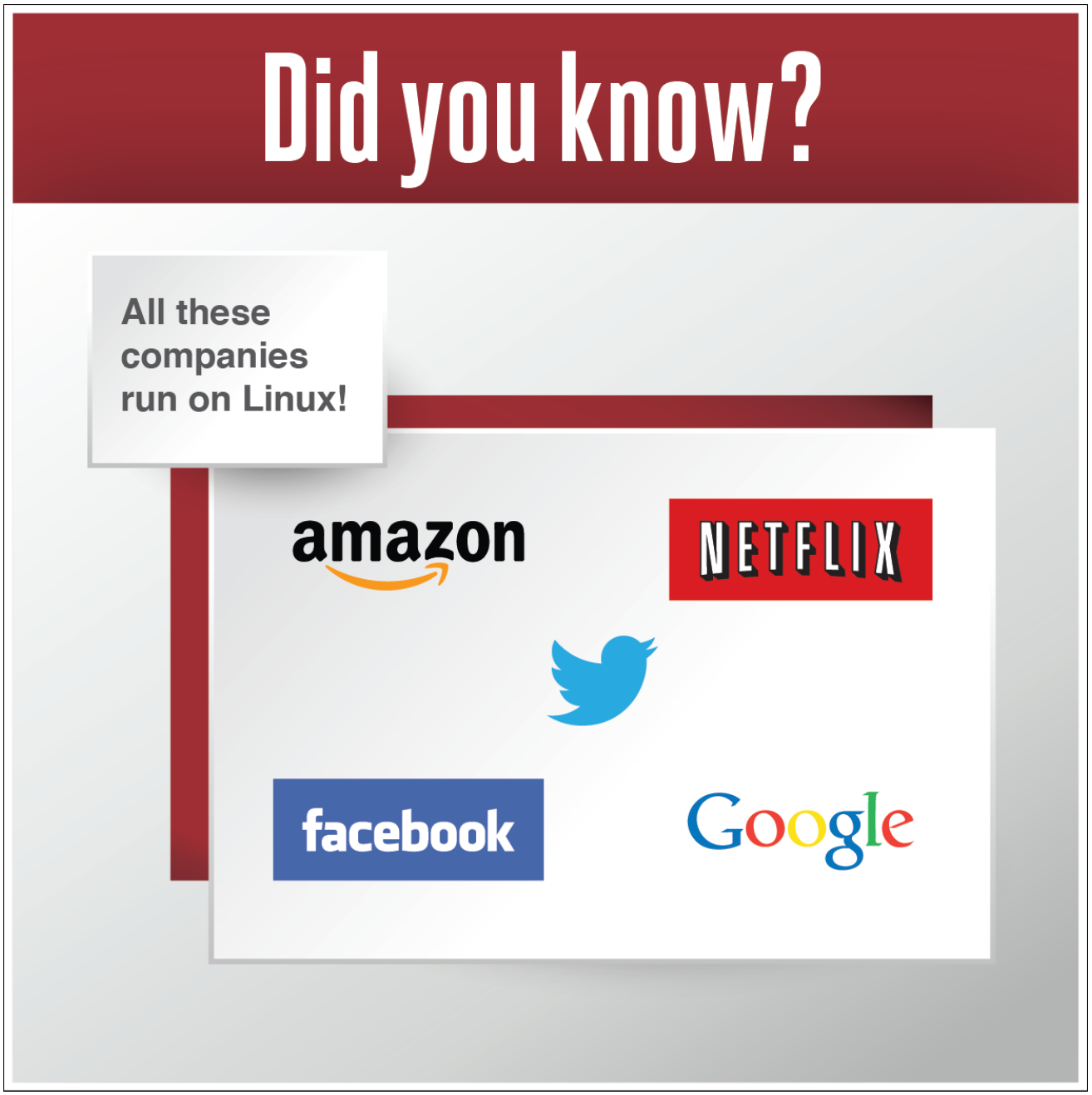
APLICAÇÕES OPEN SOURCE E LICENÇAS

2.1 Introdução

Neste capítulo, nos familiarizaremos com vários aplicativos e ferramentas de código aberto. Também discutiremos sobre software de código aberto e licenças.



2.2 Principais aplicativos de código aberto

O kernel do Linux pode executar uma ampla variedade de softwares em muitas plataformas de hardware. Um computador pode atuar como um servidor, o que significa que ele lida principalmente com dados de outros computadores ou pode funcionar como um desktop, o que significa que um usuário estará interagindo diretamente com ele. A máquina pode executar software ou pode ser usada como uma máquina de desenvolvimento no processo de criação de software. Você pode até executar várias funções, pois não há distinção para o Linux sobre o papel da máquina; é apenas uma questão de configurar quais aplicativos são executados.

Uma vantagem disso é que você pode simular quase todos os aspectos de um ambiente de produção, desde o desenvolvimento, até o teste, até a verificação em hardware reduzido, o que economiza custos e tempo. Como alguém aprendendo Linux, você pode executar os mesmos aplicativos de servidor em sua área de trabalho ou um servidor virtual de baixo custo rodando em um grande provedor de serviços de Internet. É claro que você não será capaz de lidar com o volume que um provedor de grande porte teria, já que eles terão hardware muito mais caro. Mas você pode simular quase qualquer configuração sem precisar de um poderoso hardware ou licenciamento de servidor.

O software Linux geralmente se enquadra em uma das três categorias:

- **Software de servidor -** software que não tem interação direta com o monitor e o teclado da máquina em que é executado. Sua finalidade é fornecer informações para outros computadores, chamados de clientes. Às vezes, o software do servidor pode não falar com outros computadores, mas apenas fica parado e "processa" os dados.
- **Software de área de trabalho -** um navegador da Web, editor de texto, reprodutor de música ou outro software com o qual você interaja. Em muitos casos, como um navegador da Web, o software está conversando com um servidor do outro lado e interpretando os dados para você. Aqui, o software de desktop é o cliente.
- **Ferramentas -** uma categoria solta de software que existe para facilitar o gerenciamento de seu sistema. Você pode ter uma ferramenta que ajude a configurar sua exibição, ou algo que forneça um shell do Linux, ou ferramentas ainda mais sofisticadas que convertam código-fonte em algo que o computador possa executar.

Além disso, vamos considerar aplicativos móveis. Um aplicativo móvel é muito parecido com um aplicativo de desktop, mas é executado em um telefone ou tablet em vez de um computador desktop.

Qualquer tarefa que você queira fazer no Linux provavelmente pode ser acomodada por qualquer número de aplicativos. Há muitos navegadores da Web, muitos servidores da Web e muitos editores de texto (os benefícios de cada um deles são o assunto de muitas guerras santas do UNIX). Isso não é diferente do mundo do código fechado. No entanto, um benefício do código-fonte aberto é que, se alguém não gostar da maneira como seu servidor da Web funciona, ele poderá começar a construir o seu próprio.

2.2.1 Aplicações de Servidor

O Linux é excelente para executar aplicativos de servidor devido à sua confiabilidade e eficiência. Ao considerar o software para servidores, a pergunta mais importante é “qual serviço estou executando?” Se você quiser servir páginas da Web, precisará de um software de servidor da Web, não de um servidor de e-mail!

Um dos primeiros usos do Linux foi para servidores web. Um servidor da Web hospeda conteúdo para páginas da Web, que são visualizadas por um navegador da Web usando o HTTP (Hypertext Transfer Protocol) ou seu sabor criptografado, HTTPS. A própria página da Web pode ser estática, o que significa que, quando o navegador da Web solicita a página, o servidor da Web apenas envia o arquivo como aparece no disco. O servidor também pode servir conteúdo dinâmico, o que significa que a solicitação é enviada pelo servidor da web para um aplicativo, o que gera o conteúdo. WordPress é um exemplo popular. Os usuários podem desenvolver conteúdo através de seu navegador no aplicativo WordPress e o software o transforma em um site totalmente funcional. Toda vez que você faz compras on-line, está vendo um site dinâmico.

O Apache é o servidor da web dominante em uso atualmente. O Apache era originalmente um projeto autônomo, mas o grupo já formou a Apache Software Foundation e mantém mais de cem projetos de software de código aberto.

Outro servidor da web é o nginx, que é baseado na Rússia. Ele se concentra no desempenho fazendo uso de kernels UNIX mais modernos e faz apenas um subconjunto do que o Apache pode fazer. Mais de 65% dos sites são alimentados por nginx ou Apache.

E-mail sempre foi um uso popular para servidores Linux. Ao discutir servidores de e-mail, é sempre útil observar as três funções diferentes necessárias para receber e-mails entre pessoas:

- **Agente de Transferência de Email (Mail Transfer Agent - MTA) -** descobre qual servidor precisa receber o email e usa o protocolo SMTP para mover o email para esse servidor. Não é incomum que um email leve vários “saltos” para chegar ao seu destino final, uma vez que uma organização pode ter vários MTAs.
- **Agente de Entrega de Correspondência (Mail Delivery Agent - MDA, também conhecido como Agente de Entrega Local) -** cuida de armazenar o email na caixa de correio do usuário. Geralmente invocado do MTA final na cadeia.
- **Servidor POP / IMAP -** O **Post Office Protocol** e o **Internet Message Access Protocol** são dois protocolos de comunicação que permitem que um cliente de email em execução no seu computador converse com um servidor remoto para obter o email.

Às vezes, um software implementará vários componentes. No mundo de código fechado, o Microsoft Exchange implementa todos os componentes, portanto, não há opção para fazer seleções individuais. No mundo do código aberto, há muitas opções. Alguns servidores POP / IMAP implementam seu próprio formato de banco de dados de correio para desempenho, portanto também incluirão o MDA se o banco de dados personalizado for desejado. Pessoas que usam formatos de arquivo padrão (como todos os e-mails em um arquivo de texto) podem escolher qualquer MDA.

O MTA mais conhecido é o sendmail. O Postfix é outro popular e pretende ser mais simples e mais seguro que o sendmail.

Se você usa formatos de arquivo padrão para armazenar e-mails, seu MTA também pode enviar e-mails. Como alternativa, você pode usar algo como o **procmail**, que permite definir filtros personalizados para processar e-mails e filtrá-los.

O Dovecot é um popular servidor POP / IMAP devido à sua facilidade de uso e baixa manutenção. Cyrus IMAP é outra opção.

Para compartilhamento de arquivos, o Samba é o vencedor claro. O Samba permite que uma máquina Linux pareça uma máquina Windows para poder compartilhar arquivos e participar de um domínio do Windows. O Samba implementa os componentes do servidor, como disponibilizar arquivos para compartilhamento e determinadas funções de servidor do Windows, e também o cliente, para que uma máquina Linux possa consumir um compartilhamento de arquivos do Windows.

Se você tem máquinas Apple em sua rede, o projeto **Netatalk** permite que sua máquina Linux se comporte como um servidor de arquivos da Apple.

O protocolo nativo de compartilhamento de arquivos para UNIX é chamado de **Network File System (NFS)**. Normalmente, o NFS faz parte do kernel, o que significa que um sistema de arquivos remoto pode ser montado como um disco comum, tornando o acesso a arquivos transparente para outros aplicativos.

À medida que sua rede de computadores fica maior, você precisará implementar algum tipo de diretório. O diretório mais antigo é chamado de Domain Name System e é usado para converter um nome como http://www.linux.com em um endereço IP como 192.168.100.100, que é um identificador exclusivo desse computador na Internet. O DNS também contém informações globais como o endereço do MTA para um determinado nome de domínio. Uma organização pode querer executar seu próprio servidor DNS para hospedar seus nomes públicos e também servir como um diretório interno de serviços. O Internet Software Consortium mantém o servidor DNS mais popular, chamado simplesmente de ***bind*** após o nome do processo que executa o serviço.

O DNS é amplamente focado em nomes de computadores e endereços IP e não é facilmente pesquisável. Outros diretórios surgiram para armazenar outras informações, como contas de usuários e funções de segurança. O protocolo **LDAP (Lightweight Directory Access Protocol)** é o diretório mais comum que também alimenta o Active Directory da Microsoft. No LDAP, um objeto é armazenado em uma árvore e a posição desse objeto na árvore pode ser usada para derivar informações sobre o objeto além daquelas armazenadas no próprio objeto. Por exemplo, um administrador Linux pode ser armazenado em uma ramificação da árvore chamada “departamento de TI”, que está sob uma ramificação chamada “Operações”. Assim, pode-se encontrar toda a equipe técnica pesquisando sob a ramificação do departamento de TI. O **OpenLDAP** é o player dominante aqui.

Uma parte final da infra-estrutura de rede é chamada de **DHCP (Dynamic Host Configuration Protocol)**. Quando um computador é inicializado, ele precisa de um endereço IP para a rede local, para que possa ser identificado exclusivamente. O trabalho do DHCP é ouvir solicitações e atribuir um endereço livre do pool DHCP. O Internet Software Consortium também mantém o servidor **ISC DHCP**, que é o player mais comum aqui.

Um banco de dados armazena informações e também permite fácil recuperação e consulta. Os bancos de dados mais populares aqui são **MySQL** e **PostgreSQL**. Você pode inserir números de vendas brutos no banco de dados e, em seguida, usar uma linguagem chamada SQL (Structured Query Language) para agregar vendas por produto e data para produzir um relatório.

2.2.2 Aplicações Desktop

O ecossistema Linux possui uma ampla variedade de aplicativos de desktop. Você pode encontrar jogos, aplicativos de produtividade, ferramentas criativas e muito mais. Esta seção é uma mera pesquisa sobre o que está por aí.

Antes de examinar aplicativos individuais, é útil observar o ambiente da área de trabalho. Um desktop Linux executa um sistema chamado **X Window**, também conhecido como **X11**. O servidor Linux X11 é o **X.org**, que fornece uma maneira de o software operar em um modo gráfico e aceitar a entrada de um teclado e um mouse. Janelas e ícones são manipulados por outro software chamado gerenciador de janelas ou ambiente de área de trabalho. Um gerenciador de janelas é uma versão mais simples do ambiente de área de trabalho, pois fornece apenas o código para desenhar menus e gerenciar as janelas do aplicativo na tela. Um ambiente de área de trabalho faz camadas em recursos como janelas de login, sessões, um gerenciador de arquivos e outros utilitários. Em resumo, uma estação de trabalho Linux somente de texto torna-se uma área de trabalho gráfica com a adição do X-Windows e de um ambiente de área de trabalho ou de um gerenciador de janelas.

Os gerenciadores de janelas incluem o **Compiz**, o **FVWM** e o **Enlightenment**, embora haja muitos mais. Ambientes de desktop são principalmente o **KDE** e o **GNOME**, ambos com seus próprios gerenciadores de janelas. Tanto o KDE quanto o GNOME são projetos maduros com uma quantidade incrível de utilitários construídos para eles, e a escolha é freqüentemente uma questão de preferência pessoal.

Os aplicativos básicos de produtividade, como um processador de texto, uma planilha eletrônica e um pacote de apresentação são muito importantes. Coletivamente, eles são conhecidos como uma suíte de escritório, em grande parte devido ao Microsoft Office, que é o player dominante no mercado.

O **OpenOffice** (às vezes chamado de OpenOffice.org) e o **LibreOffice** oferecem uma suíte de escritório completa, incluindo uma ferramenta de desenho que busca compatibilidade com o Microsoft Office em termos de recursos e formatos de arquivo. Esses dois projetos também são um ótimo exemplo de como a política influencia o código aberto.

Em 1999, a Sun Microsystems adquiriu uma empresa alemã relativamente obscura que estava criando uma suíte de escritório para Linux chamada StarOffice. Logo depois disso, a Sun renomeou o OpenOffice e o lançou sob uma licença de código aberto. Para complicar ainda mais as coisas, o StarOffice permaneceu como um produto proprietário que extraía do OpenOffice. Em 2010, a Sun foi adquirida pela Oracle, que mais tarde entregou o projeto à Apache Foundation.

A Oracle tem um histórico ruim de suporte a projetos de código aberto que adquire, o que levou logo após a aquisição pela Oracle, a bifurcação do projeto para se tornar o LibreOffice. Nesse ponto, dois grupos de pessoas desenvolveram o mesmo software. A maior parte do impulso foi para o projeto LibreOffice e é por isso que ele é incluído por padrão em muitas distribuições Linux.

Para navegar na web, os dois principais concorrentes são o **Firefox** e o **Google Chrome**. Ambos são navegadores da Web de código aberto que são rápidos, possuem muitos recursos e têm excelente suporte para desenvolvedores da Web. Esses dois pacotes são um bom exemplo de como a diversidade é boa para o código aberto - melhorias em um estimulam o outro a tentar o melhor do outro. Como resultado, a Internet tem dois navegadores excelentes que ultrapassam os limites do que pode ser feito na Web e funciona em várias plataformas.

O projeto Mozilla também foi lançado com o **Thunderbird**, um cliente de e-mail desktop completo. O Thunderbird se conecta a um servidor POP ou IMAP, exibe emails localmente e envia emails por meio de um servidor SMTP externo.

Outros notáveis clientes de e-mail são o **Evolution** e o **KMail**, que são os clientes de e-mail do projeto GNOME e KDE. A padronização por meio de POP e IMAP e formatos de e-mail local significa que é fácil alternar entre clientes de e-mail sem perder dados. E-mail baseado na Web também é outra opção.

Para os aplicações de criação, há o **Blender**, o **GIMP** e o **Audacity**, que lidam com a criação de filmes em 3D, manipulação de imagens em 2D e edição de áudio, respectivamente. Eles tiveram vários graus de sucesso em mercados profissionais. O Blender é usado para tudo, desde filmes independentes até filmes de Hollywood, por exemplo.

2.2.3 Ferramentas de Console

A história do desenvolvimento do UNIX mostra considerável sobreposição entre as habilidades de desenvolvimento de software e administração de sistemas. As ferramentas que permitem gerenciar o sistema possuem recursos de linguagens de computador, como loops, e algumas linguagens de computador são amplamente utilizadas na automação de tarefas de administração de sistemas. Assim, deve-se considerar essas habilidades como complementares.

No nível básico, você interage com um sistema Linux por meio de um shell, independentemente de estar se conectando ao sistema remotamente ou de um teclado conectado. O trabalho do shell é aceitar comandos, como manipulações de arquivos e iniciar aplicativos, e passá-los para o kernel do Linux para execução. Aqui, mostramos uma interação típica com o shell do Linux:

```
sysadmin@localhost:~$ ls -l /tmp/*.gz
-rw-r--r-- 1 sean root 246841 Mar 5 2013 /tmp/fdboot.img.gz
sysadmin@localhost:~$ rm /tmp/fdboot.img.gz
```

O usuário recebe um **prompt** que geralmente termina com um sinal de **$** para indicar uma conta sem privilégios. Qualquer coisa antes do prompt, neste caso, sysadmin @ localhost: ~, é um prompt configurável que fornece informações extras para o usuário. Na figura acima, sysadmin é o nome do usuário atual, localhost é o nome do servidor e ~ é o diretório atual (no UNIX, o símbolo de til é um formato abreviado para o diretório pessoal do usuário). Veremos os comandos do Linux em mais detalhes em capítulos posteriores, mas para concluir a explicação, o primeiro comando lista os arquivos com o comando ls, recebe algumas informações sobre o arquivo e, em seguida, remove esse arquivo com o comando rm.

O shell do Linux fornece uma linguagem rica para iterar arquivos e customizar o ambiente, tudo sem sair do shell. Por exemplo, é possível escrever uma única linha de comando que encontre arquivos com conteúdo que corresponda a um determinado padrão, extraia informações úteis do arquivo e copie as novas informações em um novo arquivo.

O Linux oferece uma variedade de shells para escolher, diferindo principalmente em como e o que pode ser personalizado, além da sintaxe da linguagem de script interna. As duas famílias principais são a **Bourne shell** e a **C shell**. O Bourne shell foi nomeado com o nome do criador (Stephen Bourne) e o C shell foi nomeado porque a sintaxe pega emprestado muito da linguagem C. Como esses dois shells foram inventados na década de 1970, existem versões mais modernas, o **Bourne Again Shell (Bash)** e o **tcsh (tee-cee-shell)**. O Bash é o shell padrão na maioria dos sistemas, embora você possa ter quase certeza de que o tcsh está disponível se essa for sua preferência.

Outras pessoas pegaram seus recursos favoritos do Bash e do tcsh e fizeram outros shells, como o **Korn shell (ksh)** e o **zsh**. A escolha dos shells é pessoal. Se você puder se familiarizar com o Bash, poderá operar efetivamente na maioria dos sistemas Linux. Depois disso, você pode se expandir e experimentar novos shells para ver se eles ajudam sua produtividade.

Ainda mais dividido que o gosto pelos shells é a escolha dos editores de texto. Um editor de texto é usado no console para editar arquivos de configuração. Os dois principais campos são **vi** (ou o mais moderno **vim**) e **emacs**. Ambos são ferramentas notavelmente poderosas para editar arquivos de texto, eles diferem no formato dos comandos e como você escreve

plugins para eles. Os plugins podem ser desde o destaque de sintaxe de projetos de software até calendários integrados.

Tanto o **vim** quanto o **emacs** são complexos e possuem uma curva de aprendizado acentuada. Isso não é útil se tudo que você precisa é de edição simples de um pequeno arquivo de texto. Portanto **pico** e **nano** estão disponíveis na maioria dos sistemas (sendo este último um derivado do primeiro) e fornecem edição de texto muito básica.

Mesmo se você escolher não usar o **vi**, você deve se esforçar para obter alguma familiaridade básica, porque o **vi** básico está em todos os sistemas Linux. Se você está restaurando um sistema Linux quebrado rodando no modo de recuperação da distribuição, certamente terá o **vi** disponível.

Se você tiver um sistema Linux, precisará adicionar, remover e atualizar o software. Em um determinado momento, isso significou baixar o código-fonte, configurá-lo, criá-lo e copiar arquivos em cada sistema. Felizmente, as distribuições criaram pacotes que são cópias compactadas do aplicativo. Um gerenciador de pacotes cuida de controlar quais arquivos pertencem a qual pacote e até mesmo baixar atualizações de um servidor remoto chamado repositório. Nos sistemas Debian, as ferramentas incluem **dpkg**, **apt-get** e **apt-cache**. Nos sistemas derivados da Red Hat, você usa **rpm** e **yum**. Nós olharemos mais em pacotes depois.

2.2.4 Ferramentas de Desenvolvimento

Não é de surpreender que, como software baseado em contribuições de programadores, o Linux tenha um excelente suporte para o desenvolvimento de software. Os shells são construídos para serem programáveis e existem poderosos editores incluídos em todos os sistemas. Há também muitas ferramentas de desenvolvimento disponíveis, e muitas linguagens modernas tratam o Linux como um cidadão de primeira classe.

As linguagens de computador fornecem um meio para um programador inserir instruções em um formato legível mais humano, e para essas instruções acabarem sendo traduzidas em algo que o computador entende. As linguagens se enquadram em um dos dois grupos: **interpretadas** ou **compiladas**. Uma linguagem interpretada traduz o código escrito em código de computador enquanto o programa é executado, e uma linguagem compilada é traduzida de uma só vez.

O próprio Linux foi escrito em uma linguagem compilada chamada **C**. O principal benefício é que a própria linguagem mapeia de perto o código de máquina gerado para que um programador experiente possa escrever código pequeno e eficiente. Quando a memória do computador era medida nos Kilobytes, isso era muito importante. Mesmo com grandes tamanhos de memória hoje, o C ainda é útil para escrever código que deve ser executado rapidamente, como um sistema operacional.

C foi estendido ao longo dos anos. Há o **C ++**, que adiciona suporte a objetos a C (um estilo diferente de programação), e o **Objective C** que tomou outro rumo e está em uso pesado nos produtos da Apple.

A linguagem **Java** dá um giro diferente na abordagem compilada. Em vez de compilar para código de máquina, o Java imagina primeiro uma CPU hipotética chamada Java Virtual Machine (JVM) e compila todo o código para isso. Cada computador host, em seguida, executa o software da JVM para converter as instruções da JVM (chamadas de bytecode) em instruções nativas.

A tradução extra com o Java pode fazer você pensar que seria lento. No entanto, a JVM é bastante simples, portanto, pode ser implementada de maneira rápida e confiável em qualquer coisa, desde um computador potente até um dispositivo de baixa energia que se conecta a uma televisão. Um arquivo Java compilado também pode ser executado em qualquer computador implementando a JVM!

Outro benefício de compilar para um destino intermediário é que a JVM pode fornecer serviços para o aplicativo que normalmente não estaria disponível em uma CPU. Alocar memória a um programa é um problema complexo, mas isso está embutido na JVM. Isso também significa que os fabricantes de JVM podem concentrar seus aprimoramentos na JVM como um todo, portanto, qualquer progresso que eles fizerem estará imediatamente disponível para os aplicativos.

Linguagens interpretadas, por outro lado, são traduzidas para código de máquina à medida que são executados. O poder extra do computador gasto com isso pode ser recuperado pelo aumento da produtividade que o programador ganha ao não ter que parar de trabalhar para compilar. Linguagens interpretadas também tendem a oferecer mais recursos do que as linguagens compiladas, o que significa que muitas vezes é necessário menos código. O próprio interpretador de linguagem geralmente é escrito em outra linguagem, como C e, às vezes, até mesmo em Java! Isso significa que uma linguagem interpretada está sendo executada na JVM, que é traduzida no tempo de execução em código de máquina real.

Perl é uma linguagem interpretada. Perl foi originalmente desenvolvido para realizar manipulação de texto. Ao longo dos anos, ele ganhou popularidade com os administradores de sistemas e ainda continua sendo aprimorado e usado em tudo, desde a automação até a construção de aplicativos da web.

PHP é uma linguagem que foi originalmente construída para criar páginas web dinâmicas. Um arquivo PHP é lido por um servidor da Web, como o Apache. Tags especiais no arquivo indicam que partes do código devem ser interpretadas como instruções. O servidor da Web reúne todas as partes diferentes do arquivo e as envia para o navegador da web. As principais vantagens do PHP são que é fácil de aprender e está disponível em praticamente qualquer sistema. Por causa disso, muitos projetos populares são construídos em PHP. Exemplos notáveis incluem WordPress (blogs), cactos (para monitoramento) e até partes do Facebook.

Ruby é outra linguagem que foi influenciada por Perl e Shell, juntamente com muitas outras línguas. Ele torna tarefas de programação complexas relativamente fáceis e, com a inclusão da estrutura Ruby on Rails, é uma opção popular para a criação de aplicativos da Web complexos. O Ruby também é a linguagem que alimenta muitas das principais ferramentas de automação, como o Chef e o Puppet, que tornam o gerenciamento de um grande número de sistemas Linux muito mais fácil.

O **Python** é outra linguagem de script que é de uso comum. Assim como o Ruby, ele torna as tarefas complexas mais fáceis e possui um framework chamado Django que torna a construção de aplicativos da Web muito fácil. O Python tem excelentes habilidades de processamento estatístico e é um favorito na academia.

Uma linguagem é apenas uma ferramenta que torna mais fácil informar ao computador o que você deseja fazer. Uma biblioteca agrupa tarefas comuns em um pacote distinto que pode ser usado pelo desenvolvedor. O **ImageMagick** é uma dessas bibliotecas que permite aos programadores manipular imagens no código. O ImageMagick também vem com algumas ferramentas de linha de comando que permitem processar imagens de um shell e aproveitar os recursos de script existentes.

O **OpenSSL** é uma biblioteca criptográfica que é usada em tudo, desde servidores da Web até a linha de comando. Ele fornece uma interface padrão para que você possa adicionar criptografia ao seu script Perl, por exemplo.

Em um nível muito mais baixo é a biblioteca C. Isso fornece um conjunto básico de funções para leitura e gravação em arquivos e exibições, que é usado por aplicativos e outros linguagens.

2.3 Entendendo Software e Licenciamento de Código Aberto

Quando falamos em comprar software, há três componentes distintos:

- **Propriedade -** Quem é o proprietário da propriedade intelectual por trás do software?
- **Transferência de dinheiro -** Como o dinheiro muda de mãos?
- **Licenciamento -** O que você ganha? O que você pode fazer com o software? Você pode usá-lo em apenas um computador? Você pode dar a outra pessoa?

Na maioria dos casos, a propriedade do software permanece com a pessoa ou empresa que o criou. Os usuários só recebem uma licença para usar o software. Esta é uma questão de lei de direitos autorais. A transferência de dinheiro depende do modelo de negócios do criador. É o licenciamento que realmente diferencia o software de código aberto do software de código fechado.

Dois exemplos contrastantes farão as coisas começarem.

Com o Microsoft Windows, a Microsoft Corporation possui a propriedade intelectual. A licença em si, o **Contrato de Licença de Usuário Final (EULA)**, é um documento legal personalizado no qual você deve clicar, indicando sua aceitação, para instalar o software. A Microsoft mantém o código-fonte e distribui apenas cópias binárias por meio de canais autorizados. Para a maioria dos produtos de consumo, você tem permissão para instalar o software em um computador e não tem permissão para fazer cópias do disco que não sejam de backup. Você não tem permissão para fazer engenharia reversa do software. Você paga por uma cópia do software, que recebe atualizações secundárias, mas não grandes atualizações.

Linux é de propriedade de Linus Torvalds. Ele colocou o código sob uma licença chamada **GNU Public License versão 2 (GPLv2)**. Essa licença, entre outras coisas, diz que o código-fonte deve ser disponibilizado para qualquer um que peça e que você tem permissão para fazer as alterações desejadas. Uma ressalva para isso é que, se você fizer alterações e distribuí-las, deverá colocar as alterações sob a mesma licença para que outras pessoas possam se beneficiar. A GPLv2 também diz que você não tem permissão para cobrar pela distribuição do código fonte além dos custos reais de fazê-lo (como copiá-lo para mídia removível).

Em geral, quando você cria algo, você também tem o direito de decidir como ele é usado e distribuído. **Software livre e de código aberto (Free and Open Source - FOSS)** refere-se ao software em que esse direito foi abandonado e você tem permissão para visualizar o código-fonte e redistribuí-lo. Linus Torvalds fez isso com o Linux - mesmo que ele tenha criado o Linux, ele não pode dizer que você não pode usá-lo no seu computador porque ele desistiu da licença GPLv2.

O licenciamento de software é uma questão política e não deve surpreender que existam muitas opiniões diferentes. As organizações criaram sua própria licença que incorpora suas visões particulares, por isso é mais fácil escolher uma licença existente do que criar a sua própria. Por exemplo, universidades como o Instituto de Tecnologia de Massachusetts (MIT) e a Universidade da Califórnia criaram licenças, assim como projetos como a Apache Foundation. Além disso, grupos como a Free Software Foundation criaram suas próprias licenças para promover sua agenda.

2.3.1 A Free Software Foundation e a Open Source I

Dois grupos podem ser considerados as forças mais influentes no mundo do código aberto: a **Free Software Foundation (FSF)** e a **Open Source Initiative (OSI)**.

A Free Software Foundation foi fundada em 1985 por **Richard Stallman (RMS)**. O objetivo da FSF é promover o **Free Software**. O Free Software não se refere ao preço, mas à liberdade de compartilhar, estudar e modificar o código-fonte subjacente. É a visão da FSF que software proprietário (software distribuído sob uma licença de código fechado) é ruim. A FSF também defende que as licenças de software devem impor a abertura das modificações. É sua opinião que, se você modificar o Free Software, deverá compartilhar suas alterações. Essa filosofia específica é chamada de *copyleft*.

A FSF também é contra patentes de software e atua como um cão de guarda para as organizações de padrões, falando quando um padrão proposto pode violar os princípios do Software Livre, incluindo itens como **Gerenciamento de Direitos Digitais (Digital Rights Management -DRM)** que podem restringir o que você poderia fazer com o serviço.

A FSF desenvolveu seu próprio conjunto de licenças, como a GPLv2 e a GPLv3, e as licenças Lesser GPL versões 2 e 3 (LGPLv2 e LGPLv3). As licenças menores são muito parecidas com as licenças regulares, exceto pelo fato de terem provisões para vincular-se a softwares não-livres. Por exemplo, na GPLv2, você não pode redistribuir software que usa uma biblioteca de código fechado (como um driver de hardware), mas a variante *Lesser* permite isso.

As mudanças entre as versões 2 e 3 são amplamente focadas no uso do Software Livre em um dispositivo fechado de hardware que foi chamado **Tivoização**. A TiVo é uma empresa que constrói um gravador de vídeo digital de televisão em seu próprio hardware e usa o Linux como base para seu software. Embora o TiVo tenha distribuído o código-fonte para sua versão do Linux, conforme exigido pela GPLv2, o hardware não executaria nenhum binário modificado com o uso de assinatura digital. Logo, Tivoização significa computadores (chamados de “appliances”) que contêm software coberto pela GPL que você não pode alterar, porque o dispositivo se desliga ao detectar software modificado. O motivo usual para a tivoização é que o software apresenta características que o fabricante acredita que muitas pessoas não vão gostar. Os fabricantes destes computadores tomam vantagem da liberdade que o Software Livre oferece, mas eles não deixam você fazer o mesmo.Aos olhos da FSF, isso foi contra o espírito da GPLv2, então eles adicionaram uma cláusula específica à versão 3 da licença, com o objetivo de bloquear a tivoização. Linus Torvalds concorda com a TiVo neste assunto e optou por permanecer na GPLv2.

A Open Source Initiative foi fundada em 1998 por **Bruce Perens** e **Eric Raymond (ESR)**. Eles acreditam que o Software Livre era politicamente muito carregado e que licenças menos extremas eram necessárias, particularmente em torno dos aspectos do copyleft das licenças da FSF. A OSI acredita que não apenas a fonte deve estar disponível gratuitamente, mas também que nenhuma restrição deve ser imposta ao uso do software, independentemente do uso pretendido. Ao contrário da FSF, o OSI não possui seu próprio conjunto de licenças. Em vez disso, o OSI tem um conjunto de princípios e adiciona outras licenças a essa lista se atender a esses princípios, chamados de ***licenças Open Source***. O software que está em conformidade com uma licença Open Source é, portanto, Software Open Source.

Algumas das licenças Open Source são a família de licenças BSD, que são muito mais simples que a GPL. Eles apenas afirmam que você pode redistribuir a origem e os binários

desde que mantenha avisos de direitos autorais e não implique que o criador original endossa sua versão. Em outras palavras, "faça o que quiser com esse software, mas não diga que você o escreveu". A licença do MIT tem o mesmo espírito, apenas com palavras diferentes.

As licenças da FSF, como a GPLv2, também são licenças Open Source. No entanto, muitas licenças open source, como BSD e MIT, não contêm as cláusulas do copyleft e, portanto, não são aceitáveis para a FSF. Essas licenças são chamadas licenças de software livre permissivas porque são permissivas em como você pode redistribuir o software. Você pode adquirir o software licenciado da BSD e incluí-lo em um produto de software fechado, desde que atribua a devida atribuição.

2.3.2 Mais termos para a mesma coisa

Em vez de se debruçar sobre os pontos mais delicados do Free Source versus Open Source, a comunidade começou a se referir a tudo como Free e Open Source Software (FOSS). A palavra inglesa “livre” pode significar “livre como no almoço” (como em nenhum custo) ou “livre como no discurso” (como em nenhuma restrição). Essa ambigüidade levou à inclusão da palavra **libre** para se referir à última definição. Assim, acabamos com **Software Free / Libre / Open Source** (FLOSS).

Embora esses termos sejam convenientes, eles escondem as diferenças entre as duas escolas de pensamento. No mínimo, quando você usa o software FOSS, sabe que não precisa pagar por isso e pode redistribuí-lo como desejar.

2.3.3 Outros Esquemas de Licenciamento

Licenças FOSS são principalmente relacionadas ao software. As pessoas colocaram trabalhos como desenhos e planos sob licenças FOSS, mas essa não era a intenção.

Quando o software foi colocado no **domínio Público**, o autor renunciou a todos os direitos, incluindo os direitos autorais sobre o trabalho. Em alguns países, esse é o padrão quando o trabalho é feito por uma agência do governo. Em alguns países, o trabalho protegido por direitos autorais se torna de domínio público depois que o autor morreu e um longo período de espera expirou.

A organização **Creative Commons (CC)** criou as **licenças Creative Commons** que tentam abordar as intenções por trás das licenças FOSS para entidades que não são de software. As licenças CC também podem ser usadas para restringir o uso comercial, se isso for o desejo do detentor dos direitos autorais. As licenças CC são:

- **Atribuição (CC BY) -** assim como a licença BSD, você pode usar o conteúdo CC BY para qualquer uso, mas deve creditar o detentor dos direitos autorais.
- **CompartilhaIgual (CC BY-SA) -** uma versão copyleft da licença de atribuição. Os trabalhos derivados devem ser compartilhados sob a mesma licença, bem como nos ideais do Free Software.
- **SemDerivações (CC BY-ND) -** você pode redistribuir o conteúdo sob as mesmas condições que o CC-BY, mas não pode alterá-lo
- **NãoComercial (CC BY-NC) -** assim como CC BY, mas você não pode usá-lo para fins comerciais
- **Nãocomercial-CompartilhaIgual (CC-BY-NC-SA) -** Cria a licença CC BY-NC, mas exige que suas alterações sejam compartilhadas sob a mesma licença.
- **NãoComercial-SemDerivações (CC-BY-NC-ND) -** Você está compartilhando o conteúdo a ser usado para fins não comerciais, mas as pessoas não podem alterar o conteúdo.
- **Todos direitos concedidos (CC0) -** Esta é a versão Creative Commons do domínio público.

As licenças acima podem ser resumidas como CompartilhaIgual ou sem restrições, e se o uso comercial ou derivações são permitidos ou não.

2.3.4 Modelos de negócios de código aberto

Se você está dando seu software de graça, como você pode ganhar dinheiro com isso?

A maneira mais simples de ganhar dinheiro é vender suporte ou garantia em torno do software. Você pode ganhar dinheiro instalando o software para as pessoas, ajudando as pessoas quando elas têm problemas ou corrigindo erros por dinheiro. Você é efetivamente um consultor.

Você também pode cobrar por um serviço ou assinatura que aprimore o software. O projeto de gravador de vídeo digital Open Source MythTV é um excelente exemplo. O software é gratuito, mas você pode pagar para ligá-lo a um serviço de listagem de TV para saber a que horas determinados programas de televisão estão ligados.

Você pode empacotar hardware ou adicionar software de código fechado para vender junto com o software livre. Eletrodomésticos e sistemas embarcados que usam Linux podem ser desenvolvidos e vendidos. Muitos firewalls e dispositivos de entretenimento do consumidor seguem esse modelo.

Você também pode desenvolver software de código aberto como parte de seu trabalho. Se você criar uma ferramenta para tornar sua vida mais fácil em seu trabalho regular, poderá convencer seu empregador a permitir que você a abra. Pode ser uma situação em que você estava trabalhando no software enquanto era pago, mas o licenciamento como fonte aberta permitiria que outras pessoas com o mesmo problema fossem ajudadas e até mesmo contribuíssem.

Na década de 1990, Gerald Combs estava trabalhando em um provedor de serviços de Internet e começou a escrever sua própria ferramenta de análise de rede porque ferramentas semelhantes na época eram muito caras. Mais de 600 pessoas já contribuíram para o projeto, chamado Wireshark. Hoje em dia, muitas vezes ele é considerado melhor do que as ofertas comerciais e levou a uma empresa que está sendo formada em torno de Gerald para apoiar o Wireshark e vender produtos e suporte que o tornam mais útil. Mais tarde, essa empresa foi comprada por um grande fornecedor de rede que apoia seu desenvolvimento.

Outras empresas obtêm um valor tão grande do software livre que acham que vale a pena contratar pessoas para trabalhar no software em tempo integral. O mecanismo de busca do Google contratou o criador da linguagem de computador Python, e até Linus Torvalds é contratado pela Linux Foundation para trabalhar no Linux. A empresa de telefonia americana AT&T obtém tanto valor dos projetos Ruby e Rails para suas propriedades nas Páginas Amarelas que eles têm um funcionário que não faz nada além de trabalhar para esses projetos.

Uma maneira final das pessoas ganharem dinheiro indiretamente por meio do código aberto é que essa é uma maneira aberta de avaliar as habilidades de uma pessoa. Uma coisa é dizer que você realizou determinadas tarefas no seu trabalho, mas mostrar sua criação e compartilhá-la com o mundo permite que os empregadores em potencial vejam a qualidade do seu trabalho. Da mesma forma, as empresas descobriram que o fornecimento aberto de partes não críticas de seu software interno atrai o interesse de pessoas de maior calibre.