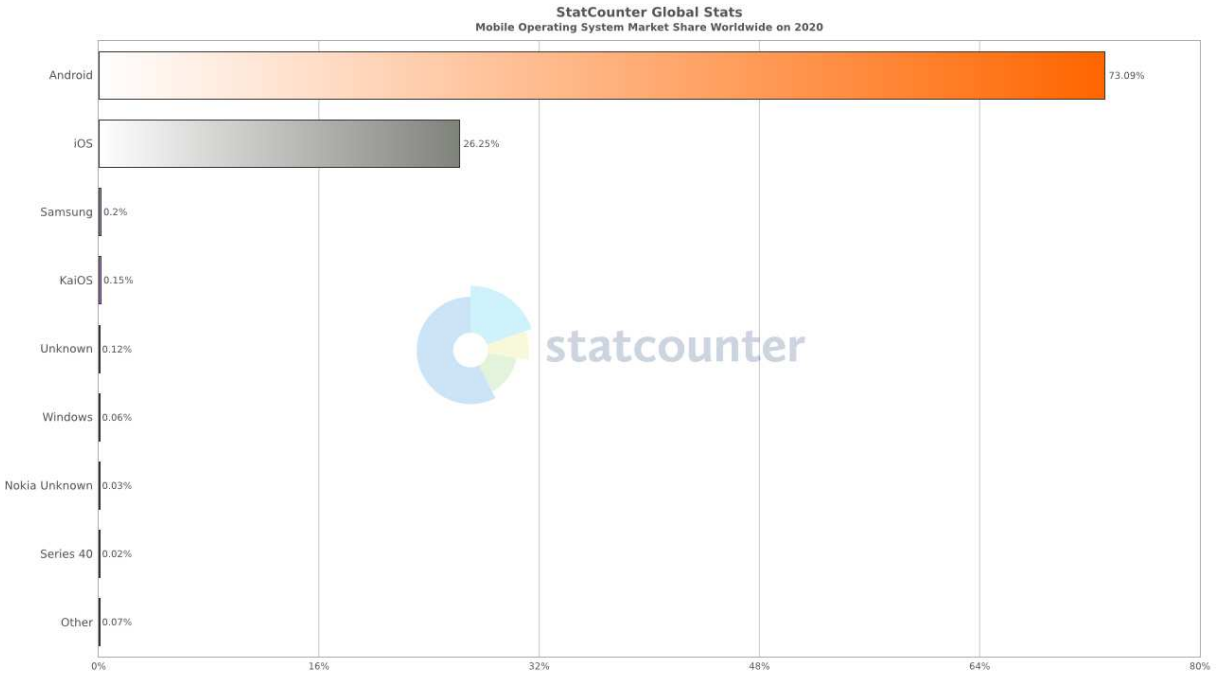
> "Este é um marco na história da tecnologia e o fim de uma era", comentou Aodhan Cullen, CEO da StatCounter. "Isso marca o fim da liderança mundial da Microsoft no mercado de sistemas operacionais, que detém desde os anos 1980. Também representa um grande avanço para o Android, que detinha apenas 2,4% do uso global da Internet há apenas cinco anos."

Atualmente, o Android é o sistema operacional móvel mais utilizado no mundo (considerando apenas sistemas conectados à Internet). Logo abaixo, temos um gráfico referente ao ano de 2020:



Dispositivos móveis

Esse grande domínio do sistema operacional Android deve-se aos dispositivos móveis como os smartphones. Abaixo temos um gráfico referente ao ano de 2020 para esses dispositivos:



Conheça mais estatísticas em: https://gs.statcounter.com/.

Desktop Linux

Atualmente, o Linux ocupa cerca de 2% dos desktops. Mesmo não dominando os computadores domésticos, ainda são desenvolvidas diversas distribuições para a utilização do usuário comum, muitas delas são até mesmo mais simples do que o próprio sistema de janelas da Microsoft.

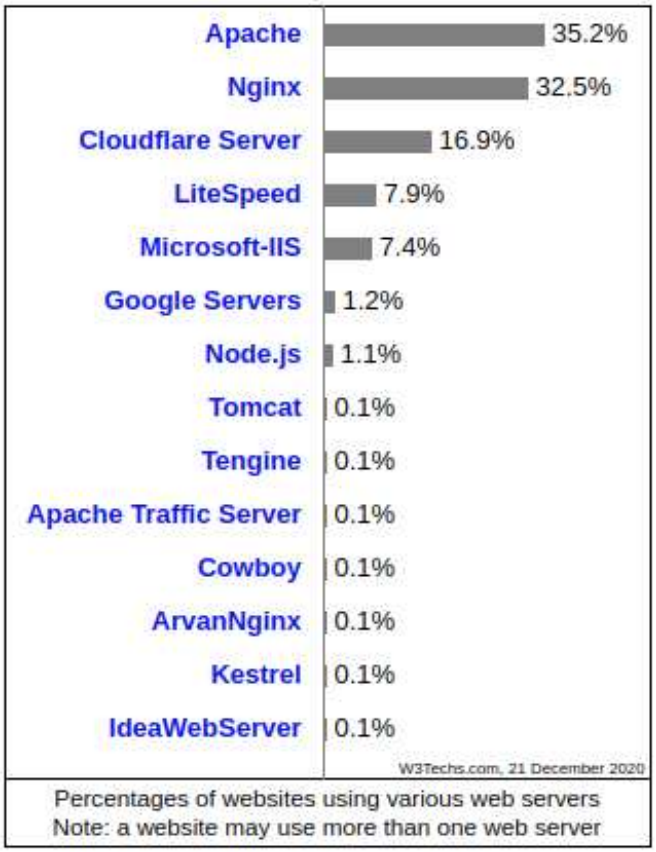
Ainda que as estações de trabalho não sejam muito utilizadas, a comunidade vem desenvolvendo interfaces cada vez mais amigáveis. A seguir temos uma breve listagem:

- Elementary OS
- MX Linux
- Linux Mint
- Deepin
- Manjaro OS
- Pop!_OS
- Ubuntu
- Fedora

Conheça as distribuições mais utilizadas em: https://distrowatch.com.

Servidores web

Em 2020, a W3Techs fez o relatório de Market Share dos servidores web mais utilizados baseado nos sistemas operacionais:

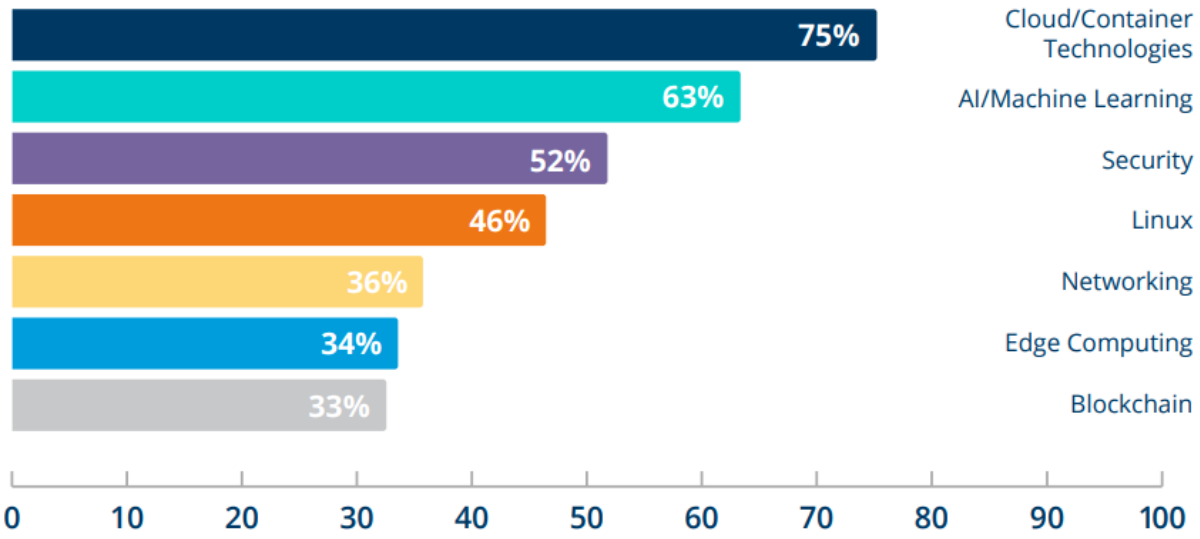


Profissões no mundo OpenSource

Alguns dados sobre o relatório de vagas de trabalho no mercado do código aberto em 2020 gerado pela Linux Foundation e pela edX:

- Cerca de 93% dos gerentes de contratação relatam dificuldade em encontrar talentos na área de código aberto e 63% dizem que suas organizações começaram a apoiar projetos de código aberto com contribuições sob a forma de código ou outros recursos pelo motivo explícito de recrutar indivíduos com essas habilidades de software, um salto significativo dos 48% conhecidos em 2018. DevOps também se tornou a principal função que os gerentes de contratação estão procurando preencher (65% estão procurando contratar talentos DevOps), movendo a demanda por desenvolvedores para o segundo lugar.
- Pela primeira vez na história deste relatório, 74% dos empregadores agora estão se oferecendo para pagar pelas certificações de seus funcionários.
- As certificações crescem em importância: 52% dos gerentes de contratação são mais propensos a contratar alguém com uma certificação, contra 47% há dois anos.
- A tecnologia de nuvem está aquecida: em termos de domínio de conhecimento, os gerentes de contratação relatam que conhecimentos em tecnologias de nuvem aberta tem o impacto mais significativo, e 70% deles está mais propenso a contratar um profissional com essas habilidades, contra 66% em 2018.

Tecnologias da mais alta importância para profissionais de código aberto em 2020:



Alguns dos muitos cargos que estão em alta no mundo Open Source:

- Analista DevOps
- Analista SRE
- Administrador de Servidores
- Desenvolvedor de Software
- Engenheiro de Software
- Analista de Banco de Dados
- Engenheiro de Dados
- Cientista de Dados
- Engenheiro de Machine Learning
- Analista de Segurança
- Gerente de TI
- Diretor Chefe de Tecnologia

2

Certificações Linux

> "A premissa de qualquer certificação é que ela ajude um gerente de contratações a organizar a pilha de currículos". – Randy Russel, Diretor de contratações e certificação da Red Hat.

Centros certificadores

Existem diversas carreiras e centros de formação para certificar profissionais em Linux. Atualmente os mais buscados são: LPI (*Linux Professional Institute*), RedHat e Linux Foundation, que vem crescendo com o tempo. Até mesmo a Microsoft fechou parceria com a Linux Foundation no mesmo ano em que disponibilizou o SQLServer para plataformas Linux.

Além dos centros citados acima, também temos outros como OpenSuse, CompTIA+ e a própria Exin que disponibilizou uma excelente certificação de DevOps.

Linux Professional Institute

Alguns pontos importantes a serem observados sobre o LPI:

- É uma organização sem fins lucrativos;
- Certificação independente de fornecedores;
- O conteúdo dos exames são baseados no padrão LSB (*Linux Standard Base*);
- Nas palavras de Jon Maddog Hall: **"É a maior certificação mundial em Linux"**;
- As provas são aplicadas pelo LPI Brasil ou pelo centro de treinamento VUE;
- As provas são aplicadas em português apenas nos níveis 1 e 2;
- O tempo de validade é de 5 anos;
- Para obter as certificações de nível 1 e 2 são realizadas duas provas;
- Para obter a certificação nível 3 é realizada apenas uma prova.

Red Hat Certificate

- Dentre as versões Linux comerciais, a Red Hat tem o mais popular programa de certificação do mercado;
- As provas são práticas;
- O conteúdo dos exames são baseados sempre na versão mais recente do Linux Red Hat Enterprise.

O programa de certificação da RedHat possui as seguintes certificações:

- Red Hat Certied System Administrator
- Red Hat Certied Engineer
- Red Hat Certied Architect

Linux Foundation

- Metodologia de aplicação de provas práticas;
- É possível escolher sua distribuição de preferência;
- Não depende de escolas autorizadas para execução das provas;
- Foi o primeiro centro certificador a fechar parceria com a Microsoft.

Sobre o programa de certificação da Linux Foundation, temos as seguintes certificações: * SysAdmin Linux Professional * Engineer Linux Professional

E por onde eu começo?

- Cursos
- Estudos autodidatas

- Simulados
- Fóruns
- Listas de discussão

3

História do Linux

Muitos desconhecem a real história e origem do sistema operacional comumente chamado de Linux quando tem seu primeiro contato com o sistema. Antes de explicarmos o que é de fato o Linux, é preciso conhecer a origem e função do projeto GNU, assim como saber observar e manipular o sistema operacional de uma forma mais técnica e detalhada.

Um sistema operacional é formado por um conjunto de funções que, trabalhando em perfeita harmonia, manipulam o hardware através de uma interface de comunicação e assim oferecem todos os recursos úteis para o seu gerenciamento. Conhecer detalhadamente as camadas que envolvem um sistema operacional é um diferencial no aprendizado.

O projeto GNU surgiu em meados de 1983, tendo como seu idealizador Richard Stallman, que tinha como objetivo criar um sistema operacional que fosse completamente livre e seguisse os padrões do Unix, que na época, cobrava licenças de alto custo. Cada ferramenta, função etc., deveria seguir apenas esta ideologia: ser livre.

Nesse contexto, quando fazemos referência à palavra livre, isso significa que é no sentido de liberdade ao código-fonte e forma de usar o software. Essa necessidade surgiu pelo fato de que praticamente todos os softwares da época estavam tomando o caminho oposto, ou seja, do software proprietário. Nesse tipo de software, o usuário geralmente detém apenas a permissão de uso sobre o software, sem nenhum acesso ao código-fonte.

O sistema GNU estava sendo desenvolvido por uma comunidade imensa, que seguiu os passos e as ideologias de Richard Stallman. Porém, com o tempo, o projeto foi ganhando tal consistência e volume que necessitou urgentemente de um órgão para gerenciar todo o seu desenvolvimento. Nasceu, então, a *Free Software Foundation* (FSF).

Todas as ferramentas desse sistema operacional estavam sendo preparadas do zero, como o editor de texto (Emacs), o ambiente gráfico (servidor X), compiladores (GCC, G++) e também um kernel (GNU/Hurd). Assim, um novo sistema operacional livre surgia, mas com um detalhe muito importante: o desenvolvimento desse sistema livre deveria seguir os padrões de sistemas operacionais Unix, que é, sem dúvida, um padrão muito respeitado e conhecido por seu alto grau de segurança e confiabilidade.

Certamente, muitos devem se perguntar o que houve com o sistema operacional GNU, que não existe oficialmente. Para se ter um sistema operacional completo, é preciso ter todas as ferramentas e funções em perfeito funcionamento, como no desenvolvimento de um carro, onde cada peça terá sua função essencial e específica. Um dos grandes motivos do sistema operacional GNU não ter sido lançado foi pelo formato do desenvolvimento do seu kernel, o GNU/Hurd, que seguiu um modelo complexo, não se adequando ou obtendo a estabilidade necessária para assumir a posição de um kernel maduro e confiável.

Eis que, no ano de 1991, o finlandês Linus Torvalds criou um kernel denominado Linux, o qual pôde ser unido aos programas GNU. Dessa junção, nasceu o sistema GNU/Linux.

Costumamos nos referir aos sistemas GNU/Linux apenas como sistemas Linux, denominação que não faz justiça aos desenvolvedores de software e ferramentas GNU, afinal, estamos nos referindo apenas ao nome do kernel criado por Linus.

Os sistemas operacionais que muitos usam atualmente, através das inúmeras distribuições existentes e muito conhecidas, têm a contribuição das diversas ferramentas e utilitários desenvolvidos pelo projeto GNU no decorrer dos anos.

Graças ao projeto GNU, o software livre se tornou uma ideologia respeitada e seguida mundialmente e, principalmente, defendida juridicamente, tornando-se o grande motivo para que pessoas em todo o mundo possam trocar informações e aprimorar sempre as tecnologias de software desenvolvidas. Muitos outros projetos ainda são mantidos pelo projeto GNU, como o Gnash (Flash Player livre), o Coreboot (BIOS em formato de software livre), o cliente VOIP Skype com base em software livre etc.

Mais informações do projeto GNU podem ser encontradas em www.gnu.org.

Curiosidade: GNU é um acrônimo recursivo para a frase *Gnu Is Not Unix* (Gnu não é Unix), fazendo uma grande oposição ao formato proprietário que a maioria dos sistemas Unix seguia.

A origem do Linux

Depois de uma breve introdução sobre o projeto GNU, que tinha a intenção de criar um sistema operacional livre que seguisse os padrões do Unix, é hora de entender e conhecer, de uma forma mais técnica, o que é de fato o Linux, e principalmente sua origem.

Primeiramente, é importante entender que, tecnicamente falando, o Linux não é um sistema operacional, mas sim um kernel, ou a grosso modo, o centro de execução do seu sistema operacional. Cada sistema operacional, independentemente da plataforma, possui o seu próprio kernel, com suas características peculiares.

A história do Linux é um tanto aventureira, pois surgiu pelas mãos de um jovem estudante Finlandês chamado Linus Benedict Torvalds, que estudava Ciências da Computação na Universidade de Helsinki.

Linus Torvalds, como é mais conhecido, começou o desenvolvimento do Linux a partir de um projeto pessoal. Teve como base um sistema operacional Unix-like chamado Minix, que foi desenvolvido por Andrew Tanenbaum em meados de 1987. O Minix era um sistema operacional básico, porém seguia os padrões do Unix e era muito utilizado no meio acadêmico para auxiliar no ensino de sistemas operacionais. Linus Torvalds iniciou seu projeto a partir do Minix e pretendia criar uma versão do Minix melhor do que o original.

> Unix-like é um termo utilizado para referenciar o padrão de sistemas operacionais Unix, onde temos um padrão robusto, seguro e estável. Basicamente, um sistema Unix-like tem o suporte fundamental aos recursos de multiusuário e multitarefa.

Foi a partir dessa ideia que Linus Torvalds começou o desenvolvimento do seu projeto, conseguindo, na versão 0.01, melhorar o kernel do Minix, acrescentando suporte a recursos que estavam indisponíveis na versão anterior. Deu-se a esse sistema o nome Linux, que é uma junção de Linus com Unix, embora o próprio Linus tenha desgostado dessa ideia inicialmente, batizando o sistema de Freax. Ari Lemmke, que trabalhava com Linus Torvalds, deu o nome Linux ao repositório dos arquivos do sistema Freax, sem consultar Linus que posteriormente acabou aceitando dar este nome ao novo sistema. Inicialmente, o desenvolvimento do kernel Linux foi simplesmente um passatempo para Linus Torvalds.

O kernel Linux é um grande exemplo de software livre licenciado utilizando-se a GPL. A iniciativa ganhou milhares de adoradores e conta atualmente com muitos desenvolvedores que trabalham de forma minuciosa, adicionando diariamente novas funções e corrigindo bugs, através da criação de correções (patches). Em sua primeira versão, o kernel Linux já conseguia trabalhar em conjunto com algumas ferramentas desenvolvidas pelo projeto GNU, como o compilador GCC e o interpretador de comandos bash. A junção das milhares de ferramentas desenvolvidas pelo projeto GNU com o kernel Linux forma todas as camadas de preenchimento para que seja possível um sistema operacional completo. Mais informações sobre o kernel Linux podem ser encontradas em www.kernel.org.

4

Licenças Open Source

Free Software Foundation

A *Free Software Foundation* surgiu em 1985, com o propósito de organizar o projeto GNU em um formato técnico e jurídico. A FSF é uma organização sem fins lucrativos, que se mantém com a doação de empresas que apoiam a mesma causa: defender o software livre. Hoje, o foco principal da FSF é manter a GPL (*General Public License*), que é a licença utilizada para defender juridicamente um software livre, assim como disseminar a ideologia de software livre pelo mundo. Mais informações sobre a FSF podem ser encontradas em www.fsf.org.

Software livre

Até o momento, o termo software livre, que foi idealizado por Richard Stallman e defendido pela Free Software Foundation através da GPL, foi muito citado. Antes de entrar nos detalhes do termo software livre, é preciso entender o que é essa ideia que moveu milhões de pessoas por todo o mundo.

Primeiramente, é preciso estar atento à tradução de free software, pois a palavra free pode ter sentidos diferentes: livre e grátis. Quando nos referimos ao software livre, estamos nos referindo à sua liberdade e não ao preço, ou seja, o usuário terá a liberdade de estudar, executar, alterar, melhorar, copiar e até mesmo distribuir um software livre.

Essa ideia toda foi minuciosamente detalhada em quatro tipos de liberdades, conforme veremos a seguir:

Liberdade número 0: é a liberdade de executar o software para qualquer propósito ou finalidade;

Liberdade número 1: é estudar o funcionamento do software, podendo fazer alterações e adaptá-las conforme a necessidade. Um pré-requisito para atingir essa liberdade é ter o acesso total ao código-fonte;

Liberdade número 2: o usuário tem a liberdade de redistribuir cópias do software para quem quiser;

Liberdade número 3: é a liberdade para modificar o programa e, então, disponibilizar as melhorias para o público, de forma que toda a comunidade possa se beneficiar disso. O acesso ao código-fonte é uma pré-condição para isso.

A junção dessas quatro liberdades é a essência do software livre, no qual a ideia principal é sempre prorrogar a liberdade de um software, de forma que um usuário jamais possa se apoderar de um software livre e torná-lo algo como um software proprietário. Para tal função, existe a GPL.

GNU GPL (General Public License)

A GPL (*General Public License*) teve sua primeira versão (GPLv1) lançada em janeiro de 1989. Ela foi rapidamente substituída devido a correção de vulnerabilidades pela GPL versão 2 (GPLv2), lançada em junho de 1991. Atualmente, também temos a GPL na versão 3 (GPLv3), que foi lançada oficialmente em junho de 2007 e permanece como referência até hoje.

A GPL, que é uma importante ferramenta jurídica criada pela *Free Software Foundation*, tinha inicialmente, como principal função, defender a liberdade dos softwares desenvolvidos pelo projeto GNU. Posteriormente, foi utilizada e empregada por centenas de outros softwares envolvidos em projetos de terceiros e particulares por todo o mundo.

A *Licença Pública Geral* segue, desde a sua primeira versão, as quatro liberdades citadas anteriormente, sendo também sua intenção poder defender o software livre de forma prorrogada. O documento oficial da GPL, válido oficialmente como ferramenta de proteção ao software livre, existe somente em inglês.

Outras licenças também foram criadas com a finalidade de manter a defesa da liberdade do código aberto, do conhecimento e da informação, como foi iniciado pela GPL. Algumas delas estão na tabela a seguir:

Licença	Site
* GFDL (GNU Free Documentation License)	www.gnu.org/copyleft/fdl.html
* OPL (Open Publication License)	www.opencontent.org/openpub/

Licença	Site
* CC (Creative Commons)	creativecommons.org/about/licenses
* BSD (Berkley Software Distribution)	www.freebsd.org/copyright/license.html
* SPL (Sun Public License)	java.sun.com/spl.html

Para mais informações sobre as versões da GPL, acessar http://www.gnu.org/licenses/licenses.html.

5

Evolução do Linux: distribuições

Com a junção das ferramentas GNU e o kernel Linux, temos tecnicamente um sistema operacional GNU/Linux. Em suas primeiras versões, instalar um sistema GNU/Linux era extremamente complexo, pois era necessário praticamente instalar e compilar manualmente quase todas as ferramentas, incluindo o próprio kernel. Ou seja, o GNU/Linux era geralmente utilizado por usuários avançados.

Foi a partir dessa dificuldade que surgiram as primeiras distribuições GNU/Linux – também chamadas de distros –, que tinham a finalidade de facilitar a instalação do sistema, disponibilizando todas as ferramentas necessárias, com um kernel Linux e um instalador para automatizar o processo de instalação. Tudo isso era disponibilizado através de um CD, disquete ou outro tipo de mídia.

Uma distribuição Linux é um pacote que consiste em um kernel Linux, mais uma seleção de aplicativos mantidos por uma empresa ou comunidade de usuários. O objetivo de uma distribuição é otimizar o kernel e os aplicativos que são executados no sistema operacional para um determinado tipo de uso ou grupo de usuários. As distribuições frequentemente incluem ferramentas próprias para a instalação de software e administração do sistema. Por essa razão, certas distribuições são usadas principalmente em ambientes desktop, por serem mais fáceis de usar, enquanto outras são mais comumente instaladas em servidores para usar os recursos disponíveis da maneira mais eficiente possível.

Hoje existe uma quantidade imensa de distribuições disponíveis para instalação, com praticamente todos os níveis de dificuldade e usos específicos.

Em 1993, Patrick Volkerding reuniu o kernel Linux e diversos aplicativos, dando origem à distribuição denominada Slackware, tida como a primeira a ser disponibilizada de forma pública,

em CDs já pré-compilados e prontos pra instalação por qualquer usuário em qualquer tipo de computador.

A maioria das distribuições existentes hoje em dia são ramificações da Slackware e de outras três distribuições: Debian, Red Hat e SUSE. As mais antigas e tradicionais são Slackware e Debian, que surgiram em meados de 1993.

No site DistroWatch.Com (distrowatch.com), é possível conferir a lista de distribuições existentes, bem como notícias e últimas atualizações de cada uma delas. O site também disponibiliza links para aquisição das distribuições.

Distribuições livres e corporativas

A maioria das distribuições é mantida por comunidades de colaboradores localizados em todo o mundo e por corporações. Dada essa característica, as distribuições podem ser divididas em duas categorias: livres e corporativas. O primeiro tipo é disponibilizado por comunidades de colaboradores que não visam absolutamente nenhum lucro sobre suas distribuições, como é o caso do Slackware, Debian, Knoppix, Gentoo e CentOS.

Dentro das distribuições livres, subdividem-se ainda as convencionais, tradicionalmente distribuídas em algum tipo de mídia para instalação em disco rígido, e as distribuições Live, cuja execução se dá a partir da própria mídia, sem instalação no HD e, portanto, sem risco ao sistema operacional original da máquina no qual está sendo executada. Quase todos os componentes do sistema já vêm previamente configurados numa distribuição Live, o que a torna conveniente e fácil de usar. A Knoppix é um exemplo bem conhecido de distribuição Live, e dele surgiram o Kurumin e o Kalango, já traduzidos para o português e adaptados à nossa realidade.

Já as distribuições corporativas, como Ubuntu, Suse, Red Hat e Mandriva, são administradas por empresas que cobram pelo suporte prestado. É importante notar que, embora um custo esteja envolvido, isso não contradiz os princípios da licença GPL, pois a liberdade de software é mantida. Qualquer usuário tem acesso ao código-fonte das distribuições corporativas, de forma que, o que é vendido não é o produto, mas uma prestação de serviço de assistência e suporte caso estes se façam necessários.

Distribuições from scratch e derivadas

As distribuições também podem ser categorizadas de acordo com a maneira como são desenvolvidas. Assim, são classificadas como from scratch (do zero) ou derivadas de alguma outra distribuição.

A diferença básica entre os dois tipos é que as distribuições from scratch não derivam de nenhuma já existente, sendo todas as suas características desenvolvidas específica e exclusivamente para elas, como é o caso da Debian, Red Hat, Slackware e Gentoo, enquanto as derivadas baseiam-se em alguma distribuição anterior, recebendo ajustes e visando um objetivo funcional definido. Exemplos desse tipo são os mencionados Kubuntu e Kurumin, além de Ubuntu, DreamLinux, BrDesktop e Slax.

Muitas distribuições independentes foram lançadas ao longo dos anos. Algumas delas se baseiam em Red Hat ou Ubuntu, outras são projetadas para aprimorar uma propriedade específica de um sistema ou hardware. Existem distribuições construídas com funcionalidades específicas, como o QubesOS, um ambiente de desktop extremamente seguro, ou o Kali Linux, que oferece um ambiente para explorar vulnerabilidades de software e é usado principalmente para testes de invasão. Recentemente, diversas distribuições Linux minúsculas foram projetadas para serem executadas especificamente em containers Linux, como o Docker. Existem também distribuições construídas especificamente para componentes de sistemas embarcados e até mesmo para dispositivos inteligentes.

Red Hat Enterprise Linux

A distribuição Red Hat Linux, criada em 1993, é pioneira em distribuições GNU/Linux corporativas. Tem grande aceitação por parte das empresas pelo fato de oferecer suporte técnico e grande compatibilidade com as tecnologias mais utilizadas, tendo conquistado no mercado corporativo o posto de uma das maiores produtoras de soluções open source do mercado. Sua interface gráfica padrão é a GNOME e utiliza o sistema de pacotes RPM (*RedHat Package Manager*) para a instalação de aplicativos.

Derivada da Red Hat Linux, a Red Hat Enterprise Linux é uma versão corporativa de distribuição original. Outro projeto relacionado é o Fedora Project, projeto patrocinado pela Red Hat, cuja proposta é ser uma distribuição para a comunidade. O CentOS é uma derivação idêntica ao Red Hat Enterprise Linux até sua versão 8. Veremos mais detalhes sobre isso em tópico posterior.

Vale ressaltar que a Red Hat possui algumas das mais conceituadas certificações Linux, focadas exclusivamente em seus produtos.

Para mais informações sobre essa distribuição, acessar https://www.redhat.com.

CentOS

O CentOS, abreviação de *Community Enterprise Operating System*, é uma distribuição Linux de classe corporativa derivada de códigos fonte gratuitamente distribuídos pela Red Hat Enterprise Linux e mantida pelo CentOS Project. Esta distribuição sempre foi uma alternativa para empresas que não desejam pagar por uma distribuição fechada, mas esse cenário pode mudar com o anúncio da desativação do projeto CentOS em 2021 e a mudança de foco para o projeto CentOS Stream, onde teremos uma distribuição para mostrar o que está por vir

no Red Hat Linux. A numeração das versões sempre foi baseada na numeração do Red Hat Enterprise Linux. Por exemplo, o CentOS 7 é baseado no Red Hat Enterprise Linux 7, mas com a chegada do CentOS Upstream isso mudará, pois será uma distribuição do tipo *rolling release* (lançamento contínuo) sem lançamento de versões finais.

Embora o CentOS seja bem visado para seu uso em servidores devido a sua estabilidade e robustez, ele não é um sistema exclusivo de servidores. Tem essa fama devido ao seu foco em recursos estáveis (semelhantes ao que o Debian GNU/Linux faz) e as aplicações são focadas em estações de trabalho e redes por padrão. Ele pode ser usado para uso comum, porém, é necessário incluir nele os repositórios adicionais que não vem ativos por padrão. Os mais comuns são: RPMFusion e EPEL.

A partir do CentOS 5, cada versão é suportada por 10 anos (por meio de atualizações de segurança, assim como as versões anteriores eram suportadas por 7 anos). Uma nova versão do CentOS é lançada aproximadamente a cada 2 anos e cada versão do CentOS é atualizada regularmente aproximadamente a cada 6 meses, para oferecer suporte a hardware mais recente. Isso resulta em um ambiente seguro, de baixa manutenção, confiável, previsível e reproduzível. Como o RHEL exige que os usuários comprem uma licença, alguns preferem optar pelo CentOS, que fornece atualizações gratuitamente. Uma das desvantagens é que os lançamentos do CentOS sempre demoram um pouco mais do que os lançamentos do RHEL para serem disponibilizados enquanto a comunidade testa os pacotes e faz alterações de marca e similares.

> Este ciclo está em processo de mudança, devido ao foco no novo projeto CentOS Stream.

Fonte: https://wiki.centos.org/About

> O CentOS Stream servirá como o ramo *upstream* (desenvolvimento) do Red Hat Enterprise Linux.

> O CentOS 8 terá seu suporte encerrado em 31 de Dezembro de 2021.

> O CentOS 7 terá seu suporte encerrado em 30 de Junho de 2024, mantendo assim o ciclo de suporte RHEL 7.

A Red hat orienta aos usuários do CentOS 8 a realizar a mudança para o Centos Upstream quando o suporte acabar:

> *"Quando o CentOS Linux 8 (a reconstrução do RHEL8) terminar, sua melhor opção será migrar para o CentOS Stream 8, que é um pequeno delta do CentOS Linux 8 e tem atualizações regulares como as versões tradicionais do CentOS Linux. Se você estiver usando o CentOS Linux 8 em um ambiente de produção e estiver preocupado com o fato de o CentOS Stream não atender às suas necessidades, recomendamos que você entre em contato com a Red Hat sobre as opções disponíveis."*

Para mais informações sobre essa distribuição, acessar `www.centos.org`.

Oracle Linux

A Oracle Linux é uma distribuição Linux empacotada e distribuída gratuitamente pela Oracle, disponível parcialmente sob a GNU General Public License desde o final de 2006. É compilada a partir do código-fonte do Red Hat Enterprise Linux (RHEL), substituindo a marca Red Hat pela Oracle.

Oferece acesso a algumas das inovações mais avançadas do Linux, como Ksplice (extensão do Kernel Linux que permite que patches de segurança sejam aplicados a um kernel em execução sem a necessidade de reinicializações), e DTrace (estrutura de rastreamento dinâmico abrangente criada originalmente pela Sun Microsystems para solucionar problemas de kernel e de aplicativo em sistemas de produção em tempo real).

Para mais informações sobre essa distribuição, acessar www.oracle.com/linux.

Slackware

Criada por Patrick Volkerding, a distribuição livre Slackware Linux foi a primeira a ser distribuída em CD e é um sistema Unix-like multitarefa completo de 32-bits. Utiliza o sistema de pacotes `tgz`, orientado por menus, e sua interface padrão é a KDE. É compatível com 486 sistemas, incluindo os servidores x86 mais modernos, possui extensa documentação online e um programa de instalação fácil de usar.

A instalação completa da Slackware proporciona ao usuário o Sistema X, os ambientes de desenvolvimentos C/C++, o Perl, um servidor de notícias, um servidor de e-mail, um servidor web e um servidor FTP. Possui ainda o *GNU Image Manipulation Program*, o navegador Mozilla Firefox, utilitários de rede, além de muitos outros programas.

Para mais informações sobre essa distribuição, acessar www.slackware.com.

Debian GNU/Linux

Com interface padrão Xfce, a distribuição livre Debian é atualmente uma das maiores distribuições e uma das principais bases para outras distribuições derivadas. Faz uso do sistema de pacotes DEB – Debian Package e é executada em quase todos os computadores pessoais, inclusive os mais antigos, sendo que cada nova versão normalmente fica compatível com mais

máquinas.

A Debian, criada em 1993 por Ian Murdock, foi uma das primeiras distribuições criadas, com o intuito de ser desenvolvida abertamente, seguindo os moldes do Linux em si. Embora possa ser baixada virtualmente e instalada normalmente com uma conexão rápida, é tradicionalmente distribuída para instalação em CDs, que podem ser comprados pelo preço somente da mídia.

Para mais informações sobre essa distribuição, acessar www.debian.org.

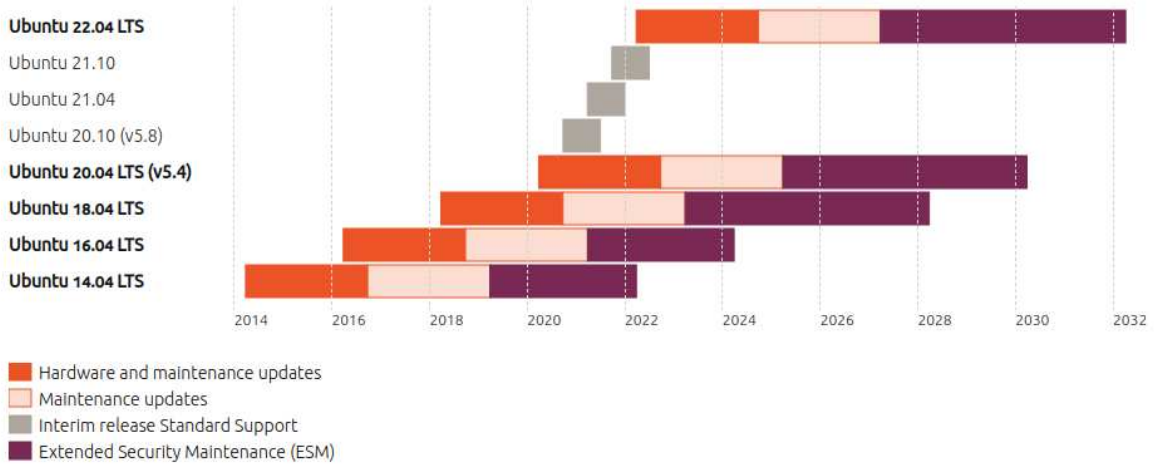
Ubuntu

Baseado no Debian, o Ubuntu, cujo nome significa "*Humanidade para os outros*" ou "*Sou o que sou pelo que nós somos*" em africano, conta com diversas ferramentas, como servidores web, ferramentas de programação, processador de texto e leitor de e-mails. Distribuído de forma convencional e Live, o Ubuntu é usado em laptops, desktop e servidores, e esses dois últimos contam com atualizações de segurança gratuitas por, no mínimo, 18 meses.

O Ubuntu, que utiliza o sistema de pacotes DEB, serviu de derivação para outras distribuições, como Kubuntu, Xubuntu e Lubuntu, só para citar algumas. Sua interface padrão é a GNOME ou KDE. Esta última é adotada pela distribuição Kubuntu.

As versões LTS ou '*Long Term Support*' são publicadas a cada dois anos, no mês de abril. As versões LTS são as versões de 'nível empresarial' do Ubuntu e são as mais utilizadas. Estima-se que 95% de todas as instalações do Ubuntu sejam lançamentos LTS.

A cada seis meses entre as versões LTS, a Canonical publica uma versão provisória do Ubuntu, sendo a versão 20.10 o exemplo mais recente. Essas são versões de qualidade de produção e são suportadas por 9 meses, com tempo suficiente fornecido para os usuários atualizarem, mas essas versões não recebem o compromisso de longo prazo recebido das versões LTS.



Os lançamentos provisórios apresentam novos recursos de projetos de código aberto da Canon-

ical e upstream, e servem como um campo de testes para esses novos recursos. Muitos desenvolvedores executam lançamentos provisórios porque fornecem compiladores mais novos ou acesso a kernels e bibliotecas mais recentes e são frequentemente usados em processos de DevOps rápidos, como pipelines de CI/CD, onde a vida útil de um artefato é provavelmente menor do que o período de suporte do lançamento provisório. Versões provisórias recebem manutenção de segurança total para a árvore principal (*main*) do projeto durante sua vida útil.

Para mais informações sobre essa distribuição, acessar www.ubuntu.com.

Linux Mint

Mint é uma distribuição derivada e compatível com o Ubuntu. Possui duas versões: uma baseada em Ubuntu (com o qual é totalmente compatível e partilha dos mesmos repositórios) e outra versão baseada em Debian. Suporta muitos idiomas, incluindo a língua portuguesa e utiliza o Cinnamon como seu principal ambiente de desktop. Esforça-se para ser um "sistema operacional moderno, elegante e confortável, além de poderoso e fácil de usar" e possui suporte multimídia pronto para o uso, incluindo alguns softwares proprietários e vem com uma variedade de aplicativos gratuitos e de código aberto. O Linux Mint diferencia-se do Ubuntu e do Debian por incluir drivers e codecs proprietários por padrão e por alguns recursos que permitem fazer através da interface gráfica (GUI – *Graphical User Interface*) configurações que em ambos os sistemas são feitas através do modo texto (CLI – *Command Line Interface*). Utiliza por padrão o desktop Cinnamon, um derivado do GNOME, com um menu no painel inferior junto à barra de tarefas (o MintMenu), similar ao menu do KDE ou o menu "Iniciar" do Windows. O propósito da distribuição é providenciar um sistema Linux que funcione "out-of-the-box", isto é, que esteja pronto para uso assim que terminada sua instalação.

Para mais informações sobre essa distribuição, acessar www.linuxmint.com.

Fedora

Outro sistema operacional baseado em Linux é o Fedora, de distribuição livre. Tem como interface padrão a GNOME e o gerenciamento de pacotes é feito de forma padrão pelo gerenciador de pacotes YUM. O YUM automaticamente baixa e gerencia a instalação do que for exigido pelo sistema, administrando pacotes instalados, atualizando ou removendo pacotes desnecessários.

Patrocinado pela Red Hat, o Fedora tem uma média de lançamento de versões novas a cada seis meses pelo Fedora Project, que é uma parceria formada por membros da comunidade de Software Livre. Diversas aplicações estão disponíveis para essa distribuição, como players de vídeo e áudio, que é o caso do MPlayer e do Amarok e de pacotes de ferramentas para escritório como o OpenOffice, além de uma coleção de jogos.

Para mais informações sobre essa distribuição, acessar fedoraproject.org.

Arch Linux

Uma distribuição no estilo *rolling release* (lançamento contínuo) com foco em usuários avançados e mantida por uma comunidade de voluntários. Com o sistema rolling release, os usuários podem ter acesso às últimas atualizações estáveis. Por exemplo, enquanto um usuário de Arch tem acesso imediato a última versão do GNOME, um de Ubuntu precisa esperar que a Canonical libere, na próxima versão do sistema operacional, essa atualização. As imagens de instalação lançadas pela equipe do Arch são apenas capturas instantâneas de imagens de disco atualizadas dos principais componentes do sistema.

Para mais informações sobre essa distribuição, acessar archlinux.org.

Manjaro Linux

Uma distribuição derivada do Arch Linux que inclui uma instalação gráfica e outras ferramentas para facilitar seu uso por usuários com menos conhecimento do Linux. O Manjaro Linux, ou apenas Manjaro (pronunciado "man-djá-ro" como em Kilimanjaro), é uma distribuição Linux com o ambiente gráfico padrão Xfce. Um sistema operacional de software livre, principalmente voltada para computadores pessoais destinadas a facilidade de uso. Como sua base de inspiração, o Arch Linux, o Manjaro usa um modelo rolling release. A distribuição tem como objetivo ser uma nova distribuição amigável ao usuário (*user friendly*), mantendo a poderosa base Arch, mais notavelmente o gerenciador de pacotes Pacman e a compatibilidade com o AUR (Repositório dos Usuários do Arch).

Para mais informações sobre essa distribuição, acessar manjaro.org.

SUSE

Uma das principais concorrentes da Red Hat, a distribuição corporativa SUSE foi adquirida pela empresa Novell em 2003. Atua no mercado corporativo oferecendo soluções para servidores e desktop. Sua interface padrão é a GNOME e adota o RPM como gerenciador de pacotes, incorporando suas próprias variações a esse sistema. A fim de sustentar uma alta compatibilidade com produtos de outras empresas, a SUSE cultiva muitas parcerias.

Para mais informações sobre essa distribuição, acessar https://www.suse.com/.

OpenSUSE

Com a interface padrão GNOME ou KDE e o sistema de pacotes RPM, a distribuição OpenSUSE é uma distribuição livre, patrocinada pela Novell e disponibilizada gratuitamente pelo openSUSE.org, visando a disseminar o uso do Linux. Tanto o OpenSUSE quanto as outras distribuições da SUSE Linux Enterprise têm como ferramenta de instalação e configuração o YaST.

Para mais informações sobre essa distribuição, acessar www.opensuse.org.

Knoppix

Distribuição do tipo Live, a Knoppix é executada a partir de um CD ou DVD, e pode se mostrar produtiva para desktops, sistemas educacionais, sistemas de recuperação ou ainda adaptada para versões demonstrativa de softwares. Consiste na plataforma GNU/Linux, com detecção automática dos dispositivos de hardware e suporte para vários periféricos, incluindo dispositivos USB e placas gráficas.

Seu sistema de pacotes é o DEB e tem como interface padrão a KDE. Em razão de sua ferramenta de remasterização, a Knoppix contribuiu significantemente para que as distribuições do tipo Live ganhassem popularidade. Essa ferramenta facilitou a tarefa de produzir novas distribuições.

Para mais informações sobre essa distribuição, acessar www.knoppix.org.

Gentoo

Gentoo é o que se chama de metadistribuição, devido à sua quase ilimitada adaptabilidade. Não só sua interface padrão fica à escolha do usuário, como o sistema pode ser otimizado e personalizado para praticamente qualquer aplicativo ou necessidade, provando-se eficiente como servidor seguro, workstation de desenvolvimento, desktop profissional, sistema de jogos, e solução embutida, para citar alguns exemplos.

O sistema operacional é baseado no Linux ou no FreeBSD e seu sistema de pacotes é o emerge, que interage com o Portage, um sistema de distribuição de software, construção de pacotes e instalação. Mantido por mais de 300 desenvolvedores e milhares de usuários e distribuído de forma Live, o Gentoo conta com ferramentas de documentação, infraestrutura, engenharia de lançamento, portagem de software e garantia de qualidade, entre outras.

Para mais informações sobre essa distribuição, acessar www.gentoo.org.

Ciclo de vida de algumas distribuições Linux

Debian GNU/Linux

O Debian sempre tem pelo menos três versões em manutenção ativa: estável (stable), testing e instável (unstable).

- **Estável (stable)**: a distribuição em sua versão estável (stable) contém a última versão oficialmente lançada do Debian. Esta é a versão de produção do Debian, a que nós primeiramente recomendamos que seja usada. A atual distribuição estável (stable) do Debian é a versão 10, codinome buster. Ela foi inicialmente lançada como versão 10 em 6 de julho de 2019 e sua última atualização, a versão 10.3, foi lançada em 8 de fevereiro de 2020.
- **testing**: a distribuição na versão testing contém pacotes que ainda não foram aceitos

na versão estável (stable), mas que estão na fila para tal. A principal vantagem de usar esta distribuição é que ela possui versões mais recentes de software. A atual distribuição testing é a *bullseye*.
- **Instável (unstable)**: a distribuição na versão instável (unstable) é onde o desenvolvimento ativo do Debian ocorre. Geralmente, esta distribuição é utilizada por desenvolvedores e por aqueles que gostam de viver no limite. A distribuição instável (unstable) é sempre chamada de *sid*.

Fonte para consulta: https://www.debian.org/releases/index.pt.html

Evolução do Linux: dispositivos embarcados

Os sistemas embarcados são uma combinação de hardware e software projetados para cumprir uma função específica dentro de um sistema maior. Normalmente fazem parte de outros dispositivos e ajudam a controlá-los. Podem ser encontrados em aplicações automotivas, médicas e até militares. Devido a essa ampla variedade de aplicações, Uma variedade de sistemas operacionais baseados no kernel do Linux foi desenvolvida para uso em sistemas embarcados. Uma parte significativa dos dispositivos inteligentes usa um sistema operacional baseado no kernel do Linux.

Assim, em sistemas embarcados temos software embarcado, cujo objetivo é acessar o hardware e torná-lo utilizável. Dentre as principais vantagens do Linux sobre qualquer software embarcado proprietário estão a compatibilidade entre plataformas de diferentes fornecedores, desenvolvimento, suporte e ausência de taxas de licença. Dois dos mais populares projetos de software embarcado são o Android, usado principalmente em telefones celulares por diferentes fabricantes, e o Raspbian, que é usado principalmente no Raspberry Pi.

O Linux se tornou tão popular como uma plataforma de software para a Internet das Coisas (IoT), que você pode encontrar não apenas uma, mas muitas distribuições Linux para utilizar em seus projetos. Distribuições populares do Linux para IoT:

- **AndroidThings**: sistema operacional baseado em Android com suporte para APIs do Android e serviços do Google.
- **Debian Tinker**: Debian para sistemas embarcados.
- **OpenWrt**: distribuição baseada em Linux para dispositivos embarcados (usados principalmente em dispositivos como roteadores sem fio).
- **Raspbian**: Raspbian é um derivado do sistema operacional Debian Linux que funciona no hardware Raspberry Pi (inclui mais de 35K pacotes).
- **Tizen**: a pilha de Linux Embarcado da Samsung que alimenta grande parte dos dispositivos de consumidor e IoT da Samsung.
- **Ubuntu Core**: o Ubuntu Core é uma versão incorporada do Ubuntu que é executada em sistemas com recursos mínimos (como o Raspberry Pi).
- **Yocto**: o Yocto não é por si só uma distribuição, mas sim um projeto de código aberto

que ajuda a criar distribuições personalizadas do Linux para sistemas embarcados e IoT.

6

Conhecendo o Linux

Os sistemas GNU/Linux herdaram muitas características do padrão Unix, o que refletirá na maneira como executaremos os comandos a partir do interpretador de comandos. O sistema operacional MAC OS X, da Apple, por ser Unix-like, também herda tais características.

- **Representação por arquivos**: esta é uma herança fundamental do sistema Unix, em que tudo que estiver presente no sistema será representado por um arquivo. Para tanto, teremos várias classificações para tipos de arquivos, que representarão dispositivos de armazenamento de dados, memória física, portas seriais ou paralelas, arquivos especiais, arquivos de texto comuns, binários e até mesmo diretórios. É importante dizer que um diretório também é um arquivo, um arquivo especial que aponta para outros arquivos e subdiretórios presentes dentro do sistema. No momento de gerenciar o sistema, é importante saber lidar com todos esses tipos de arquivos e tomar as devidas precauções na hora de manipulá-los.
- **Aspecto case-sensitive**: uma característica forte em sistemas Unix-like é a presença da característica case-sensitive, em que letras maiúsculas serão completamente diferentes de letras minúsculas e vice-versa. No início, quando não se tem prática e costume com essa característica, é um tanto desconfortável se acostumar, mas depois, com o domínio, ela é útil e prática. A funcionalidade case-sensitive estará presente no sistema através de nomes de arquivos, diretórios, comandos etc. Por exemplo, o comando LS é completamente diferente de ls.
- **Permissão para executar arquivos**: em sistemas Unix-like, para que um arquivo possa ser executado, inclusive pelo administrador do sistema, deve existir, obrigatoriamente, permissões de execução. Esse arquivo pode ser um arquivo binário de um comando qualquer, um shell script etc. A permissão de execução protege o sistema contra arquivos auto-executáveis e faz com que os sistemas Unix-like sejam praticamente imunes a vírus ou arquivos com funções mal-intencionadas.

- **Arquivos e diretórios ocultos**: por padrão, a forma utilizada para identificar arquivos ou diretórios ocultos no sistema é iniciar seus respectivos nomes por um ponto (.). Uma vez que na linha de comandos não há como utilizar efeitos como os usados nos ambientes gráficos, para visualizar tais arquivos ou diretórios ocultos, basta utilizar uma simples opção com o comando de visualização `ls`, que será abordado posteriormente neste treinamento.
- **Divisão em camadas**: o sistema é todo dividido e trabalhado em camadas, não sendo necessária a execução obrigatória de um ambiente gráfico, por exemplo. Teremos o kernel como o centro de execução do nosso sistema, que controlará todo o hardware e fará a interface de controle e gerenciamento dos usuários, aplicações e processos em execução.
- **Superusuário (root)**: o superusuário `root` é, por padrão, o único usuário com permissões completas de administração e gerenciamento do sistema. Sua utilização deve ser restrita e extremamente cautelosa, exatamente pela liberdade que possui ao executar qualquer tipo de tarefa, como excluir um diretório importante, por exemplo. O login do superusuário é `root` e o que define esse login como superusuário não é simplesmente o nome `root`, mas sim a UID (*User Identification*), que é o número de identificação que cada usuário terá no sistema. O superusuário possui UID de número 0 (zero).
- **Aspecto multitarefa / multiusuário**: uma característica herdada dos primórdios Unix é poder executar várias tarefas ao mesmo tempo, com vários usuários trabalhando simultaneamente no sistema.

Principais aplicações desktop Open-Source

- **Firefox**: um dos navegadores mais populares do mundo e utilizado por milhões de pessoas está presente em ambas as plataformas, a experiência de usar o Mozilla Firefox no Windows e no Linux é basicamente a mesma, então em uma possível migração isso não fará muita diferença. Claro que o Firefox não é o único navegador multiplataforma, temos outras opções equivalentes como Chrome, Vivaldi, Opera entre outros.
- **Thunderbird**: com o Thunderbird você baixa ou apenas acessa seus emails do Zimbra, Gmail, Hotmail, Yahoo, iG, UOL, Oi Mail, iBest, Terra, BOL ou qualquer outro provedor com suporte a SMTP/POP3/IMAP. O Thunderbird permite que você adicione novo recursos a medida que você precisar através de extensões. As extensões são ferramentas poderosas que auxiliam você a criar um cliente de e-mail de acordo com suas necessidades. O Thunderbird oferece recursos necessários a empresas, tais como S/MIME, assinaturas digitais, criptografia de mensagens, suporte a certificados e dispositivos de segurança.
- **GIMP**: é um programa de código aberto voltado principalmente para criação e edição de imagens e em menor escala também para desenho vetorial. O GIMP foi criado pela comunidade como uma alternativa livre ao Adobe Photoshop. Foi um projeto universitário que amadureceu bastante e hoje alcança expressiva popularidade, sendo utilizado por hobbistas e profissionais.

- **Inkscape**: é um software livre para editoração eletrônica de imagens e documentos vetoriais. Utiliza o método vetorial, ou seja, gera imagens a partir de um caminho de pontos definindo suas coordenadas, de forma transparente ao usuário. Imagens vetoriais têm maior aplicação em desenho técnico ou artístico e são, geralmente, mais leves e não perdem a qualidade ao sofrer transformações, como redimensionamento ou giro. O Inkscape trabalha nativamente com o formato SVG (`Scalable Vectorial Graphics`), um formato aberto de imagens vetoriais, nomeadamente, uma subdefinição (DTD) da linguagem XML definido pela W3C. O aplicativo também exporta para o popular formato da Internet PNG e importa vários formatos vectoriais ou bitmap, como por exemplo: TIFF, GIF, JPG, AI, PDF, PS, entre outros.
- **Kdenlive**: KDE Non-Linear Video Editor é um editor de vídeo open-source baseado no framework MLT e KDE. O projeto foi iniciado por Jason Wood em 2002, e hoje é mantido por um pequeno time de desenvolvedores. Com o lançamento do Kdenlive 15.04.0 passou a ser oficialmente parte do projeto oficial do KDE. Pacotes do Kdenlive estão livremente disponíveis para Linux, FreeBSD e Mac OS X sob os termos da GNU General Public License versão 2 ou posterior.
- **Virtualbox**: é um software de virtualização desenvolvido pela empresa Innotek depois comprado pela Sun Microsystems que posteriormente foi comprada pela Oracle que, como o VMware Workstation, visa criar ambientes para instalação de sistemas distintos. Ele permite a instalação e utilização de um sistema operacional dentro de outro, assim como seus respectivos softwares, como dois ou mais computadores independentes, mas compartilhando fisicamente o mesmo hardware. O VirtualBox pode ser instalado em vários sistemas operacionais hospedeiros, incluindo: Linux, MacOS e Windows. O VirtualBox pode ser usado tanto por usuários comuns, quanto por desenvolvedores de sistemas ou profissionais de TI.
- **LibreOffice**: é uma suíte de aplicativos livre para escritório disponível para Windows, Unix, Solaris, Linux e Mac OS X. A suíte utiliza o formato OpenDocument (ODF – *OpenDocument Format*) – formato homologado como ISO/IEC 26300 e NBR ISO/IEC 26300 – e é também compatível com os formatos do Microsoft Office, além de outros formatos legados. Alguns deles não são suportados pelas versões mais recentes do Microsoft Office, mas ainda podem ser abertos pelo LibreOffice. O LibreOffice é composto pelos seguintes aplicativos:
  - **Writer** - Editor de Texto;
  - **Calc** - Planilha;
  - **Impress** - Apresentações;
  - **Draw** - Desenho Vetorial;
  - **Math** - Fórmulas matemáticas;
  - **Base** - Banco de Dados;
- **VLC**: é um software de media player portátil gratuito e de código aberto, multiplataforma e também um servidor de mídia de streaming desenvolvido pelo projeto VideoLAN. O VLC está disponível para sistemas operacionais de desktop e plataformas móveis, como Android e iOS. O VLC também está disponível em plataformas de distribuição digital, como a App Store da Apple ou na Google Play. O VLC suporta muitos métodos de compactação de áudio e vídeo e formatos de arquivo, incluindo DVD-Video,

CD de vídeo e protocolos de streaming. É capaz de transmitir mídia por redes de computadores e transcodificar arquivos multimídia. A distribuição padrão do VLC inclui muitas bibliotecas de decodificação e codificação gratuitas, evitando a necessidade de encontrar/calibrar plugins proprietários.

- **Audacity**: é um software livre de edição digital de áudio disponível principalmente nas plataformas: Windows, Linux e Mac e ainda em outros Sistemas Operacionais. O código fonte do Audacity está sob a licença *GNU General Public License*. O Audacity é muito popular entre os podcasters por seus recursos de edição, sua grande disponibilidade em múltiplas plataformas, suporte e licença abertas, que permite ao programa ser gratuito.
- **ImageMagick**: o ImageMagick é uma ferramenta de linha de comando usada para converter e editar a maioria dos tipos de arquivos de imagem. Também pode ser usado para criar documentos PDF a partir de arquivos de imagem e vice e versa.
- **K3B**: é um programa de computador que funciona como uma interface gráfica para a gravação de CD-ROMs e DVDs e funciona normalmente em sistemas operacionais da família Unix, tais como o Linux e o FreeBSD. Utiliza-se, para gravar mídias, dos programas `cdrecord`, `cdrdao` e `growisofs`. O K3B faz parte do projeto KDE. Permite criar, além de CD-ROMs e DVDs de dados, CDs de áudio, CDs de vídeo e cópias exatas de CDs e DVDs.
- **Blender**: é um programa de computador de código aberto, desenvolvido pela Blender Foundation, para modelagem, animação, texturização, composição, renderização, e edição de vídeo. Está disponível sob a GNU GPL, versão 2 ou posterior. O programa é multiplataforma, estando portanto disponível para diversos sistemas operacionais. O Blender implementa ferramentas similares às de outros programas proprietários.

O programa mais popular para a reprodução de vídeo é o VLC, mas alguns usuários preferem outras alternativas, como o `smplayer`. A reprodução de música local também traz muitas opções, como o Audacious, o Banshee e o Amarok, que também podem gerenciar uma coleção local de arquivos de áudio.

Principais aplicações em servidores Open-Source

- **Apache HTTP Server**: o servidor HTTP Apache (do inglês Apache HTTP Server) ou Servidor Apache ou HTTP Daemon Apache ou somente Apache, é um servidor web livre criado em 1995 por Rob McCool. É a principal tecnologia da *Apache Software Foundation*, responsável por mais de uma dezena de projetos envolvendo tecnologias de transmissão via web, processamento de dados e execução de aplicativos distribuídos. É um servidor do tipo HTTPD, compatível com o protocolo HTTP versão 1.1. Suas funcionalidades são mantidas através de uma estrutura de módulos, permitindo inclusive que o usuário escreva seus próprios módulos – utilizando a API do software.
- **NGINX**: Nginx (lê-se "engine x") é um servidor leve de HTTP, proxy reverso, proxy de e-mail IMAP/POP3, feito por Igor Sysoev em 2005, sob a licença BSD-like 2-clause.
- **Lighttpd**: o servidor Lighttpd (pronuncia-se "lighty"), é um servidor web otimizado para

ambientes de velocidade crítica. Foi originalmente escrito por Jan Kneschke como uma prova de conceito do problema c10k – como lidar com 10.000 conexões em paralelo em um servidor, ganhando popularidade mundial. Traz um gerenciamento eficaz da `cpu-load` e um conjunto de recursos avançados como FastCGI, SCGI, Auth, Output-Compression, URL-Rewriting etc.

- **Apache Tomcat**: o Tomcat é um servidor web Java, mais especificamente, um container de servlets. O Tomcat implementa, dentre outras de menor relevância, as tecnologias Java Servlet e JavaServer Pages (JSP).
- **Node.js**: interpretador de JavaScript assíncrono com código aberto orientado a eventos, criado por Ryan Dahl em 2009, focado em migrar a programação do Javascript do cliente (frontend) para os servidores, criando aplicações de alta escalonabilidade (como um servidor web), manipulando milhares de conexões/eventos simultâneas em tempo real em uma única máquina física.
- **Mariadb**: MariaDB Server é um dos bancos de dados relacionais de código aberto mais populares. Destinado a permanecer como um software de código-fonte aberto e gratuito sob a *GNU General Public License*, foi criado pelos desenvolvedores originais do MySQL que o bifurcaram devido a preocupações sobre sua aquisição pela Oracle Corporation em 2009. É parte da maioria das ofertas de nuvem e o padrão na maioria das distribuições Linux.
- **PostgreSql**: também conhecido como Postgres, este é um poderoso sistema de banco de dados relacional de objetos de código aberto que usa e estende a linguagem SQL combinada com muitos recursos que armazenam e escalonam com segurança as cargas de trabalho de dados mais complicadas. Tem uma forte reputação por sua arquitetura comprovada, confiabilidade, integridade de dados, conjunto de recursos robustos, extensibilidade e dedicação da comunidade de código aberto por trás do software para fornecer soluções inovadoras e de alto desempenho de maneira consistente.
- **Postfix**: o Postfix é um agente de transferência de e-mails (MTA) livre e de código aberto que encaminha e entrega e-mails e tem como objetivo ser uma alternativa segura ao Sendmail, muito utilizado em servidores UNIX. O Postfix foi originalmente escrito em 1997 por Wietse Venema no Centro de Pesquisa IBM Thomas J. Watson e teve a sua primeira versão lançada em 1998.
- **Bind**: o BIND – *Berkeley Internet Name Domain* – ou, como chamado previamente, *Berkeley Internet Name Daemon* é o servidor para o protocolo DNS mais utilizado na Internet, especialmente em sistemas do tipo Unix, onde ele pode ser considerado um padrão, de fato.
- **Squid**: o Squid é um servidor proxy que suporta HTTP, HTTPS, FTP e outros. Ele reduz a utilização da conexão e melhora os tempos de resposta fazendo cache de requisições frequentes de páginas web numa rede de computadores. Ele pode também ser usado como um proxy reverso.
- **Hadoop**: plataforma Java de software de computação distribuída voltada para clusters e processamento de grandes volumes de dados, com atenção e tolerância a falhas. Foi inspirada no MapReduce e no GoogleFS (GFS). Trata-se de um projeto da Apache de alto nível, construído por uma comunidade de contribuidores e utilizando a linguagem de programação Java.

- **Kubernetes**: Kubernetes (comumente estilizado como K8s) é um sistema de orquestração de contêineres open-source que automatiza a implantação, o dimensionamento e a gestão de aplicações em contêineres. Ele foi originalmente projetado pelo Google e agora é mantido pela Cloud Native Computing Foundation. Ele funciona com uma variedade de ferramentas de conteinerização, incluindo Docker.

Principais linguagens de programação Open-Source

- **Python**: linguagem de programação de alto nível, interpretada, de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi lançada por Guido van Rossum em 1991. Atualmente possui um modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos *Python Software Foundation*.
- **PHP**: um acrônimo recursivo para *Pre Hypertext Preprocessor*, originalmente *Personal Home Pages*) é uma linguagem interpretada livre, usada originalmente apenas para o desenvolvimento de aplicações presentes e atuantes no lado do servidor, capazes de gerar conteúdo dinâmico na web.
- **Java**: o Java é uma linguagem de programação orientada a objetos desenvolvida na década de 90 por uma equipe de programadores chefiada por James Gosling, na empresa Sun Microsystems. Em 2008 o Java foi adquirido pela empresa Oracle Corporation. Diferentemente das linguagens de programação modernas, que são compiladas para código nativo, a linguagem Java é compilada para um bytecode que é interpretado por uma máquina virtual (*Java Virtual Machine*, mais conhecida pela sua abreviação JVM).

Apresentação dos principais gerenciadores de janelas usados no Linu

Com os gerenciadores de janela (*window managers*), tanto a aparência quanto a localização das janelas do X são administradas. Um dos diferenciais do GNU/Linux é permitir que o próprio usuário gerencie as janelas, em vez de conceder esse controle ao sistema, como é feito pelos sistemas Apple e Microsoft, que oferecem apenas uma aparência básica.

No GNU/Linux, um computador pode ter vários gerenciadores de janela, cabendo ao usuário usar o que mais lhe convier. Entre os gerenciadores, temos: AfterSteps, Enlightenment, Kwin (ambiente KDE), Blakbox, Evilwn, IceWM, FluxBox, SawFish, FVWM, Ion, Metacity (ambiente GNOME), WMN, OpenClasses (Sun), xfce e twm.

Os gerenciadores de janela são diferentes entre si no que diz respeito a diversos fatores, tais como:

- Recursos necessários do computador;
- Customização de funcionalidades e de aparência;
- Configuração de menus;

- Interface gráfica;
- Capacidade de uso de vários computadores.

Muitas distribuições GNU/Linux atuais disponibilizam ferramentas de acessibilidade voltadas para usuários com necessidades especiais, como problemas motores e visuais.

7

Tópicos para revisão do capítulo

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes deste capítulo.

- Um sistema operacional é formado por um conjunto de funções que, trabalhando em perfeita harmonia, manipulam o hardware e, através de uma interface de comunicação, oferecem todos os recursos úteis para o seu gerenciamento;
- A *Free Software Foundation* é uma organização sem fins lucrativos cujo foco principal é manter a GPL;
- O Linux segue os princípios do software livre, ou seja, o usuário terá a liberdade de estudá-lo, executá-lo, alterá-lo, melhorá-lo, copiá-lo e até mesmo distribuí-lo;
- Tecnicamente, o Linux não é um sistema operacional, mas sim um kernel, ou melhor, o centro de execução do seu sistema operacional. Cada sistema operacional, independentemente da plataforma, possui o seu próprio kernel, com suas características peculiares;
- Existe uma quantidade imensa de distribuições disponíveis para instalação, com praticamente todos os níveis de dificuldade e focos específicos, como Debian, Slackware, Red Hat, SUSE, entre outras;
- A certificação LPI é uma das certificações mais procuradas por profissionais da área no mundo, por ser uma certificação internacional que é reconhecida por empresas, empregadores e profissionais do mundo de TI, além de ter grande destaque entre outras certificações Linux.

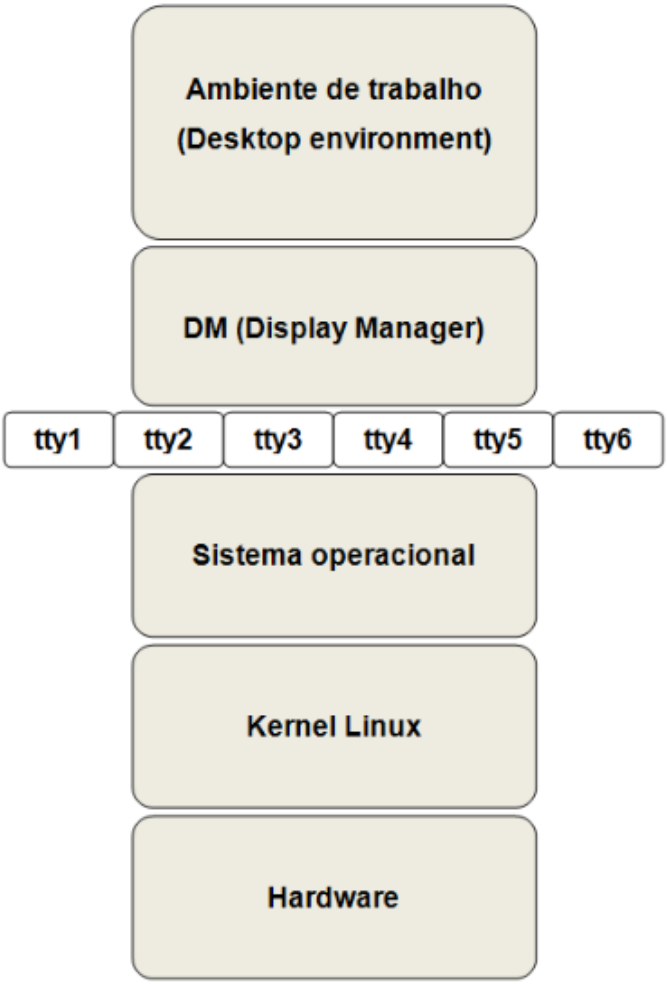
No presente capítulo, aprenderemos a executar comandos para realizar tarefas básicas no sistema **GNU/Linux**, como, por exemplo, a inicialização de uma sessão. No entanto, antes de executá-los, é importante compreender as características que esse sistema herda do padrão Unix, a estrutura do sistema operacional **GNU/Linux** e os conceitos de shell e bash. Este último é o interpretador de comandos comumente utilizado por sistemas Unix-like, como já

vimos no capítulo anterior.

8

Estrutura do sistema operacional

A estrutura de um sistema **GNU/Linux** é composta por porções denominadas camadas (*layers*). A figura a seguir demonstra a estrutura em camadas do sistema operacional **GNU/Linux**:



Para compreender de fato o que é esse sistema, é fundamental saber do que se trata cada uma de suas camadas. Vejamos:

- **Ambiente de trabalho (desktop environment)**: hospeda todos os programas cujo funcionamento depende de um ambiente gráfico;
- **DM (display manager)**: aqui são gerenciados os logins na interface gráfica e selecionado o tipo de ambiente gráfico que será executado;
- **ttyn**: terminais virtuais que interpretam os comandos introduzidos por um usuário e os convertem para uma linguagem inteligível pelo computador. Nessa camada os comandos são executados e as configurações são definidas;
- **Sistema Operacional**: camada que auxilia e hospeda todos os aplicativos das camadas superiores, mencionadas anteriormente;
- **Kernel Linux**: mediadora entre as camadas superiores e o hardware, essa camada é o núcleo do sistema operacional;
- **Hardware**: camada que compreende os dispositivos do sistema disponíveis para uso, como CD-ROM, teclado, monitor, entre outros.# Introdução

Sessões

Uma sessão é composta pelos dados gerados após o login de um usuário válido cadastrado no sistema e efetuado com sucesso no ambiente. Esse login dará acesso direto ao interpretador de comandos padrão determinado no sistema, que geralmente será o bash. As sessões são iniciadas através dos terminais, que são caracterizados como terminal virtual somente em modo texto ou pseudoterminal inicializado a partir de um ambiente gráfico. A única diferença entre eles é a forma de acesso. A seguir, falaremos sobre esses terminais.

Terminal virtual em modo texto

O conjunto teclado/monitor conectado à máquina compreende o que chamamos de terminal, ou console. Por meio de terminais virtuais, o GNU/Linux permite que diversos usuários se mantenham simultaneamente conectados, pondo em prática o aspecto multiusuário/multitarefa, característico do padrão Unix.

Com programas como `rlogin`, `ssh`, `rsh`, `rdesktop` e `telnet`, podemos acessar terminais virtuais remota ou localmente.

Um terminal virtual em modo texto é prático e rápido, pois não é necessário esperar o ambiente gráfico carregar para iniciar uma sessão a partir do mesmo. Geralmente, em servidores, o recurso de ambiente gráfico não está habilitado. Em casos como esse, assim que o processo de inicialização do sistema for concluído, o primeiro terminal estará disponível aguardando um login para o início de uma sessão.

É possível ter acesso a outros terminais que trabalham de forma completamente independente. O GNU/Linux apresenta precisamente **63 terminais** em modo texto, mas, por questões de consumo de memória RAM, somente 6 são disponibilizados por padrão.

Para acessá-los é muito simples. Basta manter pressionada a tecla **ALT** e, em seguida, pressionar a tecla de função correspondente ao terminal que desejamos acessar (teclas F1 até F6). Se estivermos no primeiro terminal, temos o seguinte:

Terminal	Nome	Tecla de atalho para acesso
Primeiro	tty1	(CTRL +) ALT + F1
Segundo	tty2	(CTRL +) ALT + F2
Terceiro	tty3	(CTRL +) ALT + F3
Quarto	tty4	(CTRL +) ALT + F4
Quinto	tty5	(CTRL +) ALT + F5
Sexto	tty6	(CTRL +) ALT + F6

Nota sobe CTRL + ALT +F(n): caso o terminal padrão não seja um terminal em modo texto,

é preciso utilizar também a tecla ALT.

Teremos então, na primeira linha, o nome e a versão do sistema operacional, o nome do computador e o nome do terminal em que estamos localizados. Na figura que mostra o terminal a seguir, temos **Debian GNU/Linux 10 debian tty1**:

Onde:

- Debian GNU/Linux 10 é o nome e a versão do sistema operacional;
- debian é o nome da máquina;
- tty1 indica o primeiro terminal.

Pseudoterminal

Um pseudoterminal é, basicamente, um terminal virtual inicializado a partir do ambiente gráfico. Tem a mesma função do terminal em modo texto, porém depende da execução do ambiente gráfico para que seja carregado. Os gerenciadores de janelas, como o KDE ou o GNOME, têm aplicações específicas que emulam um terminal no ambiente gráfico, como konsole no KDE, ou o próprio terminal no GNOME.

Execução dos primeiros comandos

Vejamos, a seguir, como executar os primeiros comandos a partir do shell do GNU/Linux, a fim de iniciar e encerrar uma sessão, desligar e reiniciar o sistema e obter ajuda a respeito da utilização de comandos.

Início de uma sessão (login)

Depois de conhecer melhor os terminais, que são a porta de entrada para o sistema, é hora de iniciar uma sessão em modo texto (fazer o login), pois esta exigirá, em um sistema configurado de modo seguro, que seja feita uma autenticação com um usuário previamente cadastrado. Em

qualquer um dos terminais virtuais em modo texto, basta inserir o login e a senha, lembrando-se da característica case-sensitive presente no sistema. Então, a sessão é iniciada, como mostrado na figura a seguir:

Assim que a autenticação for executada com sucesso, o prompt principal do interpretador de comandos utilizado é apresentado, e o sistema já pode ser operado.

Sobre o prompt principal citado anteriormente, teremos muitas informações caso o interpretador de comandos seja o bash. Caracteres como `,`, `:`, `-` e `@` são utilizados apenas para separar informações e formar o prompt principal. O caractere `$` identifica que o usuário atual não tem permissões ou poderes de superusuário para executar determinadas operações .

```
1 username@mycomputer:~$
```

Para executar comandos com poderes administrativos, temos várias maneiras e uma delas é alterar o nível de usuário de comum para superusuário. Para isso, podemos utilizar o comando `sudo -i`. Ao digitar `sudo -i` na linha de comando e teclar `ENTER`, é solicitada a senha de seu usuário.

```
1 username@mycomputer:/$ sudo -i
2 Senha:
3 root:/#
```

Outra maneira, seria alterar o login para o usuário `root`:

```
1 username@mycomputer:/$ su -
2 Senha:
3 root:/#
```

O comando `su -` (*substitute user*), tem a finalidade de alterar a sessão para outro usuário,

neste caso como não informamos nenhum após o `-`, por padrão será o usuário `root`.

Em vez de um `$`, teremos o caractere `#`, indicando que o usuário agora é um superusuário.

Existem dois comandos, **whoami** e **who am i** que lhe permite saber quem você é em determinado momento. A sequência de comandos abaixo esclarece o uso e finalidade destes dois comandos claramente:

```
1 whoami
2 who am i
```

O comando `whoami` indica quem você é no momento `root`. Se você utilizou o comando `su` para tornar-se outro usuário o comando `whoami` informa quem você realmente é **aluno**, pois foi com ele que você se logou na máquina antes de trocar de usuário.

Ele também pode ser utilizado para trocar de usuário, ele não pedirá a senha se você for usuário `root`:

```
1 su - aluno
```

Todas essas informações sobre o prompt principal poderão ser alteradas através de variáveis de ambiente.

Encerrando uma sessão (logout)

Após uma sessão em modo texto ser inicializada, é possível fechá-la (fazer o logout) com os comandos `logout` ou `exit`. A seguir veja o shell com o comando `exit` digitado:

```
1 username@mycomputer:~$ exit
```

A sequência de teclas `CTRL + D` também pode ser utilizada para finalizar uma sessão.

Desligamento do sistema

Sob o nível de usuário `root`, o desligamento do computador pode ser realizado por um dos seguintes comandos:

```
1 shutdown h now
```

ou:

```
1 halt -p
```

ou:

```
1 poweroff
```

ou:

```
1 init 0
```

Ainda que o computador possa ser desligado diretamente pelo botão liga/desliga, sem a utilização de nenhum dos comandos citados, esse procedimento deve ser evitado sempre que possível, pois os programas que não forem fechados e os dados não gravados podem gerar falhas no sistema de arquivos e perda de dados.

Dessa forma, para garantir a segurança do sistema, o usuário deve escolher um dos três comandos de desligamento disponíveis, que automaticamente executam esses processos de encerramento da maneira adequada. Salvar manualmente os arquivos antes do desligamento e considerar o uso de um nobreak são outras medidas de segurança possíveis.

O comando shutdown poderá ser utilizado para desligar ou reiniciar o sistema.

Para desligar o sistema, utiliza-se a sintaxe `shutdown -h <tempo>`, onde `<tempo>` define o momento (em minutos) em que ocorrerá o desligamento e `h` significa `halt` ou seja, desligar o sistema.

Para desligar o sistema imediatamente, digitamos o comando `now` (agora) conforme a seguinte linha:

```
1 shutdown -h now
```

Ao utilizar o parâmetro `now`, os processos atualmente executados são encerrados antes do desligamento da máquina. Outro meio de desligar o sistema imediatamente é utilizar o valor `0` (zero).

```
1 shutdown -h 0
```

Para desligar o sistema daqui a 10 minutos, por exemplo, digitamos o seguinte:

```
1 shutdown -h 10
```

Para desligar o sistema exatamente às 17:00, por exemplo, digitamos o seguinte:

```
1 shutdown -h 17:00
```

Os comandos `halt` ou `poweroff` também podem ser utilizados para desligar o sistema de forma simples e direta, mas sem a opção de poder especificar o tempo. Em computadores antigos, o usuário precisará apertar o botão liga/desliga, quando a mensagem `power down` aparecer, após o envio do comando `halt`.

Reiniciar o sistema

É recomendável não reiniciar o sistema através dos botões liga/desliga e RESET, pois, como já mencionado, o desligamento abrupto do computador pode causar não só a perda de dados não salvos adequadamente, mas também falhas nos sistemas de arquivos. Existem comandos específicos para reiniciar o sistema de forma segura, e os botões citados devem ser evitados e usados somente em último caso.

Para reiniciar o sistema, podemos utilizar, como no desligamento, o comando `shutdown`. No entanto, utilizamos o parâmetro **–r** (proveniente de `reboot`) em vez de `-h`, como mostra esta sintaxe:

```
1 shutdown -r <tempo>
```

O momento em que deve ocorrer a reinicialização deve ser especificado em `<tempo>`. No exemplo a seguir, o comando `shutdown` reiniciará o sistema daqui a 10 minutos:

```
1 shutdown -r 10
```

Para reiniciar imediatamente o sistema, digitamos a seguinte linha:

```
1 shutdown -r now
```

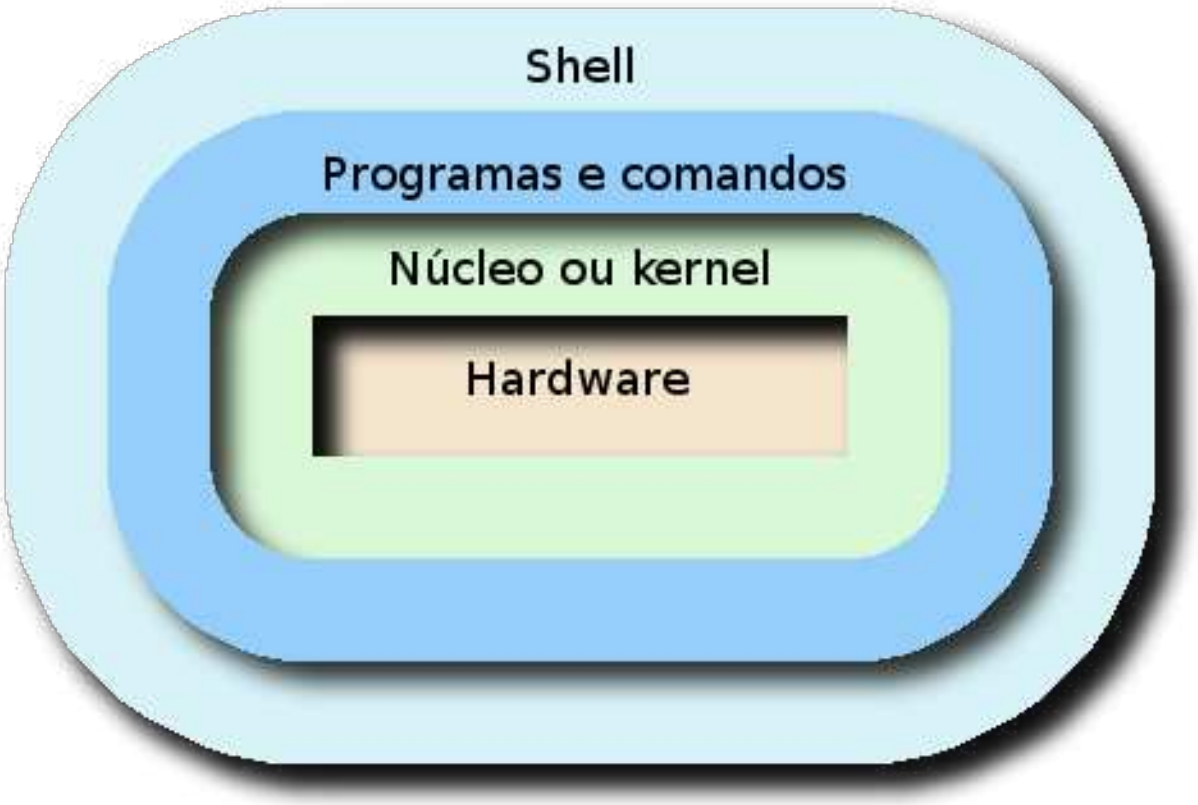
O comando `reboot` ou a sequência de teclas `CTRL + ALT+ DEL` também pode ser utilizado de forma simples para reiniciar o sistema, mas sem a opção de poder especificar o tempo. Antes de reiniciar o sistema, todos os arquivos abertos devem ser salvos, para evitar acidentes.

9

O que é um Shell

Sabemos que o principal meio de interação do usuário com um sistema GNU/Linux é o terminal de comandos, também conhecida como **shell**. Portanto e possível afirmar que o **shell** é uma camada de acesso ao sistema básico, o sistema operacional do computador, que pode ser acessada tanto pelo modo gráfico, quanto pelo modo texto. O **shell** pode ser personalizado para atender as necessidade do usuário.

Pode-se definir um idioma padrão, personalizar e automatizar processos. Nos tópicos a seguir, veremos como fazer essa personalização. A figura abaixo ilustra como podemos posicionar a **shell** dentro do sistema.



Tipos de shell

Há diversos tipos de shell disponíveis para distribuições **GNU/Linux**. O bash é o tipo utilizado na maioria das distribuições. A seguir, temos a descrição dos principais tipos de shell:

- **sh (Bourne shell)**: desenvolvido por Stephen Bourne, por isso seu nome. É o shell original, bem simples, com poucas ferramentas, mas ainda utilizado em sistemas UNIX e ambientes relacionados ao UNIX.
- **bash (Bourne Again shell)**: trata-se do shell padrão do Linux, conveniente tanto para usuários iniciantes, por ser intuitivo e flexível, quanto para usuários avançados e profissionais, por possuir ferramentas variadas e eficientes. Compatível com todos os comandos do `sh`.
- **csh (C shell)**: o preferido de alguns programadores. A sintaxe desse shell é semelhante à linguagem de programação C.
- **tcsh (Turbo C shell)**: versão melhorada do `csh`, que interpreta linguagens de comando e pode ser usada tanto como um shell de login interativo quanto como um processador de comandos de shell scripts. É completamente compatível com o `csh`.
- **ksh (Korn shell)**: este shell é o Bourne Shell evoluído, portando todos os comandos que funcionavam no Bourne Shell funcionarão aqui, com a vantagem de ter mais opções.
- **zsh (Z shell)**: além de um shell desenvolvido para uso interativo, que engloba várias

funções úteis do **bash**, **ksh** e **tcsh**, o **zsh** é também uma linguagem de script eficiente.

Alteração do shell atual

Para alterar o shell atual, utilizamos o comando `chsh` com a opção `-s` ou `--shell`, seguido do nome do shell que se deseja utilizar.

A linha a seguir altera o shell para o `sh`:

```
1 chsh -s /bin/sh
```

10

Variáveis

Podemos definir variáveis como nomes que contêm algum valor e, ainda, como espaços de memória que armazenam valores. Sua função é o fornecimento de dados variáveis úteis e necessários a usuários e programas. Tais variáveis apresentam a seguinte forma:

```
1 NOME=VALOR
```

ou seja:

```
1 4LINUX=joatham
```

Esse tipo de variável que acabamos de definir é conhecida como **escala** e pode receber valores numéricos ou caracteres.

Para acessar o endereço de memória atribuído à variável `4LINUX`, em **shell**, devemos utilizar o operador `$` (cifrão) antes do nome da variável, ou seja, se desejarmos mostrar na tela o valor da variável `4LINUX` devemos imprimir o conteúdo armazenado no endereço de memória `$4LINUX`:

```
1 echo $4LINUX
```

Interessante observar o comando `echo`, que por sua vez é usado para imprimir algo na tela ou direcionar para um arquivo. Isso é bastante útil para automação.

Nesse caso na linha de comando o `echo` é útil para inspecionar variáveis de ambiente, que são parâmetros guardados em memória e que definem o ambiente em uso.

Deixa eu te mostrar como se faz.

Para imprimir algo na tela:

```
1 echo algo
```

Vamos definir a variável **comando** com o valor igual a `ls`:

```
1 comando=ls
```

Para verificarmos o valor da variável podemos digitar:

```
1 echo $comando
```

Ou

```
1 echo "$comando"
```

Para escrevermos na tela: **$comando**, digite:

```
1 echo $comando
```

E por fim para executarmos o valor da variável **comando**, digite:

```
1 echo $comando
```

Ou

```
1 echo $($comando)
```

Variáveis Locais e de Ambiente (globais)

Quando falamos em variáveis em **shell** temos que ter em mente a divisão entre variáveis locais e de ambiente (ou globais).

- **Variáveis locais**: são as variáveis disponíveis somente para o shell atual. Isso significa que sua visibilidade é restrita ao ambiente para o qual foram definidas, ou seja, elas não ficam disponíveis para o restante do sistema.
- **Variáveis globais (de ambiente)**: são as variáveis disponíveis tanto para o shell atual como para os subprocessos que as utilizam, de forma que sua visibilidade é aberta a outros ambientes além daqueles em que elas são definidas.

Vejamos algumas variáveis de ambiente na tabela abaixo:

Variável	Definição
HOME	Responsável por identificar o diretório do usuário.
HOSTTYPE	Faz referência à plataforma que está sendo utilizada.
SHELL	Permite identificar o shell que está sendo utilizado.
TERM	Define o tipo de terminal que está sendo utilizado.
USER	Predefine o nome da conta como variável global.
PATH	Determina quais diretórios devemos pesquisar e, ainda, qual sequência deverá ser seguida para encontrar um determinado comando.
PS1	Representa as informações exibidas no prompt.
PS2	Representa o prompt estendido.
MAIL	Esta variável informa como o correio eletrônico está definido.
LOGNAME	É um sinônimo da variável USER.
OSTYPE	Com a utilização desta variável, é possível definir o tipo de sistema operacional que está sendo utilizado.

Como definir variáveis

A única diferença técnica entre variáveis locais e de ambiente é a forma de sua definição.

Para definir uma variável local, basta atribuir um valor a um nome de variável.

Para definir uma variável de ambiente o procedimento adiciona o comando `export` antes da definição.

Abaixo mostramos exemplos de definição de variável local e de ambiente:

```
1 LOCAL="sem export na frente"
2 export GLOBAL="com export na frente"
```

Vejamos o seguinte exemplo:

```
LOCAL="Variável Local"
echo $LOCAL
Variável Local
export GLOBAL="Variável Global"
echo $GLOBAL
Variável Global
```

No exemplo que acabamos de ver, foi criada uma variável local chamada `LOCAL`, a qual estará disponível apenas a esta seção. Para visualizar o retorno da variável, foi utilizado o comando `echo`. Em seguida, por meio do comando `export`, foi criada uma variável global, que estará disponível para todas as seções do sistema. Por fim, para verificar o retorno da variável global criada, o comando `echo` também foi utilizado.

Uma vez definidas as **variáveis**, podemos visualizá-las utilizando os comandos `set` e `env` ou `printenv` para variáveis locais e de ambiente, respectivamente. Com isso, se tivéssemos definido as variáveis `LOCAL` e `GLOBAL` e executássemos o comando `set`, veríamos as definições de ambas. Mas, se executássemos o comando `env`, veríamos apenas a definição da variável `GLOBAL`

```
magica="abracadabra"
echo $magica
set
clear
env
```

Abra um terminal filho:

```
bash
```

Não há nada na variável, pois ela não foi exportada:

```
echo $magica
exit
```

Exporte a variável:

```
export magica
set
clear
```

```
4 env
```

Abra um terminal filho:

```
1 bash
```

Agora existe um valor para a variável:

```
1 echo $magica
```

Utilizamos o comando `unset` para excluir variáveis. Sua sintaxe é a seguinte:

```
1 unset <variavel>
```

Exclusão de variáveis

É importante lembrar que esse comando, quando usado para uma variável global, só tem efeito na sessão atual, de forma que a variável continuará acessível em outras sessões.

```
1 unset magica
2 echo $magica
```

Para ficar permanente para todos e funcionar em qualquer terminal deve-se colocarem um dos arquivos:

```
1 /etc/profile
2 /etc/environment
```

Para ficar permanente para o usuário e funcionar em qualquer terminal deve-se colocar em um dos arquivos:

```
1 ~/.bashrc
2 ~/.bash_profile
3 ~/.bash_login4
4 ~/.profile
```

Alterar o prompt de comando

Variáveis de ambiente (as globais) são muito s pois definem o comportamento da"shell" e de muitos outros programas"

Para que possamos alterar o prompt de comandos, devemos alterar os valores que estão armazenados dentro da variável `PS1`.

Na tabela a seguir, serão descritos alguns valores que podem ser utilizados em conjunto com a variável `PS1`:

Valor	Descrição
h	Faz referência ao nome do host.
w	Caminho completo do diretório corrente.
W	Refere-se ao diretório atual.
u	Exibe o nome do usuário.
t	Exibe a hora do sistema.
d	Refere-se à data do sistema.

**$*$ $*$ $|Especificaovalor$‘`para usuários comuns, e o valor#para o usuárioroot`‘.

No exemplo adiante, alteramos a variável `PS1` para que o prompt exiba o nome do usuário, o nome do host e a hora atual:

```
1 export PS1='[u@h-t]'
2 [root@mycomputer-16:55:59]
```

Volte a configuração anterior:

```
1 source /etc/profile
```

11

Arquivos de configuração do shell

Alguns arquivos são automaticamente lidos pelo shell no momento do login e do logout de um usuário ou ainda no momento da chamada do shell a partir de uma sessão que já tenha sido iniciada. Dependendo de como for iniciado o bash, arquivos diferentes serão executados de forma automática e, a partir desse conhecimento, o usuário pode programar a criação de variáveis, aliases, funções e outras personalizações do bash.

Caso o bash seja invocado durante um login, ele lerá os arquivos `/etc/profile`, `~/.bash_profile` , `~/.bash_login` e `~/.profile`, nessa ordem, caso eles existam. O primeiro aplica-se a todos os usuários do sistema, ao passo que os três últimos, ocultos por iniciarem com um ponto (.), estão no diretório `/home` do usuário em questão. Ao terminar a sessão, o bash executará automaticamente o arquivo `~/.bash_logout`, se ele existir.

Quando não se tratar de um shell invocado numa sessão de login, como é o caso de terminais em sessões X, os arquivos executados automaticamente, caso existam, serão o `/etc/bash.bashrce` `~/.bashrc`.

Vejamos separadamente alguns arquivos de configuração importantes.

Arquivo `/etc/profile`:

O arquivo `/etc/profile` é executado automaticamente no momento do login e lido antes dos arquivos de configurações pessoais do usuário. É responsável, ainda, por definir as variáveis de ambiente para os usuários em geral e armazenar os comandos a serem executados, quando o usuário efetuar o login no sistema. Sempre que o arquivo for carregado por meio de um shell que solicita login, ele procurará os arquivos na sequência descrita no início deste tópico e realizará a execução dos comandos nele contidos, caso eles existam.

Arquivo `/etc/environment`:

O arquivo `/etc/environment` é utilizado por sistemas Debian para definir variáveis de ambiente. Uma vez definidas, todas as variáveis são exportadas, automaticamente, na inicialização do sistema. Esse arquivo substitui o arquivo `/etc/profile` em algumas distribuições.

Arquivo `/.bashrc`:

Executado por shells invocados em sessões já iniciadas, e não de login, esse arquivo possui características semelhantes às do arquivo `~/.bash.profile`.

Utilização de aliases (apelidos)

Alguns comandos têm uma sintaxe indiscutivelmente grande e digitá-la inteira toda vez que o comando for necessário tende a se tornar uma tarefa inconveniente, especialmente quando o comando é usado diversas vezes com os mesmos argumentos e opções.

Uma forma bastante útil de agilizar a entrada de um comando que seja usado normalmente com os mesmos parâmetros é atribuir um alias, que provém da palavra *apelido* em inglês. Em linha de comando ou em um arquivo próprio para isso, atribui-se um termo pequeno e de digitação fácil para executar um determinado comando com parâmetros específicos toda vez que este alias for usado.

No exemplo a seguir, executamos propositalmente `l`, que é um comando inexistente. Em seguida, com o comando alias, especificamos que ao digitar `l` o comando a ser executado será igual a `ls --color -l -a`. Então, ao executarmos novamente o comando `l`, ele passa a responder como o comando `ls` e com os parâmetros `--color -l -a`**.

Isso será particularmente útil para administradores de sistema, por exemplo, que precisam executar comandos específicos diariamente, consultar processos em execução etc., de forma que seja possível automatizar o trabalho, ou pelo menos, torná-lo menos moroso.

```
aluno1:/# l
sh: l: command not found
aluno1:/# alias l="ls --color -l -a"
aluno1:/# l
total 92
drwxr-xr-x  21 root root  4096 Nov 15 22:28 .
drwxr-xr-x  21 root root  4096 Nov 15 22:28 ..
-rw-r--r--   1 root root     0 Nov 15 22:28 ?[01
drwxr-xr-x   2 root root  4096 Nov 14 14:35 bin
drwxr-xr-x   3 root root  4096 Nov 14 14:55 boot
lrwxrwxrwx   1 root root    11 Out 13 22:35 cdrom -> media/cdrom
drwxr-xr-x  14 root root  3500 Nov 15 20:17 dev
drwxr-xr-x 111 root root  4096 Nov 15 22:28 etc
drwxr-xr-x   8 root root  4096 Nov 15 18:31 home
lrwxrwxrwx   1 root root    28 Out 13 22:40 initrd.img -> boot/initrd.img-2.6.26-2-686
drwxr-xr-x  16 root root 12288 Nov 14 14:43 lib
drwx------   2 root root 16384 Out 13 22:35 lost+found
drwxr-xr-x   3 root root  4096 Nov 14 14:11 media
drwxr-xr-x   2 root root  4096 Ago 31  2009 mnt
drwxr-xr-x   2 root root  4096 Out 13 22:36 opt
dr-xr-xr-x 103 root root     0 Nov 15 20:16 proc
drwxr-xr-x  21 root root  4096 Nov 15 17:28 root
drwxr-xr-x   2 root root  4096 Nov 14 14:43 sbin
drwxr-xr-x   2 root root  4096 Set 16  2008 selinux
drwxr-xr-x   2 root root  4096 Out 13 22:36 srv
drwxr-xr-x  11 root root     0 Nov 15 20:16 sys
drwxrwxrwt   9 root root  4096 Nov 15 20:19 tmp
drwxr-xr-x  11 root root  4096 Out 20 23:15 usr
drwxr-xr-x  16 root root  4096 Nov 14 14:53 var
lrwxrwxrwx   1 root root    25 Out 13 22:40 vmlinuz -> boot/vmlinuz-2.6.26-2-686
aluno1:/#
```

Para remover o alias de um comando, utiliza-se o comando `unalias`, como mostra este exemplo da figura a seguir, que exclui o alias `l` criado no exemplo anterior:

```
aluno1:/# l
total 92
drwxr-xr-x  21 root root  4096 Nov 15 22:28 .
drwxr-xr-x  21 root root  4096 Nov 15 22:28 ..
-rw-r--r--   1 root root     0 Nov 15 22:28 ?[01
drwxr-xr-x   2 root root  4096 Nov 14 14:35 bin
drwxr-xr-x   3 root root  4096 Nov 14 14:55 boot
lrwxrwxrwx   1 root root    11 Out 13 22:35 cdrom -> media/cdrom
drwxr-xr-x  14 root root  3500 Nov 15 20:17 dev
drwxr-xr-x 111 root root  4096 Nov 15 22:28 etc
drwxr-xr-x   8 root root  4096 Nov 15 18:31 home
lrwxrwxrwx   1 root root    28 Out 13 22:40 initrd.img -> boot/initrd.img-2.6.26-2-686
drwxr-xr-x  16 root root 12288 Nov 14 14:43 lib
drwx------   2 root root 16384 Out 13 22:35 lost+found
drwxr-xr-x   3 root root  4096 Nov 14 14:11 media
drwxr-xr-x   2 root root  4096 Ago 31  2009 mnt
drwxr-xr-x   2 root root  4096 Out 13 22:36 opt
dr-xr-xr-x 103 root root     0 Nov 15 20:16 proc
drwxr-xr-x  21 root root  4096 Nov 15 17:28 root
drwxr-xr-x   2 root root  4096 Nov 14 14:43 sbin
drwxr-xr-x   2 root root  4096 Set 16  2008 selinux
drwxr-xr-x   2 root root  4096 Out 13 22:36 srv
drwxr-xr-x  11 root root     0 Nov 15 20:16 sys
drwxrwxrwt   9 root root  4096 Nov 15 20:19 tmp
drwxr-xr-x  11 root root  4096 Out 20 23:15 usr
drwxr-xr-x  16 root root  4096 Nov 14 14:53 var
lrwxrwxrwx   1 root root    25 Out 13 22:40 vmlinuz -> boot/vmlinuz-2.6.26-2-686
aluno1:/# unalias l
aluno1:/# l
sh: l: command not found
aluno1:/#
```

Arquivos para exibição de mensagens

Existem três arquivos relacionados à exibição de imagens para usuários, que não afetam a parte operacional do sistema. O primeiro deles é o `/etc/issue`, que exibe uma mensagem para um usuário antes do login no sistema. Caso queira que a mensagem apareça depois do login, o arquivo utilizado é o `/etc/motd`. Para visualizar mensagens exibidas em logins remotos, o arquivo a ser verificado é o `/etc/issue.net`.

Arquivo `.bash_history`:

O `.bash_history` é um arquivo responsável por armazenar o histórico de linhas de comando. Por padrão, até 500 comandos podem ser armazenados nesse arquivo, que pode ser verificado por comandos que exibem o seu conteúdo. Por meio das setas direcionais no teclado, para cima e para baixo, podemos verificar quais comandos já foram utilizados.

O arquivo `.bash_history`também armazena sua configuração em algumas variáveis. Vejamos quais são elas na tabela abaixo:

Variável	Descrição
HISTFILE	Armazena o nome do arquivo utilizado para armazenar históricos.
HISTSIZE	Armazena o número máximo de comandos que o arquivo suportará.
HISTFILESIZE	Armazena o número máximo de linhas que o arquivo do histórico de comandos suportará.

Histórico de comando

Do ponto de vista prático, esse recurso traz grande praticidade ao usuário, pois não é necessário redigitar diversas vezes um mesmo comando.

Para manter o histórico dos comandos utilizados, basta digitar `history` na linha de comando e então ENTER. Por meio das teclas direcionais para cima e para baixo do teclado, temos acesso ao histórico. Veja um exemplo da execução do comando `history`:

```
1 history
2
3 93 su -
4 94 lpsci
5 95 sudo -s
6 96 aptitude search chromium
7 97 history
```

No exemplo a seguir, utilizamos o comando `history` para exibir os últimos 10 comandos digitados:

```
1  history 10
2    493  reset
3    494  set | grep LOCAL
4    495  set | grep GLOBAL
5    496  echo $GLOBAL
6    497  set | grep LOCAL
7    498  unset LOCAL
8    499  set | grep LOCAL
9    500  sudo -s
10   501  df -h
11   502  history 10
```

Comando fc

FC significa **Find Command** ou **Fix Command** pois ele executa as duas tarefas, encontrar e corrigir comandos. Para listar os comandos já digitados, guardados no `history`, digite:

```
1 fc -l
```

Por padrão mostra os últimos 16 comandos. Para visualizar uma lista de comandosdo 2 ao 6 faça:

```
1 fc -l 2 6
```

Para visualizar os ultimos 20 comandos:

```
1 fc -l -20
```

Para visualizar todos os comandos desde o ultimo comecando com h:

```
1 fc -l h
```

12

Caminhos de Diretorios

Como vimos, há várias boas interfaces gráficas de usuário, ou GUIs, disponíveis para tornar os usuários mais produtivos.

Mas de vez em quando você pode ter que fazer alguma coisa a partir da linha de comando do Linux, e é importante saber onde você está na estrutura de arquivos do Linux.

Linux são hierárquicos, ou seja, há um diretório de nível superior chamado **Raiz**, que é identificado como apenas uma barra `/`. A **Raiz** tem sub-diretórios organizados sob ele, como `/home`,`/bin` e `/usr`.

Os chamados "caminhos" de diretório, são locais no sistema onde são armazenadas localizações físicas de arquivos, pastas, scripts e demais recursos do sistema e consistem em caminhos absolutos e caminhos relativos.

Caminho absoluto é o caminho completo de um arquivo ou subdiretório desde a raiz. Por exemplo: `/proc/cpuinfo`.

Onde, `cpuinfo` é um arquivo que está abaixo do diretório `proc` e o diretório `proc` está abaixo do diretório `/` (raiz).

É importante ressaltar que o **caminho absoluto** começará sempre com `/`. Desta forma, ao acessar um arquivo ou um diretório por meio do caminho absoluto não importa em qual diretório atual (corrente) você esteja.

Já o **caminho relativo** é usado quando não é indicado o diretório raiz para acessar um subdiretório ou um arquivo qualquer. Para que o caminho relativo funcione você precisa saber em qual diretório está localizado atualmente no sistema e ter uma boa noção de onde ficam

localizados os principais diretórios e arquivos. Abaixo seguem alguns atalhos para acessar determinados diretórios.

Caminhos de diretórios:

1. `.` – diretório corrente
2. `..` – diretório pai
3. `/` – diretório raiz
4. `\-`– diretório anterior

O comando `pwd` (*print name of current/working directory*) significa: imprima o nome do diretório corrente no qual estou trabalhando agora, e é usado para saber em qual diretório você está no momento.

```
1 cd /home
2 pwd
3 cd /tmp
4 pwd
5 cd ~
6 pwd
7 /bin/ls
```

As linhas 1, 3 e 7 dos comandos acima são absolutas. Observe que todos estes caminhos começam com barra (`/`). O que permite você executar o comando `ls` sem indicar o caminho absoluto é a existência da variável de ambiente `$PATH`, que armazena os diretórios nos quais o sistema deve buscar os executáveis. O diretório `/bin` está presente na variável de ambiente `$PATH` de todo usuário comum.

Para visualizar o conteúdo da variável de ambiente `$PATH` do usuário corrente execute o comando:

```
1 echo $PATH
2 /usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
```

Observe que cada diretório presente na variável de ambiente `$PATH` é separado por dois pontos (`:`). Por enquanto, ainda não é nosso objetivo entrar em detalhes sobre as variáveis de ambiente do sistema.

Caminho relativo:

```
1 pwd
2 /tmp
3 cd ..
```

```
4 pwd
5 /
6 cd -
7 pwd
8 /tmp
```

As linhas 3 e 6 dos comandos acima usam caminho relativo. O resultado destes comandos dependem de qual diretório você está no momento, por isso o nome caminho relativo.

Caminho relativo:

```
1 cd /bin
2 ./ls
3 echo "Certificação LPIC-1"
4 echo $PATH
5 /usr/share/sbin:/usr/local/bin:/usr/sbin:/usr/bin:
6 /sbin:/bin
```

Caminho relativo:

```
1 cd /usr/share/doc
2 pwd
3 /usr/share/doc
4 cd ../../
5 pwd
6 /usr
```

A linha 4 dos comandos acima usa um caminho relativo e sobe dois níveis na árvore de diretórios.

Acessando os diretórios

Vamos aprender agora alguns comandos essenciais para a nossa movimentação dentro do sistema.

O comando `cd` é utilizado para mudar o diretório atual de onde o usuário está. Ir para o diretório `home` do usuário logado:

```
1 cd
2 cd ~
```

Ir para o início da árvore de diretórios, ou seja, o diretório `/`:

```
1 cd /
```

Ir para um diretório específico:

```
1 cd /etc
```

Sobe um nível na árvore de diretórios:

```
1 cd ..
```

Retorna ao diretório anterior:

```
1 cd -
```

Entra em um diretório específico:

```
1 cd /usr/share/doc
```

Sobe 2 níveis da árvore de diretórios:

```
1 cd ../../
```

Comando ls

O comando `ls` é utilizado para listar o conteúdo dos diretórios. Se não for especificado nenhum diretório, ele irá mostrar o conteúdo daquele onde estamos nomomento. Lista o conteúdo do diretório atual:

```
1 ls
```

Para listar o conteúdo do diretório corrente com saída colorida faça:

```
1 ls --color
```

Para listar todo o conteúdo (inclusive os arquivos ocultados) do diretório corrente:

```
ls -a
```

O arquivos ocultados no Linux começam com um . (ponto).

O asterisco é um coringa que representa nenhum ou mais caracteres.

```
ls /dev/sd*
/dev/sda   /dev/sda2  /dev/sda4  /dev/sdb   /dev/sdb2
/dev/sda1  /dev/sda3  /dev/sda5  /dev/sdb1
```

```
ls /etc/host*
/etc/host.conf  /etc/hostname  /etc/hosts  /etc/hosts.allow  /etc/hosts.deny
```

O ponto de interrogação representa um único caractere ao contrário do asterisco. Observe a saída do comando abaixo. Temos dois pontos de interrogação que serão substituídos por dois caracteres:

```
ls /dev/s??
/dev/sda  /dev/sdb  /dev/sg0  /dev/sg1  /dev/sg2  /dev/sr0

/dev/shm:
pulse-shm-1570592226  pulse-shm-2671603769  pulse-shm-525798260
pulse-shm-1851567380  pulse-shm-3911461625  pulse-shm-665496239
pulse-shm-252084306   pulse-shm-4228683900  pulse-shm-882637142

/dev/snd:
by-path  controlC0  hwC0D2  pcmC0D0c  pcmC0D0p  pcmC0D2p  seq  timer
```

O comando abaixo mostrará todos arquivos que começam com `sd` e terminam com `a`, `b` ou `c`:

```
ls /dev/sd[abc]
/dev/sda  /dev/sdb
```

O comando abaixo mostrará todos os arquivos que começam com `sda`, mas que não é seguido por 0 ou 1:

```
ls /dev/sda[!01]
```

```
2 /dev/sda2  /dev/sda3  /dev/sda4  /dev/sda5
```

Atalhos de teclado utilizados na linha de comando

Ao digitar comandos no shell do sistema GNU/Linux, podemos fazer uso de atalhos de teclado para agilizar tarefas, tais como apagar e copiar caracteres, mover o cursor para o local desejado, abrir uma nova linha de comando, entre outras descritas a seguir:

- **HOME** ou **CTRL + A**: leva o cursor para o início da linha de comandos.
- **END** ou **CTRL + E**: leva o cursor para o fim da linha de comandos.
- **BACKSPACE**: apaga o caractere à esquerda do cursor.
- **CTRL + U**: apaga tudo o que estiver à esquerda do cursor, porém permite colar o conteúdo apagado com o atalho CTRL + Y.
- **DELETE**: apaga o caractere à direita do cursor.
- **CTRL + K**: apaga tudo o que estiver à direita do cursor, porém permite colar o conteúdo apagado com o atalho CTRL + Y.
- **CTRL + L**: funciona como o comando `clear`, limpa a tela e joga o cursor para a primeira linha. Com o atalho **SHIFT + PAGE UP**, ainda é possível visualizar o conteúdo.
- **CTRL + C**: abre uma nova linha de comando na posição do cursor.
- **CTRL + R**: busca algum caractere específico no último comando digitado.
- **CTRL + D**: assim como o comando `exit`, sai do shell.

13

Tópicos para revisão do capítulo

Atente para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes deste capítulo.

- Os sistemas GNU/Linux herdaram muitas características do padrão Unix, como a representação por arquivos de tudo que estiver no sistema, a característica case-sensitive, a necessidade de permissões de execução de arquivos para todos os usuários, a divisão do sistema em camadas, entre outras;
- Existem basicamente três tipos de usuários no Linux: o usuário `root` ou superusuário, que possui funções administrativas, os usuários comuns e os usuários de sistema;
- O shell é o mediador entre o usuário e o computador num sistema Unix-like, já que é ele o responsável por interpretar e executar os comandos enviados por um usuário;
- O shell usado no sistema operacional GNU é chamado `bash`, e apresenta mais funcionalidades e praticidades na hora de realizar tarefas na linha de comandos do que seu antecessor, o interpretador de comandos `sh`;
- Uma sessão é composta por um login efetuado com sucesso por um usuário previamente cadastrado no sistema;
- Iniciar, desligar e reiniciar o sistema adequadamente são processos que fazem parte dos primeiros comandos que devem ser compreendidos pelo usuário, já que sua execução inadequada pode gerar danos, como falhas no sistema de arquivos e perda de dados;
- Há diversos tipos de shell disponíveis para distribuições GNU/Linux. O bash é o tipo utilizado na maioria das distribuições, mas existem também outros, como `sh`, `csh` e `zsh`;
- Podemos definir variáveis como nomes que contêm algum valor e, ainda, como espaços de memória que armazenam valores. Sua função é o fornecimento de dados úteis e necessários a usuários e programas;
- Alguns arquivos são automaticamente lidos pelo shell no momento do login e do logout de um usuário ou ainda da inicialização do shell a partir de uma sessão que já tenha sido

iniciada. Com eles, o usuário pode programar a criação de variáveis, aliases (apelidos), funções e outras personalizações do bash;
- Para alterar o shell atual, utilizamos o comando `chsh` com a opção `-s` ou `--shell`, seguido do nome do shell que se deseja utilizar;
- É possível atribuir um alias para executar um determinado comando com parâmetros específicos toda vez que o termo atribuído for usado;
- Os arquivos `/etc/issue`, `/etc/motd` e `/etc/issue.net` operam na exibição de mensagens para usuários. Os dois primeiros exibem mensagens antes e depois do login, respectivamente, e o terceiro permite visualizar mensagens exibidas em logins remotos;
- O arquivo `.bash_history` é um arquivo responsável por armazenar o histórico de linhas de comando. Por padrão, até 500 comandos podem ser armazenados nesse arquivo.
- Por meio do comando `history`, é possível acessar uma lista com o histórico de comandos anteriormente utilizados.

14

Como obter ajuda

O ritmo de geração de conhecimento e informação tem sido vertiginoso nos últimos **sessenta anos**, especialmente na área tecnológica. Por isso é fundamental saber onde buscar informações para manter-se sempre atualizado. Neste capítulo, vamos aprender a consultar as documentações existentes e como buscar informações sobre o que precisamos.

O Sistema Operacional **GNU/Linux** possui uma vasta biblioteca de documentação. Antes de recorrermos a ajuda de outras pessoas, devemos lembrar que podemos ter a respostas que precisamos no próprio sistema, bem a nossa frente, ao teclar de um simples comando. Essa documentação em grande parte dos casos é de extrema qualidade.

O **GNU/Linux** cresceu porque a comunidade contribui para o sistema e sua documentação. Essa comunidade não tem medo ou receio de compartilhar informações e disponibiliza o que foi desenvolvido no próprio sistema. É muito importante reforçar que no Software Livre, as pessoas nunca ocultam seu **know-how**, ou seja, você pode perguntar a vontade, desde que saiba o que e onde perguntar.

A documentação do GNU/Linux pode ser vista também como fonte de conhecimento, onde pode-se aprender muito sobre cada um dos serviços e comandos disponíveis.

Essa ajuda é provida por meio dos manuais, as famosas **Man Pages**.

Abaixo vamos começar a nos familiarizar com a documentação existente e as formas nas quais ela é apresentada.

15

Formas de documentação

Existem diversas formas de se documentar um projeto, dentre elas temos os **Howto's**, os **manuais** e as **documentações**.

How-to's

Os **How-to's** (do inglês "Como Fazer"), são documentos que focam uma necessidade específica, como montar um "firewall", instalar uma "webcam", configurar placas de som, configurar um servidor web e muitos outros. Normalmente esses documentos são instalados junto com suas respectivas aplicações ou podem ter um pacote específico para a documentação daquela aplicação. Os **how-to's** também são conhecidos como **cook-books** - livro de receitas.

O diretório de **How-to's** do GNU/Linux é o `/usr/share/doc`. Se desejamos saber como configurar um **firewall**, podemos consultar os arquivos do diretório:

```
1 cd /usr/share/doc/iptables/
```

Muitas vezes o uso de **how-to's** ou **cook-book's**, não agrega um bom conhecimento, pois trata-se somente de uma lista de afazeres para chegar a um objetivo. Quando o software é atualizado, todo aquele conhecimento fica dependente de um novo **how-to**.

Manuais

Diferente dos **How-to's** os manuais não vão te mostrar um passo a passo ou mesmo te dar uma lista de afazeres. O principal objetivo do manual é te mostrar como as funcionalidades daquele software podem ser usadas. Com o manual o aprendizado para a utilização da ferramenta é facilitado, já que o mesmo possui alguns exemplos de usabilidade. Esses manuais podem ser encontrados através do comando **man**, o qual veremos ainda nesse capítulo, um pouco mais adiante.

Documentação

A palavra documentação é muito intensa. Quando falamos em documentar uma ferramenta, estamos na realidade abrangendo uma série de outros itens importantes, dentre eles os **How-to's** e os manuais. Com a documentação de um projeto é possível entender absolutamente tudo sobre o mesmo, ou seja, essa documentação deve mostrar todas as partes relacionadas ao projeto.

Podemos, por exemplo, citar a documentação de um projeto de rede, onde deve constar não só documentos como **how-to's** e manuais, mas sim todas as especificações dos componentes, bem como cabos, "switch's" e "routers" dentre outros detalhes muito importantes.

Como esse tipo de documentação é muito especifica, devemos consultar o site de cada projeto individualmente.

Existem diversos comandos de ajuda no GNU/Linux, vamos abordar cada um deles logo abaixo:

16

Comando help

O comando `help` provê ajuda para comandos internos do interpretador de comandos, ou seja, o comando `help` fornece ajuda rápida. Ele é muito útil para saber quais opções podem ser usadas com os comandos internos do interpretador de comandos (shell). Para visualizar uma ajuda rápida para todos os comandos internos do sistema, podemos fazer da seguinte forma:

```
1 help
```

Caso desejemos visualizar a ajuda rápida para somente um comando interno, usamos esta outra sintaxe:

```
1 help [comando]
```

> O comando `help` somente mostra a ajuda para comandos internos.

```
1 help type
```

O comando `type` mostra se cada nome de comando é um comando do UNIX, um comando interno, um alias, uma palavra-chave do shell ou uma função de shell definida.

Verifique o tipo do comando help que conheceremos a seguir:

```
1 help help
```

Para comandos externos, o `help` aparece como parâmetro. Por exemplo:

```
1 [comando] --help
```

Desse modo, caso desejemos visualizar uma ajuda rápida sobre um comando externo, devemos fazer da seguinte forma:

```
1 ls --help
```

O parâmetro `--help` pode ser utilizado em qualquer comando para ter uma consulta rápida dos parâmetros que ele pode nos oferecer. É importante entender que `--help` é na verdade um parâmetro individual de cada comando, logo se um comando não tiver esse parâmetro existem outros meios para se obter ajuda. Não se esqueça de estudar as diferenças entre comandos internos e externos.

17

Comando apropos

O comando `apropos` ajuda o usuário quando ele não se lembra do comando exato, mas conhece algumas palavras-chave relacionadas ao comando que definem seu uso ou funcionalidade. Ele pesquisa a página de manual do Linux com a ajuda da palavra-chave fornecida pelo usuário para encontrar o comando e suas funções.

Sintaxe:

```
apropos [OPÇÃO ..] PALAVRA-CHAVE ...
```

Situacao 1: suponhamos que você não saiba como compactar um arquivo, então você poderia digitar o seguinte comando no terminal e ele mostrará todos os comandos relacionados e sua breve descrição ou funcionalidade.

```
apropos compress
```

Depois de executar o comando acima, você observará uma série de comandos listados no terminal que tratam não apenas de como compactar um arquivo, mas também de expandir um arquivo compactado, pesquisar um arquivo compactado, comparar um arquivo compactado etc.

Situacao 2: o comando `apropos` também suporta várias palavras-chave se fornecidas como um argumento, ou seja, também podemos fornecer mais de uma palavra-chave para uma busca melhor. Assim, se duas palavras-chave forem fornecidas, o comando `apropos` exibirá toda a lista do comando que contém a primeira palavra-chave em sua descrição de página de manual

ou a segunda palavra-chave.

```
1 apropos email
```

Entrada 1 (com uma palavra-chave)

```
1 apropos email address
```

Entrada 2 (com várias palavras-chave)

Opções:

- **-d**: esta opção é usada para emitir mensagens de depuração. Quando esta opção é usada, o terminal retorna diretórios man, caminho global, diretório do caminho, avisos, etc. de cada comando que está relacionado à palavra-chave de pesquisa.
- **-v** : esta opção é usada para imprimir mensagens de aviso detalhadas.
- **-e**, **–exact** : esta opção é usada para pesquisar cada palavra-chave para correspondência exata. Se nenhuma opção for usada, o comando apropos retorna a lista de todos os comandos cuja descrição na descrição da página do manual corresponde à palavra-chave ou que estão de alguma forma relacionados à palavra-chave fornecida no argumento. No entanto, quando a opção `-e` é usada, o apropos retorna apenas o comando cuja descrição corresponde exatamente à palavra-chave.
- **-w**, **–wildcard** : esta opção é usada quando a (s) palavra (s) -chave contém curingas. `apropos` irá pesquisar independentemente o nome da página e a descrição que corresponde à (s) palavra (s) -chave.
- **-a** , **–and** : esta opção é usada quando queremos que todas as palavras-chave correspondam. Ele não retorna nada se qualquer uma das palavras-chave fornecidas não tiver correspondência na página do manual ou na descrição. Na entrada abaixo, duas palavras-chave foram fornecidas e apenas dois comandos são exibidos no resultado, pois há apenas um comando que contém ambas as palavras-chave.
- **-l** , **–long** : por padrão, a saída é cortada para a largura do terminal. Esta opção é útil quando não queremos que o resultado seja truncado.
- **-C** : esta opção é usada quando não queremos usar o padrão (`/ manpath`), mas o arquivo de configuração do usuário.
- **-L** : define o local para esta pesquisa.
- **-m**, **–systems** : esta opção usa páginas de manual de outros sistemas. Esta opção é útil quando queremos pesquisar a descrição da página do manual de outro sistema operacional acessível.
- **-M** , **–manpath** : define o caminho de pesquisa das páginas de manual para `PATH` em vez do `$MANPATH` padrão .

- **-s** , **–sections** , **–section** : Esta opção é usada quando queremos pesquisar apenas seções particulares (separadas por dois pontos) que são fornecidas no argumento.
- **-?** , **–Help** : esta opção exibe a lista de ajuda.
- **-V** , **–version** : usado para imprimir a versão do programa.
- **-r** , **–regex** : esta opção interpreta cada palavra-chave como um regex (expressão regular). A palavra-chave será comparada independentemente com o nome e a descrição da página.

E por fim, uma forma equivalente ao `apropos` é usar o comando `man` juntamente com a opção `-k`

```
1 man -k editor
```

18

Comando whatis

O comando `whatis` tem basicamente a mesma função do comando `apropos`, porém as buscas do comando `whatis` são mais específicas. O `apropos` busca as páginas de manuais e descrições de maneira mais genérica. Se digitarmos a palavra **passwd** ele nos trará tudo que tiver **passwd**, seja como nome ou parte do nome do manual ou na descrição. Já o `whatis` nos trará somente o manual com nome exato da palavra pesquisada.

A sintaxe utilizada no comando `whatis` é a seguinte:

```
1 whatis [comando]
```

Você sabe que tem um programa chamado `vim`, mas não sabe o que ele faz?

```
1 whatis vim
```

> Uma forma equivalente ao `whatis` é usar o comando `man` juntamente com a opção `-f`:

```
1 man -f vim
```

Para localizar as **man pages**, o comando `apropos` e `whatis` utilizam o mesmo banco de dados construído com o comando `catman` ou `makewhatis` (executado pelo administrador do sistema, **root**).

Para construir o banco de dados do comando `apropos` e `whatis` devemos executar o comando

abaixo:

Debian:

```
1 catman
```

CentOS:

```
1 makewhatis -v
```

19

Comando man

O comando `man` sem dúvidas é o comando mais usado para obtenção de documentação no Linux, ele é o responsável por trazer os manuais mais completos sobre determinado comando, arquivo de configuração, bibliotecas, entre outros nos quais estamos trabalhando.

Os manuais do sistema são divididos nos seguintes níveis:

- **man 1** -> Programas e executáveis disponíveis ao usuário;
- **man 2** -> Rotinas de sistema Unix e C;
- **man 3** -> Rotinas de bibliotecas da linguagem C;
- **man 4** -> Arquivos especiais (dispositivos em `/dev`);
- **man 5** -> Arquivos de configuração e convenções;
- **man 6** -> Games;
- **man 7** -> Diversos (macros textuais, por exemplo, `regex`);
- **man 8** -> Comandos administrativos;
- **man 9** -> Rotinas internas do kernel.

> É comum o exame da LPI cobrar mais dos níveis 1, 5 e 8 dos manuais! Então lembre se de estudar binários, arquivos de configuração e comandos administrativos.

Sintaxe do comando `man`:

```
1 man [ comando ]
```

ou

```
man [ seção ] [ comando ]
```

> Uma curiosidade: as informações sobre as seções do comando `man` podem ser encontradas em seu próprio manual, digitando o comando `man man`.

Se for necessário visualizar o manual do comando `passwd`, podemos fazer da seguinte forma:

```
man passwd
```

Para navegar pelo manual, o comando `man` abre um arquivo que está compactado na pasta `/usr/share/man/man1` para o `passwd`. Outros níveis de manuais, dependem do comando ou arquivo. O `passwd` é conhecido no sistema **GNU/Linux** como um comando que adiciona ou modifica a senha do usuário e, também, como o arquivo de usuários do sistema `/etc/passwd`).

Veremos agora o manual do arquivo de usuários `passwd`:

```
man 5 passwd
```

Podemos consultar quais manuais estão disponíveis dentro do próprio diretório do `man`:

```
ls /usr/share/man/
```

Dentro desse diretório é possível ver todas as divisões dos manuais: os níveis, os idiomas e mais. Todos os níveis de manuais possuem sua determinada introdução que pode ser vista com o comando:

```
man <nivel> intro
```

Podemos ver que para visualizar o manual do arquivo de usuário `passwd` precisamos informar em qual nível de manual ele se encontra, pois já existe um `passwd` no nível 1, que é o comando, então ele aparece primeiro quando digitamos `man passwd` sem indicar o nível.

Esse manual do arquivo `passwd` está compactado na pasta **/usr/share/man/man5**.

20

Comando info

As **info pages** são como as páginas de manuais, porém são utilizadas com navegação entre as páginas. Elas são acessadas pelo comando `info`. Este é útil quando já sabemos o nome do comando e só queremos saber qual sua respectiva função.

A navegação nas **info pages** é feita através de nomes marcados com um **(*) **(hipertextos) que, ao pressionarmos** Enter**, nos leva até a seção correspondente, e** Backspace** volta à página anterior. Algo parecido com a navegação na Internet.

Podemos também navegar pelas páginas com as teclas: * **n (next/próximo);** * **p (previous/anterior);** * **u (up/sobe um nível)**.

Para sair do comando `info`, basta pressionar a tecla `q`.

Se for necessário exibir a lista de todos os manuais de comandos/programas disponíveis, execute o comando abaixo sem nenhum argumento. Assim:

```
1 info
```

Para exibir as informações somente de um determinado comando, usaremos a seguinte sintaxe:

```
1 info [comando]
```

Visualizar informações do comando vim:

```
1 info vim
```

Alternativas para consulta

Para obter uma melhor visualização, duas ferramentas de documentação foram desenvolvidas:

- **yelp** -> Ferramenta gráfica para visualização de manuais de aplicativos gráficos do GNOME; (fornecido pelo pacote yelp);
- **xman** -> "Front-end" para o comando `man`, que facilita a consulta das **man pages**; (fornecido pelo pacote `x11-apps`).

21

Comando whereis

O comando `whereis` é utilizado para mostrar a localização do binário do comando, do arquivo de configuração (caso exista), e a localização das páginas de manuais do determinado comando ou arquivo.

Se compararmos o comando `whereis` com o comando `find`, eles parecerão semelhantes entre si, pois ambos podem ser usados para os mesmos fins, mas o comando `whereis` produz o resultado com mais precisão, consumindo menos tempo comparativamente.

- **Exemplo 1:** digamos que queremos encontrar a localização do comando `apropos` e, em seguida, precisamos executar o seguinte comando no terminal:

```
1 whereis apropos
```

- **Exemplo 2:** para encontrar a localização do comando `lshw`.

```
1 whereis lshw
```

Opcoes:

- **-b**: esta opção é usada quando queremos apenas pesquisar binários.

> **Exemplo:** para localizar o binário de um comando do Linux, digamos `gunzip`.

```
1 whereis -b gunzio
```

- **-m**: esta opção é usada quando queremos apenas pesquisar por seções manuais.

> **Exemplo:** Para localizar a página de manual do comando falso.

```
1 whereis -m false
```

- **-s**: esta opção é usada quando queremos apenas pesquisar fontes.
- **-u**: esta opção pesquisa entradas incomuns. Um arquivo de origem ou um arquivo binário é considerado incomum se não tiver nenhuma existência no sistema de acordo com `[-bmsu]` descrito junto com **–u** . Assim, `whereis -m -u *`pede os arquivos no diretório atual que possuem entradas incomuns.

> **Exemplo**: Para exibir os arquivos do diretório atual que não possuem arquivo de documentação.

```
1 whereis -m -u *
```

- **-B**: esta opção é usada para alterar ou limitar os locais onde o `whereis` procura por binários.

> **Exemplo**: para localizar o binário de lesspipe no caminho, `/bin`.

```
1 whereis -B /bin -f lesspipe
```

- **-M**: esta opção é usada para alterar ou limitar os locais onde o `whereis` procura por seções manuais.

> **Exemplo**: para verificar a página de manual de introdução que está apenas em um local específico, ou seja, `/usr/share/man/man1`.

```
1 whereis -M /usr/share/man/man1 -f intro
```

- **-S**: esta opção é usada para alterar ou de outra forma limitar os locais onde whereis procura pelas fontes.

> **Exemplo**: Para localizar todos os arquivos em `/usr/bin` que não estão documentados em `/usr/man/man1` com fonte em `/usr/src`

```
1 whereis -u -M /usr/share/man/man1 -S /usr/src -f *
```

- **-f**: esta opção simplesmente termina a última lista de diretórios e sinaliza o início dos nomes dos arquivos. Deve ser usado quando qualquer uma das opções `-B` , `-M` ou `-S` for usada.
- **-V**: exibe informações sobre a versão e sai.
- **-h** : exibe esta ajuda e sai.

22

Comando which

O comando `which` é bem semelhante ao comando `whereis`, entretanto este só mostra a localização do binário do comando.

Para visualizar a localização do binário do comando, utilizamos a seguinte sintaxe:

```
1 which <comando>
```

Localização do binário do comando `vim`:

```
1 which vim
```

Tópicos para revisão do capítulo

Atente-se para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes deste capítulo.

- O comando `help` somente mostra a ajuda para comandos internos.
- `apropos` é um comando dos sistemas operacionais unix-like que mostra informações sobre um assunto a partir de um banco de dados.
- `apropos` usa o mesmo banco do comando `whatis`.
- Uma forma equivalente ao `whatis` é usar o comando `man` juntamente com a opção **-f**.
- É comum o exame cobrar mais dos níveis 1, 5 e 8 dos manuais! Então lembre-se de

estudar binários, arquivos de configuração e comandos administrativos.

- Nos sistemas operacionais do tipo Unix, o comando `info` lê a documentação armazenada no formato info, desenvolvido pela **Free Software Foundation**.
- O comando `whereis` permite localizar arquivos binário, fonte e também as páginas de manual de comandos especificados no Linux.
- O comando `man` fornece ajuda sobre o funcionamento de comandos com sua respectiva sintaxe e opções;
- O comando `which` localiza o arquivo executável associado a um determinado comando

23

FHS, Hierarquia dos Diretórios

Introdução teórica

Quem já teve algum contato com o **GNU/Linux**, mesmo que superficial, deve ter percebido a presença de vários diretórios (pastas) no sistema. Entretanto, eles estão organizados seguindo o padrão `POSIX`, com o qual você pode não estar muito familiarizado. Neste capítulo, vamos conhecer a organização, e explorar a estrutura de diretórios de um sistema **GNU/Linux**.

Desde que o **GNU/Linux** foi criado, muito se tem feito para seguir um padrão em relação à estrutura de diretórios. O primeiro esforço para padronização de sistemas de arquivos para o GNU/Linux foi o **FSSTND - Filesystem Standard**, lançado no ano de 1994.

Cada diretório do sistema tem seus respectivos arquivos que são armazenados conforme regras definidas pela **FHS - Filesystem Hierarchy Standard** ou **Hierarquia Padrão do Sistema de Arquivos**, que define que tipo de arquivo deve ser guardado em cada diretório. Isso é muito importante, pois o padrão ajuda a manter compatibilidade entre as distribuições existentes no mercado, permitindo que qualquer software escrito para o **GNU/Linux** seja executado em qualquer distribuição desenvolvida de acordo com os padrões **FHS**.

Atualmente, o **FHS** está na sua versão 2.3, e é mantido pelo **Free Standard Group**, uma organização sem fins lucrativos formada por grandes empresas como HP, IBM, RedHat e Dell.

> É vital entender bem sobre a **FHS** para prova, é através dela que nós devemos fazer nossas atividades com o **GNU/Linux** em nosso dia-a-dia.

Estrutura de Diretórios GNU/Linux

A estrutura de diretórios também é conhecida como "Árvore de Diretórios" porque tem a forma de uma árvore. Mas, antes de estudarmos a estrutura de diretórios, temos que entender o que são diretórios.

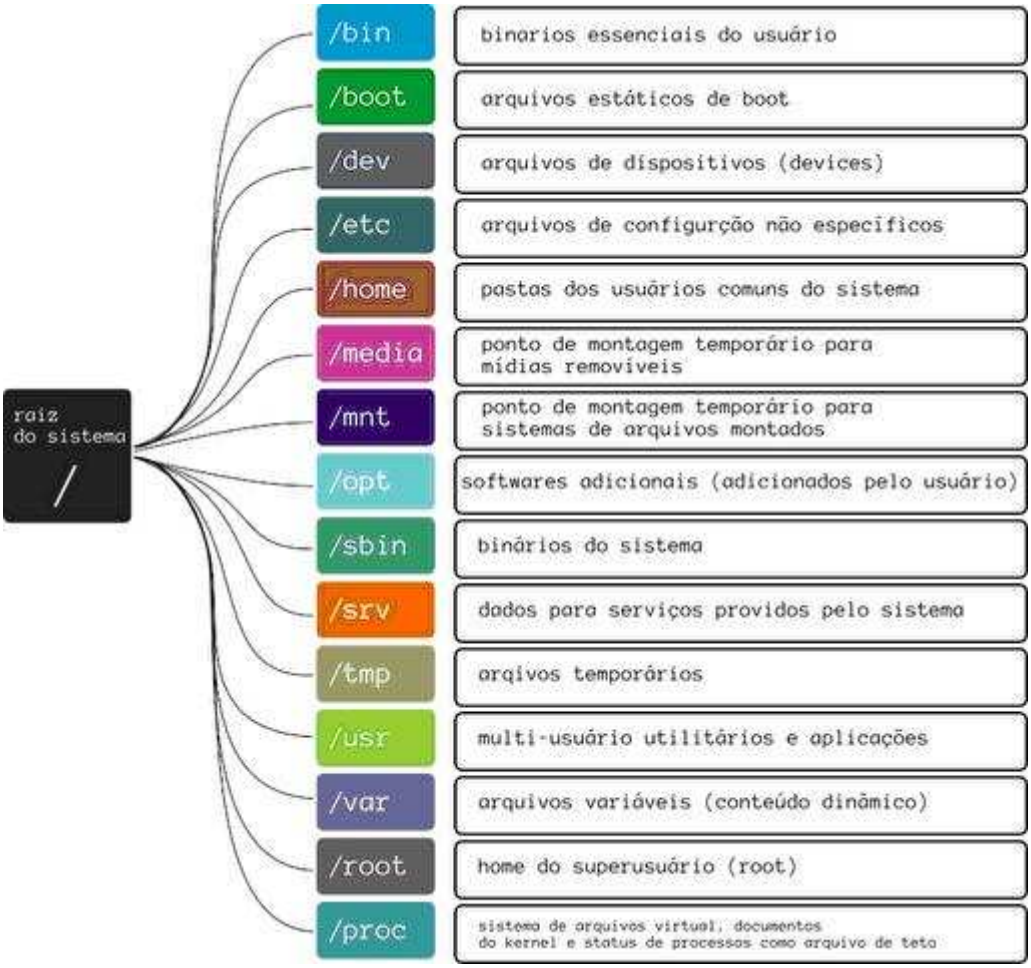
Um diretório é o local onde os arquivos são guardados no sistema. O objetivo é organizar os diferentes arquivos e programas. Pense nos diretórios como sendo as gavetas de um armário. Cada gaveta guarda, normalmente, um tipo diferente de roupa, enquanto cada diretório guarda um certo tipo específico de arquivo.

O arquivo pode ser um texto, uma imagem, planilha, etc. Os arquivos devem ser identificados por nomes para que sejam localizados por quem deseja utilizá-los.

Um detalhe importante a ser observado é que o **GNU/Linux** segue o padrão `POSIX` que é **case sensitive**, isto é, ele diferencia letras maiúsculas e minúsculas nos arquivos e diretórios.

Sendo assim, um arquivo chamado **Joatham** é diferente de um outro chamado **JOATHAM** e diferente de um terceiro, chamado **joatham**. Inteligente isso, não é?

A árvore de diretórios do GNU/Linux tem a seguinte estrutura:



A seguir segue uma listagem da estrutura padrão dos diretórios e o que contém em cada um deles.

- `/bin`: comandos (binários) essenciais acessíveis a qualquer usuário;
- `/sbin`: comandos (binários) essenciais administrativos;
- `/boot`: arquivos estáticos do gerenciador de inicialização e imagem do kernel;
- `/dev`: arquivos dispositivos (discos rígidos, placa de som, placa de vídeo, mouse, etc., exceto placa de rede);
- `/etc`: arquivos de configuração de sistema e de programas em geral;
- `/lib`: bibliotecas essenciais do sistema;
- `/media`: ponto de montagem para dispositivos removíveis;
- `/mnt`: ponto de montagem para sistemas de arquivos temporários;
- `/opt`: pacotes de software adicionais (proprietários);
- `/srv`: dados providos por serviços de rede do sistema;
- `/var`: dados variáveis (por exemplo: arquivos de log, cache, spool de impressão etc.);
- `/home`: diretórios pessoais dos usuários;
- `/root`: diretório pessoal do usuário `root`;
- `/usr`: hierarquia secundária. É chamado de hierarquia secundária pois é o maior diretório

do sistema após o diretório raiz. Todos os subdiretórios em localizados dentro do diretório `/usr` não são essenciais e tem uma estrutura muito parecida com `/` do sistema:

- `/usr/sbin`: binários administrativos não essenciais;
- `/usr/bin`: binários não essenciais;
- `/usr/lib`: bibliotecas não essenciais ao sistema;
- `/usr/share/man`: documentações (manuais);
- `/usr/src`: códigos-fonte, por exemplo: do kernel
- `/proc`: armazena informações dinâmicas sobre o sistema operacional, como, processos, informações sobre entrada e saída, interrupções, modelo do processador, versão do kernel, estatísticas sobre quanto tempo a máquina está ligada, quais sistemas de arquivos suportados etc. Este diretório não existe no disco rígido e sim na memória RAM;
- `/sys`: sistema de arquivos virtual como o `/proc` mas que armazena e permite modificações nos dispositivos conectados ao sistema;
- `/lost+found`: armazena arquivos recuperados pelo sistema;
- `/run`: informações sobre o sistema em execução desde o último boot, usuários logados e processos servidores em execução;
- `/tmp`: arquivos temporários que são limpos durante a inicialização;

Diretório /

```
1 ls --color /
```

> A opção `--color` do comando `ls` serve para deixar colorido a listagem, ex: * Azul -> Diretório. * Branco -> Arquivo regular. * Verde -> Arquivo executável. * Azul claro -> Link simbólico. * Vermelho -> Arquivo compactado. * Rosa -> Imagem.

Este é o principal diretório do **GNU/Linux**, e é representado por uma `/` (barra). É no diretório raiz que ficam todos os demais diretórios do sistema. Estes diretórios, que vamos conhecer agora, são chamados de **subdiretórios** pois estão dentro do diretório `/`.

Diretório /bin

```
1 ls /bin
```

O diretório `/bin` guarda os comandos essenciais para o funcionamento do sistema.

Esse é um diretório público, sendo assim, os comandos que estão nele podem ser utilizados por qualquer usuário do sistema. Entre os comandos, estão: * `/bin/ls`; * `/bin/cp`; * `/bin/mkdir`; * `/bin/cat`;

Qualquer usuário pode executar estes comandos:

```
1
2 (joathamkali)-[~]
3 $ /bin/ls /boot/grub
4
5 (rootkali)-[/home/joatham]
6 # /bin/ls /boot/grub
```

Diretório /boot

```
1 ls /boot
```

No diretório `/boot` estão os arquivos estáticos necessários à inicialização do sistema, e o gerenciador de **boot**. O gerenciador de **boot** é um programa que permite escolher e carregar o sistema operacional que será iniciado.

Diretório /dev

```
1 ls /dev
```

No diretório `/dev` ficam todos os arquivos de dispositivos. O **GNU/Linux** faz a comunicação com os periféricos por meio de **links** especiais que ficam armazenados nesse diretório, facilitando assim o acesso aos mesmos.

Para verificar que seu mouse é reconhecido como um arquivo, tente olhar o conteúdo do arquivo `/dev/input/mice`:

```
1 cat /dev/input/mice
```

Repare que os dados são binários e não é possível ler o arquivo com o comando `cat`. Caso seu terminal fique com caracteres estranhos utilize o comando `reset` para resetar o shell:

```
1 reset
```

Para visualizar o conteúdo do arquivo `/dev/input/mice` execute o comando `od` que é utilizado para visualizar o conteúdo de um arquivo nos formatos: **hexadecimal**, **octal**, **ASCII** e nome dos caracteres. Este comando pode ser útil para um programador que deseja criar um programa conforme o movimento do mouse.

```
1 od /dev/input/mice
```

Observe o conteúdo do seu HD:

```
1 hexdump /dev/sda
```

O comando `hexdump` tambem é utilizado para visualizar o conteúdo de um arquivo nos formatos: **hexadecimal**, **octal**, **decimal**, **ASCII**.

Diretório /etc

```
1 ls /etc
```

No diretório `/etc` estão os arquivos de configuração do sistema. Nesse diretório vamos encontrar vários arquivos de configuração, tais como: **scripts** de inicialização do sistema, tabela do sistema de arquivos, configuração padrão para **logins** dos usuários, etc.

```
1 cat /etc/passwd
```

Vamos pegar uma linha de exemplo:

```
1 joatham:x:1000:1000:joatham:/home/joatham:/bin/bash
```

Vamos dividir esta linha em **campos**, onde cada campo é separado por : (dois pontos), então:

Campo	Significado
joatham	Login do Usuario
x	Aqui diz que a senha esta no arquivo `/etc/shadow`. Se estivesse *, a conta estaria desabilitada e se estivesse sem nada (::), a conta nao teria senha.
1000	**UID (User IDentification)**, o numero de identificacao do usuario.

Campo	Significado
1000	**GID (Group IDentification)**, o numero de identificacao do grupo do usuario.
joatham	**GECOS** Comentarios do usuario, como nome, telefone e etc.
/home/joatham	O diretorio **HOME** do usuario.
/bin/bash	Shell do usuario, ou seja, o programa que ira interpretar os comandos que o usuario executar.

Vamos conhecer o arquivo `/etc/shadow`:

```
1 more /etc/shadow
```

O comando `more` assim como o `cat` serve para ver o conteúdo de um arquivo que é, geralmente, texto. A diferença entre o `more` e o `cat` é que o `more` faz uma pausa a cada tela cheia exibindo uma mensagem `--More`, dando uma oportunidade ao usuário ler a tela.

Aperte **enter** para ir para a próxima linha ou espaço para ir para a próxima página e para sair digite **q**.

Uma alternativa ao uso do comando `more` seria o uso do comando `less`, que implementa as mesmas funcionalidades que `more` e mais algumas, como a possibilidade de rolar a tela para cima e para o lado quando o texto ocupa mais de **oitenta colunas**.

A utilização dos comandos `less` e `more` se faz de maneira semelhante.

```
1 less /etc/shadow
```

Vamos pegar uma linha de exemplo:

```
1 joatham:$1$Tcnt$Eisi0J9Wh3fCEsz1:11983:0:99999:7:30::
```

Este arquivo possui as senhas criptografadas dos usuários do sistema. Existe uma entada no arquivo para cada usuário do sistema com os seguintes campos:

Campo	Significado
joatham	Login do Usuario.
1Tcnt$Eisi0J9Wh3ACEsn1	A senha criptografada.
11983	O numero de dias que existem de 01/01/1970 ate a data da ultima modificacao da senha.
0	Numero de dias antes do sistema permitir uma nova modificacao na senha.
99999	Numero maximo de dias que o usuario pode ficar com uma mesma senha
7	Numero de dias, antes da expiracao da senha, quando o usuario e informado da necessidade de alterar a senha.
30	Numero de dias apos a expiracao da senha, quando a conta passa a ser considerada inativa (o valor -1 significa que a conta fica inativa no mesmo dia da data de expiracao).
Numero de dias que existem de 01/01/1970 ate a expiracao da senha (o valor -1 significa que nao ha data de expiracao).	
campo reservado para uso futuro	

Apenas o usuário `root` **(administrador do sistema)** tem permissão para acessar o arquivo `/etc/shadow`.

O comando `pwconv` é usado para criar o arquivo shadow a partir do arquivo `/etc/passwd`, enquanto o comando `pwunconv` executa a operação inversa. Execute:

```
1 pwunconv
```

Verifique que não existe mais o arquivo `/etc/shadow`:

```
1 cat /etc/shadow
```

Verifique que as senhas criptografadas estão agora no arquivo `/etc/passwd` através do comando `getent`:

```
1 getent passwd
```

Para voltar as senhas criptografadas, execute:

```
1 pwconv
```

Agora as senhas estão protegidas novamente!! Antigamente estes comandos eram utilizados para sistemas que não vinham com as senhas protegidas no `/etc/shadow` por padrão, hoje em dia praticamente todas as distribuições trazem o arquivo como padrão, então utilizamos o comando para execução de scripts para facilitar a captura de senhas, como por exemplo a migração de um servidor de e-mail, onde queremos manter a senha antiga do usuário

Diretório /lib

```
1 ls /lib
```

No diretório `/lib` estão as bibliotecas compartilhadas e módulos do kernel. As bibliotecas são funções que podem ser utilizadas por vários programas.

Cada kernel têm seus próprios módulos, que ficam em: `/lib/modules/<versão do kernel>/kernel` Separados por tipos em subdiretórios.

Para saber sua versão do kernel execute:

```
1 uname -r
```

Para visualizar os tipos de módulos:

```
1 ls /lib/modules/$(uname -r)/kernel
```

Diretório /media

```
1 ls /media
```

Ponto de montagem para dispositivos removíveis, tais como:

- hd
- cd
- dvd
- pendrive
- cameras

Diretório /mnt

```
1 ls /mnt
```

Este diretório é utilizado para montagem temporária de sistemas de arquivos, tais como compartilhamentos de arquivos entre **Windows e GNU/Linux**, **GNU/Linux e GNU/Linux**, etc.

Diretório /opt

```
1 ls /opt
```

Normalmente, é utilizado por programas proprietários ou que não fazem parte oficialmente da distribuição.

Diretório /sbin

```
1 ls /sbin
```

O diretório `/sbin` guarda os comandos utilizados para inicializar, reparar, restaurar e/ou recuperar o sistema. Isso quer dizer que esse diretório também contém comandos essenciais, mas os mesmos são utilizados apenas pelo usuário administrador **root**.

Entre os comandos estão:

- `hatl`
- `ifconfig`
- `init`
- `iptables`

Os usuários comuns não podem executar comandos do `/sbin` que alterem o sistema, apenas alguns para visualização.

Exemplo: Visualizar IP configurado na placa **eth0**:

```
1
2 (joathamkali)-[~]
3 $ /sbin/ifconfig eth0
```

Criar uma alias da interface **eth0**

```
1
2 (joathamkali)-[~]
3 $ /sbin/ifconfig eth0:1 10.10.10.10
```

Diretório /srv

```
1 ls /srv
```

Diretório para dados de serviços fornecidos pelo sistema, cuja aplicação é de alcance geral, ou seja, os dados não são específicos de um usuário. Por exemplo:

- `/srv/www` (servidor web)
- `/srv/ftp` (servidor ftp)

Diretório /tmp

```
1 ls /tmp
```

Diretório para armazenamento de arquivos temporários. É utilizado principalmente para guardar pequenas informações que precisam estar em algum lugar até que a operação seja completada, como é o caso de um **download**.

Enquanto não for concluído, o arquivo fica registrado em `/tmp`, e, assim que é finalizado, é encaminhado para o local correto.

Diretório /usr

```
1 ls /usr
```

O diretório `/usr` contém programas que não são essenciais ao sistema e que seguem o padrão **GNU/Linux**, como, por exemplo, navegadores, gerenciadores de janelas, etc.

Diretório /var

```
1 ls /var
```

O diretório `/var` contém arquivos de dados variáveis. Por padrão, os programas que geram arquivos de registro para consulta, mais conhecidos como **logs**, ficam armazenados nesse diretório. Além do **log**, os arquivos que estão aguardando em filas, também ficam localizados em `/var/spool`.

Os principais arquivos que se utilizam do diretório `/var` são:

- Mensagens de e-mail
- Arquivos a serem impressos

```
1 ls /var/spool
```

- Arquivos de logs

```
1 ls /var/log
```

Diretório /proc

```
1 ls /proc
```

O `/proc` é um diretório virtual, mantido pelo kernel, onde encontramos a configuração atual do sistema, dados estatísticos, dispositivos já montados, interrupções, endereços e estados das

portas físicas, dados sobre as redes, etc.

Utilize os paginadores `more` ou `less` para visualizar alguns arquivos:

```
1 more /proc/cpuinfo
```

Diretório /sys

```
1 ls /sys
```

Pode-se dizer que esse diretório é um primo do diretório `/proc`. Dentro do diretório `/sys` podemos encontrar o quase o mesmo conteúdo do `/proc`, mas de uma forma bem mais organizada para nós administradores.

Diretórios /home e /root

```
1 ls /home /root
```

Os diretórios `/root` e `/home` podem estar disponíveis no sistema, mas não precisam obrigatoriamente possuir este nome. Por exemplo, o diretório `/home` poderia se chamar `/casa`, que não causaria nenhum impacto na estrutura do sistema.

O `/home` contém os diretórios pessoais dos usuários cadastrados no sistema.

O `/root` é o diretório pessoal do super usuário `root`.

Como sabemos o **root** é o administrador do sistema, e pode alterar as configurações do sistema, configurar interfaces de rede, manipular usuários e grupos, alterar a prioridade dos processos, entre outras.

Dica: Utilize uma conta de usuário normal em vez da conta `root` para operar seu sistema.

> Uma razão para evitar usar privilégios `root` regularmente, é a facilidade de se cometer danos irreparáveis; além do que, você pode ser enganado e rodar um programa **Malware** (programa que obtém poderes do super usuário) comprometendo a segurança do seu sistema sem que você saiba.

24

Aprendendo Comandos do GNU/Linux

Comandos são instruções passadas ao computador para executar uma determinada tarefa. No mundo **NIX (GNU/Linux,Unix)**, o conceito de comandos é diferente do padrão **MS-DOS**. Um comando é qualquer arquivo executável, que pode ser ou não criado pelo usuário.

Uma das tantas vantagens do **GNU/Linux** é a variedade de comandos que ele oferece, afinal, para quem conhece comandos, a administração do sistema acaba se tornando um processo mais rápido.

O **Shell** é o responsável pela interação entre o usuário e o sistema operacional, interpretando os comandos.

É no **Shell** que os comandos são executados.

Extraindo mais do comando ls

O comando `ls` possui muitos parâmetros, veremos aqui as opções mais utilizadas. A primeira delas é o `-l` que lista os arquivos ou diretórios de uma forma bem detalhada (quem criou, data de criação, tamanho, dono e grupo ao qual cada um pertence):

```
1 ls -l /
2 crw-r--r--  1 root root     10, 235 Mar  3 11:54 autofs
```

```
3 drwxr-xr-x  2 root root         140 Mar  3 11:54 block
4 lrwxrwxrwx  1 root root           3 Mar  3 11:54 cdrom -> sr0
5 brw-rw----  1 root disk      8,   0 Mar  3 11:54 sda
6 -rw-r--r--  1 root root       501 Aug  6  2019 updatedb.conf
```

Veja que a saída desse comando é bem detalhada. Falando sobre os campos, para o primeiro caractere temos algumas opções:

```
1 d --> Indica que se trata de um diretório.
2 l --> Indica que se trata de um **link** ( como se fosse um atalho também vamos
      falar sobre ele depois).
3 - --> Hífen, indica que se trata de um arquivo regular.
4 c --> Indica que o arquivo é um dispositivo de caractere (sem buffer).
5 b --> Indica que o arquivo é um dispositivo de bloco (com buffer).
6 u --> "Sinônimo para o tipo c" indica que o arquivo é um dispositivo de
      caractere (sem buffer).
7 s --> Indica que o arquivo é um socket.
8 p -> Indica que o arquivo é um fifo, named pipe.
```

FIFO - Sigla para **First In, First Out**, que em inglês significa primeiro a entrar, primeiro a sair. São amplamente utilizados para implementar filas de espera. Os elementos vão sendo colocados no final da fila e retirados por ordem de chegada. Pipes `|` são um exemplo de implementação de FIFO.

Buffer - É uma região de memória temporária, usada para escrita e leitura de dados. Normalmente, os buffers são utilizados quando existe uma diferença entre a taxa em que os dados são recebidos e a taxa em que eles podem ser processados.

Socket - É um meio de comunicação por software entre um computador e outro. É uma combinação de um endereço IP, um protocolo e um número de porta do protocolo.

O campo **rw-r–r–** lista as permissões, enquanto os campos **root** indicam quem é o usuário e grupo dono desse diretório que, no nosso caso, é o administrador do sistema, o usuário **root**. O número antes do dono indica o número de **hard links**, um assunto abordado apenas em cursos mais avançados.

O campo **501** indica o tamanho do arquivo, e o campo **Mar 3 11:54** informa a data e hora em que o diretório foi criado. Finalmente, no último campo temos o nome do arquivo ou diretório listado, que, no nosso exemplo, é o **updatedb.conf**.

Com relação aos diretórios, é importante ressaltar que o tamanho mostrado não corresponde ao espaço ocupado pelo diretório e seus arquivos e subdiretórios. Esse espaço é aquele ocupado pela entrada no sistema de arquivos que corresponde ao diretório. A opção `-a` lista todos arquivos, inclusive os ocultos:

```
1 ls -a / root
2 .. aptitude . bashrc . profile
```

Veja que, da saída do comando anterior, alguns arquivos são iniciados por . **(ponto)**. Esses arquivos são ocultos. No Linux, arquivos e diretórios ocultos são iniciados por um . **(ponto)**. Listar arquivos de forma recursiva, ou seja, listar também os subdiretórios que estão dentro do diretório `/`:

```
1 ls -R /
```

Como listar os arquivos que terminam com a palavra **.conf** dentro do diretório `/etc`?

```
1 ls /etc/*.conf
```

Como buscar no diretório raiz `/` todos os diretórios que terminem com a letra **n**?

```
1 ls -ld /*n
```

Criar arquivo

Para criar um arquivo, podemos simplesmente abrir um editor de texto e salvá-lo. Mas existem outras formas. Uma das formas mais simples é usando o comando `touch`:

```
1 cd /tmp
2 touch procedimentos.txt
3 touch contas.pdf contas2.pdf contas3.pdf contas4.pdf
```

Curingas

O significado da palavra curinga no dicionário é o seguinte: carta de baralho, que em certos jogos, muda de valor e colocação na sequência. No sistema **GNU/Linux** é bem parecida a utilização desse recurso. Os curingas são utilizados para especificar um ou mais arquivos ou diretórios.

Eles podem substituir uma palavra completa ou somente uma letra, seja para listar, copiar, apagar, etc. São usados cinco tipos de curingas no **GNU/Linux**:

Campo	Significado
*	Utilizado para um nome completo ou restante de um arquivo/diretório;
?	Esse curinga pode substituir uma ou mais letras em determinada posição;
!	exclui da operação;
[padrão]	É utilizado para referência a uma faixa de caracteres de um arquivo /diretório.
[a-z][0-9]	Usado para trabalhar com caracteres de a at é z seguidos de um caractere de 0 at é 9.
[a,z][1,0]	Usado para trabalhar com os caracteres a e z seguidos de um caractere 1 ou 0 naquela posição.
[a-z,1,0]	Faz referência do intervalo de caracteres de a at é z ou 1 ou 0 naquela posição.
[^ abc]	Faz referência a qualquer caracter exceto a, b e c.
{padrão}	Expande e gera strings para pesquisa de padrões de um arquivo /diretório.
X{ab,01}	Faz referência a sequência de caracteres Xab ou X01.
X{a-e,10}	Faz referência a sequência de caracteres Xa Xb Xc Xd Xe X10.

> A barra invertida serve para escapar um caracter especial, ela é conhecida também como **backslash**.

A diferença do método de expansão dos demais, é que a existência do arquivo ou diretório é opcional para resultado final. Isto é útil para a criação de diretórios.

Os 5 tipos de curingas mais utilizados **(*, ?, [ ], , !)** podem ser usados juntos.

Vejamos alguns exemplos:

Vamos criar 5 arquivos no diretório `/srv` utilizando o método de expansã

```
cd /srv
touch curriculo{1,2,3}.txt curriculo{4,5}.new
```

Podemos listá-los assim:

```
ls
curriculo1.txt curriculo2.txt curriculo3.txt curriculo4.new curriculo5.new
```

Vamos listar todos os arquivos do diretório `/srv`. Podemos usar o curinga "*" para visualizar todos os arquivos do diretório:

```
ls *
curriculo1.txt curriculo2.txt curriculo3.txt curriculo4.new curriculo5.new
```

Para listarmos todos os arquivos do diretório `/srv` que tenham **new** no nome:

```
ls *new*
curriculo4.new curriculo5.new
```

Listar todos os arquivos que começam com qualquer nome e terminam com **.txt**:

```
ls *.txt
curriculo1.txt curriculo2.txt curriculo3.txt procedimentos.txt
```

Listar todos os arquivos que começam com o nome **curriculo**, tenham qualquer caractere no lugar do curinga, e terminem com **.txt**:

```
ls curriculo?.txt
curriculo1.txt curriculo2.txt curriculo3.txt
```

Para listar todos os arquivos que começam com o nome **curriculo**, tenham qualquer caractere entre o número **1-3** no lugar da 4ł letra e terminem com **.txt**. Neste caso, se obtém uma filtragem mais exata, pois o curinga especifica qualquer caractere naquela posição e **[ ]** especifica um intervalo de números ou letras que será usado:

```
ls curriculo[1-3].txt
```

```
2 curriculo1.txt curriculo2.txt curriculo3.txt
```

Para listar todos **.txt** exceto o **curriculo2.txt**:

```
1 ls curriculo[!2].txt
2 curriculo1.txt curriculo3.txt
```

Para listar os arquivos **curriculo4.new** e **curriculo5.new** podemos usar os seguintes métodos:

```
1 ls *.new
2 ls *new*
3 ls curriculo?.new
4 ls curriculo[4,5].*
5 ls curriculo[4,5].new
```

Existem muitas outras sintaxes possíveis para obter o mesmo resultado. A mais indicada será sempre aquela que atender à necessidade com o menor esforço possível.

A criatividade nesse momento conta muito. No exemplo anterior, a última forma resulta na busca mais específica. O que pretendemos é mostrar como visualizar mais de um arquivo de uma só vez. O uso de curingas é muito útil e pode ser utilizado em todas as ações do sistema operacional referentes aos arquivos e diretórios: copiar ,apagar, mover e renomear.

Criando diretórios

O comando `mkdir` é utilizado para criar um diretório no sistema. Um diretório é uma pasta onde você guarda seus arquivos. Exemplo:

Criar o diretório **Suporte**:

```
1 cd /srv
2 mkdir Suporte
```

Criar o diretório **Financeiro** e o subdiretório **Contas a Pagar**:

```
1 mkdir -p Financeiro/Contas\ a\ Pagar
```

A opção `-p` permite a criação de diretórios de forma recursiva. Para que um subdiretório exista, o seu diretório diretamente superior tem que existir. Portanto a criação de uma estrutura como

Rh/Processos/Cv's exigiria a execução de três comandos `mkdir`.

Algo como:

```
1 mkdir Rh
2 mkdir Rh/Processos
3 mkdir Rh/Processos/Cv\s
```

A opção `-p` permite que toda essa estrutura seja criada em uma única linha. Assim:

```
1 mkdir -p Rh/Processos/Cv\s
```

Removendo arquivos/diretórios

O comando `rm` é utilizado para apagar arquivos, diretórios e susiretórios estejam eles vazios ou não.

Exemplos:

Remover os arquivos com extensão **txt**:

```
1 cd /srv
2 ls
3 rm curriculo?.txt
4 ls
```

Remover o arquivo **curriculo4.new** pedindo confirmação:

```
1 rm -i curriculo4.news
2 rm: remover arquivo comum vazio curriculo4.new?
```

A opção `-i` força a confirmação para remover o arquivo **curriculo4.new**.

Remover o diretório **Rh**:

```
1 rm -r Rh
```

A opção `-r` ou `-R` indica recursividade, ou seja, a remoção deverá ser do diretório Rh e de todo o seu conteúdo.

> Observação: Muita atenção ao usar o comando `rm`! Uma vez que os arquivos e diretórios removidos não podem mais ser recuperados!

O comando `rmdir` é utilizado para remover diretórios vazios.

Exemplos:

Remover o diretório **Suporte**:

```
1 rmdir Suporte
```

Copiar arquivos/diretórios

O comando `cp` serve para fazer cópias de arquivos e diretórios. Perceba que para lidar com diretórios a opção `-r` ou `-R` tem que ser usada:

```
1 cp arquivo-origem arquivo-destino
2 cp arquivo-origem caminho/diretório-destino/
3 cp -R diretório-origem nome-destino
4 cp -R diretório-origem caminho/diretório-destino/
```

Uma opção do comando `cp` muito útil em nosso dia-a-dia é a opção `-p`, que faz com que a cópia mantenha os **meta-dados** dos arquivos, ou seja, não modifica a data e hora de criação, seus donos e nem suas permissões. Utilizar como **root**:

```
1 su - aluno
2 $ touch teste
3 $ ls -l
4 $ exit
5 cd / home / aluno
6 cp -p teste teste2
7 cp teste teste3
8 ls -l teste2 teste3
```

Mover ou renomear arquivos/diretórios

O comando `mv` serve tanto para renomear um arquivo quanto para movê-lo:

```
mv arquivo caminho/diretório-destino/
mv arquivo novo-nome
mv diretório novo-nome
mv diretório caminho/diretório-destino/
```

A movimentação de um arquivo é uma ação de cópia seguida de uma remoção.

Renomeando arquivo:

```
mv teste teste4
```

Movendo arquivo:

```
mv teste4 /tmp
```

Renomeando diretório:

```
cd /srv
mv Financeiro financeiro
```

Movendo diretório:

```
mv financeiro /srv/Rh/
```

25

Localização no sistema

Comando find

O comando `find` procura por arquivos/diretórios no disco. Ele pode procurar arquivos pela sua data de modificação, tamanho, etc. O `find`, ao contrário de outros programas, usa opções longas por meio de um `-`.

Sintaxe do comando `find`:

> find [diretório] [opções/expressão]

- **-name [expressão]**: Procura pela [expressão] definida nos nomes de arquivos e diretórios processados.

```
1 find /etc -name *.conf
```

- **-maxdepth[num]**: Limita a recursividade de busca na árvore de diretórios. Por exemplo, limitando a 1, a busca será feita apenas no diretório especificado e não irá incluir nenhum subdiretório.

```
1 find /etc -maxdepth 1 -name *.conf
```

- **-amin[num]**: Procura por arquivos que foram acessados [num] minutos atrás. Caso seja antecedido por -", procura por arquivos que foram acessados entre [num] minutos

atrás e o momento atual.

```
1 find ~ -amin -5
```

- **-atime[num]**: Procura por arquivos que foram acessados [num] dias atrás. Caso seja antecedido por -, procura por arquivos que foram acessados entre [num] dias atrás e a data atual.

```
1 find ~ -atime -10
```

- **-uid[num]**: Procura por arquivos que pertençam ao usuário com o **uid 1000** [num].

```
1 find / -uid 1000
```

- **-user[nome]** : Procura por arquivos que pertençam ao usuário `aluno` [nome].

```
1 find / -user aluno
```

- **-perm[modo]**:

Procura por arquivos que possuem os modos de permissão [modo]. Os [modo] de permissão podem ser numérico (octal) ou literal.

```
1 find / -perm 644
```

- **-size[num]**: Procura por arquivos que tenham o tamanho [num]. O tamanho é especificado em bytes. Você pode usar os sufixos k, M ou G para representar o tamanho em Quilobytes, Megabytes ou Gigabytes, respectivamente. O valor de [num] Pode ser antecedido de + ou - para especificar um arquivo maior ou menor que [num].

```
1 find / -size +1M
```

- **-type[tipo]**: Procura por arquivos do [tipo] especificado. Os seguintes tipos são aceitos:

> **b - bloco**;

> **c - caractere**;

> **d - diretório**;
>
> **p - pipe**;
>
> **f - arquivo regular**;
>
> **l - link simbólico**;
>
> **s - socket**.

```
1 find /dev -type b
```

Mais alguns exemplos que podem te ajudar no seu dia-a-dia..

O comando abaixo busca todos os arquivos ou diretórios que possuem permissão 777:

```
1 find / -perm 777
```

O comando abaixo busca no diretório `/root` apenas arquivos com permissão 777:

```
1 find /root -type f -perm 777
```

O comando abaixo busca no diretório `/` apenas arquivos com tamanho maior do que 10M:

```
1 find / -size +10M
```

O comando abaixo busca no diretório `/` arquivos com permissão 600 e executa o comando `ls` no formato longo para obter mais detalhes sobre os arquivos:

```
1 find / -perm 600 -exec ls -l {} \;
```

O comando abaixo busca no diretório `/` arquivos com permissão 600 e executa o comando `ls` no formato longo para obter mais detalhes sobre os arquivos:

```
1 find / -perm 600 -print0 | xargs -0 ls -l
```

O comando abaixo buscará no diretório raiz apenas arquivos vazios:

```
1 find / -type f -empty
```

O comando abaixo buscará no diretório raiz apenas diretórios vazios:

```
1 find / -type d -empty
```

O comando abaixo buscará no diretório raiz todos os arquivo modificados nos 50 dias anteriores:

```
1 find / -mtime 50
```

O comando abaixo buscará no diretório raiz todos arquivos que foram modificados entre 50 e 100 dias atrás.

```
1 find / -mtime +50 -mtime -100
```

O comando abaixo buscará no diretório raiz todos arquivos que foram acessados nos últimos 50 dias:

```
1 find / -atime 50
```

O comando abaixo buscará no diretório corrente apenas arquivos que foram acessados no último dia (nas últimas 24 horas) e executará o comando `ls` no formato longo (detalhado) na possível lista de arquivos que será gerada:

```
1 find . -type f -atime -1 -exec ls -l {} \;
```

O comando abaixo buscará no diretório pessoal do usuário que estiver executando todos os arquivos modificados nos últimos 60 minutos (1 hora):

```
1 find ~ -cmin -60
```

O comando abaixo buscará no diretório raiz todos os arquivos com extensão `.txt` e retirará a permissão de execução de todos eles:

```
find / -name "*.txt" -exec chmod -x {} ";"
```

O comando abaixo buscará arquivos que tem todas as permissões citadas em `-perm`, neste caso 4000:

```
find / -perm -4000
```

O comando abaixo buscará um arquivo com o nome `passwd` até dois níveis (subdiretórios) abaixo da raiz:

```
find / -maxdepth 2 -name passwd
```

O comando abaixo buscará no diretório corrente (ponto (`.`) após o `find`) diretórios vazios e os removerá:

```
find . -type d -empty -exec rmdir {} \;
```

Comando xargs

Outra forma de procurar por arquivos e/ou diretórios e executar um comando é através do comando xargs que obtém como a entrada a saída ok do comando antes do pipe e envia como stdin do próximo comando, no caso o `ls -ld`:

```
find / etc -type d | xargs ls -ld
```

Vamos agora listar diretórios utilizando o `xargs`:

```
ls / | xargs - n1
ls / | xargs - n2
ls / | xargs - n3
```

Outros testes com o `xargs`:

```
ls / > teste_xargs.txt
cat teste_xargs.txt
cat teste_xargs.txt | xargs -n 2
```

```
4 xargs -n 3 < teste_xargs.txt
```

Você percebeu que no primeiro comando ele listou o diretório, jogando na tela um nome de cada vez. O segundo comando fará o mesmo só que com dois nomes na mesma linha, e o terceiro com 3 nomes.

```
1 echo "linux:macos:freebsd:openbsd" > /tmp/teste.txt
2 cat /tmp/teste.txt | xargs -d: -n 2
3 linux macos
4 freebsd openbsd
```

Comando locate

O comando `locate` é um comando rápido de busca de arquivos, porém não usa busca recursiva na sua árvore de diretórios. Ele utiliza uma base de dados que é criada pelo comando `updatedb`, para que a busca seja mais rápida. Por padrão, a atualização da base de dados é agendado no `cron` do sistema para ser executada diariamente.

Para utilizá-lo, primeiro é necessário criar a sua base de dados usando a seguinte sintaxe:

```
1 updatedb
```

Quando esse comando é executado pela primeira vez costuma demorar um pouco. Isso deve-se a primeira varredura do disco para a criação da primeira base de dados.

Para o comando `locate`, usamos a seguinte sintaxe:

```
1 locate howto
```

A saída do comando será algo parecido com:

```
1 /usr/share/doc/hashcat/extra/tab_completion/howto.txt
2 /usr/share/doc/smartmontools/badblockhowto.html
3 /usr/share/doc/util-linux/howto-build-sys.txt
4 /usr/share/doc/util-linux/howto-compilation.txt
5 /usr/share/doc/util-linux/howto-contribute.txt.gz
```

26

Tópicos para revisão do capítulo

Atente-se para os tópicos a seguir. Eles devem ser estudados com muita atenção, pois representam os pontos mais importantes deste capítulo.

- Os diretórios no Linux são organizados de acordo com o padrão FHS – *Filesystem Hierarchy Standard*
- O comando `touch` cria um arquivo vazio e também pode ser usado para modificar a data de criação/modificação de um arquivo ou diretório. O comando `file` mostra o tipo de um arquivo.
- O comando `mkdir` é usado para criar diretórios. O parâmetro `-p` é usado para não sobrescrever um diretório caso ele já exista.
- O comando `rm` é usado para remover arquivos ou diretórios. O parâmetro `-i` faz uma pergunta para confirmar a remoção do arquivo. O argumento `-r` serve para remover o conteúdo do diretório e o parâmetro `-f` força a remoção dos arquivos.
- O comando `find` faz busca por arquivos e diretórios por meio de suas características, por exemplo, dono, grupo, permissões, tamanho, tipo de arquivo etc.
- A função do comando `xargs` é construir listas de parâmetros e passá-la para a execução de outros programas ou instruções.
- O comando `locate` é usado para buscar em uma base dados gerada pelo comando `updatedb`.