

CERTIFICAÇÃO LINUX ESSENTIALS

MATERIAL PREPARATÓRIO PARA CERTIFICAÇÃO



LUCIANO ANTONIO SIQUEIRA

2019

Certificação Linux Essentials

Material preparatório para a certificação LPI Linux Essentials

Luciano Antonio Siqueira

Atribuição-Compartilhual 4.0 Internacional (CC BY-SA 4.0)

Aviso Legal

Curso Linux Essentials de [Luciano Antonio Siqueira](#) está licenciado com uma [Licença Creative Commons - Atribuição-CompartilhaIgual 4.0 Internacional](#).



Atribuição-CompartilhaIgual 4.0 Internacional (CC BY-SA 4.0)

DISTRIBUIÇÃO GRATUITA

Sobre o autor

Luciano Antonio Siqueira é formado em Matemática Aplicada e Computacional pela Universidade de São Paulo e em Psicologia pela Universidade Estadual Paulista. Trabalha como autor e desenvolvedor em Software Livre, publicou diversos livros na área e já ministrou cursos de Certificação Linux no Brasil, Uruguai, Colômbia e Jamaica. O autor trabalha junto ao *Linux Professional Institute* na produção dos exames e conteúdos em português. Para entrar em contato, escreva para luciano@lcnsqr.com.

Introdução

A Certificação LPI Linux Essentials é o programa de entrada para aqueles interessados em obter uma certificação profissional em Linux. Este livro é um material preparatório para o **exame de certificação versão 1.6**, que em 2019 se tornou a versão vigente do exame.

Menos abrangente que a Certificação LPI nível 1, o propósito da Certificação Linux Essentials é definir o conhecimento básico necessário para o uso competente de um desktop ou dispositivo móvel com o sistema operacional Linux instalado.

A certificação é oferecida pelo Linux Professional Institute, o primeiro e maior organismo de certificação Linux do mundo neutro em relação a fornecedores. Com o apoio de uma rede global de afiliados, o LPI trabalha para aumentar a conscientização sobre o poder das tecnologias de código aberto e, ao mesmo tempo, ajudar a garantir que os profissionais de TI tenham as habilidades necessárias para serem competitivos no ambiente de trabalho global. Mais informações sobre o LPI e como obter suas diferentes certificações podem ser obtidas em lpi.org.

A partir da minha bem sucedida experiência com os livros preparatórios para as certificações LPIC-1 e LPIC-2 – que podem ser encontrados em lcsqr.com/livros – escrevi o material que segue dentro da mesma perspectiva. De maneira sucinta, procurei explicar cada aspecto exigido na lista de objetivos para a certificação, que pode ser encontrada no último capítulo. O exame para a Certificação Linux Essentials é composto de 40 questões. Em cada um dos tópicos e subtópicos há a indicação do seu respectivo peso, indicando quantas perguntas sobre o assunto constarão na prova.

O primeiro tópico, *A comunidade Linux e a Carreira Open Source*, é basicamente a exposição do que é Software Livre/Código Aberto e sua história. Já os demais tópicos abordam o objetivo maior de qualquer aprendizado em Linux: trabalhar com a linha de comando e conhecer as principais ferramentas e configurações de um sistema Linux/Unix.

Para um usuário habituado à interface de janelas e operação com o mouse, utilizar a linha de comando pode parecer ultrapassado e difícil. Espero que com a leitura desse material ele possa mudar essa perspectiva, compreendendo que para algumas tarefas é bastante conveniente abdicar de certas facilidades para obter maior controle e profundidade.

O material está estruturado de acordo com os tópicos exigidos na Certificação. Os pesos que aparecem abaixo de cada tópico correspondem à quantidade de perguntas sobre aquele tema que o candidato vai encontrar na prova. Comandos e listagens de código são apresentadas em formato fixo. É importante observar que comandos de *root* possuem o caractere # a frente, enquanto que comandos de usuários comuns possuem o caractere \$. Mais detalhes são fornecidos ao longo do texto.

Qualquer distribuição pode ser utilizada para praticar os conteúdos abordados, o que é muito importante para o aprendizado. Recomendo utilizar a distribuição **Debian**, www.debian.org, ou **CentOS**, centos.org. Instruções de instalação podem ser encontradas nas respectivas páginas das distribuições. Como opção, é possível experimentar um ambiente Linux básico diretamente no navegador, no endereço <https://vfsync.org/vm.html>.

Espero que o material seja útil para os todos os interessados. Envie suas dúvidas, sugestões e críticas para luciano@lcsqr.com.

Atenciosamente,

Luciano Antonio Siqueira

Atualizações

Visite lcnsqr.com para obter atualizações deste livro e outros conteúdos relacionados.

Conheça outros livros de Luciano Antonio Siqueira:



Certificação LPI-1 (101-102)

Curso completo para a Certificação LPI nível 1

SAIBA MAIS

lcnsqr.com/livros



Certificação LPI-2 (201-202)

Curso completo para a Certificação LPI nível 2

SAIBA MAIS

lcnsqr.com/livros

Tópico 1: A comunidade Linux e a Carreira Open Source

Peso: 7

1.1: Evolução do Linux e Sistemas Operacionais Populares

Peso: 2

O Linux é um sistema operacional de computadores criado no modelo de desenvolvimento de *Código Aberto*. Apesar do termo ser utilizado para designar todo o conjunto de componentes que formam um sistema operacional, Linux refere-se mais precisamente ao componente central do sistema, denominado *kernel* (ou *núcleo*). Seu criador, o programador finlandês Linus Torvalds, publicou a primeira versão do Linux em 5 de outubro de 1991. Sua intenção foi criar um kernel para o **projeto GNU**.

Sistema Operacional

Um sistema operacional é o conjunto de programas encarregado de controlar os componentes do computador, criando o ambiente no qual os aplicativos utilizados pelo usuário podem funcionar. Sem um sistema operacional, cada aplicativo teria de lidar diretamente com os componentes do computador, o que dificultaria seu desenvolvimento e utilização simultânea (multitarefa).

Projeto GNU

Em 1983, o programador americano Richard Stallman iniciou o projeto GNU para criar um sistema operacional de código aberto que funcionasse do mesmo modo que o sistema operacional Unix. O Unix foi um sistema que definiu conceitos técnicos utilizados por diversos sistemas operacionais inspirados nele.

Unix

O Unix original foi desenvolvido na década de 1970 pelos pesquisadores Ken Thompson e Dennis Ritchie (entre outros), no centro de pesquisas Bell Labs da empresa AT&T. O projeto GNU é uma dessas variantes do Unix original, porém com a premissa de ser desenvolvido em Código Aberto.

Código Aberto

Em seu sentido mais amplo, Código Aberto é um modelo de desenvolvimento que promove:

- Acesso universal, via licença livre, ao projeto ou especificação de um produto.
- Redistribuição universal deste projeto ou especificação, incluindo melhorias feitas posteriormente.

A partir da década de 1990, o projeto GNU – equipado com o kernel Linux – deu origem a inúmeros sistemas operacionais de código aberto, cada um organizando, aprimorando e criando novos programas à sua maneira. Esses sistemas operacionais são chamados **Distribuições Linux**.

Distribuições Linux

Além de conter o kernel Linux e programas GNU, uma distribuição Linux normalmente agrega outros recursos para tornar sua utilização mais simples. Além de oferecerem um conjunto

completo de aplicativos prontos para uso, as distribuições mais populares podem atualizar e instalar novos programas automaticamente. Esse recurso é chamado **gestão de pacotes**. O gestor de pacotes da distribuição elimina o risco de instalar um programa incompatível ou mal intencionado.

Outra vantagem das distribuições é seu custo. Um usuário experiente pode copiar e instalar legalmente a distribuição sem precisar pagar por isso. Existem distribuições pagas, mas que pouco diferem daquelas sem custo no que diz respeito a facilidade e recursos. As principais distribuições Linux são:

- [Debian](#). A principal característica do Debian é seu sistema de gestão de pacotes, o **dpkg**. Os pacotes do dpkg têm o sufixo `.deb`. [Instruções para cópia e instalação](#).
- [CentOS](#). O CentOS é uma versão gratuita da distribuição comercial [Red Hat Enterprise Linux](#). Seu sistema de pacotes chama-se **RPM**.

Existem muitas outras distribuições importantes além dessas duas, como [Ubuntu](#), [Fedora](#), [Linux Mint](#), [openSUSE](#), etc. Apesar de cada distribuição ter suas peculiaridades, um usuário com alguma experiência em Linux será capaz de trabalhar com todas elas. Muitas utilizam o mesmo sistema de gestão de pacotes. Por exemplo, as distribuições Debian, Ubuntu e Linux Mint utilizam o dpkg (essas duas últimas foram criadas a partir da Debian). Já a Red Hat, CentOS e Fedora utilizam RPM. Outras, como a OpenSUSE e Slackware, têm sistemas de gestão de pacotes menos comuns, mas semelhantes ao dpkg e o RPM em sua finalidade.

Além dessas distribuições de uso geral, existem distribuições voltadas para um público específico. Por exemplo, a [Scientific Linux](#) é uma distribuição voltada para produção científica. A distribuição [Kali Linux](#) é uma distribuição baseada no Debian especializada em ferramentas de segurança e teste de penetração.

Arquiteturas

Desde a década de 1990, a arquitetura de computadores mais comum é a **x86**, utilizada tanto em servidores quanto em computadores pessoais. Porém, existem muitas outras arquiteturas, e o Linux pode ser instalado na maioria delas.

Muitas dessas arquiteturas são encontradas em máquinas industriais e até eletrodomésticos. Para esses casos pode ser encontrada ou criada uma versão especializada do Linux, voltada apenas para o equipamento em questão. Esse tipo de sistema recebe o nome de **sistema embarcado**.

Uma arquitetura que merece destaque é a arquitetura **ARM**, na qual se baseiam praticamente todos os *smartphones* atuais. O sistema [Android](#), utilizado em bilhões de aparelhos celulares pelo mundo, é um tipo de Linux. Apesar de haver controvérsia quanto a chamar o Android de distribuição Linux, é inegável que o kernel desse sistema operacional é Linux. Por ter seu código de programação aberto, é até possível utilizar variantes do Android, como o [LineageOS](#).

Além de smartphones, outros aparelhos são baseados em arquitetura ARM, como o [Raspberry Pi](#). O Raspberry Pi é basicamente um computador completo reduzido a uma única placa de circuito impresso, com o tamanho ligeiramente maior que um cartão de crédito. Todos os componentes básicos de um computador tradicional estão presentes no Raspberry Pi, como processador, memória, saída de vídeo HDMI, áudio, rede ethernet, WIFI, portas USB, etc. Contudo, o Raspberry Pi não possui uma interface de disco rígido interna *IDE* ou *SATA*, de modo que o sistema operacional é armazenado num cartão SD e inserido na entrada presente no aparelho. O Raspberry Pi não é tão potente quanto um computador moderno, mas pode desempenhar muitas das funções menos exigentes.

Existem muitos modelos de computadores desse tipo, conhecidos como *Single Board Computer*, ou simplesmente *SBC*, e quase sempre são equipados com uma distribuição Linux própria para arquitetura ARM. No caso do Raspberry Pi, a distribuição Linux mais utilizada é a [Raspbian Linux](#), uma distribuição baseada no Debian e compatível com a arquitetura ARM. Esses dispositivos costumam ser utilizados em aplicações de controle e monitoramento, muitas vezes se comunicando entre si e com um servidor central. Quando essa comunicação se dá via Internet, esse tipo de utilização recebe o nome de *Internet das coisas* ou simplesmente **IoT**, sigla em inglês de *Internet of Things*.

Computação em Nuvem

Dada a grande variedade modalididades de troca de informações pela Internet, se tornou mais conveniente do ponto de vista da infraestrutura de comunicação que essas informações utilizadas e geradas por aplicações sejam armazenadas remotamente. A esse tipo de abordagem de distribuição das informações dá-se o nome de **Computação em Nuvem**, pois os recursos podem estar fragmentados em diferentes localidades físicas e não mais num único ponto.

É comum que a contratação de uma **IaaS** (do inglês *Infrastructure as a Service, Infraestrutura como Serviço*) para implementar uma aplicação de grande porte siga o modelo de computação em nuvem. Usando computadores de grande capacidade interligados com Internet de alta velocidade, o provedor da IaaS oferece recursos – como espaço em disco – que podem estar em qualquer uma dessas máquinas separadas geograficamente. Esse modelo permite expandir, reduzir ou criar recursos sob demanda. Os dados também podem estar espelhados em diferentes pontos geográficos para garantir sua segurança e disponibilidade.

A natureza dinâmica desse tipo de hospedagem não permite que o contratante escolha arbitrariamente o sistema operacional que vai utilizar para implementar o serviço. Uma distribuição Linux, por exemplo, poderá ser instalada a partir de uma lista de opções pré-definidas pelo provedor, que já estão preparadas para operar em sua infraestrutura. O processo de instalação costuma ser feito via navegador, numa área dedicada ao contratante, que escolhe as quantidades e tipos de recursos que vai utilizar. Outras configurações, como restrições de *firewall* e cópia de chaves *SSH* também costumam ser feitas nessa mesma interface no navegador. Depois de instalado, o sistema operacional pode ser acessado da mesma maneira que uma máquina remota convencional.

1.2: Principais Aplicações Open Source

Peso: 2

Um aplicativo é um programa cuja finalidade não está diretamente associada com o funcionamento do computador, mas com alguma tarefa de interesse do usuário. As distribuições Linux oferecem diversas opções de aplicativos para as mais diferentes finalidades: aplicativos de escritório, Internet, edição de áudio e vídeo, etc. Sempre há mais de uma opção de aplicativo para uma mesma finalidade, ficando a cargo do usuário escolher a que mais lhe agrada.

Cada distribuição oferece uma coleção de aplicativos já instalados por padrão. Além desses aplicativos pré-instalados, todas as distribuições populares possuem um gestor de pacotes com uma vasta coleção de aplicativos disponíveis para instalação, chamada *repositório de pacotes*. Apesar de serem os mesmos programas, cada distribuição possui o próprio sistema de instalação de pacotes. O Debian, Ubuntu e Linux Mint, por exemplo, utilizam as ferramentas *dpkg* e *apt-get* para instalar pacotes. Já as distribuições Red Hat, Fedora e CentOS utilizam as ferramentas *rpm*, *yum* e *dnf*. Como o empacotamento dos aplicativos é diferente para cada distribuição, é muito importante instalá-los a partir dos repositórios correspondente à distribuição utilizada. O gestor de pacotes da distribuição se encarrega de usar somente os pacotes corretos e fazer as atualizações necessárias, não sendo necessário se preocupar com essas diferenças.

Aplicativos de Escritório

Aplicativos de escritório são editores de texto, de apresentações, planilhas eletrônicas e outros documentos que auxiliam o usuário a realizar seu trabalho. É comum que esses aplicativos estejam agrupados em coleções chamadas suítes de escritório.

Durante muito tempo, a suíte mais utilizada no Linux foi o pacote **OpenOffice.org**. Após ser transferido para diferentes empresas, o desenvolvimento do OpenOffice.org foi separado em dois projetos distintos: O *LibreOffice* e o *Apache OpenOffice*.

Ambos possuem os mesmos recursos básicos e são compatíveis com os formatos de documentos do *Microsoft Office*. O LibreOffice mantido pela **Document Foundation** e Apache OpenOffice mantido pela **Apache Software Foundation**. Os aplicativos principais de cada um, separados por finalidade são:

- **Writer:** Editor de textos.
- **Calc:** Planilha eletrônica.
- **Impress:** Apresentações.
- **Draw:** Desenho vetorial.
- **Math:** Fórmulas matemáticas.
- **Base:** Banco de dados.

Tanto o LibreOffice quanto o Apache OpenOffice são programas de código aberto, sendo a licença do primeiro a **LGPLv3** e, do segundo, a **Apache License 2.0**. A maioria das distribuições Linux adota o LibreOffice por padrão.

Aplicativos para Web

Para grande parte dos usuários, a principal finalidade de um computador é usar Internet. Hoje, as páginas da Internet podem substituir diversos aplicativos, com a vantagem de poderem ser acessadas de qualquer localidade, sem necessidade de instalação de novos programas. Esse fato tornou o navegador no principal aplicativo do sistema operacional.

Os principais navegadores de Internet no ambiente Linux são o **Google Chrome** e o **Mozilla Firefox**. O Chrome é um navegador fornecido pelo Google, mas é baseado no navegador de código aberto **Chromium**, que pode ser instalado a partir do gestor de pacotes da distribuição e é totalmente compatível com o Chrome. Mantido pela Mozilla, uma fundação sem fins lucrativos, as origens do Firefox remontam ao precursor **Netscape**, tendo sido desenvolvido a partir da liberação do código fonte deste. Desde então, os desenvolvedores do Mozilla Firefox têm participado ativamente do desenvolvimento de padrões abertos e novas tecnologias da Internet, essenciais para a Web moderna.

MDN Web Docs

Uma das melhores fontes para aprender a desenvolver aplicativos para a Internet é a [MDN Web Docs](#). Mantido pela Mozilla, o site contém materiais de referência e tutoriais para iniciantes que abrangem todas as tecnologias utilizadas na Web. É fonte de referência para todos os níveis de desenvolvedores.

Além do Firefox, a Mozilla desenvolve outros aplicativos, como o **Thunderbird**, um cliente de email. Com a ascensão do webmail, o uso dos clientes de email declinou, mas muitos optam por utilizá-los devido aos recursos adicionais e integração com outros aplicativos.

Multimídia

Apesar do avanço das aplicações Web, muitas tarefas ainda demandam aplicativos tradicionais. É o caso das aplicações que manipulam conteúdos multimídia, como imagens, áudio e vídeo.

Alguns dos aplicativos multimídia de código aberto mais populares são:

- **Blender:** Para criação de animações 3D. Também pode exportar modelos tridimensionais utilizados por impressoras 3D.
- **Gimp:** Criação e edição de imagens, com muitos recursos. Apesar de sua abordagem ser diferente, pode ser comparado ao programa *Adobe Photoshop*. O Gimp pode editar a maioria dos arquivos de imagens do tipo *bitmap*, como *JPEG*, *PNG*, *GIF*, *TIFF*, etc.
- **Inkscape:** Editor de imagens vetoriais, semelhante ao programa *Corel Draw*. O Inkscape trabalha com o formato *SVG*, que é o padrão aberto para gráficos vetoriais e compatível para exibição em navegadores.
- **Audacity:** Edição de áudio. Pode ser utilizado para ouvir e converter diferentes formatos

de áudio como MP3, WAV, OGG, etc.

- **ImageMagick**: Converte e edita praticamente todos os formatos de imagem.

Mais do que os editores de mídia, a maioria dos usuários necessita de um bom tocador de música e vídeo. Para vídeo, os mais populares são o **VLC** e o **smplayer**. Para música, há muitas opções, como o **Audacious**, **Banshee** e o **Amarok**.

Programas para Servidores

Quando um computador abre uma página de Internet, na verdade faz uma conexão com outro computador e solicita determinadas informações a este. Nesse caso, o primeiro computador é chamado **cliente** e o segundo é chamado **servidor**.

O computador servidor – que pode ser um equipamento especial ou um computador doméstico – necessita de programas específicos para cada tipo de informação que vai fornecer. No caso de páginas de Internet, a maior parte dos servidores espalhados pelo mundo utiliza programas de código aberto. Esse tipo de servidor é chamado **servidor HTTP** (*Hyper Text Transfer Protocol*), e os servidores HTTP de código aberto mais populares são o [Apache](#), o [Nginx](#) e o [lighttpd](#).

Além de páginas de Internet, programas servidores podem fornecer diversos tipos de informações aos clientes. Também é comum que programas servidores se comuniquem com outros programas servidores. É o caso dos programas servidores HTTP e os programas servidores de *banco de dados*.

Banco de Dados

Um banco de dados é um conjunto organizado de informações. Um programa servidor de banco de dados armazena os conteúdos em formatos que permitem consultar, armazenar e relacionar rapidamente um grande volume de informações.

Quando uma página de Internet é acessada, o servidor HTTP identifica qual foi o endereço solicitado, consulta no banco de dados todos os textos, imagens e outras informações relacionadas àquele endereço e envia essas informações de volta para o programa cliente (nesse caso, um navegador como o Firefox). De forma parecida, quando um usuário faz um cadastro na Internet, o servidor HTTP coleta essas informações enviadas pelo cliente e as armazena num banco de dados.

Os programas servidores de banco de dados de código aberto são muito utilizados, principalmente os chamados bancos de dados relacionais. Tanto na Internet como em outras aplicações, dois dos bancos de dados relacionais mais utilizados são o [MySQL](#) (que vem sendo substituído pelo [MariaDB](#)) e o [PostgreSQL](#).

Compartilhamento de Dados

Existem outros tipos de comunicação entre computadores além do modelo cliente/servidor. Em algumas redes locais, como em escritórios e residências, é conveniente que os computadores hora atuem como servidores, hora atuem como clientes.

Isso é necessário quando se quer acessar arquivos de um outro computador dentro da rede – por exemplo, entre um computador portátil e um computador de mesa – sem a necessidade de copiar os arquivos para um meio intermediário, como chaveiros USB ou mídias de CD/DVD.

Entre computadores Linux, costuma-se compartilhar arquivos e espaço em disco com o **NFS** (*Network File System*). O NFS é o mecanismo de compartilhamento de arquivos padrão em ambientes do tipo Unix.

Se houverem computadores na rede com outros sistemas operacionais, é mais conveniente utilizar uma comunicação que seja comum a todos. No Linux, isso é conseguido ao utilizar o [Samba](#). Além de poder acessar arquivos ou espaço de ambientes Windows, o Samba pode

simular um servidor de arquivos desse tipo, permitindo o compartilhamento para computadores com Windows ou outros computadores Linux com Samba.

Outros tipos de comunicação entre dois computadores são aquelas que precisam ser intermediadas por um terceiro. É o caso do correio eletrônico (email). Ao enviar um email, a mensagem é armazenada em um servidor até que o usuário destinatário o acesse para ler a mensagem. Esse processo tem a vantagem de dispensar que os computadores remetente e destinatário estejam disponíveis ao mesmo tempo, além de permitir a comunicação entre computadores que não estão diretamente ligados numa mesma rede. O principal programa em código aberto servidor de emails é o **Postfix**, um dos mais utilizados no mundo.

Quando o objetivo é implementar a modalidade de computação em nuvem, que agrega várias funções de armazenamento e compartilhamento remoto, existem opções como o **Nextcloud** e o **ownCloud**. Ambos oferecem soluções semelhantes a serviços como o *Google Drive* ou o *Dropbox*. Porém, diferente destes serviços, é possível instalar o Nextcloud e o ownCloud num servidor particular, evitando armazenar dados sensíveis em servidores de terceiros. A instalação é feita sobre um servidor Web convencional, como o Apache. Além do recurso de compartilhamentos de arquivos, eles podem oferecer outros recursos disponíveis na forma de *plugins* que podem ser ativados ou desativados de acordo com a necessidade.

Há também programas servidores de muitos outros tipos, cada um para finalidades bastante específicas. Em organizações muito grandes, pode ser utilizado um servidor que fornece informações para a identificação de funcionários ou contatos de fornecedor. Esse tipo de informação pode ser enviada aos programas clientes por um servidor chamado **OpenLDAP**. O OpenLDAP é um servidor de código aberto para o protocolo chamado *Lightweight Directory Access Protocol*.

Administração de Redes

Tanto os programas clientes quanto os programas servidores necessitam que a conexão de rede esteja operando sem problemas. Para isso, existem programas do sistema operacional que atuam no estabelecimento e manutenção da conexão de rede. Dois serviços essenciais para o funcionamento da rede são o **DHCP** – *Dynamic Host Configuration Protocol* – e o **DNS** – *Domain Name Server*. O primeiro, DHCP, é o responsável por automaticamente estabelecer a conexão assim que um cabo de rede é inserido ou uma rede sem fio é acessada. O segundo, DNS, é o responsável por traduzir um nome como *www.lpi.org* para um número de identificação (número IP) que será utilizado pelo cliente para localizar o servidor e estabelecer a conexão.

Linguagens de Programação

Todos os programas, sejam servidores, clientes, aplicativos e o próprio sistema operacional, são feitos utilizando uma ou mais linguagens de programação. Os programas também são arquivos, mas o sistema operacional os trata como uma sequência de instruções e condições que devem ser atendidas pelo processador e demais dispositivos.

Existem inúmeras linguagens de programação. Uma das maiores virtudes do Linux é oferecer programas para aprender e desenvolver praticamente qualquer linguagem de programação. Algumas das linguagens mais populares são:

- **JavaScript:** É uma linguagem de programação utilizada principalmente em páginas de Internet. Originalmente, seu propósito era realizar somente tarefas simples, como a validação de formulários antes de enviar as informações para o servidor. Depois de passar por muitos aprimoramentos nas últimas décadas, hoje o JavaScript é considerado uma linguagem de primeira grandeza e é utilizado para criar aplicações extremamente sofisticadas, tanto no ambiente Web quanto em aplicativos móveis e em servidores.
- **C:** A linguagem de programação C está associada aos sistemas Unix, mas pode ser utilizada para escrever programas para quase todo tipo de computador ou dispositivo. As maiores virtudes da linguagem C são sua flexibilidade e desempenho. Tanto supercomputadores de alta capacidade quanto microcontroladores em aparelhos domésticos podem ser programados em linguagem C. Um mesmo programa escrito em linguagem C pode ser compilado em diferentes plataformas, necessitando pouca ou nenhuma alteração. Porém, depois de compilado, o programa só funcionará no sistema

operacional e na plataforma para o qual foi compilado.

- **Java:** A maior virtude do Java está em sua portabilidade, o que permite escrever um mesmo programa para ser utilizado em diferentes sistemas operacionais. Apesar do nome, o Java não tem relação com o JavaScript.
- **Perl:** A linguagem Perl é uma linguagem de programação muito utilizada para manipular conteúdos de texto. Com seus recursos de expressões regulares e formatação, é possível tratar textos e gerar relatórios ou outros documentos, como páginas de Internet (HTML).
- **Shell:** O Shell não é apenas uma linguagem de programação, mas um ambiente de interação com o computador. Pela linha de comando do Shell, é possível gerar pequenos programas – *scripts* – que automatizam tarefas complicadas ou recorrentes.
- **Python:** A linguagem Python é popular entre estudantes e profissionais que não estão diretamente ligados a área de programação. Apesar de possuir recursos avançados, a linguagem Python é um bom começo para quem deseja aprender a programar.
- **PHP:** A linguagem PHP é a mais utilizada em servidores HTTP. A maior parte das páginas disponíveis na Internet não são arquivos armazenados no servidor, mas conteúdos gerados dinamicamente pelo servidor a partir de diversas fontes, como bancos de dados. A linguagem PHP é uma das mais utilizadas para criar os programas que vão coletar informações e gerar os conteúdos das páginas. Da sua associação com um servidor Linux, o programa servidor HTTP Apache e o banco de dados MySQL, surgiu a sigla **LAMP**, *Linux, Apache, MySQL e PHP*, a combinação de programas mais popular para operar conteúdos da Internet.

Antes de se tornar um programa, todo programa é um arquivo de texto, chamado **código-fonte**. Em *linguagens compiladas* – como C e Java – o código-fonte precisa ser previamente convertido em um *arquivo binário* para que possa ser utilizado. Essa operação é feita pelo programa *compilador*.

Nas *linguagens interpretadas* – como JavaScript, Perl, Python e PHP – o programa não precisa ser compilado previamente, tornando mais simples seu desenvolvimento e modificação. Por ser convertido para instruções binárias toda vez que é executado, um programa em linguagem interpretada tende a ser mais lento do que um equivalente em linguagem compilada.

1.3: Entendendo o Software Open Source e suas Licenças

Peso: 1

O modelo de negócios com programas de código aberto pode despertar algumas dúvidas por ser diferente do modelo de venda de cópias com restrições. Muitos consumidores de programas de computador estão tão habituados com a ideia de comprar uma cópia para poder utilizar um programa que têm dificuldade em entender outras modalidades de comércio.

Ao comprar uma licença de uso de programas em código fechado (também chamado *código proprietário*), o usuário tem permissão de utilizar uma cópia do programa mas não tem acesso ao seu código fonte. O fabricante toma essa atitude para inserir mecanismos de proteção contra cópias ilegais e para esconder de seus concorrentes a maneira como seu programa funciona internamente.

Esse procedimento pode não fazer grande diferença para um usuário comum. Já para uma empresa de tecnologia, um profissional da computação ou um pesquisador, pode trazer grandes prejuízos.

Defensores do código aberto acreditam ser fundamental que haja acesso universal ao código fonte do programa. Tal como no conhecimento científico, acreditam que o avanço tecnológico é muito maior quando especialistas analisam e confrontam diferentes métodos e abordagens sobre um problema. Além da abertura do código, são formulados padrões abertos para garantir que qualquer interessado com capacidade possa acompanhar, interferir e até criar suas próprias soluções.

Nomes

Nem todo programa de código aberto é gratuito e nem todo programa gratuito é de código aberto. Os termos *Software Livre* e *Código Aberto* são utilizados para designar os programas cujo código fonte pode ser estudado, modificado e compartilhado sem restrições. Em inglês, a ambiguidade do termo *Free Software* – que pode ser entendido tanto como software livre quanto software gratuito – pode levar a conclusão de que trata-se simplesmente de programas que podem ser obtidos sem custo. Apesar de muitas vezes ser aplicável, essa percepção esconde os aspectos principais do código aberto. Já a expressão código aberto transmite a ideia do acesso ao funcionamento interno, mas não muito sobre sua modificação e compartilhamento.

Existem outros termos que se pretendem mais didáticos. Da união do termo software livre e código aberto surgiu **FOSS**: *Free Open Source Software*, ou Programa Livre de Código Aberto. Para reduzir a ambiguidade de *free*, também é utilizado o termo **FLOSS**: *Free/Libre Open Source Software*. Este último é mais utilizado quando se quer ressaltar a gratuidade e a liberdade de estudar, modificar e compartilhar. Ao longo de todo esse texto é utilizado o termo *código aberto* para designar FOSS e FLOSS.

Modelo de Negócios

Se no código aberto não há necessariamente a venda de cópias dos programas compilados, a empresa ou o profissional são remunerados de outras maneiras. Algumas delas são:

- *Financiamento externo*. Grandes empresas financiam fundações destinadas a desenvolver e produzir projetos de código aberto. Empresas como *Google* e *IBM* dependem de programas de código aberto e por isso os financiam.
- *Serviços*. Profissionais e empresas costumam vender serviços associados ao código aberto, como suporte técnico, treinamento e consultorias. Empresas locais podem oferecer os mesmos serviços de grandes multinacionais, pois os programas não são exclusivos de nenhuma delas. A **certificação LPI** é um dos instrumentos utilizados por clientes para localizar profissionais qualificados para prestar serviços.
- *Assinaturas*. Enquanto que o programa é oferecido sem custo, é possível cobrar pela hospedagem e manutenção no servidor. Aplicativos são oferecidos diretamente na Internet e o usuário paga uma taxa para utilizá-lo, o que muitas vezes é mais conveniente do que contratar um servidor e instalar o programa por conta própria.

Além dessas opções, não é impedido a um fornecedor cobrar pela organização e compilação dos programas num CD, por exemplo. Contudo, ele pode estar obrigado a fornecer também os códigos fonte dos programas. Este fator vai ser determinado pelo tipo de licença utilizada pelo programa em questão.

Licenças

Em linhas gerais, um programa de código aberto é aquele que possui código fonte disponível sem restrições. Contudo, há divergências quanto ao que é ou deixa de ser uma restrição. É possível que um programa seja parte fechado e parte aberto? O criador do programa pode impedir que seu código seja utilizado para determinadas finalidades? Essas questões são definidas na licença adotada pelo programa.

Governos, empresas e usuários precisam ter segurança contratual quanto a tecnologia que adotam. Tanto cliente quanto fornecedor podem sair prejudicados se as regras de aquisição e utilização não estiverem claras. Modos de licenciamento mais antigos não são adequados.

Uma patente favorece economicamente o criador do programa, mas impede o aprimoramento e compartilhamento da tecnologia. Outras formas de propriedade intelectual, como marcas registradas e *copyright*, garantem a autoria e estratégias de comercialização, mas por si só não atendem as peculiaridades do código aberto. Licenças específicas foram e ainda são elaboradas para esse novo modelo de produção.

Licenças GNU

A licença de código aberto mais tradicional é a **GPL**: *GNU General Public License*. Como o nome implica, essa licença está diretamente relacionada ao projeto GNU, mas pode ser utilizada por qualquer projeto de código aberto. A GPL gira em torno de quatro liberdades principais:

- A liberdade de utilizar o programa para qualquer fim.
- A liberdade de modificar o programa para atender suas necessidades.
- A liberdade de compartilhar o programa com amigos e vizinhos.
- A liberdade de compartilhar as modificações realizadas.

A última liberdade está associada ao conceito de **copyleft**. Ou seja, o programa possui *copyright inverso*. No lugar de limitar o uso do código fonte, o desenvolvedor está liberado a manter o código aberto. Especificamente no caso da licença GPL, o desenvolvedor pode usar e distribuir um software baseado em código GPL somente se esse código modificado também for licenciado sob a GPL. Para a *Free Software Foundation*, essa característica define se um programa de código aberto pode ser chamado de *software livre*.

Free Software Foundation

O projeto GNU e seus subprojetos são mantidos pela [Free Software Foundation](#), ou simplesmente FSF. A FSF é uma entidade sem fins lucrativos que promove a liberdade dos usuários de computador e a defesa dos direitos dos usuários de software livre.

A licença GPL completa pode ser obtida em <http://www.gnu.org/licenses/gpl.html>. A Free Software Foundation também oferece outros tipos de licenças livres:

- *GNU Lesser General Public License (LGPL)*: Uma versão menos rigorosa que a GPL. É indicada para bibliotecas – componentes de programas – abertas que pretendem substituir versões proprietárias equivalentes. Um programa proprietário pode utilizar uma biblioteca LGPL sem necessidade de tornar-se código aberto ou livre.
- *GNU Affero General Public License (AGPL)*: Em contextos onde o programa não é fornecido da maneira tradicional, como em sistemas na Internet, a licença AGPL garante o acesso ao seu código.
- *GNU Free Documentation License (FDL)*: Destinada para conteúdos de manuais e livros. Assim como no caso dos programas, essa licença garante a cópia, redistribuição com ou sem modificações, comercialmente ou não.

As licenças GNU não são as únicas licenças de código aberto. Como a Free Software Foundation, a [Open Source Initiative](#) é uma entidade sem fins lucrativos que promove a difusão do código aberto. Enquanto a FSF é menos flexível quanto ao conceito de código livre e publica sua própria licença, a Open Source Initiative procura difundir todas as modalidades de programas não proprietários.

Open Source Initiative

A principal finalidade da [Open Source Initiative](#) (ou simplesmente OSI) é publicar a definição formal do que é código aberto. Resumidamente, código aberto é o código fonte que pode ser *utilizado, modificado e compartilhado* sem restrições. A OSI não publica licenças, mas avalia licenças disponíveis e as aprova ou desaprova como licenças de código aberto. Segundo a OSI, a própria GPL é uma licença genuinamente de código aberto.

Para uma licença ser aprovada pela OSI, ela deve atender aos critérios específicos da definição de código aberto mantida pela OSI. São eles:

1. *Distribuição Livre*. A licença não poderá restringir nenhuma das partes de vender ou dar o programa como um componente de uma seleção de programas de diferentes fontes. A licença não pode exigir pagamento de *royalty* ou outra taxa pela venda.

2. *Código Fonte*. O programa precisa incluir o código fonte, e precisa permitir a distribuição tanto em código fonte quanto em formato compilado. Quando o código fonte não acompanhar o produto, será necessário existirem meios suficientemente divulgados para obter o código fonte, preferivelmente a cobrar não mais que seu custo de reprodução ou copiando via Internet sem custo adicional.
3. *Trabalhos Derivados*. A licença deve permitir modificações e trabalhos derivados, e precisa permitir que estes sejam distribuídos sob os mesmos termos da licença original
4. *Integridade do Código Fonte do Autor*. A licença pode impedir a distribuição do código fonte modificado *apenas* se permitir a distribuição de emendas (*patch files*) com a finalidade de modificar o programa durante a compilação. A licença deve permitir explicitamente a distribuição de programas compilados a partir do código fonte modificado. A licença pode exigir que trabalhos derivados tenham um nome ou versão diferentes do original.
5. *Nenhuma Discriminação Contra Pessoas ou Grupos*. A licença não pode discriminar nenhuma pessoa ou grupo de pessoas.
6. *Nenhuma Discriminação Contra Campos de Atuação*. A licença não pode impedir o uso do programa em determinados campos de atuação. Por exemplo, ela não pode impedir que o programa seja utilizado em um negócio ou em pesquisas genéticas.
7. *Distribuição da Licença*. As regras anexadas ao programa se aplicam a quem o programa for redistribuído, sem necessidade de utilizar uma licença adicional para essas partes.
8. *A Licença não Pode Ser Específica a um Produto*. As regras atribuídas ao programa não podem depender de seu vínculo a uma distribuição em particular. Se o programa é extraído da distribuição e utilizado ou distribuído dentro dos termos de sua licença, todas as partes a quem o programa é redistribuído possuirão os mesmos direitos da licença original do programa.
9. *A Licença Não Pode Restringir Outro Programa*. A licença não pode impor restrições a outros programas distribuídos com o programa licenciado. Por exemplo, a licença não pode exigir que todos os programas distribuídos na mesma mídia sejam de código aberto.
10. *A Licença Precisa Ser Tecnicamente Neutra*. Nenhuma disposição da licença pode estar associado a uma tecnologia em particular ou modalidade de interface.

Esses são os critérios para uma licença ser chamada de código aberto. Portanto, nem todo programa que oferece algum tipo de acesso ao seu código fonte pode ser chamado de código aberto. Existem muitas licenças aprovadas, dentre as quais destacam-se:

- *Apache License 2.0*
- *BSD 3-Clause New* or Revised license**
- *BSD 2-Clause Simplified* or FreeBSD license**
- *GNU General Public License (GPL)*
- *GNU Library or Lesser* General Public License (LGPL)**
- *MIT license*
- *Mozilla Public License 2.0*
- *Common Development and Distribution License*
- *Eclipse Public License*

Enquanto que algumas licenças, como a GPL, proíbem que o programa seja modificado e redistribuído sem que o usuário tenha acesso ao código fonte, outras não são tão rigorosas. Um programa com a licença BSD, por exemplo, poderá ser modificado e redistribuído sem o código fonte, desde que a licença BSD original seja abandonada.

Todas as licenças de código aberto aprovadas pela Open Source Initiative podem ser

consultadas em <http://opensource.org/licenses>.

Outras Licenças

A cultura do trabalho colaborativo se expandiu para além do desenvolvimento tecnológico. Hoje, é comum que textos originais, obras de arte e outros trabalhos sejam publicados com licenças livres. Destacam-se as licenças [Creative Commons](#), que permitem ao autor decidir de que maneira sua obra poderá ser utilizada, se poderá ser modificada e compartilhada.

1.4: Habilidades em TIC e Trabalhando no Linux

Peso: 2

A menos que se viva em uma comunidade extremamente isolada, a Tecnologia da Informação e Comunicação – **TIC** – está presente em todos os aspectos da vida de uma pessoa. Do lazer ao trabalho, é difícil localizar uma atividade que não envolva direta ou indiretamente o uso de um recurso computacional.

Sendo irrevogável a presença da tecnologia, é importante conhecer mais sobre ela. Se essa recomendação é válida para qualquer pessoa, torna-se uma exigência quando se trata de um profissional de TIC.

O Computador

Apesar da tecnologia estar tão próxima, sua complexidade e diversidade pode tornar difícil a compreensão de conceitos importantes. O próprio conceito de computador passou por muitas transformações ao longo de décadas. No início do século XX, o computador era a pessoa encarregada de calcular e revisar longas séries de operações matemáticas, para os mais diversos fins. Nessa época, o matemático inglês **Alan Turing** desenvolveu a ideia de uma máquina que poderia executar o trabalho de um computador.

O conceito, de maneira simplificada, propunha a ideia de uma máquina capaz de ler os valores e as operações a serem realizadas com eles. Esses valores e operações seriam armazenados sequencialmente, como furos numa fita. Os furos e a ausência deles formariam um padrão binário de representação numérica, capacitando o computador a lidar com toda informação que possa ser expressa numericamente.

A agulha que lê os valores e operações a partir dos furos na fita é responsável por mudar a operação e copiar os valores para os registradores do processador, que efetua a operação. O resultado da operação é armazenado como furos no próximo espaço vazio da fita, que por sua vez poderá ser utilizado numa nova operação.

Tal conceito pode ser aplicado num mecanismo meramente mecânico, mas a medida que a complexidade aumenta, sua construção torna-se inviável. O avanço da eletrônica e dos *chips* – minúsculos conjuntos de transistores utilizados para realizar operações lógicas – tornou possível o desenvolvimento de processadores muito complexos e velozes. Outros dispositivos eletrônicos, como memórias RAM e discos rígidos, exercem a função da fita perfurada. O conceito da máquina de Turing permanece o mesmo nos computadores modernos, agora operando com sinais eletrônicos e meios magnéticos e óticos.

O inconveniente é que, à medida em que se aumenta a versatilidade e componentes do computador, mais complexa se torna sua operação. Por isso também é importante a elaboração de meios que tornem essa operação mais acessível.

Computação Pessoal

Durante muito tempo, computadores eram restritos à grandes empresas e instituições de pesquisa. A partir do final dos anos 1970, equipamentos domésticos passaram a ser fabricados e surgiu o usuário comum.

O usuário comum, apesar de não compreender o funcionamento de um computador em sua totalidade, é capaz de interagir com ele se existir um conjunto simplificado de instruções de operações.

Interagindo com o computador

Esse conjunto simplificado de instruções é a chamada *interface computador/usuário* ou simplesmente *interface*. A interface traduz a solicitação feita pelo usuário ao computador, que por sua vez gera informações que serão traduzidas pela interface para serem apresentadas ao usuário.

Existem muitos tipos de interface de usuário. Em uma de suas formas mais simples, a entrada de informações pode ser feita por um painel com alguns botões e o resultado da computação ser exibido com luzes piscantes. Em contrapartida, já são comuns interfaces que identificam o movimentos sutis do usuário e exibem sofisticadas imagens em telas de alta resolução.

Das muitas interfaces já criadas, talvez a mais consolidada seja a interface de texto. Apesar de muitos usuários estarem mais acostumados a utilizar um apontador controlado por um *mouse* sobre ilustrações exibidas na tela, a interface de texto é a muito utilizada por profissionais do universo Unix e outros sistemas operacionais.

Terminal

A interface de texto é chamada **terminal** ou **console**. Estes nomes vêm da época em que era utilizado um equipamento chamado terminal, que era conectado ao computador para operá-lo. O console era uma tela simples que exibia informações de texto sobre o computador. Também é comum se referir à interface de texto como **linha de comando**.

Hoje, existe um programa chamado *emulador de terminal* que pode ser utilizado para trabalhar linha de comando. Os diferentes ambientes de desktop gráfico oferecem várias opções de emulador de terminal, como o *Konsole*, o *Terminal do Gnome* ou o *xterm*. Outra possibilidade é alternar para o console de texto que permite fazer login para a linha de comando, usando a combinação de teclas **Ctrl + Alt + F2**. A principal vantagem da linha de comando é sua simplicidade e versatilidade. Também são mais simples da perspectiva do uso dos recursos de hardware, pois é necessário muito menos níveis de tradução entre os comandos de texto e as instruções compreendidas pelo computador. Mesmo que a princípio pareça contraditório, a linha de comando pode ser mais versátil que um aplicativo gráfico. Os comandos podem ser combinados entre si, permitindo um controle apurado e produção de resultados elaborados.

Essas características tornam a linha de comando uma boa interface para um usuário profissional. Enquanto uma interface gráfica exige uma série de programas adicionais e equipamentos mais complexos, o console é parte de todo sistema operacional Unix e pode ser utilizado mesmo quando uma interface gráfica não está disponível.

Contudo, existem tarefas onde aplicativos em interfaces gráficas são mais apropriados. É o caso do navegador de Internet, que hoje é uma das principais interfaces de operação de um computador. As páginas de Internet muitas vezes funcionam como aplicativos, num ambiente chamado *Plataforma Web*.

A plataforma Web

Muitas atividades pessoais e profissionais são desempenhadas no navegador. Apesar da praticidade, alguns cuidados devem ser tomados quando utilizando este ambiente. Os principais navegadores oferecem recursos para preservar a segurança e privacidade do usuário.

Segurança

Ao utilizar qualquer tipo de comunicação em rede, sempre existe a possibilidade de que a informação transmitida e recebida esteja sendo interceptada. Para evitar que um interceptador possa ler as informações, é importante verificar se o protocolo **HTTPS** está sendo utilizado pelo navegador. Sempre que informações sensíveis, como senhas e outros dados pessoais, estejam sendo transmitidas, o termo **https://** deve aparecer ao lado do endereço da página. Assim, as informações serão criptografadas e apenas a origem e o destino da comunicação serão capazes de ler as informações.

A criptografia praticamente elimina a possibilidade de uma senha ou qualquer outra informação sensível ser interceptada. Mesmo assim, existe a possibilidade de senhas serem

roubadas pela quebra de algoritmos defasados ou pelo armazenamento incorreto das senhas no provedor de serviços. Para mitigar o dano em caso de roubo de senhas, é recomendável usar uma senha diferente em cada serviço. Desse modo, a senha roubada poderá ser utilizada somente na conta comprometida. O gerenciador de senhas presente nos navegadores modernos armazenam as diferentes senhas de modo seguro, mas é muito importante que a senha mestra desses gerenciadores seja forte, de preferência maiores que 15 caracteres, contendo letras, números e outros símbolos e não sejam compartilhadas ou armazenadas em nenhum meio inseguro.

Privacidade

Além dos ataques por interceptação, é possível que outra pessoa com acesso ao computador possam obter as informações de outros usuários que utilizaram o navegador. Para evitar que o navegador preserve informações como dados pessoais, senhas, histórico de visitas, etc, recomenda-se utilizar o *modo de navegação privativo*. Este recurso, disponível em todos os navegadores modernos, é indispensável quando se utiliza um computador de outra pessoa ou de uso público.

As janelas no modo privativo também não armazenam *cookies* de maneira persistente. Os cookies são pequenos marcadores que o servidor Web usa para identificar cada navegador individualmente. Enquanto que cookies são úteis para certos fins, como preservar uma cesta de compras numa loja online ou armazenar preferências do usuário, eles também podem ser usados para identificar e monitorar o usuário sem o seu consentimento.

Para além da computação pessoal, outros conceitos relativos às Tecnologias da Informação e Comunicação devem ser considerados por um profissional da área. Anterior à computação pessoal, a computação corporativa possui características próprias e exige mais conhecimento.

Computação corporativa

Para um profissional de TIC, não basta escolher os equipamentos e programas que atendem às suas necessidades pessoais, mas aqueles que atendem às necessidades de sua atividade profissional ou da empresa para qual trabalha.

Num computador pessoal, além de ser desejável que os equipamentos e programas não apresentem falhas, também é desejável que estejam atualizados com os recursos mais recentes. Distribuições como a *Linux Mint* e *Fedora* procuram oferecer as versões mais recentes dos programas em seus pacotes. Contudo, muitos dos recursos mais recentes não foram exaustivamente testados, e podem causar alguma instabilidade que será corrigida tão logo seja detectada.

Num computador de uso corporativo, também é desejado que estejam presentes os recursos mais recentes. Contudo, instabilidades devem ser completamente rechaçadas. Por exemplo, uma instabilidade que torne um servidor inacessível, mesmo que por alguns instantes, pode causar grandes prejuízos.

É para esse tipo de uso que existem as chamadas *distribuições corporativas*. Nessas distribuições, como *Debian*, *CentOS* e *Ubuntu Server*, há somente programas exaustivamente testados, o que praticamente elimina a ocorrência de falhas.

Além da possibilidade de implementar uma solução interna para as necessidades da corporação ou entidade, o profissional de TIC pode transferir o ônus de operar a infraestrutura de TI para um provedor externo, que muito provavelmente empregará as mesmas distribuições corporativas que seriam utilizadas internamente, porém em seus próprios servidores remotos. Estes equipamentos são, via de regra, mais eficientes do que aqueles que uma empresa de pequeno e médio porte poderia manter. Esse tipo de provedor de serviços de TIC, muitas vezes denominado *provedor de nuvem*, costuma oferecer diferentes níveis de serviços:

- **IaaS:** *Infrastructure as a Service*, Infraestrutura como serviço. O provedor é responsável por oferecer o hardware e a conectividade de rede, cabendo ao contratante a administração do sistema operacional.
- **PaaS:** *Platform as a Service*, Plataforma como serviço. O provedor oferece um ambiente de desenvolvimento e implementação já preparado, onde o contratante já tem a

disposição os servidores Web e de banco de dados, por exemplo.

- **SaaS:** *Software as a Service*, Software como serviço. A aplicação final é fornecida pelo provedor, bastando ao contratante apenas utilizar a aplicação pronta. É o caso de serviços como *webmail* corporativo e *groupwares* colaborativos.

Uma das vantagens que um provedor de nuvem oferece é a escalabilidade sob demanda do serviço. Diante da necessidade, é possível solicitar ao provedor que aumente ou reduza a capacidade contratada. Isso é possível pois os serviços em nuvem costumam utilizar tecnologias de *virtualização* que permitem a alocação dinâmica de recursos. Muitos dos provedores de nuvem empregam tecnologias de virtualização oferecidas pelo próprio Linux, como o **KVM** ou **Xen**.

Virtualização

Virtualização é a capacidade de executar simultaneamente mais de um sistema operacional num mesmo computador. Esse recurso possibilita um melhor aproveitamento dos recursos de hardware em máquinas de grande porte. Mesmo que estejam rodando na mesma máquina, sistemas virtualizados são completamente independentes entre si e têm as mesmas finalidades e modos de utilização de um sistema tradicional não virtualizado.

As soluções de código aberto são muito diversas e versáteis, mas podem não ser a resposta para todas as questões. Certas necessidades demandam sistemas operacionais ou programas proprietários, que não possuem equivalente em código aberto. A ausência de pagamento pelo programa não implica que não haverão outros custos. Além do custo da implementação, é provável que será necessário contratar suporte, manutenção e treinamento. Somente um estudo de cada caso pode definir a opção mais adequada.

Tópico 2: Localizar-se num Sistema Linux

Peso: 9

2.1: O Básico da Linha de Comando

Peso: 3

A maneira tradicional de interagir com um computador com Linux – especialmente um servidor de rede – é usando a *linha de comando*. A linha de comando apresenta o *prompt do shell* indicando que o sistema está pronto para receber instruções. Normalmente, o prompt terminado com o caractere `$` indica que é um usuário comum que está utilizando o sistema. Quando terminado com o caractere `#`, indica tratar-se do usuário *root*.

O usuário root

O usuário root pode realizar todo tipo de operações no sistema. Por questões de segurança e privacidade, toda tarefa e arquivo têm um usuário “dono”. Cada usuário pode interferir apenas nas tarefas e arquivos que lhe pertencem. Já o usuário root pode interferir em qualquer tarefa ou arquivo.

A linha de comando é aberta com um programa chamado *emulador de terminal*. Normalmente, é chamado apenas de *terminal* no menu de aplicativos.

O shell Bash

No ambiente de linha de comando, o *shell* é o programa que faz a intermediação entre o usuário e os recursos do computador, como se fosse um ambiente de programação em tempo real para executar tarefas. O shell padrão na maioria das distribuições Linux é o **Bash** (*Bourne Again Shell*), ao qual os procedimentos aqui apresentados referem-se.

Um comando do shell pode consistir somente no nome do programa que se quer executar. Além daqueles oferecidos pelos demais programas instalados no computador, estão disponíveis diversos comandos embutidos do shell. Um dos comandos embutidos mais simples é o `echo`, que simplesmente exibe um conteúdo na tela:

```
$ echo Linux Essentials
Linux Essentials
```

O sinal `$` é mostrado apenas para indicar o prompt do shell, não deve ser digitado. Após escrever o comando `echo Linux Essentials`, o comando deve ser enviado pressionando a tecla *Enter*. A resposta do comando será exibida logo abaixo do comando. Para evitar que o comando `echo` quebre a linha (coloque o prompt na linha de baixo) no final da saída, utiliza-se `echo -n Linux Essentials`. Para que o comando `echo` utilize caracteres especiais, deve ser informada a opção `-e`. Por exemplo, para criar uma quebra de linhas entre as duas palavras:

```
$ echo -e "Linux\nEssentials"
Linux
Essentials
```

O termo `\n` representa uma quebra de linha.

Alguns comandos podem usar tantas opções e argumentos que chegam a ocupar mais de uma linha. Para quebrar um comando em mais de uma linha, se utiliza o sinal `\` (barra invertida) antes de cada quebra de linha. Por exemplo, para quebrar o comando `find /usr/share/doc -ipath '*howto*' -type d -ctime -100 -exec ls -ld '{}' ';'` em duas linhas:

```
$ find /usr/share/doc -ipath '*howto*' -type d \
-ctime -100 -exec ls -ld '{}' ;'
```

O comando `find` é utilizado para localizar arquivos no sistema de arquivos e é um exemplo de programa que pode receber muitos argumentos e opções. Um argumento ou parâmetro de comando é um termo esperado pelo programa para realizar a ação desejada. No exemplo anterior, foi utilizado o argumento `/usr/share/doc` para indicar onde o `find` deveria iniciar a busca.

Uma opção de comando é uma letra ou palavra, normalmente precedida de um traço, que modifica o comportamento de um comando. No exemplo anterior, duas opções utilizadas foram `-ipath '*howto*'` e `-type d`, que indicam alguns filtros do `find`.

O esperado para a maioria dos comandos é que produzam a saída com o resultado imediatamente após o pressionamento da tecla *Enter*. Dependendo das circunstâncias, a saída do comando pode demorar mais que o esperado. Caso um programa esteja demorando muito para terminar ou não se quer esperar sua conclusão por qualquer outro motivo, pressionar a combinação de teclas **Ctrl + c** fará com que o programa seja interrompido e o prompt do shell fique novamente disponível.

Variáveis

As variáveis usadas no shell são semelhantes às usadas em linguagens de programação. Uma variável é um nome que guarda um valor, que pode ser letras ou números. Nomes de variáveis são limitados a caracteres alfanuméricos, ou seja, pode conter letras e números, mas deve sempre começar com uma letra. A atribuição de um valor para uma variável é feito com o sinal de igual, onde o nome da variável fica à esquerda e o conteúdo da variável fica à direita:

```
$ lpi="Linux Professional Institute"
```

Não devem haver espaços antes ou depois do sinal de igual. Se houverem espaços no conteúdo da variável, é importante utilizar as aspas duplas ou simples para não confundir o shell. O valor de uma variável pode ser exibido colocando o sinal `$` à frente do nome:

```
$ echo $lpi
Linux Professional Institute
```

Variáveis podem ser criadas por usuários comuns ou pré-definidas pelo sistema operacional. Tanto as variáveis pré-definidas quanto variáveis definidas para a sessão atual podem ser utilizadas por programas para obter configurações importantes do sistema. Quando definidas numa sessão interativa do shell, as variáveis também são chamadas de *variáveis de ambiente*. O próprio shell utiliza variáveis de ambiente para definir diversas de suas configurações.

O shell interpreta a primeira palavra fornecida como um comando interno ou o nome de um programa. A localização para o programa precisa ser fornecida, a menos que este esteja localizado em um dos diretórios contidos na variável global `PATH`. Se o programa está no diretório atual e fora dos diretórios contidos em `PATH`, seu nome deve ser precedido por `./`, por exemplo `./script.sh`.

O conteúdo da variável `PATH` pode ser exibido com o comando `echo`:

```
$ echo $PATH
/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin
```

Caso o programa ou script esteja em um diretório que não consta na variável `PATH`, o usuário pode incluí-lo para seu uso próprio temporariamente, até o fim da sessão do shell. Isso evita a necessidade de especificar o caminho para o programa, como no caso do `./` em `./script.sh`, ou um caminho mais longo. Caso o programa esteja no diretório atual, primeiro pode ser utilizado o comando `pwd` para obter o caminho do diretório atual por extenso:

```
$ pwd
/home/luciano
```

O caminho `/home/luciano` pode ser incluído na variável `PATH` com o comando `export`:

```
$ export PATH=/home/luciano:$PATH
```

Dessa forma, até o fim da sessão atual do shell, a variável `PATH` terá um novo diretório:

```
$ echo $PATH
/home/luciano:/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin
```

Essa ação costuma ser necessária somente se o usuário escreve os próprios programas. Por padrão, todos os programas ficam nos diretórios originais definidos na variável `PATH`. A localização específica de um comando pode ser determinada com outro comando, `which`:

```
$ which man
/usr/bin/man
```

Outra maneira de determinar a localização de um comando no sistema de arquivos é utilizar o comando `type`:

```
$ type man
man é /usr/bin/man
```

Caso o comando informado seja um comando interno do Bash, a saída produzida pelo `type` será diferente:

```
$ type echo
echo é um comando interno do shell
```

No caso do programa `man`, o Bash foi capaz de identificar sua localização pois o diretório `/usr/bin` consta na variável `PATH`.

Todas as variáveis globais são escritas em letras maiúsculas. Em ambientes Unix, há diferença entre letras maiúsculas e minúsculas. Portanto, utilizar *Path* ou *path* não irá funcionar. As variáveis globais são criadas no carregamento do sistema e exportadas com o comando `export` para que estejam disponíveis em todas as sessões futuras do shell.

Existem diversas outras variáveis globais definidas ao iniciar o shell que podem orientar o comportamento dos programas. A variável `HISTFILE`, por exemplo, armazena o caminho para o histórico de comandos digitados. Por padrão, o histórico é armazenado no arquivo `.bash_history`, no diretório pessoal do usuário. O histórico é útil para recuperar comandos previamente digitados. Ao pressionar a seta para cima, os comandos digitados anteriormente são exibidos. Todo histórico de comandos é exibido com o comando `history`.

Substituição de comandos

A saída de um comando pode ser armazenada dentro de uma variável, usando as aspas invertidas:

```
$ agora=`date`
```

O comando `date` exibe a data e a hora do sistema. Nesse caso, a data e a hora foram armazenadas na variável `agora`, cujo conteúdo pode ser exibido com o comando `echo`:

```
$ echo $agora
seg fev 25 21:38:15 -03 2019
```

É possível também usar a saída de um comando como argumento para outro, usando aspas invertidas:

```
$ echo "A data atual é `date`"  
A data atual é seg fev 25 21:38:54 -03 2019
```

Alternativamente, pode ser usado o formato `$()` no lugar das aspas simples. O mesmo resultado dos exemplos anteriores seria obtido com `$(date)` no lugar de ``date``.

Englobamento

As operações com arquivos e diretórios permitem o uso de caracteres curinga, que são padrões de substituição de caracteres. O caractere `*` substitui qualquer sequência de caracteres:

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

O caractere `?` substitui apenas um caractere:

```
$ ls /dev/sda?  
/dev/sda1 /dev/sda2 /dev/sda3 /dev/sda4
```

O uso de colchetes permite indicar uma lista de caracteres:

```
$ ls /dev/hd[abc]  
/dev/hda /dev/hdb /dev/hdc
```

Chaves indicam uma lista de termos separados por vírgula:

```
$ ls /dev/{hda,fd0}  
/dev/fd0 /dev/hda
```

O uso de exclamação antes de um curinga o exclui da operação:

```
ls /dev/tty[!56789]  
/dev/tty0 /dev/tty1 /dev/tty2 /dev/tty3 /dev/tty4
```

Curingas precedidos de barra invertida (`\`) não realizam substituição. São denominados caracteres *escapados*. Entre aspas duplas, apenas os caracteres especiais (```) e `$` têm efeito. Entre aspas simples, nenhum caractere especial tem efeito.

Comandos sequenciais

A grande maioria das tarefas depende da execução de mais de um comando. Para executar três comandos em sequência, independente do resultado de cada um, utiliza-se o formato:

```
comando1 ; comando2 ; comando3
```

Executar o comando seguinte apenas se o anterior foi bem sucedido (se retornou 0):

```
comando1 && comando2 && comando3
```

Executar o comando seguinte apenas se o anterior não foi bem sucedido (se retornou diferente de 0):

```
comando1 || comando2 || comando3
```

Todos os comandos obedecem à essas formas padronizadas de utilização. No entanto, cada comando possui uma finalidade e opções diferentes, sendo necessário consultar a sua documentação para conhecer detalhadamente sua utilização.

2.2: Utilizando a Linha de Comando para Obter Ajuda

Peso: 2

Em função do grande número de comandos disponíveis no ambiente Linux, é bastante difícil memorizar a finalidade de todos os comandos e opções. Para facilitar a digitação de um comando ou de um caminho existente, o shell oferece o recurso de autocompletar, acionado com a tecla `[TAB]`. Após digitar as primeiras letras de um comando ou caminho de diretório, a tecla `[TAB]` completa a linha de acordo com os comandos e caminhos encontrados.

Muitas das dúvidas a respeito da utilização de um comando podem ser sanadas ao fornecer a opção `--help` ao próprio comando. Contudo, a fonte de informação mais completa sobre um programa encontra-se em seu manual. Praticamente todos os comandos e arquivos de configuração no Linux acompanham um manual. Esse manual está acessível por intermédio do comando `man`, que demonstra em detalhes as funções do item em questão. Para ver um manual, basta usar o comando `man`, tendo o comando ou arquivo como argumento.

O comando info

O comando `info` é uma espécie de alternativa aos manuais `man`. Além do comando `man`, pode haver documentação disponível pelo `info`. Em geral, informações disponíveis em páginas *info* também estão disponíveis em páginas de manual, porém de forma menos detalhada. Por padrão, os arquivos desse tipo de documentação são armazenadas em `/usr/share/info`.

Em sua maioria, os manuais têm a seguinte organização:

- **Nome:** Assunto do manual seguido por uma descrição breve;
- **Sinopse:** A sintaxe do comando;
- **Descrição:** Descrição detalhada;
- **Opções:** Revisão de todas as opções e suas funções;
- **Arquivos:** Arquivos relacionados ao assunto;
- **Veja também:** Outros manuais relacionados ao tópico.

A navegação dentro do manual pode ser feita com as setas do teclado e com as teclas *Page Up* e *Page Down*. Também é possível fazer uma busca textual ao pressionar a tecla `/` (barra), que abre um campo na parte de baixo do terminal para digitar o termo a ser buscado. A busca é então realizada ao pressionar `Enter`. Para localizar a próxima ocorrência, basta pressionar a tecla `n`. Para encontrar a ocorrência anterior, basta pressionar a tecla `?`. Pressionar a tecla `q` finalizar o leitor do manual e retorna para o prompt do shell.

Procurar por manuais

É possível buscar ocorrências de um termo na seção *nome* dos manuais com o comando `apropos`. Esse comando retorna a uma descrição breve para cada ocorrência encontrada e o nome do respectivo comando ou arquivo.

Uma maneira de localizar os manuais de referência para um determinado programa é usar o comando `whatis`. O banco de dados do comando `whatis` armazena a seção *nome* dos manuais do sistema. O banco de dados geralmente é atualizado por um agendamento de sistema. Para cada recurso de manual localizado, o `whatis` mostra uma breve descrição:

```
$ whatis man
man (1)      - an interface to the on-line reference manuals
man (7)      - macros to format man pages
```

Os números entre parênteses referem-se à seção a qual pertence o manual. As seções existentes são listadas a seguir:

- **Seção 1:** Programas disponíveis ao usuário;
- **Seção 2:** Funções de Sistema Unix e C;
- **Seção 3:** Funções de bibliotecas da linguagem C;
- **Seção 4:** Arquivos especiais (dispositivos em /dev);
- **Seção 5:** Convenções e formatos de arquivos;
- **Seção 6:** Jogos;
- **Seção 7:** Diversos (macros textuais etc.);
- **Seção 8:** Procedimentos administrativos (daemons, etc).

Para acessar um item em uma seção específica, o número da seção precede o nome do item. Por exemplo, acessar o manual de `printf` na seção número 3:

```
man 3 printf
```

Por padrão, os arquivos dos manuais são armazenadas em `/usr/man` e `/usr/share/man`, em subdiretórios correspondentes à cada seção. Outros locais podem ser especificados com a variável `MANPATH`.

O comando `whereis` pode ser utilizado para localizar o arquivo do manual. Além disso, ele exibe o diretório onde está o comando e, se houver, o código fonte:

```
$ whereis find
find: /usr/bin/find /usr/bin/X11/find /usr/share/man/man1/find.1.gz
```

Para localizar arquivos e diretórios em geral, pode ser utilizado o comando `locate`. Um nome ou parte do caminho é fornecido como argumento, que exibirá todas as ocorrências correspondentes.

Outras documentações

Projetos GNU geralmente incluem documentação como guias e exemplos de utilização. Esses documentos não têm um formato universal, mas podem ser oferecidos em texto puro, documentos HTML ou PDF. Estes arquivos podem ser encontrados em `/usr/share/doc`, em diretórios correspondentes aos programas.

2.3: Utilizando Diretórios e Listando Arquivos

Peso: 2

À exceção do usuário administrador `root`, cujo diretório é `/root`, o diretório inicial é o diretório pessoal do usuário, localizado em `/home`, que leva o nome do usuário (utilizado para entrar no sistema). Para mudar de diretório, é utilizado o comando `cd`, seguido do caminho absoluto ou relativo para o diretório desejado.

Os caminhos para diretórios e arquivos podem ser acessados tanto por seu caminho absoluto quanto pelo relativo. Caminhos absolutos são aqueles iniciados pela barra da raiz (`/`) e caminhos relativos são aqueles que tomam por referência o diretório atual. O ponto (`.`) refere-se ao diretório atual, e os dois pontos (`..`) referem-se ao diretório que contém o diretório atual.

Sem argumentos, o comando `cd` leva para o diretório pessoal do usuário. Outra maneira de indicar o diretório pessoal é utilizar o sinal `~`. Este pode ser utilizado, por exemplo, para ir até um diretório no diretório pessoal, `cd ~/Documentos`, ou para indicar um arquivo no diretório pessoal, `ls ~/Documentos/doc.txt`.

O comando `ls` é usado para listar arquivos e conteúdo de um diretório. A opção `-l` exibe detalhes sobre o(s) arquivo(s), `-s` mostra o tamanho em bytes e `-d` mostra as propriedades de um diretório, não seu conteúdo. Exemplo de saída de `ls -l`:

```
$ ls -l /
total 64
lrwxrwxrwx 1 root root 7 fev 7 2018 bin -> usr/bin
dr-xr-xr-x 7 root root 4096 fev 26 10:06 boot
drwxr-xr-x 20 root root 4300 fev 26 10:10 dev
```

```

drwxr-xr-x. 194 root root 12288 fev 26 10:10 etc
drwxr-xr-x. 4 root root 4096 fev 7 2018 home
lrwxrwxrwx 1 root root 7 fev 7 2018 lib -> usr/lib
lrwxrwxrwx 1 root root 9 fev 7 2018 lib64 -> usr/lib64
drwx----- 2 root root 16384 nov 15 2016 lost+found
drwxr-xr-x. 2 root root 4096 fev 7 2018 media
drwxr-xr-x. 2 root root 4096 set 18 00:04 mnt
drwxr-xr-x. 4 root root 4096 fev 7 2018 opt
dr-xr-xr-x 209 root root 0 fev 26 10:08 proc
dr-xr-xr-x. 27 root root 4096 fev 26 10:53 root
drwxr-xr-x 53 root root 1560 fev 26 11:08 run
lrwxrwxrwx 1 root root 8 fev 7 2018 sbin -> usr/sbin
drwxr-xr-x. 2 root root 4096 fev 7 2018 srv
dr-xr-xr-x 13 root root 0 fev 26 10:10 sys
drwxrwxrwt 14 root root 320 fev 26 12:13 tmp
drwxr-xr-x. 14 root root 4096 ago 28 08:41 usr
drwxr-xr-x. 26 root root 4096 ago 30 12:43 var

```

Diversas informações são exibidas à esquerda do nome de cada item, como as permissões de acesso, o usuário dono do arquivo e a data de modificação.

Arquivos que começam com um ponto “.” não são exibidos pelo comando `ls`. Para que esses arquivos também sejam exibidos, é necessário utilizar o `ls` com a opção `-a`. Também é possível utilizar englobamento nas listagens. Por exemplo, para listar todos os arquivos que terminem com o sufixo `.txt`, utiliza-se `ls *.txt`.

Para obter uma lista com os conteúdos de todos os subdiretórios, pode ser utilizado o comando `ls -R` ou o comando `find`. O principal comando de localização de arquivos em linha de comando é o `find`, cuja sintaxe básica é *find diretório critério*.

O argumento *diretório* indica onde o `find` deve iniciar a busca, e o *critério* pode ser o nome do arquivo ou diretório a ser procurado ou uma regra para a busca. Existem dezenas de critérios de busca, os mais comuns são:

- `-type x`: A letra *x* define o tipo do arquivo (*d* para diretório, *f* para arquivo comum e *l* para link simbólico).
- `-name nome`: O nome do arquivo. Pode ser um nome parcial, como `'foto*'`, para localizar todos os nomes que começam com o termo *foto*. É recomendável utilizar as aspas simples nesse caso.
- `-user usuário`: O usuário dono do arquivo.
- `-atime -/+n`: Arquivo acessado antes ou após *n*. *n* corresponde à expressão *n * 24 horas*. Ou seja, se for utilizado `-atime -1` serão listados os arquivos acessados a menos de 24 horas. Se for utilizado `-atime +2`, serão listados os arquivos acessados a mais de 48 horas.
- `-ctime -/+n`: Arquivo criado antes ou após *n*. Vale o mesmo princípio do `-atime`.
- `-mtime -/+n`: Arquivo modificado antes ou depois de *n*. Vale o mesmo princípio do `-atime`.
- `-amin -/+n`: Arquivo acessado antes ou depois de *n*. *n* corresponde à quantidade de minutos. Por exemplo, `-amin -15` irá listar todos os arquivos acessados a menos de 15 minutos. Com `-amin +30`, serão listados os arquivos acessados a mais de 30 minutos.
- `-cmin -/+n`: Arquivo criado antes ou depois de *n*. Vale o mesmo princípio de `-amin`.
- `-mmin -/+n`: Arquivo modificado antes ou depois de *n*. Vale o mesmo princípio de `-amin`.
- `-newer arquivo`: O arquivo procurado foi criado ou modificado após *arquivo*.
- `-perm modo`: O arquivo procurado tem permissão especificada igual a modo, como as letras *r*, *w* e *x*.
- `-perm -modo`: O arquivo procurado tem todas as permissões listadas em modo.
- `-perm +modo`: O arquivo procurado tem qualquer das permissões listadas em modo.

O `find` pode ser utilizado, por exemplo, para obter uma lista com todos os arquivos criados no diretório pessoal ou em seus subdiretórios nas últimas 24 horas:

```
$ find /usr/lib -type f -ctime -1
```

A opção `-find f` determina que sejam buscados somente arquivos (excluindo diretórios e outros arquivos especiais) e a opção `-ctime -1` determina que a data de criação dos arquivos seja inferior a 1 dia em relação a data atual.

2.4: Criar, mover e apagar arquivos

Peso: 2

O comando `cp` é utilizado para copiar arquivos. Suas opções principais são:

- `-i`: Modo interativo. Pergunta antes de sobrescrever um arquivo.
- `-p`: Copia também os atributos do arquivo original.
- `-r`: Copia recursivamente o conteúdo do diretório de origem.

É importante saber que ao copiar um diretório recursivamente, o uso da barra `/` no final do diretório de origem fará com que apenas o conteúdo do diretório seja copiado para o destino; não usar a barra fará com que o diretório de origem e seu conteúdo sejam copiados no destino.

O comando `mv` move e renomeia arquivos e diretórios. Usado com a opção `-i`, ele pede confirmação antes de sobrescrever um arquivo de destino.

Um arquivo vazio pode ser criado com o comando `touch`. Por exemplo, `touch arquivo.txt` cria o arquivo vazio `arquivo.txt`. O comando `touch` também pode ser utilizado apenas para alterar a data de um arquivo. Usado sem argumentos, `touch` altera a data e a hora de criação e modificação de um arquivo para os valores atuais do sistema. Para alterar apenas a data de modificação, usa-se a opção `-m`, e para alterar apenas a data de acesso, usa-se a opção `-a`. Outros valores de tempo podem ser passados com a opção `-t`.

O comando `mkdir` cria diretórios. Para criar uma árvore de diretórios recursivamente, sem necessidade de criar um a um, usa-se a opção `-p`:

```
mkdir -p caminho/completo/para/diretório
```

Para alterar as permissões do diretório no ato da criação, as mesmas são transmitidas ao `mkdir` com a opção `-m`. Diretórios vazios podem ser apagados pelo comando `rmdir`. Com a opção `-p`, o `rmdir` remove o diretório indicado e os diretórios superiores, desde que estejam vazios.

Para apagar um arquivo, o comando é `rm`. Para apagar diretórios com conteúdo, usa-se `rm -r`. Para forçar a remoção, a opção `-f` é utilizada.

Tópico 3: O Poder da Linha de Comando

Peso: 9

3.1: Armazenamento de arquivos na linha de comando

Peso: 2

Praticamente toda operação realizada no computador envolve a manipulação de arquivos, que muitas vezes precisarão ser transportados ou armazenados em diferentes locais. Nesses casos, é conveniente gerar um arquivo contendo diretórios e outros arquivos, de modo a facilitar seu transporte e otimizar a ocupação de espaço em disco.

No Linux, o principal comando para agregar diferentes arquivos é o comando `tar`. Originalmente desenvolvido para armazenar cópias de segurança em fita, hoje o `tar` também é utilizado para facilitar o armazenamento e distribuição de arquivos em diferentes mídias.

Utilizando o tar

Para criar um arquivo contendo todo o diretório `/etc` e seu conteúdo com o `tar`, podemos usar o comando:

```
tar cvf etc.tar /etc
```

Diferente de outros comandos, a inclusão do traço antes das opções do `tar` é facultativa. As opções fornecidas no exemplo representam:

- `c`: Criar um arquivo;
- `v`: Mostrar cada arquivo conforme é incluído;
- `f`: Especifica o arquivo a ser criado, informado em seguida.

O último argumento é o diretório(s) ou arquivo(s) a ser incluído. Para extrair o conteúdo de um arquivo `tar`, a opção usada é a `x`:

```
tar xvf etc.tar
```

Os arquivos serão extraídos com a árvore de diretórios completa. Este arquivo `.tar`, apesar de agregar vários arquivos, não está comprimido.

Compressão

Os principais comandos de compressão no Linux são o `gzip` e o `bzip2`. Para compactar um arquivo com `gzip`:

```
gzip etc.tar
```

Para comprimir com `bzip2`:

```
bzip2 etc.tar
```

Será criado automaticamente o arquivo `etc.tar.gz` ou `etc.tar.bz2`. A principal diferença entre as duas modalidades de compressão é o algoritmo utilizado. O `gzip` é mais rápido, enquanto que o `bzip2` costuma oferecer melhores taxa de compressão.

A compressão pode ser especificada diretamente com o comando `tar`. Para realizar a compressão com `gzip`, é utilizada a opção `z`:

```
tar czvf etc.tar.gz /etc
```

Para usar *bzip2*, é utilizada a opção *j*:

```
tar cjvf etc.tar.bz2 /etc
```

A descompressão pode ser feita com os comandos *gunzip* (ou *gzip -d*) e *bunzip2* (ou *bzip2 -d*), mas também pode ser feita diretamente com o comando *tar* e com as opções *z* e *j*, respectivamente.

Arquivos zip

Outro formato popular para comprimir arquivos é formato *zip*. Esse tipo de arquivo também pode ser criado pela linha de comando, com o comando *zip*. Assim como o comando *tar*, o comando *zip* aceita caracteres especiais para selecionar os arquivos que serão incluídos:

```
zip documentos.zip *.pdf
```

Neste exemplo, será criado o arquivo *documentos.zip* contendo todos os arquivos do diretório atual terminados com o sufixo *pdf*. Para extrair o conteúdo deste arquivo, é utilizado o comando *unzip documentos.zip*, que criará os arquivos extraídos no diretório atual. Para extrair os arquivos para outra pasta, deve ser utilizada a opção *-d* *diretório*, onde *diretório* é o diretório que receberá os arquivos extraídos.

3.2: Procurar e Extrair Informações de Arquivos

Peso: 3

Um dos comandos mais básicos para examinar o conteúdo de arquivos de texto é o comando *less*. O *less* simplesmente exibe na tela o conteúdo de um arquivo informado como argumento. Por exemplo, o conteúdo do arquivo */proc/cpuinfo*, que contém detalhes sobre o processador do computador, será exibido com o comando *less /proc/cpuinfo*.

Caso o tamanho do arquivo seja maior que o espaço disponível em tela, as teclas de direção para cima e para baixo do teclado podem ser utilizadas para navegação no arquivo. O *less* também permite fazer buscas de texto ao pressionar a tecla */* (barra) no teclado, que permite digitar o texto buscado na parte de baixo da tela e iniciar a procura ao pressionar *Enter*. Para localizar a próxima ocorrência, basta pressionar a tecla *n*. Para encontrar a ocorrência anterior, basta pressionar a tecla *p*. Pressionar a tecla *q* finalizar o *less* e retorna para o prompt do shell.

Redirecionamento

Processos Unix (e Linux) por padrão abrem três canais de comunicação, que os permitem receber e emitir dados. Esses dados, que podem ser um texto legível ou dados binários como áudio e vídeo, podem ser redirecionados de e para outros arquivos ou processos. No caso de programas interativos, o canal de entrada (chamado *standard input* ou *stdin*) costuma ser o próprio teclado. Os canais de saída-padrão (*standard output* ou *stdout*) e de saída de erro (*standard error* ou *stderr*) costumam ser a tela do computador. Os valores numéricos para esses canais são **0** para *stdin*, **1** para *stdout* e **2** para *stderr*. Esses descritores são automaticamente definidos pelos dispositivos virtuais localizados no diretório */dev*: */dev/stdin*, */dev/stdout* e */dev/stderr*.

O fluxo dos dados para redirecionamentos e canalizações numa linha de comando acontece da esquerda para a direita. Para redirecionar a saída-padrão de um comando para um arquivo, utiliza-se o caractere *>* após este, que deve indicar o arquivo que receberá os dados em questão. Por exemplo, pode-se redirecionar o conteúdo do arquivo */proc/cpuinfo* com o comando:

```
cat /proc/cpuinfo > ~/cpu.txt
```

A finalidade do comando *cat* é semelhante à do *less*, mas o *cat* envia um conteúdo para a saída padrão de modo não interativo. No exemplo, a saída do *cat* é o conteúdo do arquivo */proc/cpuinfo*, que foi redirecionado para o arquivo *cpu.txt* no diretório pessoal do usuário. Se o

arquivo `~/cpu.txt` já existe, será sobrescrito. Para adicionar os valores sem apagar o conteúdo existente, usa-se `>>` no lugar de `>`.

O conteúdo redirecionado por padrão é o de *stdout*. Para especificar *stderr*, usa-se `2>`. Para redirecionar ambos simultaneamente, usa-se `&>`.

Canalização

Parte da filosofia Unix é que cada programa tenha uma finalidade específica e evite tentar desempenhar todas as tarefas sozinho. Nesse contexto, é possível encadear a execução de comandos individuais, cada um desempenhando sua função, para obter um resultado combinado. Esse encadeamento consiste em enviar a saída de um comando para a entrada de outro comando utilizando o caractere de canalização `|`, chamado *pipe*.

Por exemplo, o conteúdo do arquivo `/proc/cpuinfo` pode ser direcionado para o comando `wc` com o comando:

```
$ cat /proc/cpuinfo | wc
208      1184      6096
```

O conteúdo do arquivo `/proc/cpuinfo` foi redirecionado para a entrada padrão do comando `wc`, que faz a contagem do número de linhas, palavras e caracteres de um arquivo ou do conteúdo recebido pela entrada padrão. O mesmo resultado do exemplo é obtido ao substituir o uso do `cat` pelo operador de redirecionamento `<`:

```
$ wc < /proc/cpuinfo
208      1184      6096
```

O redirecionamento com `<` é específico para enviar o conteúdo de um arquivo para a entrada padrão de um programa. Neste caso, o fluxo dos dados segue da direita para a esquerda. Via de regra, os comandos que lidam com conteúdos de texto dispensam o uso desse operador, bastando informar a localização do arquivo.

Várias canalizações podem ser feitas em sequência. A seguir, duas canalizações usadas numa mesma linha de comando:

```
$ cat /proc/cpuinfo | grep 'model name' | uniq
model name      : Intel(R) Xeon(R) CPU           X5355  @ 2.66GHz
```

O conteúdo do arquivo `/proc/cpuinfo` foi canalizado com o comando `cat /proc/cpuinfo` para o comando `grep 'model name'`, que selecionará apenas as linhas contendo o termo *model name*. Por tratar-se de um computador com vários processadores, há várias linhas *model name* iguais. A última canalização é do comando `grep 'model name'` para o comando `uniq`, que reduz linhas repetidas em sequência para apenas uma ocorrência.

Expressões regulares

Expressões regulares são elementos de texto e operadores que formam um padrão, usado para encontrar e opcionalmente alterar um padrão correspondente. As expressões regulares são suportadas por uma grande variedade de programas que lidam com texto e que compartilham o os principais operadores, listados a seguir:

- `^`: Começo de linha.
- `$`: Fim de linha.
- `.`: Qualquer caractere.
- `*`: Qualquer sequência com tamanho igual ou superior a zero do caractere precedente.
- `?`: Uma ou nenhuma ocorrência do caractere precedente.
- `[]`: Qualquer caractere que esteja presente nos colchetes.

Um dos principais programas que utiliza expressões regulares para realizar buscas em texto é o `grep`, cuja aplicação mais comum é filtrar e facilitar a inspeção de arquivos muito longos. Pode ser utilizado, por exemplo, para analisar o conteúdo do arquivo `/etc/services`, que contém

as definições de portas associadas a serviços de rede:

```
$ cat /etc/services | grep '^..tp *.../tcp'
lmtp      24/tcp          # LMTP Mail Delivery
smtp      25/tcp          mail
tftp      69/tcp
http      80/tcp          www www-http # WorldWideWeb HTTP
sftp      115/tcp
nnntp     119/tcp          readnews untp # USENET News Transfer Protocol
qmtip     209/tcp          # Quick Mail Transfer Protocol
bftp      152/tcp          # Background File Transfer Program
pftp      662/tcp          # PFTP
dctp      675/tcp          # DCTP
vatp      690/tcp          # Velneo Application Transfer Protocol
```

É importante que a expressão esteja indicada com as aspas simples, para evitar que seja indevidamente interpretada pelo Bash.

O operador colchetes permite indicar uma sequência implícita de caracteres. Por exemplo, o operador [a-z] corresponde a qualquer letra do alfabeto, de a a z:

```
$ fdisk -l | grep '^Disco /dev/sd[a-z]'
```

Disco /dev/sda:	29,8 GiB,	32017047552 bytes,	62533296 setores
Disco /dev/sdb:	298,1 GiB,	320072933376 bytes,	625142448 setores
Disco /dev/sdc:	931,5 GiB,	1000170586112 bytes,	1953458176 setores

O comando `fdisk -l` gerou as informações detalhadas a respeito de partições nos disco presentes no sistema e enviou para a entrada padrão do `grep`, que por sua vez selecionou somente as linhas iniciando com a palavra *Disco* seguidas do caminho */dev/sd[a-z]*, correspondente a qualquer caminho terminando com uma letra do alfabeto.

O `grep` aceita diversas opções que modificam seu comportamento. Algumas opções comuns são:

- `-c`: Conta as linhas contendo o padrão.
- `-i`: Ignora a diferença entre maiúsculas e minúsculas.
- `-f`: Usa a expressão regular contida no arquivo indicado por essa opção.
- `-n`: Procura somente na linha indicada por essa opção.
- `-v`: Mostra todas as linhas exceto as que correspondem ao padrão.

Dois comandos complementam as funções do `grep`: `egrep` e `fgrep`. O comando `egrep` é equivalente ao comando `grep -E`, que incorpora outras funcionalidades além das expressões regulares padrão. Com o `egrep` pode-se usar o operador *pipe*, que atua como o operador *OU*. Desse modo, a expressão *'invenção|invenções'* corresponde a todas as ocorrências do termo *invenção* ou *invenções*.

Já o `fgrep` age da mesma forma que o `grep -F`, ou seja, ele deixa de interpretar expressões regulares. É especialmente útil nos casos mais simples, quando se quer apenas localizar a ocorrência de algum texto simples. Mesmo se forem utilizados caracteres especiais, como `$` ou o ponto, estes serão interpretados literalmente, e não pelo que representam numa expressão regular.

Manipulando conteúdo de arquivos

Existem muitos outros comandos que trabalham com arquivos de conteúdo de texto, basicamente realizando tarefas de recortar, extrair e filtrar. Vários desses comandos são fornecidos pelo pacote *GNU coreutils*.

O comando `tac` tem a mesma função do `cat`, mas mostra o conteúdo de trás para frente.

O comando `sort` ordena alfabeticamente. Com a opção `-n`, ordena numericamente. A opção `-r` inverte o resultado.

O comando `head` mostra o começo de arquivos. Por padrão, as primeiras dez linhas são mostradas. A quantidade de linhas a serem mostradas é indicada pela opção `-n`. A opção `-c` especifica o número de caracteres (*bytes*) a serem mostrados.

O comando `tail` mostra o final de arquivos. Por padrão, as últimas dez linhas são exibidas. A

quantidade de linhas a serem mostradas é indicada pela opção `-n`. A opção `-c` especifica o número de caracteres (bytes) a serem exibidos. Para que o final do arquivo seja mostrado continuamente, à medida que mais texto é adicionado, usa-se a opção `-f` (de *follow*). O sinal `+` indica que a leitura deve ser feita a partir da linha especificada após o `+`.

O comando `wc` conta linhas, palavras ou caracteres, a partir das opções `-l`, `-w` e `-c`, respectivamente. Quando usado sem argumentos, mostra esses três valores na mesma sequência.

O comando `cut` delimita um arquivo em colunas, em determinado número de caracteres ou por posição de campo. Para separar por campo, a opção `-d` especifica o caractere delimitador e `-f` informa a posição do campo. Por exemplo, para mostrar os campos da posição 1 e 3 do arquivo `/etc/group`, que estão separados por `:`:

```
$ cut -d ':' -f 1,3 /etc/group
root:0
daemon:1
bin:2
sys:3
adm:4
(...)
```

Para exibir outro delimitador no lugar do delimitador original, usa-se a opção `--output-delimiter`:

```
$ cut -d ':' -f 1,3 --output-delimiter ' = ' /etc/group
root = 0
daemon = 1
bin = 2
sys = 3
adm = 4
(...)
```

O comando `paste` concatena arquivos lado a lado, na forma de colunas:

```
$ cat um.txt
1      a1      a2      a3
2      b1      b2      b3
3      c1      c2      c3

$ cat dois.txt
1      x1      x2      x3
2      y1      y2      y3
3      z1      z2      z3

$ paste um.txt dois.txt
1      a1      a2      a3      1      x1      x2      x3
2      b1      b2      b3      2      y1      y2      y3
3      c1      c2      c3      3      z1      z2      z3
```

Todos esses comandos podem ser combinados com canalizações e redirecionamentos. Mesmo quando se utiliza um processador de textos para editar um arquivo, os comandos podem ser úteis para realizar tarefas mais elaboradas ou repetitivas.

3.3: Converter comandos em um script

Peso: 4

Scripts são arquivos que agem como programas, passando instruções a um interpretador para realizar determinada tarefa. Diferente de programas compilados, scripts são arquivos de texto que podem ser manipulados em qualquer editor de texto puro. No Linux, é bastante comum escrever *shell scripts*, que automatizam a execução de tarefas na linha de comando, das mais simples às mais sofisticadas. Cada linha do arquivo script será interpretada como um comando digitado no shell e deve respeitar as mesmas regras dos comandos digitados manualmente.

Editores de texto

Existem diversos editores de texto para linha de comando. O mais tradicional é o **vi**, mas existem alternativas como o **pico** e o **nano**. É recomendável conhecer minimamente o vi, pois é o editor encontrado em qualquer ambiente Unix.

Edição com Vi

O vi é considerado um editor para usuários experientes. Mesmo se comparado a outros editores de terminal, suas peculiaridades o tornam pouco intuitivo para usuários iniciantes.

A interface do vi se resume à tela onde o texto é apresentado e é manipulado, com um cursor indicando onde a ação é executada. Todas as operações são realizadas a partir de comandos do teclado. No vi existem os chamados modos de execução, nos quais as ações de teclado se comportam de maneira distinta. Há três modos de execução básicos no vi: O *modo de navegação*, o *modo de inserção* e o *modo de comando*.

O **modo de navegação** é o modo inicial do vi. Nele as teclas do teclado atuam basicamente para navegação e seleção de blocos de texto. Geralmente, os comandos são letras únicas. Se precedido por um número, o comando será repetido correspondentemente ao valor desse número.

A utilização do modo de navegação só faz sentido em um texto já existente ou após digitar algum conteúdo em um documento novo. Para abrir um arquivo, basta fornecer seu caminho como argumento ao comando vi. A seguir, alguns comandos de navegação importantes.

- **h**: Posicionar o cursor uma posição à esquerda.
- **j**: Posicionar o cursor uma posição abaixo.
- **k**: Posicionar o cursor uma posição acima.
- **l**: Posicionar o cursor uma posição à direita.
- **i, I**: Entra no modo de inserção na posição atual do cursor ou no início da linha atual.
- **a, A**: Entra no modo de inserção depois do cursor ou no fim da linha.
- **o, O**: Adiciona linha e entra no modo de inserção na linha posterior ou anterior a do cursor.
- **Esc**: Sair do modo de inserção e voltar para o modo de navegação.
- **0, \$**: Ir para o início ou para o fim da linha.
- **G**: Posicionar o cursor no final do documento. Se precedido de um número, ir até a linha correspondente.
- **(,)**: Ir para o início ou fim da sentença.
- **{, }**: Ir para o início ou fim do parágrafo.
- **w, W**: Ir até o início da próxima palavra. Com **W**, ir até a próxima palavra descartando caracteres especiais.
- **v, V**: Seleciona texto com as teclas de navegação. O uso de **V** seleciona linhas completas.
- **s, S**: Entra no modo de inserção, substituindo o caractere sob o cursor ou a linha toda.
- **c**: Apaga o conteúdo selecionado e entra no modo de inserção.
- **r**: Substitui o caractere sob o cursor.
- **x**: Apaga o caractere sob o cursor.
- **t**: Posiciona o cursor antes do caractere informado.
- **y, yy**: Copia o conteúdo selecionado ou a linha toda.
- **d, dd**: Apaga o conteúdo selecionado ou toda a linha atual. O conteúdo removido é copiado.
- **p, P**: Cola o conteúdo copiado, após ou antes do cursor.
- **u, [Ctrl][r]**: Desfazer, refazer
- **/, ?**: Buscar no texto a partir da posição atual ou antes da posição atual.
- **ZZ**: Fecha e salva, se necessário.
- **ZQ**: Fecha e não salva.

Diversas teclas de navegação podem ser combinadas. Por exemplo, para apagar todo o conteúdo a partir da posição atual do cursor até o próximo ponto, basta pressionar as teclas **dt..**

A finalidade do **modo de inserção** é simplesmente inserir texto. A tecla **[Esc]** sai do modo de inserção e volta para o modo de navegação.

O **modo de comando** é acionado ao pressionar a tecla **:** no modo de navegação. Usado para fazer buscas, alterações, salvar, sair, executar comandos no shell, alterar configurações do vi,

etc. Para retornar ao modo de navegação, usa-se o comando `visual` ou simplesmente a tecla **[Enter]** com a linha vazia. A seguir, alguns comandos importantes do modo de comando:

- `!:` Permite executar comandos do shell. Para retornar ao vi, executar o comando `exit` ou pressionar `ctrl + d`.
- `:quit` ou `:q`: Fecha o editor.
- `:quit!` ou `:q!`: Fecha sem gravar.
- `:wq`: Salva e fecha.
- `:exit` ou `:x` ou `:e`: Fecha e grava, se necessário.
- `:visual`: Volta para o modo de comando.

Existem versões do vi que possuem mais recursos, como o *vim* e até versões com interface gráfica, como o *gvim*. Contudo, o vi é suficiente para escrever arquivos de texto não formatado e scripts.

Início do script

A primeira linha do arquivo de script deve especificar o interpretador, que é indicado pelos caracteres `#!` (termo conhecido como *she-bang*). Para um script com instruções para o shell Bash, a primeira linha deverá ser `#!/bin/bash`. Assim, o interpretador para todas as instruções subsequentes será o programa `/bin/bash`. Com exceção da primeira linha, todas as demais linhas começando com `#` são ignoradas e podem ser utilizadas como lembretes e comentários.

Variáveis especiais

Os argumentos passados para um script e outras informações úteis são retornados pela variável especial `$x`, em que `x` determina qual valor retornar:

- `$*`: Todos os valores passados como argumentos.
- `$#`: O número de argumentos.
- `$0`: O nome do arquivo de script.
- `$n`: O valor do argumento na posição `n`. A variável `$1` corresponde ao primeiro argumento, `$2` ao segundo argumento e assim sucessivamente.
- `$!`: PID do último programa executado.
- `$$`: PID do shell atual.
- `$?`: Código numérico de saída do último comando. No padrão Unix, um código de saída igual a `0` significa que o último comando foi executado sem erros. O mesmo vale para o código de saída de scripts do shell.

Para solicitar valores ao usuário durante a execução do script, é utilizada a instrução `read`:

```
echo "Informe valor solicitado:"
read RESPOSTA
```

O valor retornado será armazenado na variável `RESPOSTA`. Caso uma variável de retorno não seja especificada, o nome padrão da variável de retorno, `REPLY`, será utilizado. Para armazenar a saída de um comando em uma variável, são utilizadas as aspas invertidas:

```
os=`uname -o`
```

Para exibir o conteúdo da variável, é necessário incluir o `$` à frente de seu nome. As variáveis podem ser utilizadas para exibir valores ou internamente, para armazenar dados que serão avaliados pelo programa para tomada de decisões.

Tomada de decisão

A principal característica de qualquer programa é a execução de determinadas ações dependendo de circunstâncias pré-determinadas. Para essa tarefa, existe o operador **if**, que executa um comando ou uma lista de comandos se uma condição for verdadeira. A instrução **test** avalia se a condição é verdadeira ou falsa. Seu uso é geralmente associado ao operador **if**, como no exemplo a seguir, que exibe `ok` se o arquivo `/bin/bash` for executável:

```
if test -x /bin/bash ; then
    echo "ok"
fi
```

O exemplo a seguir mostra outra maneira de realizar a mesma tarefa:

```
if [ -x /bin/bash ] ; then
    echo "ok"
fi
```

A instrução `else` é opcional à estrutura `if` e determina o bloco de instruções a executar caso a afirmação avaliada seja falsa. Exemplo:

```
if [ -x /bin/bash ] ; then
    echo "ok"
else
    echo "não ok"
fi
```

O final da estrutura `if` deve ser sempre sinalizado com `fi`.

Existem opções do `test` para várias finalidades. A seguir, algumas opções de avaliação da instrução `test` para arquivos e diretórios, supondo que um caminho para um arquivo ou diretório foi armazenado na variável `$caminho`:

- `-d $caminho`: Verdadeiro se o caminho existir e for um diretório.
- `-c $caminho`: Verdadeiro se o caminho existir.
- `-f $caminho`: Verdadeiro se o caminho existir e for um arquivo comum.
- `-L $caminho`: Verdadeiro se o caminho existir e for um link simbólico.
- `-r $caminho`: Verdadeiro se o caminho existir e puder ser lido (acessado).
- `-s $caminho`: Verdadeiro se o caminho existir e seu tamanho for maior que zero.
- `-w $caminho`: Verdadeiro se o caminho existir e puder ser escrito.
- `-x $caminho`: Verdadeiro se o caminho existir e for executável.
- `$caminho1 -ot $caminho2`: Verdadeiro se `caminho1` for diferente de `caminho2`.

Opções de avaliação de `test` para conteúdo de texto, supondo que a variável `$texto` contenha algum conteúdo:

- `-n $texto`: Verdadeiro se o tamanho de texto for diferente de zero.
- `-z $texto`: Verdadeiro se o tamanho de texto for zero.
- `$texto1 == $texto2`: Verdadeiro se `texto1` for igual a `texto2`.
- `$texto1 != $texto2`: Verdadeiro se `texto1` for diferente de `texto2`.

Opções de avaliação de `test` para números, supondo que `$num1` e `$num2` contenham valores numéricos:

- `$num1 -lt $num2`: Verdadeiro se `num1` for menor que `num2`.
- `$num1 -gt $num2`: Verdadeiro se `num1` for maior que `num2`.
- `$num1 -le $num2`: Verdadeiro se `num1` for menor ou igual a `num2`.
- `$num1 -ge $num2`: Verdadeiro se `num1` for maior ou igual a `num2`.
- `$num1 -eq $num2`: Verdadeiro se `num1` for igual a `num2`.
- `$num1 -ne $num2`: Verdadeiro se `num1` for diferente de `num2`.

Uma variação da instrução `if` é a instrução `case`. A instrução `case` prosseguirá se um item indicado for encontrado em uma lista de itens divididos pelo caractere barra vertical "|". Supondo que a variável `$num` contenha o número 3:

```
case $num in (1|2|3|4|5)
    echo "Número $num encontrado na lista,";
    echo "portanto case finalizou e";
    echo "executou esses comandos";
esac
```

O final da estrutura `case` deve ser sempre sinalizado com o termo `esac`.

Instruções de repetição

É bastante comum o desenvolvimento de scripts cuja finalidade é executar determinada tarefa repetidamente, obedecendo a uma condição pré-determinada. Para esse fim existem as chamadas instruções de repetição ou laço.

A instrução `for` executa uma ou mais ações para cada elemento de uma lista. Neste caso, cada número gerado pelo comando `seq`:

```
for i in $(seq 5); do
    echo "Copiando parte $i";
    scp luciano@lensqr.com:~/parte_$i ./;
done
```

O comando `seq 5` gera a sequência numérica de 1 a 5, tomados um a um pelo `for` e atribuídos a variável `i`. Para cada item da lista - nesse caso, para cada número - será executada a sequência de comandos dentro do bloco até o termo `done`.

A instrução `until` executa a sequência de comandos até que uma afirmação seja verdadeira. Por exemplo, implementar a mesma repetição feita anteriormente com `for` agora com `until`:

```
i=1;
until [ $i -gt 5 ]; do
    echo "Copiando parte $i";
    scp luciano@lensqr.com:~/parte_$i ./;
    i=$((i+1));
done
```

O `until` geralmente é maior que seu equivalente em `for`, mas pode ser mais adequados em algumas situações. Seu critério de encerramento é mais versátil que o contador do `for`, pois aceita qualquer parâmetro do `test`.

A instrução `while` é semelhante à instrução `until`, mas executa uma ação até que a afirmação deixe de ser verdadeira:

```
i=1;
while [ $i -le 5 ]; do
    echo "Copiando parte $i";
    scp luciano@lensqr.com:~/parte_$i ./;
    i=$((i+1));
done
```

Este último exemplo produz o mesmo resultado da implementação com `until` no exemplo anterior.

Execução do script

Caso o script vá ser compartilhado para execução por outros usuários, é importante que estes tenham acesso de leitura ao mesmo. O script do Bash pode ser executado invocando o comando `bash` tendo o caminho do script como argumento. Por exemplo, para executar o script `meuscript.sh` presente no diretório atual:

```
$ bash meuscript.sh
```

Alternativamente, o script pode ter a permissão de execução ser executado como um programa convencional. Para atribuir a permissão de execução a um script, é utilizado o comando `chmod`:

```
$ chmod +x meuscript.sh
```

Dessa forma, o arquivo do script `meuscript.sh` localizado no diretório atual poderá ser executado diretamente com o comando `./meuscript.sh`.

Tópico 4: O Sistema Operacional Linux

Peso: 8

4.1: Escolhendo um Sistema Operacional

Peso: 1

Todo sistema operacional possui as mesmas finalidades básicas, que são controlar o hardware, gerenciar programas e oferecer uma interface de operação ao usuário. Contudo, diferentes tipos de sistemas operacionais empregam diferentes abordagens.

Se considerarmos os computadores pessoais tradicionais, os sistemas operacionais mais utilizados são o Microsoft Windows e o Mac OS X. Em menor medida, estão as distribuições Linux. Considerando servidores, dispositivos embarcados e móveis, a participação do Linux é a maior.

Principalmente no âmbito da computação pessoal, eleger o melhor sistema operacional é muito mais uma questão pessoal do que um critério objetivo. De uma maneira ou de outra, tudo o que um sistema operacional pode oferecer pode ser encontrado em outro. De fato, os principais sistemas operacionais possuem muitas diferenças importantes.

Microsoft Windows

O Windows é o sistema operacional mais popular entre usuários domésticos. É produzido pela Microsoft e totalmente proprietário. Apesar de poder ser comprado diretamente pelo usuário, é mais comum que esteja pré-instalado pelo fabricante do computador. Também é muito utilizado irregularmente, instalado a partir de cópias piratas. Suas principais vantagens são:

- Familiar. A maioria dos usuários de computador conhece a interface do Windows.
- Compatibilidade. O Windows pode ser instalado em praticamente qualquer computador contemporâneo à sua versão de lançamento. Praticamente todo dispositivo fabricado é feito para funcionar com Windows.
- Programas. Existem incontáveis programas para Windows. Alguns muito específicos possuem versão somente para Windows e não há similares para outros sistemas operacionais.

Algumas de suas desvantagens são:

- Restritivo. Legalmente, não é possível instalar e compartilhar o Windows sem comprar uma licença.
- Inflexível. Um técnico ou desenvolvedor vai encontrar dificuldades se deseja realizar tarefas de modo diferente ao padrão do Windows.
- Programas mal-intencionados. Existem muitos programas mal-intencionados e vírus desenvolvidos para Windows.

O desempenho do Windows varia conforme o computador onde está instalado. Versões mais antigas, como o Windows 7, ainda são utilizadas por ter um bom desempenho mesmo em computadores já obsoletos.

Mac OS X

O OS X é o sistema operacional produzido pela Apple para sua linha de computadores pessoais. O OS X difere de suas versões anteriores principalmente na base do sistema, desenvolvida a partir do *FreeBSD*, um sistema Unix de código aberto. Apesar disso, toda a interface e aplicativos do OS X são proprietários, o que é permitido pela licença do *FreeBSD*. Algumas das vantagens do OS X são:

- Desempenho. O sistema operacional e o computador são produzidos pela Apple e estão integrados harmoniosamente.
- Interface. Os desenvolvedores de aplicativos para OS X devem seguir rígidos padrões estéticos para não prejudicar a consistência da interface.

Algumas de suas desvantagens:

- Compatibilidade. Mesmo sendo possível instalar o OS X em computadores de outros fabricantes, essa prática não é recomendada pela Apple e nem sempre produz resultados satisfatórios.
- Restritivo. Como o Windows, não é possível instalar e compartilhar o OS X sem comprar uma licença.

Mesmo sendo tecnicamente um Unix, o OS X mantém esse aspecto em segundo plano, privilegiando a administração do sistema em sua interface proprietária. É possível utilizar diversos comandos tradicionais de Unix no terminal do OS X.

Linux

Dada a sua natureza aberta de desenvolvimento, não existe um único sistema operacional Linux, mas diversas variações chamadas distribuições. Contudo, as principais distribuições possuem interfaces semelhantes. Na operação via linha de comando, há pouca diferença entre uma distribuição e outra.

As distribuições Linux têm procurado oferecer uma experiência familiar ao usuário habituado à interface de janelas, mas sempre mantendo a tradição da operação via linha de comando. A interface de janelas oferece facilidade de uso, mas a linha de comando é mais eficiente para certos tipos de tarefas, principalmente para usuários mais experientes executando tarefas administrativas.

Algumas vantagens do Linux em relação ao Windows e OS X são:

- Liberdade e versatilidade. Preservadas as condições definidas na licença de cada componente, o Linux pode ser utilizado como melhor convier ao usuário.
- Aprendizado. Usuários interessados no funcionamento do sistema operacional ou em aprender a programar encontram imensas fontes de informação num sistema Linux, tanto na forma de código fonte quanto de documentação.
- Programas. Um usuário de Linux tem a sua disposição uma grande variedade de programas a partir do instalador de pacotes da distribuição. Usuários do OS X e Windows já possuem instaladores semelhantes, mas no Linux os instaladores estão consolidados há mais tempo e a instalação de programas “por fora” é algo raro e não recomendado. Essa característica assegura a compatibilidade e a segurança do sistema operacional.
- Compatibilidade. Enquanto que com o OS X não é usual a instalação em equipamentos diferentes e com o Windows será necessário instalar diversos *drivers* de dispositivo à parte, uma distribuição Linux recém instalada está pronta para uso e raramente necessita de ajustes. No Linux, os *drivers* são chamados de *módulos* do kernel e são carregados automaticamente quando o dispositivo é identificado.

Dentre as desvantagens do Linux, destacam-se:

- Falta de padronização. Existem diferenças significativas entre as distribuições. A execução de uma tarefa em uma distribuição não será necessariamente igual em outra distribuição.
- Suporte. Alguns fabricantes de dispositivos e programas não fornecem suporte para Linux, obrigando o usuário a manter um Windows ou OS X exclusivamente para esses casos.

A diversidade de tipos de Linux pode ser entendida tanto como uma vantagem como uma desvantagem. Enquanto este fato torna mais difícil aprender sobre as diversas peculiaridades, é conveniente a existência de distribuições para as mais diversas finalidades e preferências. O desenvolvimento do Linux é tão dinâmico que em muitos casos o usuário pode optar por uma versão mais consolidada, chamada *estável*, ou uma versão mais avançada mas menos testada, chamada *beta*.

Algumas dessas variações não são exclusividade do Linux. É comum no Linux e entre outros sistemas operacionais a oferta de diferentes versões de um mesmo sistema operacional para diferentes finalidades, como versões para servidor ou para ambiente doméstico. Também existem distribuições que oferecem versões estáveis com suporte estendido, como é o caso do Ubuntu LTS (*long term support*) e seus derivados.

4.2: Entendendo o Hardware do Computador

Peso: 2

O funcionamento interno dos computadores modernos envolve muita complexidade para tornar sua operação mais simples e para se obter melhores resultados. É por conta do aprimoramento da tecnologia que existem diferentes tipos de conexões e dispositivos que podem ser conectados ao computador.

Periféricos

Genericamente, um periférico é qualquer equipamento externo conectado ao computador. Muitos periféricos podem até ser considerados computadores, alguns inclusive dotados de sistema operacional. É o caso de roteadores ou *switches* de rede, muitos utilizando alguma variação de Linux. Porém, diferente de computadores tradicionais, são equipamentos com uma finalidade específica. No caso dos roteadores, sua função é conectar computadores para formar uma rede local (LAN: *Local Area Network*).

Um computador moderno conta com os seguintes tipos de conexões para periféricos:

- **USB:** *Universal Serial Bus*, é o tipo de conexão mais comum. Utilizado por impressoras, teclado, mouse, telefones, etc. No Linux, o comando `lsusb` lista todos os dispositivos USB identificados pelo sistema.
- Entrada/Saída áudio analógico: Para caixas de som e microfone.
- **VGA/DVI:** Para conectar um monitor.
- **HDMI:** Saída integrada de vídeo e áudio, para enviar som e imagem para televisores.
- **Ethernet:** Conexão de rede cabeada.

Existem outras conexões pouco utilizadas ou em desuso, como a serial e a porta paralela. Além das conexões externas, a placa-mãe possui diversas conexões para componentes ligados internamente.

Conexões internas

A placa-mãe é a responsável por interligar todos os componentes do computador. Diversos componentes que antigamente eram separados hoje são integrados na placa-mãe, como controladores de disco e dispositivos de áudio e vídeo. Além disso, diversas conexões tornaram-se obsoletas, como as conexões ISA e AGP.

Além das conexões para memória e processador, as conexões internas mais utilizadas são:

- **PCI:** Para conectar placas expansoras, como placas de vídeo.
- **IDE:** Utilizada para discos rígidos e DVD/CDROM.
- **SATA:** Para discos rígidos mais modernos.

Antes mesmo que o sistema operacional esteja encarregado de controlar o computador, o BIOS (*Basic Input/Output System*, ou Sistema Básico de Entrada/Saída) da placa-mãe identifica e realiza testes simples nos itens fundamentais de hardware, como processador, memória e disco. Caso algum dos componentes internos não esteja ligado corretamente, o BIOS emitirá uma mensagem notificando sobre o erro. Caso não encontre erros, o BIOS executa o carregador de boot localizado no início do disco rígido pré-definido que, por sua vez, carregará o sistema operacional.

Dispositivos de armazenamento

No Linux, todo dispositivo de armazenamento encontrado é identificado por um arquivo dentro do diretório `/dev/` e são genericamente chamados *dispositivos de bloco*. O nome utilizado para o arquivo depende do tipo do dispositivo (IDE, SATA, SCSI etc) e das partições nele contidas.

Discos rígidos, mesmo os externos e os SSD (discos de estado sólido), são nomeados `/dev/sda`, `/dev/sdb`, `/dev/sdc`, etc, na ordem correspondente à que são identificados pelo sistema. A primeira partição do disco `/dev/sda` será `/dev/sda1`, a segunda partição neste disco será `/dev/sda2`, assim por diante. O mesmo vale para os demais dispositivos de armazenamento de bloco, como *pendrives*. É na partição onde fica o sistema de arquivos, que é responsável por organizar o armazenamento de arquivos. Os sistemas de arquivos mais populares no Linux são o `ext3`, `ext4`, `xfs` e `btrfs`.

Em versões mais antigas do kernel, os dispositivos IDE eram identificados como `/dev/hda`, `/dev/hdb`, etc. Nos sistemas Linux mais recentes, desde a versão 2.6 do Kernel, mesmo os discos IDE serão identificados como se fossem discos SATA. Nesse caso, os nomes serão criados com o prefixo `sd`, mas ainda será respeitado o esquema de nomes por *master* ou *slave* (no primeiro canal IDE, `sda` para master e `sdb` para slave, por exemplo).

Dispositivos de CD/DVD e disquetes também têm arquivos correspondentes em `/dev/`. Um leitor/gravador de CD/DVD conectado ao primeiro canal IDE será identificado como `/dev/sr0`. Se conectado ao segundo canal IDE será identificado como `/dev/sr1`. Um antigo dispositivo de disquete *floppy disk* é identificado pelo arquivo `/dev/fd0`.

4.3: Onde a Informação é Armazenada

Peso: 3

Como todo sistema operacional moderno, o Linux guarda um registro de ocorrência de todos os eventos relevantes no sistema. Para verificar como ocorreu o processo de carregamento do sistema é usado o comando `dmesg`, que exibe as mensagens no chamado *kernel ring buffer*. Essas mensagens do carregamento são sobrescritas toda vez que o sistema é reiniciado e são armazenadas em `/var/log/dmesg`, além de outras mensagens do kernel, que podem ser cheçadas dentro do arquivo `/var/log/messages`.

O Linux também mantém diversas informações relevantes a respeito dos processos do sistema. Em linhas gerais, um processo é um programa em execução. Cada processo possui um número único de identificação chamado *PID*. Esse número pode ser usado para mudar a prioridade de um processo ou para finalizá-lo.

Monitoramento de processos

Diversos comandos podem ser usados para inspecionar processos e são especialmente úteis para localizar e finalizar processos dispensáveis ou suspeitos. São eles:

- `ps`: Mostra os processos ativos de maneira detalhada;
- `top`: Monitora continuamente os processos, mostrando informações como uso de memória e CPU de cada processo. A tecla `[H]` fornece ajuda sobre o uso do programa. Pode ser usado para alterar a prioridade de um processo;
- `pstree`: Mostra processos ativos em formato de árvore genealógica (processos filhos ligados aos respectivos processos pais);

Um usuário só pode encerrar ou modificar a prioridade de seus próprios processos, e não pode reduzir a prioridade para abaixo de 0. O usuário `root` pode encerrar processos de qualquer usuário, e pode reduzir a prioridade para abaixo de 0 (maior prioridade).

Recursos de sistema

A administração de processos deve se basear nos recursos de hardware disponíveis. Basicamente, processos suspeitos que ocupam muita memória ou processamento podem ser finalizados em situações de emergência.

O comando `free` mostra o montante total de memória RAM, a quantidade de memória livre e o espaço de *swap*, em *kilobytes*:

```
$ free
              total        used        free      shared    buffers     cached
Mem:      16442484    15904244     538240           0     524656    1570016
-/+ buffers/cache:    13809572     2632912
Swap:      3998712     3336664     662048
```

Numa situação em que não há mais memória RAM disponível e o espaço de swap já está demasiado ocupado, existe a suspeita de que algum processo está ocupando muita memória indevidamente e deve ser finalizado.

Localização dos arquivos de sistema

Todo arquivo no sistema tem uma localização adequada, que varia conforme sua finalidade. Em sistemas Linux, o padrão que define a localização dos arquivos e diretórios chama-se *Filesystem Hierarchy Standard*, **FHS**.

FHS

O FHS (do inglês *Filesystem Hierarchy Standard* ou *Hierarquia Padrão de Sistemas de arquivos*) é o padrão de localização de arquivos adotado pela maioria das distribuições Linux. Cada um dos diretórios serve a um propósito, sendo divididos entre os que devem existir na partição raiz e os que podem ser pontos de montagem para outras partições ou dispositivos.

Os diretórios que residem obrigatoriamente na partição raiz são:

- `/bin` e `/sbin`: Contêm os programas essenciais para carregar e administrar o sistema.
- `/etc`: Arquivos de configuração específicos da máquina.
- `/lib`: Bibliotecas compartilhadas pelos programas em `/bin` e `/sbin` e módulos do kernel.
- `/mnt` e `/media`: Pontos de montagem para outras partições ou dispositivos.
- `/proc` e `/sys`: Diretórios especiais com informações de processos e hardware.
- `/dev`: Arquivos de acesso a dispositivos e outros arquivos especiais.

Os demais diretórios da raiz que podem ser pontos de montagem são:

- `/boot`: Kernel e mapas do sistema e os carregadores de boot de segundo estágio.
- `/home`: Os diretórios dos usuários.
- `/root`: Diretório do usuário root.
- `/tmp`: Arquivos temporários.
- `/usr/local` e `/opt`: Programas adicionais. Também podem conter as bibliotecas necessárias para os programas adicionais.
- `/var`: Dados de programas e arquivos relacionados, arquivos de log, bancos de dados e arquivos de sites. Pode conter diretórios compartilhados.

Em computadores pessoais, não há necessidade de criar partições extras ou em outros dispositivos. Já em servidores, é recomendável fazê-lo, pois esse procedimento torna a administração mais simples e pode resultar em ganho de desempenho.

4.4: Seu Computador em Rede

Peso: 2

A finalidade de um computador quase sempre está associado ao seu uso numa rede de computadores e o meio de comunicação entre computadores mais consolidado são as redes IP (*Internet Protocol*). As principais características desse tipo de rede são a utilização de um número de identificação - o número IP - para o dispositivo conectado e a possibilidade de comunicação entre diferentes redes de computadores, daí origina o nome *Internet* (entre redes). Ao conectar um computador na rede, seja via um roteador numa rede interna ou diretamente à Internet, ele necessariamente precisará obter um endereço IP para poder comunicar-se com outras máquinas.

Configuração dos clientes na rede

Na maior parte dos casos, tanto a configuração da interface quanto das rotas de rede é feita automaticamente usando o recurso chamado *DHCP*. O programa DHCP cliente envia uma requisição para a rede por meio da interface especificada e o servidor responde com informações de endereço IP, máscara de rede, broadcast etc., que serão usadas para configurar a interface local.

Fundamental para o funcionamento da rede é que a interface de rede esteja configurada corretamente. Se toda a parte estrutural da rede - roteador e cabeamento - estiver corretamente preparada e a interface de rede corretamente instalada, esta poderá ser configurada manualmente por meio do comando `ifconfig` ou do comando `ip`.

O `ifconfig` e o `ip` também podem ser usados para inspecionar as configurações de uma interface. O comando `ifconfig` sem argumentos mostra as configurações de todas as interfaces ativas do sistema. No Linux, a única interface de rede que é sempre presente é a `lo`, que

representa uma interface de comunicação local, não correspondente a nenhum item de hardware. Para verificar as configurações de uma interface específica, basta fornecer como argumento o nome da interface.

De modo semelhante, o comando `ip addr show` mostra as configurações de todas as interfaces ativas do sistema, sendo possível especificar a interface de rede específica para inspeção.

Endereço IP

Endereços IP no formato a.b.c.d - conhecidos pelo termo inglês *dotted-quad* - são a expressão, em números decimais, de um endereço de rede binário. Cada um dos quatro campos separados por pontos corresponde a um byte, algumas vezes chamado octeto. Por exemplo, o número IP 192.168.1.1 corresponde à forma binária 11000000.10101000.00000001.00000001.

Cada interface de rede numa mesma rede deverá ter um endereço IP único, mas cada computador pode possuir mais de uma interface de rede. Nesse caso, o computador pode estar conectado a diversas redes diferentes.

Endereço de rede, máscara de rede e endereço broadcast

Para que os dados possam ser encaminhados corretamente pela rede, a interface de rede precisa conhecer seu número IP, o número IP de destino e a qual rede eles pertencem.

Na maioria dos casos, a rede do IP de destino só será conhecida quando esse IP de destino estiver dentro da mesma rede interna do IP de origem. É possível identificar se um IP pertence a uma rede observando sua máscara de rede.

A máscara de rede define uma quantidade de bits. Se essa quantidade inicial de bits em um endereço é igual a de outro endereço, significa que ambos pertencem a mesma rede.

- Máscara de 16 bits: 11111111.11111111.00000000.00000000 = 255.255.0.0
- Máscara de 17 bits: 11111111.11111111.10000000.00000000 = 255.255.128.0

Na primeira máscara (16 bits), pertencerão à mesma rede os IPs cujos dois primeiros octetos do endereço não difiram entre si. Na segunda máscara (17 bits), pertencerão à mesma rede os IPs cujos dois primeiros octetos e o primeiro bit do terceiro octeto do endereço não difiram entre si. Dessa forma, dois endereços de interface 172.16.33.8 e 172.16.170.3 estarão na mesma rede se a máscara for de 16 bits, mas não se a máscara for de 17 bits.

As máscaras de rede variam dependendo do contexto da rede. Consequentemente, o endereço da rede corresponde à parte do número IP determinado pelos bits marcados da máscara de rede. Para uma máquina 172.16.33.8 com máscara de rede 255.255.0.0, o endereço da rede será 172.16.0.0.

O endereço broadcast e o número IP que designa todas as interfaces numa rede. Para um endereço de rede 172.16.0.0, o endereço broadcast será 172.16.255.255.

Um endereço IP pode demonstrar a informação de endereço da rede, máscara de rede e broadcast numa forma abreviada. Por exemplo, 192.168.1.129/25, em que número 25 após a barra indica a quantidade de bits da máscara. Conclui-se que a máscara de rede é 255.255.255.128, o endereço IP é 192.168.1.128 e o endereço de broadcast é 192.168.1.255.

IPv4 e IPv6

O padrão tradicional de 32 bits (quatro octetos de bits) dos números IP é conhecido como *IPv4*. Há outro padrão mais recente, conhecido como *IPv6*, que consiste de uma sequência de 128 bits. A vantagem óbvia do IPv6 sobre o IPv4 é a disponibilidade de uma quantidade muito maior de números IP. Enquanto o IPv4 é capaz de gerar 4.294.967.296 endereços, o IPv6 disponibiliza 2^{128} endereços.

Um endereço IPv6 normalmente é escrito na forma de oito grupos de quatro números hexadecimais, como: 2001:0db8:85a3:08d3:1319:8a2e:0370:7334.

O IPv4 ainda é muito mais difundido e é possível a intercomunicação entre os dois padrões. Porém, à medida que cada vez mais dispositivos demandarem o uso de um endereço IP, o

padrão IPv6 estará mais disseminado o vigente.

Rotas

Conhecendo o IP de destino e a qual rede ele pertence, o sistema será capaz de encaminhar os dados pela interface de rede correta. Contudo, principalmente em comunicação na Internet, dificilmente a interface local estará diretamente conectada à rede do IP de destino. Para esses casos, uma **rota padrão** é estabelecida: um endereço IP – conhecido como *gateway* – dentro de uma rede conhecida é utilizado para encaminhar esse tipo de comunicação para uma rede desconhecida. Esse processo ocorre sucessivamente, até que encontre a rede a qual onde o IP de destino é conhecido.

A tabela de rotas do sistema pode ser consultada de várias maneiras. O comando `route` tem a finalidade de manipular a tabela de rotas, mas sem argumentos apenas exibe a tabela de rotas atual:

```
$ route
Tabela de Roteamento IP do Kernel
Destino      Roteador      MáscaraGen.   Opções  Métrica  Ref   Uso  Iface
default      _gateway      0.0.0.0       UG      0        0    0   tun0
10.8.0.0     0.0.0.0       255.255.255.0 U        0        0    0   tun0
_gateway     0.0.0.0       255.255.255.255 UH      0        0    0   tun0
192.168.2.0  0.0.0.0       255.255.255.128 U       100     0    0   enp0s25
```

Nessa saída é possível identificar as diferentes rotas configuradas para as interfaces de rede. A rota padrão é identificada pelo termo *default* na primeira coluna. O comando `netstat`, utilizado para exibir várias informações a respeito da rede, também pode exibir essa saída ao ser utilizado com a opção `-r`. O comando `ip` é outro comando que pode ser utilizado para inspecionar as rotas e possui uma saída mais simplificada:

```
$ ip route show
default via 10.8.0.1 dev tun0
10.8.0.0/24 dev tun0 scope link
10.8.0.1 dev tun0 proto kernel scope link src 10.8.0.6
192.168.2.0/25 dev enp0s25 proto kernel scope link src 192.168.2.10 metric 100
```

Essas informações são especialmente úteis quando a rede está apresentando falhas de conexão. Um primeiro passo para diagnosticar o problema verificando se o endereço de destino da rota padrão está acessível.

Diagnóstico de conexão

O comando `ping` pode ser usado para fazer diagnósticos simples na rede. Utilizando o protocolo *ICMP*, ele simplesmente envia uma pequena quantidade de dados para uma máquina especificada e aguarda uma resposta. Se a máquina remota responder, significa o nível básico de conectividade da rede está funcionando corretamente.

Se o acesso a diferentes sites de Internet não estiver funcionando, por exemplo, é possível que exista algum problema de conectividade na rota padrão. Após identificar o endereço do *gateway*, o comando `ping` pode ser utilizado para verificar se o mesmo está acessível:

```
$ ping -c3 10.8.0.1
PING 10.8.0.1 (10.8.0.1) 56(84) bytes of data:
64 bytes from 10.8.0.1: icmp_seq=1 ttl=64 time=4.07 ms
64 bytes from 10.8.0.1: icmp_seq=2 ttl=64 time=3.20 ms
64 bytes from 10.8.0.1: icmp_seq=3 ttl=64 time=1.70 ms

--- 10.8.0.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.700/2.994/4.075/0.981 ms
```

Nesse exemplo, foi possível verificar que os três pacotes (definidos com a opção `-c3`) enviados para o *gateway* 10.8.0.1 foram respondidos com sucesso. Isso significa que o problema hipotético não está ocorrendo no envio de pacotes pela rota padrão.

Resolução de nomes

Problemas de conexão de rede também podem estar associados ao serviço de resolução de nomes *DNS*. O DNS é responsável por traduzir nomes como *lcnsqr.com* para um número IP e vice-versa. A configuração incorreta do DNS resulta numa rede praticamente inoperante, pois imensa maioria dos destinos remotos são acessadas por nome e não diretamente pelo seu número IP.

Para testes mais simples de resolução de nomes, o comando `host` pode ser utilizado. Por exemplo, traduzir o nome *lcnsqr.com* o número IP correspondente:

```
$ host lcnsqr.com
lcnsqr.com has address 162.243.102.48
lcnsqr.com mail is handled by 20 trilobit.lcnsqr.com.
```

De modo semelhante, é possível consultar se um nome está associado a um número IP:

```
$ host 162.243.102.48
48.102.243.162.in-addr.arpa domain name pointer trilobit.lcnsqr.com.
```

A esse tipo de associação se dá o nome de DNS reverso. Caso o comando `host` não devolva um número IP para o nome pesquisado, é possível fazer outros testes indicando um servidor DNS específico logo após o nome sendo pesquisado:

```
$ host lcnsqr.com 208.67.222.222
Using domain server:
Name: 208.67.222.222
Address: 208.67.222.222#53
Aliases:

lcnsqr.com has address 162.243.102.48
lcnsqr.com mail is handled by 20 trilobit.lcnsqr.com.
```

Se apenas ao especificar um servidor DNS ao comando `host` se obtém o endereço IP correspondente, significa que a configuração de DNS local não está correta ou o servidor DNS utilizado por padrão não está funcionando.

As configurações de DNS ficam armazenadas dentro de arquivos de texto comuns, no diretório `/etc`. Apesar de cada distribuição utilizar métodos distintos para realizar as configurações automáticas de rede, todas elas obedecem à padronização tradicional de armazenamento das configurações. Os principais arquivos de configuração relacionados ao DNS no cliente são:

- `/etc/hosts`: Associa os números IP da rede a nomes. Ele é prático para atribuir um nome mais simples para máquinas acessadas frequentemente ou para pequenas redes, onde um serviço de resolução automática de nomes não é necessário.
- `/etc/resolv.conf`: Determina os números IP dos servidores de resolução de nomes DNS.

Via de regra, o arquivo `/etc/hosts` é consultado primeiro. Em seguida, caso o nome consultado não seja encontrado no arquivo, será consultado um servidor DNS especificado no arquivo `/etc/resolv.conf`.

O arquivo `/etc/hosts` é bastante simples. Nele os números IP são posicionados ao lado dos respectivos nomes:

```
$ cat /etc/hosts
127.0.0.1    localhost
162.243.102.48  trilobit
```

Mais de um nome pode ser atribuído a um mesmo IP, atuando como um nome alternativo para o mesmo IP. Por exemplo, incluir o nome *lcnsqr* ao IP 162.243.102.48:

```
162.243.102.48  trilobit lcnsqr
```

Já no arquivo `/etc/resolv.conf` são indicados os números IP dos servidores DNS, ou seja, os computadores que fazem a tradução de um nome para um número IP e vice versa. A entrada fundamental no `/etc/resolv.conf` é **nameserver**, que define o servidor DNS. Outras entradas

nameserver podem estar indicadas, para o caso do primeiro servidor DNS estar fora do ar ou muito ocupado:

```
$ cat /etc/resolv.conf
domain lcnsqr.com
nameserver 192.168.1.1
nameserver fd6d:c2f2:544f::1
```

A linha `domain` indica o domínio local. Assim, quando for consultado um nome sem domínio, o domínio padrão será considerado. Se necessário, as entradas `nameserver` podem ser alteradas para outros endereços IP cujo funcionamento tenha sido verificado com o comando `host`. Os endereços IP 208.67.222.222 e 208.67.220.220 são servidores DNS públicos do [Opendns.com](https://opendns.com) e podem ser utilizados livremente. Outros servidores DNS públicos são 8.8.8.8 e 8.8.4.4, estes oferecidos pelo [Google](https://www.google.com/dns/).

O comando `dig` (*Domain Information Groper*) retorna informações mais avançadas para o diagnóstico de problemas em servidores DNS. Se nenhum argumento for utilizado, o comando realizará o teste-padrão no(s) servidor(es) encontrados no arquivo `/etc/resolv.conf`. Para utilizar um servidor específico, basta indicá-lo após o caractere `_@_`:

```
$ dig lcnsqr.com @8.8.8.8

; <<>> DiG 9.11.4-P2-RedHat-9.11.4-13.P2.fc28 <<>> lcnsqr.com @8.8.8.8
;; global options: +cmd
;; Got answer:
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 57487
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;lcnsqr.com.                IN A

;; ANSWER SECTION:
lcnsqr.com.                1799    IN A    162.243.102.48

;; Query time: 230 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: qua fev 27 01:17:14 -03 2019
;; MSG SIZE rcvd: 55
```

Essa resposta mostra que o nome `lcnsqr.com` foi localizado pelo servidor DNS indicado. O trecho *QUESTION* mostra qual foi o nome pesquisado e o trecho *ANSWER* mostra qual foi a resposta do servidor consultado. A letra *A* nas linhas *QUESTION* e *ANSWER* indica que o registro DNS possui um endereço IPv4 associado. Se o endereço IP associado fosse IPv6, o tipo indicado para o registro seria *AAAA*.

Outros problemas de rede

Se todas as configurações de rede estiverem em ordem e ainda assim existirem problemas de conectividade, é possível que as falhas estejam em outros pontos da rede. Um comando importante para analisar o tráfego e a resposta das máquinas remotas é o `netstat`. Por exemplo, é possível inspecionar todas as conexões do protocolo TCP ativas:

```
$ netstat -tn
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 162.243.102.48:22      189.100.34.204:43382    ESTABLISHED
```

A opção `-n` determina que sejam mostrados apenas os números IPs e a opção `-t` determina a configuração apenas das conexões do protocolo TCP. Para exibir continuamente as novas conexões, basta informar a opção `-c`.

O `netstat` também agrega algumas funções de outros comandos. Com a opção `-i`, exibe todas as interfaces de rede ativas e estatísticas relacionadas:

```
$ netstat -i
Kernel Interface table
```

Iface	MTU	Met	RX-OK	RX-ERR	RX-DRP	RX-OVR	TX-OK	TX-ERR	TX-DRP	TX-OVR	Flg
eth0	1500	0	270337	0	0	0	143119	0	0	0	BMRU
lo	16436	0	0	0	0	0	0	0	0	0	LRU

Caso o tráfego de dados mostre que algumas conexões não respondem, o problema poderá estar no ponto remoto, ou seja, em outro computador da rede. Para identificar em que ponto as conexões não seguem adiante, existe o comando `tracert`. Por exemplo, o comando `tracert 162.243.102.48` mostra as rotas percorridas por um pacote até chegar ao destino 162.243.102.48, permitindo avaliar se o problema apresentado encontra-se na rede local ou no caminho até o destino.

Tópico 5: Segurança e Permissões de Arquivos

Peso: 7

5.1: Segurança Básica e Identificação de Tipos de Usuários

Peso: 2

O Linux é um sistema operacional multiusuário. Como o nome sugere, esse tipo de sistema operacional permite que diferentes usuários utilizem o sistema de modo independente. Ou seja, diferentes pessoas podem utilizar o mesmo sistema operacional sem que seus arquivos e personalizações estejam acessíveis a outros usuários.

Diferentes usuários podem utilizar o sistema operacional ao mesmo tempo. Apesar de não ser comum um mesmo computador possuir mais de um teclado e monitor, é bastante comum que diversos usuários utilizem o mesmo sistema via rede.

Outra vantagem em manter os usuários em contas distintas é poder controlar seus privilégios dentro do sistema. Um usuário deve poder trabalhar com seus arquivos e ter acesso restrito a certos recursos do computador, mas não deve ter acesso a arquivos de terceiros ou a recursos que podem prejudicar o funcionamento do sistema operacional. O único usuário que possui acesso irrestrito dentro do sistema operacional é o usuário **root**.

O usuário root

O principal usuário de um sistema Unix é o usuário chamado **root**. Ele é criado por padrão e é utilizado para configurar o sistema operacional e corrigir eventuais problemas. Também é o único usuário cujo diretório pessoal não está no diretório padrão de usuários, `/home`, mas na raiz do sistema de arquivos, em `/root`.

Senha de root

A senha do usuário root normalmente é definida durante a instalação do sistema operacional. Em alguns casos, não é atribuída uma senha ao usuário root, e será necessário utilizar o comando `sudo` para realizar operações de root.

Devido ao seu poder irrestrito no sistema operacional, é importante ter muito cuidado quando utilizando o usuário root. Mesmo tratando-se de um computador pessoal utilizado por apenas uma pessoa, é recomendável operar o computador com um usuário convencional.

Usuários convencionais e de sistema

É o usuário root o responsável por criar usuários no sistema local. Todas as contas de usuário são armazenadas no arquivo de configuração `/etc/passwd`. O usuário root também pode associar um usuário a um grupo para lhe conferir privilégios extras. Os grupos do sistema e os usuários a eles associados são armazenados no arquivo `/etc/group`.

Todo usuário está associado a pelo menos um grupo, chamado seu *grupo principal*. O grupo principal é único, mas o usuário pode estar associado a diversos outros grupos.

Cada usuário e grupo possuem um número único associado, chamado respectivamente *UID* e *GID*. O único usuário com um número padrão é o root, com número UID 0. Os números de um usuário e de seus grupos podem ser observados com o comando `id`. Sem argumento, o comando `id` exibe o número do usuário atual e os grupos aos quais pertence:

```
$ id
uid=13130(lcnsqr) gid=207(bmac) groups=207(bmac)
```


Neste exemplo, o número (uid) do usuário *lcnsqr* é *13130*, o número de seu grupo principal (*bmac*) é *207* e está associado somente a este grupo. Os números são gerados automaticamente, geralmente sendo atribuídos números a partir de 1000 para usuários convencionais.

Usuários comuns podem executar tarefas normalmente reservadas ao root com o comando `sudo`. Por exemplo, para utilizar o comando administrativo `apt-get dist-upgrade` com o `sudo`:

```
$ sudo apt-get dist-upgrade
```

O `sudo` pode ser configurado para restringir diferentes tipos de ação a usuário específicos, solicitando ou não uma senha. Ele também pode ser usado com o comando `su` para iniciar uma nova sessão do shell como outro usuário, com o comando `sudo su usuário`, onde *usuário* é o nome de login do outro usuário. Caso a conta do root tenha uma senha definida, pode ser utilizado o comando `su` diretamente:

```
$ su -
```

Nesse caso será necessário fornecer a senha do root para prosseguir.

Além dos usuários convencionais, existem usuários especiais chamados usuários de sistema. Estes usuários não correspondem a uma pessoa, mas a alguma tarefa do sistema operacional. Essa abordagem oferece uma camada adicional de segurança, pois programas executados por um usuário não podem interferir com arquivos e processos que não lhe pertencem.

Inspeção de usuários

O usuário root pode interferir em processos de outros usuários e modificar seus arquivos. O root pode até checar quem está usando o sistema com o comando `who`. Com o comando `w` o usuário root pode verificar quem está utilizando o sistema e sua atividade no momento.

O comando `lastlog` exibe quando foi a última vez que cada usuário entrou no sistema:

```
$ lastlog -t 2
Nome de Usuário      Porta      De          Último
guilherme            pts/1      143.107.45.1  Qua Feb 27 09:14:45 -0300 2019
lucas                 pts/1      143.107.45.1  Ter Feb 26 10:23:34 -0300 2019
gustavo               pts/1      143.107.45.1  Ter Feb 26 19:47:28 -0300 2019
afranio              pts/2      143.107.45.1  Ter Feb 26 19:17:22 -0300 2019
filipe                pts/1      c-73-208-7-86.hs Ter Feb 26 15:39:21 -0300 2019
willian               pts/1      179.225.189.179 Qua Feb 27 08:57:10 -0300 2019
lcnsqr                pts/2      189.79.126.38  Qua Feb 27 09:57:19 -0300 2019
```

A opção `-t 2` foi utilizada para limitar a inspeção aos últimos dois dias. É possível verificar a partir de onde o usuário ingressou no sistema e qual foi a data e hora exatas do login.

O comando `last` possui finalidade semelhante, mas exibindo os últimos ingressos de usuários no sistema. Para verificar os últimos ingressos de um usuário específico, basta fornecer seu nome de usuário como argumento:

```
$ last lcnsqr
lcnsqr pts/2      189.79.126.38  Wed Feb 27 09:57  still logged in
lcnsqr pts/1      189.79.126.38  Tue Feb 26 17:27 - 17:32 (00:04)
lcnsqr pts/1      189.79.126.38  Tue Feb 26 16:29 - 17:15 (00:46)
lcnsqr pts/0      189.79.126.38  Wed Feb 20 16:24 - 16:27 (00:02)
lcnsqr pts/0      189.79.126.38  Wed Feb 20 16:24 - 16:24 (00:00)

wtmp begins Fri Feb 1 13:48:21 2019
```

O comando `last reboot` mostra quando o sistema foi ligado pela última vez, ou seja, desde quando está ligado. O usuário root pode verificar também se houveram tentativas mal sucedidas de ingresso no sistema com o comando `lastb`.

5.2: Criação de Usuários e Grupos

Peso: 2

Em ambientes onde mais de uma pessoa utiliza o computador ou utiliza os recursos fornecidos por ele via rede, é muito importante que cada uma delas possua restrições para que não comprometa dados sensíveis, sejam eles pertinentes ao próprio sistema, sejam pertinentes a outros usuários. Para isso, para cada usuário é criada uma conta com a qual ele acessará o sistema.

Cabe ao usuário root administrar as contas e grupos de usuários. Contudo, o próprio usuário pode modificar alguns aspectos de sua própria conta no sistema.

Conta de usuário

O comando `useradd` é usado pelo usuário root para criar uma nova conta no sistema. As principais opções do `useradd` são:

- `-c` comentário: comentário (geralmente o nome completo do usuário).
- `-d` diretório: caminho para o diretório pessoal do usuário.
- `-g` grupo: grupo inicial (GID). Precisa existir previamente no sistema.
- `-G` grupo1,grupo2: grupos adicionais, separados por vírgula.
- `-u` UID: UID (User ID) do usuário.
- `-s` shell: Shell padrão para o usuário.
- `-p` senha: senha (entre aspas).
- `-e` data: data de validade da conta.
- `-k` /etc/skel: copia o diretório modelo /etc/skel.
- `-m`: cria o diretório pessoal, se não existir.

Com a opção `-k` /etc/skel novos diretórios pessoais podem ser criados a partir de um conteúdo padrão armazenado em /etc/skel. Esse procedimento facilita a criação de várias contas de usuário a partir de um perfil pré-definido.

Para que o usuário possa acessar sua conta, o administrador precisará definir uma senha para ele. Isso pode ser feito por meio do comando `passwd` usuário. Usado sem argumentos, `passwd` altera a senha para o usuário atual, o que pode ser feito por um usuário comum. O `passwd` também pode ser utilizado pelo usuário root para bloquear a conta de um usuário. Por exemplo, a conta do usuário *luciano* pode ser bloqueada com o comando `passwd -l luciano`. A conta poderá ser desbloqueada com o comando `passwd -u luciano`.

O campo de descrição pode ser alterado com o comando `chfn` e o shell principal pode ser alterado com `chsh`. Usuários comuns podem usar estes comandos para alterar exclusivamente suas próprias contas.

Uma conta de usuário pode ser apagada com o comando `userdel`. A opção `-r` assegura que o diretório pessoal do usuário também seja apagado.

As informações de conta dos usuários do sistema são armazenadas no arquivo /etc/passwd, no formato:

```
root:x:0:0::/root:/bin/bash
luciano:x:1000:1000:Luciano Antonio Siqueira:/home/luciano:/bin/bash
```

Cada usuário é definido em uma linha, em campos separados por ":" representando, respectivamente:

- Nome de Login.
- Senha ("x" quando usando o arquivo /etc/shadow).
- Número de identificação do usuário (UID).
- Número do grupo principal do usuário (GID).
- Descrição do usuário (opcional).
- Diretório pessoal para o usuário.
- Shell inicial do usuário (se vazio, o arquivo padrão /bin/sh será usado).

Para editar diretamente o arquivo /etc/passwd, é recomendado usar o comando `vipw`, que bloqueia o arquivo /etc/passwd contra possíveis alterações concorrentes, evitando corrupção do

arquivo. A edição será feita com o editor padrão, via de regra o editor *vi*. Usado com a opção *-s*, *vi*pw abrirá para edição o arquivo */etc/shadow*.

Senhas shadow

O arquivo */etc/passwd* pode ser lido por qualquer usuário, o que pode tornar as senhas criptografadas passíveis de decodificação. Para evitar essa possibilidade, é usado um segundo arquivo, acessível apenas ao usuário *root*, o arquivo */etc/shadow*. As senhas dos usuários são armazenadas num *hash* criptografado que não pode ser facilmente decodificado.

Como no arquivo */etc/passwd*, os campos no arquivo */etc/shadow* são separados por *“:”*, correspondendo a:

- Nome de usuário, que deve corresponder a um nome válido em */etc/passwd*.
- A senha criptografada. Em branco permite login sem senha. Com um asterisco *“*”* indica que a conta está bloqueada.
- O número de dias (desde 01/01/1970) desde que a senha foi alterada.
- Número mínimo de dias até que uma senha possa ser novamente alterada. O número zero *“0”* permite alterar a senha sem tempo de espera.
- Número de dias depois dos quais a senha deverá ser alterada. Por padrão, 99999, ou 274 anos.
- Número de dias para informar ao usuário sobre a expiração da senha.
- Número de dias, depois de a senha expirar, até que a conta seja bloqueada.
- O número de dias, a partir de 01/01/1970, desde que a conta foi bloqueada.
- Campo reservado.

As informações referentes à validade da senha também podem ser modificadas por meio do programa *chage*, com as seguintes opções:

- *-m* dias: mínimo de dias até que o usuário possa trocar uma senha modificada.
- *-M* dias: número máximo de dias que a senha permanecerá válida.
- *-d* dias: número de dias decorridos em relação a 01/01/1970. Determina quando a senha foi mudada. Também pode ser expresso no formato de data local (dia/mês/ano).
- *-E* dias: número de dias decorridos em relação a 01/01/1970, a partir dos quais a conta não estará mais disponível. Também pode ser expresso no formato de data local (dia/mês/ano).
- *-I* dias: inatividade ou tolerância de dias, após a expiração da senha, para que a conta seja bloqueada.
- *-W* dias: dias anteriores ao fim da validade da senha, quando será emitido um aviso sobre a expiração da validade.

Para usuários comuns, o *chage* só pode ser usado com a opção *-l* usuário, que mostra as restrições referentes ao usuário em questão. O comando *usermod* agrega as funções de alteração de conta de usuário por meio das opções:

- *-c* descrição: descrição do usuário.
- *-d* diretório: altera diretório do usuário. Com o argumento *-m*, move o conteúdo do diretório atual para o novo.
- *-e* valor: prazo de validade da conta, especificado no formato *dd/mm/aaaa*.
- *-f* valor: número de dias, após a senha ter expirado, até que a conta seja bloqueada. Um valor *-1* cancela essa função.
- *-g* grupo: grupo principal do usuário.
- *-G* grupo1, grupo2: grupos adicionais para o usuário.
- *-l* nome: nome de login do usuário.
- *-p* senha: senha.
- *-u* UID: número de identificação (UID) do usuário.
- *-s* shell: shell padrão do usuário.
- *-L*: bloqueia a conta do usuário, colocando um sinal *!* na frente da senha criptografada. Uma alternativa é substituir o shell padrão do usuário por um script ou programa que informe as razões do bloqueio.
- *-U*: desbloqueia a conta do usuário, retirando o sinal *!* da frente da senha criptografada.

Os arquivos */etc/passwd* e */etc/shadow* armazenam somente as informações diretamente relacionadas a cada conta de usuários. Grupos de usuários são configurados em seus próprios

arquivos de configuração.

Grupos de usuários

Para criar um grupo de usuários, é usado o comando `groupadd`:

```
# groupadd estudo_c
```

O número de identificação do grupo (GID) pode ser especificado com a opção `-g`. Para excluir um grupo, é usado o comando `groupdel`:

```
# groupdel estudo_c
```

Um usuário poderá ser incluído/excluído de um grupo com o comando `gpasswd`, utilizando o argumento adequado. As opções mais comuns do `gpasswd` são:

- `gpasswd grupo`: cria uma senha para grupo.
- `gpasswd -r grupo`: apaga a senha para grupo.
- `gpasswd -a usuário grupo`: associa usuário ao grupo.
- `gpasswd -d usuário grupo`: exclui usuário de grupo.
- `gpasswd -A usuário grupo`: torna um usuário administrador de grupo.

Um usuário pode pertencer a mais de um grupo, mas apenas um grupo pode ser o principal. Para mostrar os grupos aos quais um usuário pertence, pode ser usado o comando `groups usuário`. Usado sem argumentos, o comando `groups` mostra os grupos do usuário atual.

As informações sobre os grupos existentes no sistema são armazenadas no arquivo `/etc/group`. Neste arquivo, cada grupo é definido em uma linha, em campos separados por `:`, representando:

- Nome do grupo.
- Senha para o grupo (x se utilizar `/etc/gshadow`).
- Número de identificação do grupo (GID).
- Lista de membros do grupo, separados por vírgula.

Para editar diretamente o arquivo `/etc/group`, é altamente indicado usar o comando `vigr`, que bloqueia o arquivo `/etc/group` contra possíveis alterações externas, evitando corrupção do arquivo.

O comando `groupmod` agrega algumas funções de alteração de grupos. Suas opções mais comuns são:

- `-g GID`: altera o número (GID) do grupo.
- `-n nome`: altera o nome do grupo.

A principal finalidade dos grupos é permitir que usuários executem atividades que não são permitidas fora do grupo em questão. Essas atividades estão relacionadas principalmente à leitura e escrita em arquivos e diretórios restritos. Portanto, é importante compreender como funciona o sistema de permissões de arquivos em ambientes Unix.

5.3: Controle de Permissões e Propriedade de Arquivos

Peso: 2

Em sistemas de arquivos do padrão Unix, existem regras de permissões que determinam a quem pertence um determinado arquivo ou diretório e quais usuários ou grupos podem utilizá-los. Para arquivos e diretórios há três níveis de permissão:

- Usuário dono do arquivo (**u**).
- Grupo dono do arquivo (**g**).
- Demais usuários – *outros* – (**o**).

O diretório `/var/run` contém arquivos com diferentes tipos de permissões, identificadas na

primeira coluna:

```
drwx----- 2 root      root      40 fev 26 10:08 cryptsetup
drwxr-xr-x  3 root      lp        80 fev 26 10:08 cups
srw-rw-rw-  1 root      root       0 fev 26 10:08 .heim_org.h5l.kcm-socket
drwxrwxr-x  2 lightdm   lightdm   40 fev 26 10:08 lightdm
drwxr-x---  2 root      root      40 fev 26 10:08 pptp
-rw-r--r--  1 root      root       4 fev 26 10:08 crond.pid
drwx----- 2 root      root      40 fev 26 10:51 udisks2
srw-rw-rw-  1 root      root       0 fev 26 10:08 secrets.socket
drwxr-x---  2 chrony    chrony    80 fev 27 09:00 chrony
prw-----  1 root      root       0 fev 26 10:08 dmeventd-server
drwx----- 2 rpc       rpc       60 fev 26 10:08 rpcbind
drwxrwxr-x  2 root      root      40 fev 26 10:08 netreport
-rw-r--r--  1 root      root       5 fev 27 09:00 dhclient6-enp0s25.pid
prw-----  1 root      root       0 fev 26 10:08 initctl
```

A primeira letra representa o tipo do arquivo, podendo ser:

- -: Arquivo convencional.
- d: Diretório.
- l: Link simbólico.
- c: Dispositivo especial de caracteres.
- p: Canal *fifo*.
- s: *Socket*.

As demais letras são divididas em grupos de três, determinando as permissões para o dono do arquivo, o grupo do arquivo e demais usuários, respectivamente. O arquivo `crond.pid`, por exemplo, possui permissão `rw-` para o dono do arquivo (o usuário `root`), permissão `r--` para o grupo (o grupo `root`) e permissão `r--` para os demais usuários.

Alterando permissões

As permissões são alteradas com o comando `chmod` e podem ser de leitura (**r**), escrita (**w**) e execução (**x**). Por exemplo, o grupo ao qual pertence um arquivo chamado `documentos.tar.gz` terá apenas acesso de leitura a este e para os demais usuários será retirada a permissão de leitura:

```
$ chmod g=r,o-r documentos.tar.gz
```

Para incluir permissão de escrita para o grupo do arquivo `documentos.tar.gz`:

```
$ chmod g+w documentos.tar.gz
```

Apesar de possuírem o mesmo modelo de permissões, arquivos e diretórios comportam-se de maneiras diferentes tendo as mesmas permissões. Em diretórios, a permissão **r** possibilita ler o conteúdo do diretório, a permissão **w** permite criar arquivos dentro do diretório e **x** permite listar o conteúdo do diretório.

Permissões numéricas (octais)

Permissões podem ser manejadas de modo mais sucinto através de um formato numérico, chamado *octal*. O número octal consiste em uma sequência de quatro dígitos. O primeiro dígito representa uma permissão especial, abordada adiante. Os demais representam as permissões para o usuário, grupo e outros, nessa ordem.

Cada dígito indica a presença de uma permissão a partir da soma dos valores 4, 2 e 1. Esses valores correspondem à leitura, escrita e execução. A tabela a seguir mostra todas as permissões possíveis, desde 0 (nenhuma permissão) até 7 (todas as permissões).

Dígito	Leitura (valor 4)	Escrita (valor 2)	Execução (valor 1)
0	—	—	—
1	—	—	Sim

2	—	Sim	—
3	—	Sim	Sim
4	Sim	—	—
5	Sim	—	Sim
6	Sim	Sim	—
7	Sim	Sim	Sim

Dessa forma, o comando `chmod 0664 arquivos.tar.gz` mudará as permissões do arquivo `documentos.tar.gz` para `-rw-rw-r--`, ou seja, leitura e escrita para o usuário, leitura e escrita para o grupo e somente leitura para os demais.

Para mudar recursivamente todos os arquivos dentro de um diretório especificado, utiliza-se o `chmod` com a opção `-R`.

Modificar donos e grupos de arquivos

Para alterar dono e grupo de arquivos e diretórios, utiliza-se os comandos `chown` e `chgrp`. O primeiro argumento é um nome válido de usuário ou grupo e o segundo é o arquivo ou diretório a ser alterado. Apenas o usuário `root` pode usar o comando `chown`, mas qualquer usuário pode usar o comando `chgrp` em seus arquivos e diretórios, desde que faça parte do grupo que será atribuído.

Mudar dono de arquivo usando o `chown`:

```
# chown luciano documentos.tar.gz
```

Mudar grupo de arquivo:

```
$ chgrp users documentos.tar.gz
```

Para alterar usuário e grupo simultaneamente:

```
# chown luciano.users documentos.tar.gz
```

ou

```
chown luciano:users documentos.tar.gz
```

Tanto `chown` quanto `chgrp` possuem a opção `-R` para alterar conteúdos de diretórios recursivamente.

5.4: Arquivos e Diretórios Especiais

Peso: 1

Permissões `suid` e `sgid`

Num ambiente Unix, todos os processos são vinculados ao usuário que os iniciou. Dessa forma, o programa herdar as mesmas permissões de leitura e escrita do usuário que o executou. Algumas tarefas, no entanto, exigem que o processo altere ou acesse arquivos para os quais o usuário não tem a permissão necessária. Por exemplo, alterar a própria senha exige que o arquivo `/etc/shadow` seja alterado, mas as permissões de `/etc/shadow` limitam a escrita ao usuário dono deste arquivo (o usuário `root`):

```
$ ls -l /etc/shadow
-rw-r----- 1 root shadow 1172 Mai 15 2017 /etc/shadow
```

Para contornar essa condição, existe um tipo de permissão especial, chamada **suid** (*set user id*). Arquivos executáveis que possuam a permissão suid serão executados com as mesmas permissões do dono do *comando* e não com as permissões do usuário que o executou. A permissão suid é representada pela letra **s** no lugar do x na porção referente ao dono do arquivo:

```
$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 54192 Feb 24 2017 /usr/bin/passwd
```

Para incluir a permissão suid em um arquivo executável, utiliza-se:

```
# chmod u+s comando
```

De maneira semelhante, a permissão **sgid** atua em diretórios. Ela é uma permissão de grupo, portanto aparece no campo de permissões referente ao grupo. Num diretório com a permissão sgid, todos os arquivos ali criados pertencerão ao grupo do diretório em questão, o que é especialmente útil em diretórios onde trabalham usuários pertencentes ao mesmo grupo.

Quando ativas, as permissões suid e sgid fazem aparecer a letra **s** no lugar da letra x nas permissões de dono do arquivo e grupo do arquivo, respectivamente. Se a permissão de execução também existir, aparecerá a letra **s** minúscula. Se apenas as permissões suid e sgid existirem, aparecerá a letra **S** maiúscula.

A permissão sticky

O inconveniente em usar diretórios compartilhados é que um usuário poderia apagar algum ou todo o conteúdo inadvertidamente. Para evitar que isso aconteça, existe a permissão *sticky*, que impede usuários de apagar arquivos não criados por eles mesmos. É o caso do diretório `/tmp`, cujas propriedades podem ser verificadas com o comando `ls -ld` mais a opção `-d`, que permite obter informações do próprio diretório em questão e não de seu conteúdo:

```
$ ls -ld /tmp
drwxrwxrwt 17 root root 36864 Feb 27 11:32 /tmp
```

A letra **t** nas permissões para demais usuários demonstra o uso da permissão sticky. Se apenas a permissão sticky existir, aparecerá a letra **T** maiúscula.

Para atribuir a permissão sticky num diretório chamado `trabalho`, portanto, pode ser utilizado um comando como `chmod o+t trabalho`.

Permissões especiais em formato numérico

Como as opções convencionais, as permissões especiais também podem ser manipuladas em formato octal (numérico). A permissão especial é o primeiro dos quatro dígitos da opção no formato octal. A tabela a seguir detalha essa correspondência.

Dígito	suid (valor 4)	sgid (valor 2)	sticky (valor 1)
0	—	—	—
1	—	—	Sim
2	—	Sim	—
3	—	Sim	Sim
4	Sim	—	—
5	Sim	—	Sim
6	Sim	Sim	—
7	Sim	Sim	Sim

Criar e alterar links simbólicos e hardlinks

Links são arquivos especiais que têm finalidade de atalho para outros arquivos, facilitando a maneira como são acessados. Existem dois tipos de links: o *softlink* (link simbólico) e o *hardlink* (link físico).

Hardlinks (links físicos)

Hardlinks são um ou mais nomes que um *inode* do sistema de arquivos pode ter. Todo arquivo criado é, necessariamente, um hardlink para seu inode correspondente. Novos hardlinks são criados usando o comando `ln`:

```
$ ln documentos.tar.gz docs.tar.gz
```

O que é um inode?

Um inode é o elemento básico que identifica o arquivo no sistema de arquivos. O primeiro inode de um arquivo guarda suas propriedades e indica em quais outros inodes do sistema de arquivos os dados deste arquivo estão localizados.

A opção `-i` do comando `ls` mostra o número dos inodes dos arquivos:

```
$ ls -i
6534 documentos.tar.gz 6534 docs.tar.gz
```

Ambos `documentos.tar.gz` e `docs.tar.gz` são hardlinks para o mesmo inode 6534. Hardlinks para o mesmo inode possuem mesma permissão, donos, tamanho e data, pois esses atributos são registrados diretamente nos inodes.

```
$ ls -l documentos.tar.gz
-rw-r--r-- 2 luciano luciano 188 Oct 16 22:10 documentos.tar.gz
```

O número **2** na segunda coluna de informações demonstra que há dois hardlinks para o inode correspondente ao arquivo `documentos.tar.gz`. Um arquivo só é de fato apagado do sistema de arquivos quando o último hardlink remanescente é excluído.

Hardlinks só podem ser criados dentro de um mesmo sistema de arquivos. Não é possível criar hardlinks para diretórios. Os arquivos especiais `.` e `..` são hardlinks para diretório criados exclusivamente pelo próprio sistema.

Softlinks (links simbólicos)

Links simbólicos podem apontar para qualquer alvo, inclusive em sistemas de arquivos diferentes. Para criar um link simbólico, usa-se `ln` com a opção `-s`:

```
$ ln -s documentos.tar.gz atalho.tar.gz
```

Detalhes do link:

```
$ ls -l atalho.tar.gz
lrwxrwxrwx 1 luciano luciano 17 Feb 27 11:40 atalho.tar.gz -> documentos.tar.gz
```

Um link simbólico é indicado pela letra `l` no início das permissões que, nesse caso, são sempre `lrwxrwxrwx`. O tamanho do arquivo de link é exatamente a quantidade de bytes (caracteres) do caminho alvo. A seta ao lado do nome do link simbólico indica o caminho até o alvo.

Um link simbólico para um caminho relativo será quebrado se o alvo ou o próprio link for movido. Um link simbólico para um caminho absoluto só será quebrado se o alvo for movido ou

apagado. Para atualizar a informação de alvo de um link simbólico existente, mas “quebrado”, pode-se redefinir o alvo do link com a opção -f.

Funções comuns para links simbólicos são indicar caminhos longos frequentemente usados, criar nomes mais simples para arquivos executáveis e nomes adicionais para bibliotecas de sistema.

Objetivos da Certificação Linux Essentials

Para receber o certificado Linux Essentials, o candidato deve:

- ter uma compreensão do Linux e do setor de código aberto e conhecimento dos aplicativos de código aberto mais populares;
- entender os principais componentes do sistema operacional Linux e ter a proficiência técnica para trabalhar na linha de comando do Linux; e
- ter um conhecimento básico de tópicos relacionados à segurança e administração, como gerenciamento de usuário / grupo, trabalhando na linha de comando e permissões.

Os objetivos detalhados estão listados a seguir. Essa é a versão 1.6 dos objetivos, publicada em 2019 e disponível em www.lpi.org.

Tópico 1: A comunidade Linux e a carreira Open Source (peso: 7)

1.1 A evolução do linux e sistemas operacionais populares

Peso: 2

Descrição: Conhecimento de desenvolvimento em Linux e suas principais distribuições.

Áreas chave de conhecimento:

- Distribuições
- Sistemas Embarcados
- Linux na Nuvem

Segue uma lista parcial dos arquivos, termos e utilitários utilizados:

- Android
- Debian, Ubuntu (LTS)
- CentOS, openSUSE, SUSE, Red Hat
- Linux Mint, Scientific Linux
- Raspberry Pi, Raspbian

1.2 Principais Aplicações Open Source

Peso: 2

Descrição: Reconhecimento das aplicações principais, bem como seu desenvolvimento e seus usos.

Áreas chave de conhecimento:

- Aplicações Desktop
- Aplicações em Servidores
- Linguagens de desenvolvimento
- Ferramentas de gerenciamento de pacotes e repositórios

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- OpenOffice.org, LibreOffice, Thunderbird, Firefox, GIMP
- Apache HTTPD, NGINX, MySQL, NFS, Samba
- C, Java, Perl, shell, Python, PHP, Javascript
- dpkg, apt-get, rpm, yum
- Nextcloud, ownCloud

1.3 Entendendo o Software Open Source e suas Licenças

Peso: 1

Descrição: Comunidades abertas e licenciamento de software Open Source para negócios.

Áreas chave de conhecimento:

- Filosofia do Código Aberto
- Licenciamento
- Free Software Foundation (FSF), Open Source Initiative (OSI)

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- GPL, BSD, Creative Commons, Copyleft, Permissive
- Free Software, Open Source Software, FOSS, FLOSS
- Modelos de negócios Open Source

1.4 Habilidades ICT e trabalhando no Linux

Peso: 2

Descrição: Habilidades básicas em TIC (Tecnologia da Informação e Comunicação) e trabalhando no linux.

Áreas chave de conhecimento:

- Habilidades no Desktop
- Chegando à linha de comando
- Usos industriais para Linux, computação em nuvem e virtualização

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- Usando um navegador, questões de privacidade, opções de configuração, procurando na web e salvando conteúdo
- Terminal e Console
- Senhas
- Questões e ferramentas relacionadas à privacidade
- Uso de aplicações Open Source populares em apresentações e projetos

Tópico 2: Encontrando seu caminho em um Sistema Linux (peso: 9)

2.1 O básico sobre a linha de comando

Peso: 3

Descrição: Básico de como usar a linha de comando no Linux

Áreas chave de conhecimento:

- Shell básico
- Sintaxe da linha de comando
- Variáveis
- Globbing (Englobamento)
- Uso de aspas

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- Bash
- echo
- history
- Variável de ambiente PATH
- export
- type

2.2 Usando a linha de comando para conseguir ajuda

Peso: 2

Descrição: Rodando comandos de ajuda e de navegação de vários sistemas de ajuda.

Áreas chave de conhecimento:

- Man
- Info

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- man
- info
- páginas Man
- /usr/share/doc/
- locate

2.3 Usando diretórios e listando arquivos

Peso: 2

Descrição: Navegação no diretório home e nos diretórios do sistema. Listando arquivos em vários locais.

Áreas chave de conhecimento:

- Arquivos, diretórios
- Arquivos e diretórios ocultos
- Home
- Caminhos relativos e absolutos

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- Opções comuns para ls
- Listagens recursivas
- cd
- . e ..
- home e ~

2.4 Criando, Movendo e Deletando Arquivos

Peso: 2

Descrição: Criar, mover e deletar arquivos e diretórios dentro do diretório home.

Áreas chave de conhecimento:

- Arquivos e diretórios
- Uso de maiúsculas e minúsculas
- Englobamento e uso de aspas

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- mv, cp, rm, touch
- mkdir, rmdir

Tópico 3: O Poder da Linha de Comando (peso: 9)

3.1 Empacotando arquivos na linha de comando

Peso: 2

Descrição: Empacotando arquivos no diretório home.

Áreas chave de conhecimento:

- Arquivos, diretórios
- Pacotes, compressão

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- tar
- Opções comuns do tar
- gzip, bzip2
- zip, unzip

3.2 Pesquisando e extraindo dados de arquivos

Peso: 3

Descrição: Pesquisar e extrair dados de arquivos no diretório home.

Áreas chave de conhecimento:

- Pipes da linha de comando
- Redirecionamento I/O
- Expressões regulares básicas utilizando ., [], *, e ?

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- grep
- less
- cat, head, tail
- sort
- cut
- wc

3.3 Transformando comandos em Scripts

Peso: 4

Descrição: Tornando comandos repetitivos em scripts simples.

Áreas chave de conhecimento:

- Shell scripting básico
- Reconhecimento de editores de texto comuns (vi e nano).

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- #! (shebang)
- /bin/bash
- Variáveis
- Argumentos
- loops for
- echo
- Exit status

Tópico 4: O Sistema Operacional Linux (peso: 8)

4.1 Escolhendo um Sistema Operacional

Peso: 1

Descrição: Conhecimento dos principais sistemas operacionais e distribuições Linux.

Áreas chave de conhecimento:

- Diferenças entre Windows, Mac, Linux
- Gerenciamento do ciclo de vida das distribuições

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- GUI versus linha de comando, configuração desktop
- Ciclos de manutenção, Beta e Stable

4.2 Entendendo o Hardware do Computador

Peso: 2

Descrição: Familiaridade com os componentes utilizados para construir computadores desktop e servidores.

Áreas chave de conhecimento:

- Hardware

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- Placas-mãe, processadores, fontes, drives ópticos, periféricos
- Discos-rígidos e partições, /dev/sd*
- Drivers

4.3 Onde os dados são armazenados

Peso: 3

Descrição: Onde vários tipos de informações são armazenadas em um sistema Linux.

Áreas chave de conhecimento:

- Programas e configuração, pacotes e banco de dados de pacotes
- Processos, endereços de memória, mensagens do sistema; logando-se.

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- ps, top, free
- syslog, dmesg
- /etc/, /var/log/
- /boot/, /proc/, /dev/, /sys/

4.4 Seu Computador na Rede

Peso: 2

Descrição: Consultando informações vitais de configuração de rede e determinando os requisitos básicos para um computador em uma LAN.

Áreas chave de conhecimento:

- Internet, rede, roteadores
- Consultando configuração do cliente DNS
- Consultando configuração da rede

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- route, ip route show
- ifconfig, ip addr show
- netstat, ip route show
- /etc/resolv.conf, /etc/hosts
- IPv4, IPv6
- ping
- host

Tópico 5: Segurança e Permissões de Arquivos (peso: 7)

5.1 Segurança Básica e Identificação de Tipos de Usuários

Peso: 2

Descrição: Vários tipos de usuários em um sistema Linux.

Áreas chave de conhecimento:

- Root e Usuários padrão
- Usuários do sistema

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- /etc/passwd, /etc/group
- id, who, w
- sudo, su

5.2 Criando Usuários e Grupos

Peso: 2

Descrição: Criando usuários e grupos em um sistema Linux.

Áreas chave de conhecimento:

- Comandos de usuários e de grupos
- IDs de usuários

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- /etc/passwd, /etc/shadow, /etc/group, /etc/skel/
- id, last
- useradd, groupadd
- passwd

5.3 Gerenciando permissões e donos de arquivos

Peso: 2

Descrição: Entendendo e manipulando permissões e configurações relacionadas a donos de arquivos.

Áreas chave de conhecimento:

- Permissões de arquivos/diretórios e seus donos.

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- ls -l, ls -a
- chmod, chown

5.4 Diretórios e arquivos especiais

Peso: 1

Descrição: Arquivos e diretórios especiais em um sistema Linux, incluindo permissões especiais.

Áreas chave de conhecimento:

- Usando arquivos e diretórios temporários
- Links simbólicos

Segue uma lista parcial de arquivos, termos e utilitários utilizados:

- /tmp/, /var/tmp/ e Sticky Bit

- ls -d
- ln -s