UNIP

UNIVERSIDADE PAULISTA

Sistemas Operacionais Abertos e Mobile

Autor: Prof. Michel Bernardo Fernandes da Silva
Colaboradores: Profa. Vanessa Santos Lessa
Profa. Larissa Rodrigues Damiani

Professor conteudista: Michel Bernardo Fernandes da Silva

Graduado em Engenharia Elétrica, com ênfase em Telecomunicações, pela Escola Politécnica da Universidade de São Paulo (Poli-USP), pós-graduado em Finanças e MBA Executivo no Insper e mestre em Engenharia Elétrica pela Poli-USP. Com experiência profissional de mais de 12 anos na área financeira, já atuou como professor universitário e de pós-graduação em diversas instituições, como FMU, Uninove e Anhanguera. Atualmente, está cursando doutorado em Engenharia Elétrica na Poli-USP. Desde 2013, é professor dos cursos superiores de Tecnologia em diversos *campi* da Universidade Paulista (UNIP). É autor do livro *Cibersegurança: uma visão panorâmica sobre a segurança da informação na internet*, publicado em 2023.

Dados Internacionais de Catalogação na Publicação (CIP)

S586s Silva, Michel Bernardo Fernandes da.

Sistemas Operacionais Abertos e Mobile / Michel Bernardo Fernandes da Silva. – São Paulo: Editora Sol, 2023.

156 p., il.

Nota: este volume está publicado nos Cadernos de Estudos e Pesquisas da UNIP, Série Didática, ISSN 1517-9230.

1. Sistemas operacionais. 2. Memória. 3. Proteção. I. Título.

CDU 681.3.066

U518.62 – 23

Sumário

Sistemas Operacionais Abertos e Mobile

Unidade III

Unidade IV

APRESENTAÇÃO

O objetivo geral desta disciplina é entender quais são as funções que um sistema operacional (SO) deve desempenhar em um sistema de computação, tais como gerenciamento do processador, gerenciamento de memória, memória virtual, sistemas de arquivos e sistemas de E/S (entrada e saída), aspectos de segurança, entre outros.

Um objetivo específico é acentuar as principais características que o SO deve ter, os conceitos sobre sistemas operacionais, sua arquitetura, os algoritmos utilizados nas diferentes atividades de gerência, bem como as formas de implementação em sistemas operacionais abertos e mobile. Sistemas operacionais abertos são aqueles nos quais o usuário possui acesso ao código-fonte do sistema, pode alterá-lo e distribuí-lo com suas modificações, e o Linux é o principal exemplo de sistema aberto.

Você certamente já teve contato com sistemas operacionais Android e iOS, comumente usados em smartphones, tablets, relógios inteligentes ou smartwatches, smart TVs, entre outros dispositivos móveis. Os smartphones são uma poderosa ferramenta para a vida pessoal e profissional e ainda oferecem diversas formas de interação com o usuário, tais como sensores, GPS, acelerômetros e teclados virtuais.

Com a evolução dos equipamentos digitais como smartphones e tablets, que têm diversos dispositivos de entrada/saída como câmeras, microfones, interfaces de rede, houve uma enorme evolução de sistemas operacionais utilizados nesses equipamentos. Eles são conhecidos como sistemas operacionais móveis, e os principais exemplos que estão disponíveis de forma comercial atualmente são os sistemas iOS, desenvolvido pela Apple, e o Android, criado por um consórcio de desenvolvedores denominado Open Handset Alliance e cujo principal colaborador é o Google.

Uma categoria de usuários também já utilizou sistemas operacionais abertos, como o Linux e suas diferentes distribuições. Nesses tipos, é possível customizar o código-fonte do SO, de forma a se adequar melhor em necessidades específicas.

Entretanto, na verdade, há uma interação com uma interface gráfica do SO ou Graphical User Interface (GUI) e com um interpretador de comandos denominado shell. Tanto a interface gráfica como o interpretador de comandos são programas que não fazem parte do núcleo do SO.

Esta disciplina irá utilizar diversos conceitos aprendidos nas áreas de sistemas operacionais e organização de computadores, apresentando o funcionamento e as soluções existentes em sistemas operacionais abertos e móveis.

Podemos considerar o SO como um software básico para o computador, pois sobre ele serão construídos outros softwares. O SO facilita a operação dos outros softwares aplicativos porque contribui com todas as interações necessárias com os recursos de hardware.

INTRODUÇÃO

O computador é um dispositivo físico que necessita de gerenciamento dos recursos de hardware, tais como processador, memória e dispositivos de E/S para poder funcionar. Na ausência de software, o hardware do computador fica inutilizado, resumindo-se apenas a uma caixa plástica com uma coleção de circuitos eletrônicos.

De forma geral, é possível classificar os softwares em dois tipos: programas aplicativos e programas de sistemas.

Os programas aplicativos são implementados para realizar as diferentes necessidades dos usuários, tais como assistir vídeos, fazer pesquisas na internet e enviar e-mail.

Um SO é um software extremamente complexo e com numerosos módulos a serem criados, cada um precisa ter uma delimitação clara do sistema e com entradas, saídas e funções definidas criteriosamente. Somente o núcleo do Linux, cujo código é livre e aberto, possui dezenas de milhões de linhas de código-fonte!

Por sua vez, os programas de sistemas são responsáveis por gerenciar a operação do computador. O principal deles é o SO, que controla os recursos computacionais disponíveis, como memória, processador, acesso a dispositivos de E/S e estabelece acessos ao hardware que os programas aplicativos usam para suas funções. Neste livro-texto, o foco serão os sistemas operacionais.

Segundo Deitel, Deitel e Choffnes (2005), o SO reveza sua execução com a de outros programas, de forma que aparenta estar vigiando, controlando e orquestrando todo o processo computacional. Podemos considerar o SO como um software básico sobre o qual serão desenvolvidos outros softwares.

O conhecimento profundo de sistemas operacionais é importante para os profissionais de computação, já que os mecanismos desenvolvidos pelo SO influenciam de forma direta o comportamento e o desempenho de aplicações. Mesmo dispositivos computacionais de menor porte, tais como smartphones, smartwatches ou assistentes virtuais, possuem SO para realizar suas operações. Entretanto, em alguns dispositivos, ele é projetado para funcionar sem intervenção do usuário.

Este livro-texto está dividido em quatro unidades.

Na primeira unidade, será abordada a visão básica de sistemas operacionais abertos, como Linux, e de sistemas mobile, em especial, Android e iOS. Serão descritas a sua evolução histórica, suas funções essenciais de gerência de recursos e as principais características e a organização desse tipo de software. Adicionalmente, para o Linux, serão acentuadas suas diversas distribuições e sua configuração básica desse sistema.

Na segunda unidade, serão descritos os conceitos e as técnicas aplicáveis à gerência de memória, como, swapping, memória virtual e alocação de memória, mostrando como essas atividades de gerenciamento de memória são aplicadas em sistemas abertos e mobile.

Na terceira unidade, será detalhada a gerência de dispositivos de E/S para o SO, analisando tanto o hardware quanto as interfaces de software necessárias para as operações de E/S.

Por fim, a quarta unidade estudará os requisitos de sistema de arquivos e a organização de diretórios. Adicionalmente, serão apresentados aspectos de proteção e segurança de sistemas operacionais, detalhando como esses conceitos são aplicados aos sistemas operacionais.

Aproveite a leitura e bons estudos!

Unidade I

1 SISTEMA OPERACIONAL LINUX

Inicialmente, apresentaremos o SO Linux, analisando seu histórico e a motivação para o desenvolvimento, o tipo de licença usada, as distribuições, seus principais componentes, informações sobre a utilização de Linux e locais onde podemos encontrar sua documentação. Linux é um SO de código aberto e é uma variante do Unix. Dispositivos de diferentes portes utilizam esse SO, que também é adotado por muitos desenvolvedores para gerenciar seus ambientes criar novos códigos para o mercado.

1.1 Histórico do Linux

Na década de 1960, houve um esforço para desenvolver sistemas operacionais de tempo compartilhado. A versão inicial do Unix foi criada em 1969 com a liderança de Ken Thompson no grupo de pesquisas da Bell Laboratories, feita em assembly, uma linguagem de máquina e de baixo nível para um minicomputador PDP-7 (Machado; Maia, 2013). Para que o sistema tivesse portabilidade para outras plataformas, Thompson desenvolveu uma linguagem de alto nível denominada B. Posteriormente, Dennis Ritchie participou da etapa de melhoria do sistema, que foi escrito na linguagem de programação C. Além da utilização no Bell Laboratories, o Unix começou a ser aplicado em diversas universidades.

O Unix é portável, eficiente, com alta segurança e desempenho de operações de rede. Como sistema portátil, ele pode ser executado independentemente da arquitetura do computador. Por ser multitarefas, pode executar diversas tarefas simultaneamente. Também é multiusuário, pois permite que vários usuários o operem ao mesmo tempo. O Unix é considerado o pai dos sistemas operacionais, pois serviu de base para vários sistemas subsequentes.

Observação

Um SO tem portabilidade ou é portátil quando pode ser transferido com facilidade de uma plataforma de hardware para outra com configurações diferentes.

Segundo Ferreira (2003), Richard Stallman e Linus Torvalds são responsáveis pela criação do sistema em meados de 1984. No ano de 1983, Richard Stallman iniciou o projeto que ficou conhecido como GNU, e seu objetivo era desenvolver um SO baseado no Unix, mas sem cobrança de licenças de uso e com permissão de copiar, estudar, modificar ou até distribuir seu código.

Desde o princípio, o Linux foi criado para ser multitarefa e multiusuário. Algumas ferramentas foram sendo desenvolvidas, por exemplo, editor de texto e compiladores de linguagem C. A partir de 1992, aproximou-se de sua fase quase adulta, mas faltava algo muito importante: o núcleo ou kernel do SO.

O time liderado por Stallman estava envolvido em um projeto de desenvolvimento de um código chamado Hurd, que nada mais era do que justamente um núcleo. Esse projeto já era mantido por empresas que incentivavam e apoiavam o desenvolvimento de software livre, e ele foi aproveitado no projeto de desenvolvimento do novo SO. Richard Stallman fundou em 1985 a Free Software Foundation – Fundação para Software Livre (FSF), uma organização sem fins lucrativos que se dedica à eliminação de restrições sobre a cópia, o estudo e a modificação de programas de computador (bandeiras do movimento do software livre).

Essa fundação foi importante para ajudar na concepção do novo software. Ainda segundo Ferreira (2003), a FSF também possui a responsabilidade de manter a General Public License (GPL). Essa licença defende juridicamente um software livre e mantém as quatro liberdades de uso do SO, que são:

- **Liberdade 0**: executar o programa para qualquer finalidade.
- **Liberdade 1**: estudar o funcionamento do programa, podendo fazer alterações e adaptá-lo conforme a necessidade. É preciso ter acesso ao código-fonte.
- **Liberdade 2**: redistribuir cópias do software, de modo que ajude o próximo.
- **Liberdade 3**: modificar o programa e disponibilizar as alterações.

Em 1991, Linus Torvalds criou um novo kernel chamado Linux. Este se uniu às ferramentas elaboradas pelo projeto GNU, surgindo o que é conhecido atualmente como GNU/Linux. Seu desenvolvimento teve como base o sistema Minix, criado em 1987 por Andrew Tanenbaum. O Minix baseava-se nos padrões do Unix e, na época, era usado apenas na área acadêmica para estudos e desenvolvimentos específicos.

Linux parece-se muito com qualquer outro sistema Unix; na verdade, a compatibilidade com o Unix era um objetivo importante do projeto do Linux. No entanto, o Linux é muito mais jovem do que a maioria dos sistemas Unix. Seu desenvolvimento começou em 1991, quando um estudante universitário finlandês, Linus Torvalds, começou a elaborar um kernel pequeno, mas autossuficiente, para o processador 80386, o primeiro processador verdadeiro de 32 bits no conjunto de CPUs da Intel compatíveis com PCs.

O nome Linux foi batizado assim porque é a junção do nome de seu desenvolvedor, Linus Torvalds, com Unix, o SO tomado como base para a criação do kernel do Linux. Entretanto, o Unix foi elaborado para servidores de grande porte, os mainframes.

No início de seu desenvolvimento, o código-fonte do Linux foi disponibilizado livremente – sem custo e com restrições mínimas de distribuição – na internet. Como resultado, a história do Linux tem sido de colaboração entre muitos desenvolvedores de todo o mundo, correspondendo-se quase exclusivamente pela internet. A partir de um kernel inicial que implementava parcialmente um pequeno subconjunto de serviços do sistema Unix, o Linux cresceu para incluir toda a funcionalidade esperada de um Unix moderno.

De acordo com Nemeth, **Snyder e Hein** (2007), o Linux é uma reimplementação de baixo para cima do padrão POSIX e é capaz de rodar em diversas plataformas de hardware, sendo compatível com a maior parte dos softwares Unix. As principais diferenças entre Linux e Unix são que o código-fonte do Linux é aberto, não há pagamento de licença e o sistema é desenvolvido de forma cooperativa, com colaboração proveniente de milhares de indivíduos e organizações.

Em seus primórdios, o desenvolvimento do Linux evoluiu em grande parte em torno do kernel central do SO ou núcleo. Nele há execução em modo privilegiado que gerencia todos os recursos do sistema e que interage diretamente com o hardware do computador. Naturalmente, precisamos de muito mais do que esse kernel para produzir um SO completo. Assim, temos que diferenciar o kernel do Linux de um sistema Linux completo. O kernel do Linux é um componente de software original criado a partir do zero pela comunidade Linux. O Linux, conforme o conhecemos hoje, inclui vários componentes, alguns escritos do zero, outros emprestados de outros projetos de desenvolvimento, e ainda outros criados em colaboração com outras equipes.

Saiba mais

O documentário indicado a seguir conta a história dos projetos GNU, Linux, e o movimento do software livre. Lançado em 2011 em comemoração aos 20 anos do kernel do Linux, o filme mostra uma série de entrevistas com empreendedores, programadores e hackers, tais como Richard Stallman, Michael Tiemann, Linus Torvalds, Larry Augustin e Eric S. Raymond.

REVOLUTION OS. 2012. 1 vídeo (85 min.). Publicado por Jardel Sacramento Disponível em: https://shre.ink/n9PR. Acesso em: 25 set. 2023.

Adicionalmente, vale pontuar que o Linux não é a única versão gratuita do Unix existente no mundo. Foram desenvolvidos pela Berkeley Software Distribution (BSD), originada na Universidade de Berkeley, nos EUA, os sistemas operacionais FreeBSD, NetBSD e OpenBSD. Eles são comparáveis ao Linux em relação a recursos e confiabilidade. Entretanto, há carência no suporte de terceiros como fornecedores de software.

1.2 Características do Linux

Por se tratar de um SO de código aberto e livre, o Linux é desenvolvido de forma voluntária por programadores experientes, hackers e contribuidores espalhados ao redor do mundo, com apoio de grandes empresas, como IBM, RedHat e Intel. O objetivo é a contribuição para a melhoria e se crescimento.

Alguns dos programadores experientes estavam incomodados com a baixa qualidade dos sistemas existentes, que causavam inúmeros erros (bugs) e, mais recentemente, pelo excesso de propaganda.

O Linux é capaz de conviver sem nenhum tipo de conflito com outros sistemas operacionais, por exemplo, Windows e OS/2 no mesmo computador ou dispositivo. Ele também possibilita a conectividade com outros tipos de plataformas, como Apple, Sun, Macintosh, Unix e Windows. A figura a seguir mostra o Tux, o pinguim que é mascote oficial do Linux.

Lembrete

O Linux é um sistema multitarefa, que possibilita a execução de múltiplas tarefas paralelamente. Também é multiusuário, ou seja, diversos usuários podem usar os recursos do computador simultaneamente.

Figura 1 – Tux, mascote oficial do Linux

Disponível em: https://shre.ink/nYP9. Acesso em: 25 set. 2023.

Apesar de ser difícil estimar a participação de mercado do Linux, é possível calcular com algumas premissas. O site *Statcounter*, uma ferramenta gratuita de estatísticas de visitantes, estima a participação de mercado ou market share com base no número de acessos a sites de uma amostra mensal de mais de 5 bilhões de visualizações de páginas. No Brasil, a participação do Linux para desktops foi 2,7% em junho de 2023, o Windows tinha 89,6% e o OS X dos computadores Apple tinha 4,1%. O gráfico a seguir apresenta a evolução histórica da participação de acesso com sistemas Linux entre janeiro de 2009 e junho de 2023. Para efeito de comparação, mundialmente, a participação do Linux é de 3,1% para o mês de junho de 2023.

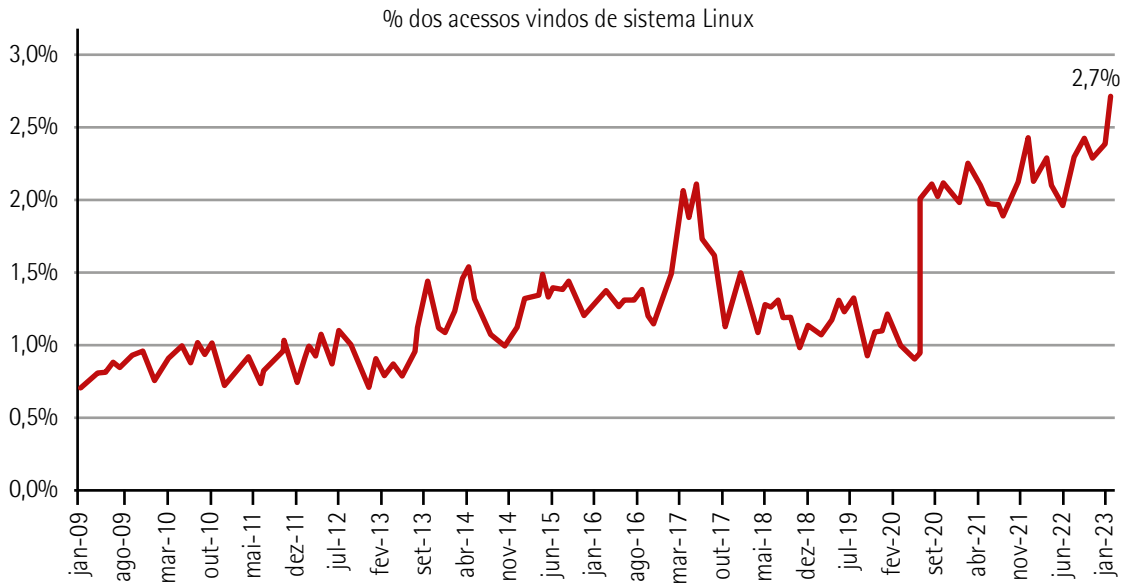

Figura 2 – Participação de Linux nos acessos

Adaptada de: https://shre.ink/nYug. Acesso em: 25 set. 2023.

A utilização de Linux é bem relevante nos servidores usados pelas empresas. Segundo o site *Enterprise Apps Today*, em 2019 sua participação no mercado global de servidores foi de 13,6%, com um aumento em relação a 2018, 12,9%. O gráfico a seguir ilustra a distribuição dos sistemas.

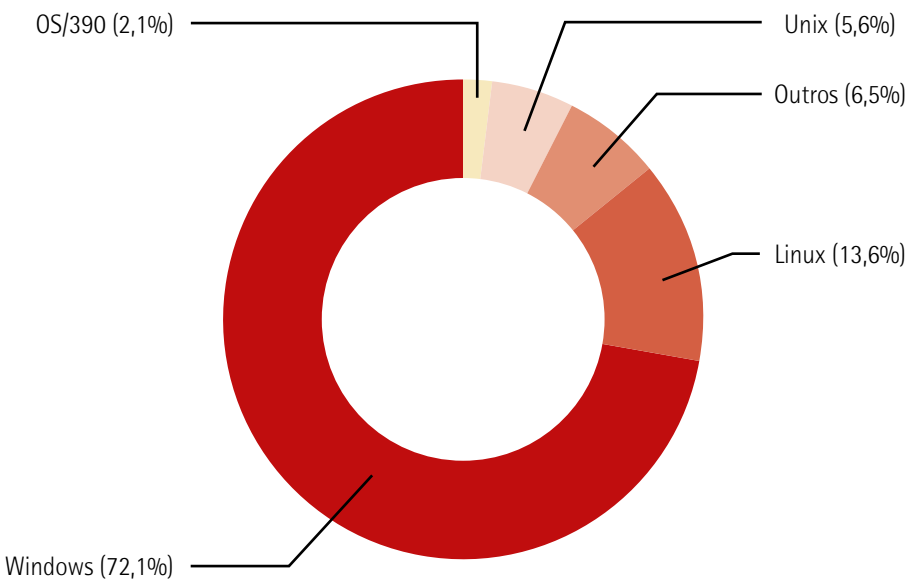

Figura 3 – Participação do mercado global de servidores por SO em 2019

Adaptada de: https://shre.ink/nYbO. Acesso em: 25 set. 2023.

Existem diversas empresas e consultores especializados no suporte ao Linux espalhados por todo o mundo, assim, uma empresa que adotá-lo não ficará desamparada em caso de falhas.

1.3 Componentes do Linux

Ele tem três corpos de código principais, assim como a maioria das implementações tradicionais do Unix: kernel, bibliotecas do sistema e utilitários do sistema.

O kernel é a parte central do sistema e é responsável pela manutenção de todas as suas abstrações importantes, incluindo aspectos como memória virtual, arquivos e processos. No kernel, todo o código é executado em modalidade privilegiada de processador com acesso integral a todos os recursos físicos do computador. No Linux, a modalidade privilegiada é conhecida como modalidade de kernel e não há código de usuário embutido no kernel.

O primeiro kernel do Linux lançado para o público foi a versão 0.01, em maio de 1991. Ele possuía diversas limitações. Como não tinha conexão de rede, somente era compatível com processadores Intel compatíveis da série 80386 e suportava poucos drivers de dispositivos.

Após três anos de desenvolvimento, foi lançada em 1994 a versão Linux 1.0, com a inclusão de conexões de rede segundo o protocolo TCP/IP, padrão do Unix e, posteriormente, utilizado como protocolo padrão da internet.

De acordo com Silberschatz, Galvin e Gagne (2015), a existência de módulos do kernel traz conveniência por várias razões. Como o código-fonte do Linux é de livre acesso, qualquer interessado que deseja escrever código do kernel é capaz de compilar um kernel modificado e reinicializar o sistema com a adição dessa nova funcionalidade.

Lembrete

Um dos componentes do SO é denominado kernel ou núcleo. Ele é responsável pela manutenção de todas as abstrações relevantes do SO, incluindo itens como processos, memória virtual e arquivos.

Por sua vez, as bibliotecas do sistema definem um conjunto padrão de funções por meio das quais as aplicações podem interagir com o kernel. Essas funções implementam grande parte da funcionalidade do SO que não precisa de todos os privilégios do código do kernel. A biblioteca mais importante do sistema é a biblioteca C, conhecida como libc. Além de fornecer a biblioteca C padrão, libc implementa o lado da modalidade de usuário da interface de chamadas de sistema do Linux, assim como outras interfaces críticas de nível de sistema.

Qualquer código de suporte do SO que não precise ser executado em modalidade de kernel é inserido nas bibliotecas do sistema e operado em modalidade de usuário. Diferentemente da modalidade de kernel, a de usuário tem acesso apenas a um subconjunto controlado dos recursos do sistema.

Os utilitários do sistema são programas que executam tarefas de gerenciamento individuais especializadas. Alguns deles são invocados apenas uma vez para inicializar e configurar certos aspectos do sistema.

Um outro tipo de utilitário, chamado daemons na terminologia do Unix, é operado permanentemente, manipulando tarefas, como responder a conexões de rede recebidas, aceitar solicitações de login provenientes de terminais e atualizar arquivos de log.

O quadro a seguir apresenta os diversos componentes do Linux completo.

Quadro 1

Programas de gerenciamento do sistema	Processos de usuário	Programas utilitários de usuário	Compiladores
Bibliotecas compartilhadas do sistema			
Kernel do Linux			
Módulos carregáveis do kernel			

Fonte: Silberschatz, Galvin e Gagne (2015, p. 433).

Embora vários sistemas operacionais modernos tenham adotado uma arquitetura de transmissão de mensagens para seus mecanismos internos do kernel, o Linux mantém o modelo histórico do Unix: o kernel é criado como um binário único, monolítico. A principal razão é o desempenho. Já que todo o código e as estruturas de dados do kernel são mantidos em um único espaço de endereçamento, nenhuma mudança de contexto é necessária quando um processo chama uma função do SO ou quando uma interrupção de hardware é acionada. Além disso, o kernel pode passar dados e fazer solicitações entre vários subsistemas usando invocações de função em C relativamente barata e uma comunicação entre processos (IPC) igualmente pouco complicada. Esse espaço de endereçamento único contém não apenas o código básico de escalonamento de processos ou scheduling e a memória virtual, mas todo o código do kernel, incluindo todo o código de drivers de dispositivos, sistemas de arquivos e conexão de rede.

A recompilação, a revinculação e a recarga do kernel inteiro representam um ciclo complexo a se percorrer quando um programador estiver desenvolvendo um novo driver. Ao utilizar módulos do kernel, não será necessária a criação de um novo kernel para testar um novo driver, pois o driver pode ser compilado por sua conta e carregado no kernel já em execução. Após a geração do novo driver, este poderá ser distribuído como módulo para que outros usuários se beneficiem sem ter que reconstruir seus kernels.

Os módulos do kernel permitem que o Linux seja configurado com um kernel mínimo padrão, sem quaisquer drivers de dispositivos adicionais embutidos. Qualquer driver de dispositivos que o usuário precisar poderá ser carregado explicitamente pelo sistema na inicialização ou carregado automaticamente pelo sistema sob demanda e descarregado quando não estiver mais em uso. Por exemplo, um driver de mouse pode ser carregado quando um mouse USB for conectado ao sistema e descarregado quando o mouse for desconectado.

De acordo com Silberschatz, Galvin e Gagne (2015), o suporte a módulos no Linux é formado por quatro componentes:

- **Sistema de gerenciamento de módulos**: permite que os módulos sejam carregados na memória e se comuniquem com o resto do kernel.

- **Carregador e descarregador de módulos**: utilitários de modalidade de usuário, funcionam com o sistema de gerenciamento de módulos para carregar um módulo na memória.

- **Sistema de registro de drivers**: permite que os módulos informem ao resto do kernel que um novo driver está disponível.

- **Mecanismo de resolução de conflitos**: permite que diferentes drivers de dispositivos reservem recursos de hardware e protejam esses recursos do uso acidental por outro driver.

1.4 Distribuições do Linux

O sistema Linux básico é um ambiente padrão para aplicações e programação de usuários, mas não impõe nenhum meio padrão de gerenciamento da funcionalidade disponível como um todo. À medida que o Linux amadurecia, surgiu a necessidade de outra camada de funcionalidade no topo do sistema, exigência que tem sido atendida por várias distribuições do Linux. Uma distribuição inclui todos os componentes padrões do Linux, mais um conjunto de ferramentas administrativas para simplificar a instalação inicial e a atualização subsequente do Linux e para gerenciar a instalação e a remoção de outros pacotes do sistema.

Uma distribuição moderna também inclui, tipicamente, ferramentas para o gerenciamento de sistemas de arquivos, criação e gestão de contas de usuário, administração de redes, navegadores da web, editores de texto e assim por diante.

Baseado nas quatro definições de liberdade estabelecidas para o uso do GNU/Linux, os usuários iniciaram um processo de personalização do sistema, programando-o de acordo com as necessidades individuais e dando início às distribuições.

Segundo Ferreira (2003), uma distribuição é um conjunto de vários softwares agrupados em mídias. Com esses instaladores customizados, é possível facilitar o trabalho do usuário e dos administradores. As distribuições Linux começaram a ficar mais populares a partir do final dos anos 1990, quando se tornaram uma alternativa livre aos sistemas operacionais que existiam na época. Estes inicialmente eram muito populares na linha de servidores; depois, começaram a participar também do mercado de desktops.

Também conhecidas como distros, cada distribuição possui suas características particulares e muitas vezes exclusivas, por exemplo, a maneira de realizar algum tipo de configuração no sistema, instalação de programas, automatização de tarefas, recuperação de sistemas danificados e monitoramento de redes de computadores. Apesar de suas diferenças, a essência de todas as distros é similar.

Assim, aprendendo a usar o GNU/Linux em determinada distribuição, não haverá dificuldades para trabalhar com outras, porque em geral todas têm a mesma sintaxe de comandos e funcionalidades.

Entretanto, cada uma possui características próprias, tais como sistema de instalação, objetivo, interface gráfica de localização de programas, nomes de arquivos de configuração e de logs, entre outros. A escolha de uma distribuição é pessoal e depende das necessidades de cada usuário (Silva, 2020a).

Conforme Oliveira *et al.* (2009), outro aspecto relativo às distribuições GNU/Linux é a questão comercial. Por que era necessário comprar uma distribuição se o GNU/Linux é um sistema livre? Na verdade, o valor pago no momento da compra de uma distribuição não é do GNU/Linux em si, mas sim de um conjunto de serviços que são disponibilizados pela distribuição.

Há alguns anos, era enviado um CD-ROM ou DVD a ser utilizado para a instalação, que foi substituído pela possibilidade de realizar downloads nos sites das distribuições que disponibilizam as imagens dos CD-ROMs e dos DVDs de instalação para serem copiadas. As distribuições também fornecem um manual de utilização e/ou instalação, um suporte hot-line, isto é, a ajuda de outros usuários, ou ainda algum software específico desenvolvido e vendido com a distribuição.

As distribuições que usam kernel Linux mais conhecidas são:

- Ubuntu.
- Debian.
- Slackware.
- Red Hat.
- Suze.
- CentOS.
- Arch Linux.
- Kali Linux.

Um exemplo de distro com um propósito específico é o Kali Linux, criado pela empresa Offensive Security. Essa distribuição é baseada em Debian e foi projetada para auxiliar principalmente profissionais da área de segurança cibernética com seus inúmeros testes de penetração (penetration testing) e ferramentas para auditoria da segurança. O site oficial dessa distribuição é www.kali.org. A figura a seguir apresenta a interface gráfica do Kali Linux.

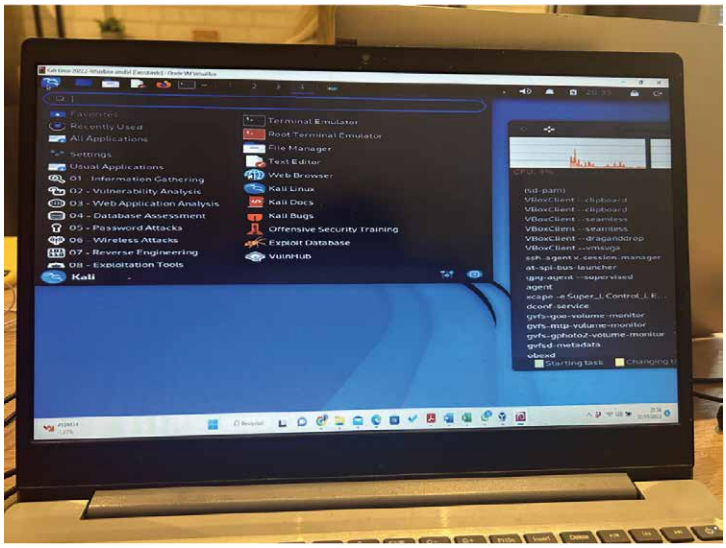

Figura 4 – Interface gráfica da distribuição Kali Linux versão 2022.2

Exemplo de aplicação

A distro Debian, cuja página principal está disponível em www.debian.org, é criada e atualizada através do esforço de voluntários distribuídos por todo o mundo, do mesmo estilo de desenvolvimento GNU/Linux. Por esse motivo, é considerada como a distribuição oficial do projeto GNU.

De acordo com Silva (2020b), a instalação da distribuição do Debian pode ser realizada tanto por meio de pendrives, CD-ROM, Tftp, Ftp, NFS, imagem Docker ou por meio da combinação de vários destes em cada etapa de instalação.

O Debian acompanha dezenas de milhares de programas distribuídos em forma de pacotes, que são mantidos e testados pela pessoa ou grupo responsável por seu empacotamento. Os pacotes são classificados como categorizados ou gerenciados por meio de um avançado sistema de gerenciamento.

Antes do lançamento de uma nova versão, são realizados extensivos testes buscando atingir um alto grau de confiabilidade do sistema. Por meio de um sistema de tratamento de falhas, são reportadas as eventuais falhas ocorridas nos pacotes desenvolvidos. Esse sistema de tratamento de falhas encaminhará a falha encontrada diretamente ao desenvolvedor responsável para avaliação e futura correção. Existem listas de discussões que tratam de forma específica as soluções das falhas existentes na distribuição e qualquer usuário pode se cadastrar nas listas de discussões que estudam especificamente a solução de falhas.

1.5 Licença GPL

Também conhecida como General Public License (GPL), garante liberdades como a execução do programa a qualquer propósito, o estudo do funcionamento do programa e adaptação às suas

necessidades, a redistribuição de cópias para auxiliar os próximos usuários e a possibilidade de aperfeiçoar o programa. Ela pode liberar suas modificações, de modo a permitir que toda a comunidade se beneficie com as alterações.

Essa licença defende juridicamente um software livre e mantém as quatro liberdades de uso do SO já mencionadas: liberdade 0, liberdade 1, liberdade 2 e liberdade 3.

O acesso ao código-fonte é uma condição básica necessária para qualquer software livre.

Observação

GPL é um dos tipos de licenças públicas mais comuns para programa de código aberto e exige que evoluções baseadas no programa original sejam licenciadas sob a mesma licença.

As distribuições livres do Linux são mantidas por comunidades de colaboradores que não objetivam o lucro econômico com o desenvolvimento do software.

A interface de módulos do kernel permite que terceiros escrevam e distribuam, em seus próprios termos, drivers de dispositivos ou sistemas de arquivos que não poderiam ser distribuídos nos termos da GPL.

1.5.1 Diferença entre software livre e software gratuito

Em um software gratuito ou freeware, o usuário pode utilizá-lo sem pagamento de licença para uso ilimitado, mas existe uma proteção dos direitos autorais. Nesse caso, os desenvolvedores mantêm todos os direitos sobre o programa, e o usuário não possui acesso ao código-fonte, portanto, não pode alterá-lo. Em alguns casos, são comercializados serviços ou opções adicionais para o usuário em softwares gratuitos.

Adicionalmente, podem existir limitações de uso do software para softwares gratuitos, como número de usuários ou de funcionalidades. Como não há acesso ao código-fonte do programa, o usuário não pode customizá-lo para suas necessidades específicas e, consequentemente, não consegue distribuir suas alterações.

Por sua vez, no software livre ou aberto é garantida a liberdade para qualquer usuário de distribuir seu código modificado.

Apesar de ser possível realizar o download e a instalação de um kernel do Linux gratuitamente e em alguns minutos, poderão existir despesas para utilização completa do sistema e de softwares desenvolvidos no Linux, como custos de aquisição, integração, modificação, desenvolvimento de bibliotecas de funções e suporte técnico.

1.6 Particularidades de sistemas Linux

No Windows, é utilizado um único banco de dados de opções de configuração com a ajuda de seu registro. Já o Linux não fornece esse recurso, pois não há registro e os arquivos de configuração de todo o sistema ficam armazenados no diretório raiz (/). Arquivos de configuração específicos do usuário, geralmente, estão localizados em diretórios ocultos na pasta inicial.

Desse modo, no Linux não há chance de falha na configuração do sistema. Se um arquivo de configuração for danificado, apenas essa função será interrompida, e o restante funcionará. A figura a seguir ilustra um exemplo de edição de arquivo de configuração no Linux, no caso, é o arquivo interfaces, que permite configurar os dispositivos de rede.

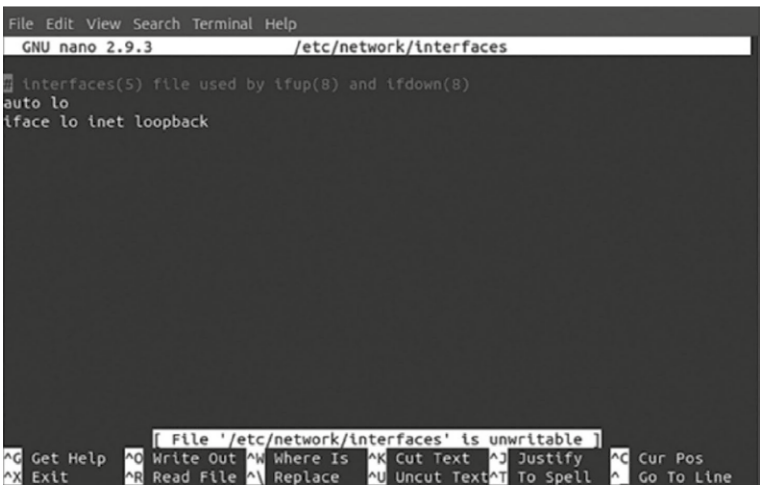

Figura 5 – Exemplo de edição de um arquivo de configuração do Linux

Fonte: Córdova Jr., Ledur e Morais (2019, p. 141).

Adicionalmente, o Linux facilita o backup de arquivos de configuração e a solução de problemas. Nele, os sistemas de arquivos não recebem letras, como no Windows, mas existe um sistema de arquivos de raiz única cujo caminho é /. O disk analyzer mostra o uso e o layout do sistema de arquivos, e o Linux monta novas unidades em pastas dentro do sistema de arquivos raiz.

O Linux introduz o conceito de repositórios de software e, quando desejamos instalar um programa, simplesmente usamos o utilitário de adicionar/remover programas, procuramos o programa e o instalamos. O gerenciador de pacotes exibirá todos os requisitos, denominados dependências, e o processo análogo ocorre durante a desinstalação de um programa.

No Linux, o administrador é chamado de root, e o acesso de root somente é utilizado quando estritamente necessário. Interfaces modernas enviam avisos ao usuário sobre a senha de root quando for preciso. No Linux, sempre devemos fazer o login como usuário comum, que não possui todos os privilégios de acesso ao kernel do SO.

Outra característica do Linux é que os usuários têm controle total das atualizações e podem instalá-las sempre que precisarem, elas levam menos tempo e sem qualquer reinicialização.

1.7 Busca de documentação dentro do SO

Uma parte importante de qualquer SO é a documentação, que envolve os manuais técnicos que descrevem o uso e o funcionamento dos programas. O sistema GNU/Linux tem uma ampla documentação técnica oficial. Os próprios comandos já possuem suas respectivas documentações, e o mais comum é usar o programa MAN (manual), seguido pelo comando desejado que se quer aprimorar.

1.7.1 Tipos de documentação

Existem diversas maneiras para consultar a documentação do sistema GNU/Linux. Além do registro que relaciona os comandos e seus parâmetros, pode-se consultar HowTo e páginas de manuais, que ajudarão o profissional a realizar determinado procedimento, por exemplo, a instalação de um servidor de compartilhamento de arquivos, um servidor para hospedagem de sites, um servidor de banco de dados etc.

- **HowTo**: refere-se a um procedimento detalhado para a realização de determinada atividade. Esse tipo de documentação pode ser encontrado nos diversos sites oficiais ou não oficiais, como blogs.

- **Páginas de manuais**: esse tipo de documentação já está instalado no próprio sistema. A cada novo comando instalado, sua respectiva documentação também o será. As manpages são organizadas por sessões que começam do valor 1 até 9.

- **Guia Foca**: apresenta de forma didática explicações sobre computadores e o sistema GNU/ Linux para usuários iniciantes, intermediários e avançados e aborda formas de administração e segurança do sistema para usuários avançados. Lançado em 1999, o nome Foca é um acrônimo de Fonte de Consulta e Aprendizado.

- **Sites de comunidades de usuários Linux**: o site vivaolinux.com.br possui uma lista de comunidades de usuário de Linux categorizadas por assunto e interesse do usuário.

2 SISTEMAS OPERACIONAIS MOBILE

Um SO ou operational system (OS), em inglês, é um programa, ou seja, um software responsável por gerenciar o hardware de um computador. Tal programa também fornece uma base para os programas aplicativos e atua como intermediário entre o usuário e o hardware do computador.

Segundo Maziero (2019), um SO móvel é utilizado em equipamentos de uso pessoal compactos, como smartphones e tablets. Nesse contexto, as principais prioridades são a gestão eficiente da energia de bateria, a conectividade nos diversos tipos de rede, por exemplo, Wi-Fi ou wireless fidelity, por radiofrequência, Bluetooth, comunicação de campo próximo ou Near Field Communication (NFC), entre outros. Adicionalmente, esse sistema deve interagir com uma grande variedade de sensores existentes no dispositivo, como GPS, sensores de luminosidade, giroscópio, acelerômetros, tela de toque ou touch screen, leitor de digitais etc.

Observação

O giroscópio é um sensor que detecta a velocidade angular do dispositivo no qual está instalado. Com isso, é possível observar se o usuário gira o smartphone em torno do seu próprio eixo e se o telefone está sendo apontado para baixo ou para cima, por exemplo. Uma aplicação do giroscópio envolve jogos que exigem a movimentação de um personagem para realizar a leitura dos movimentos físicos que serão convertidos em dados para as aplicações.

O acelerômetro é um dispositivo eletromecânico usado para detectar a inclinação do aparelho e mede as forças de aceleração. Uma aplicação dos acelerômetros é a detecção de movimento baseada em eixos, técnica adotada para fazer a contagem de passos em aplicativos de atividade física.

Caso as aplicações fossem desenvolvidas sem a utilização de um SO, o tempo de programação para o desenvolvimento do código-fonte seria muito maior, pois o programador precisaria implementar as rotinas de E/S necessárias, que são previstas pelo SO. Com isso, a complexidade do software desenvolvido será maior. Adicionalmente, o usuário deverá estar preocupado com os detalhes de hardware, tais como as configurações técnicas de cada dispositivo.

O SO é considerado um software básico para o sistema computacional e é essencial, pois permite maior dedicação aos problemas de alto nível, isto é, não ligados ao hardware e voltados para as aplicações. Além disso, traz maior portabilidade entre dispositivos diferentes, permitindo que uma mesma aplicação funcione para diversas configurações de hardware.

Machado e Maia (2013, p. 3) definem um SO como: "[...] um conjunto de rotinas executado pelo processador, de forma semelhante aos programas dos usuários". A principal função é o controle da operação do computador, por meio de gerenciamento, utilização e compartilhamento dos vários hardwares existentes, tais como memória, dispositivos de E/S e processadores.

Uma caraterística que distingue um SO e outros programas aplicativos é a forma pela qual as rotinas são processadas ao longo do tempo. No SO, a forma de execução das rotinas é não linear, pois as rotinas são executadas de forma concorrente acionadas por eventos assíncronos, isto é, eventos que podem ocorrer em qualquer instante.

Com a utilização do SO, pode-se criar uma máquina mais flexível e fazer o uso eficiente e controlado dos componentes de hardware. Além disso, o SO permite a utilização compartilhada e protegida dos diversos componentes de hardware e software por múltiplos usuários.

A seguir, faremos uma apresentação geral de um SO para dispositivos móveis e as características dos principais SOs existentes no mercado.

2.1 Sistemas operacionais para dispositivos móveis

Os SOs podem ser classificados em diversos tipos: monotarefa, desktop, servidor, embarcado, tempo real. Pode-se incluir os SOs para dispositivos móveis como um tipo de SO. Tais sistemas recebem essa classificação por serem executados em dispositivos como smartphones, assistente digital pessoal (PDA) e tablets.

De acordo com Maziero (2019), um SO é classificado como embarcado ou embedded, em inglês, quando é construído para operar sobre um hardware com recursos limitados de processamento, armazenamento e energia. Os SOs móveis podem ser diferenciados de sistemas embarcados, pois estes são mais simples e desenvolvidos para um objetivo em particular. Os dispositivos móveis possibilitam novas formas de interação com o usuário, tais como teclados virtuais, sensores como acelerômetro e giroscópio, GPS ou widgets, interfaces gráficas com o objetivo de promover o acesso a recursos de ferramentas e apps e são exibidos nas telas iniciais do celular. Um widget bastante comum apresenta a temperatura atual onde está o usuário sem ele ter que buscar essa informação em sites.

Um SO móvel é responsável por identificar e definir recursos e funções do dispositivo móvel, incluindo teclados, sincronização de aplicativos, e-mail, botão giratório e mensagens de texto. Ele é semelhante a um SO padrão (como Windows, Linux e Mac), mas é relativamente simples e leve e gerencia sobretudo as variações sem fio de conexões locais e de banda larga, multimídia móvel e vários métodos de entrada.

De acordo com Moraes *et al.* (2022), na computação móvel, a mobilidade está relacionada ao uso de dispositivos portáteis capazes de operar um conjunto de funções de aplicação, tais como conexão, obtenção e envio de dados a outros usuários e utilização de aplicações e sistemas.

É possível classificar dispositivos móveis em:

- telefones celulares;
- smartphones;
- tablets;
- laptops;
- robôs.

2.1.1 Principais sistemas operacionais móveis

Os principais são: Android, desenvolvido pela Google, iOS, implementado pela Apple, Symbian, Windows Mobile da Microsoft e BlackBerry.

Em janeiro de 2023, o Android era usado por 71,8% dos smartphones e o iOS, por 27,6%, sendo as duas principais plataformas do mercado e os SOs a serem mais detalhados (MOBILE..., 2023a).

Saiba mais

Na referência indicada a seguir, é possível observar a participação de mercado ou market share dos sistemas operacionais para smartphones no mundo. Há um gráfico que ilustra a participação de mercado de sistemas operacionais móveis.

MOBILE operating system market share worldwide: Jan 2010 – Jan 2023. *Statcounter*, 2023. Disponível em: https://shre.ink/nYiy. Acesso em: 25 set. 2023.

Segundo Lee, Schneider e Schell (2005), a mobilidade pode ser definida como a capacidade de poder se deslocar ou ser deslocado de forma fácil. Em computação móvel, mobilidade se refere ao uso de dispositivos portáteis que oferecem a capacidade de realizar, com facilidade, um conjunto de funções de aplicação, como conexão, obtenção e fornecimento de dados a outros usuários e uso de aplicações e sistemas. Assim, a mobilidade possui a portabilidade, a usabilidade, a funcionalidade e a conectividade como suas principais características.

Em função da evolução dos smartphones, a capacidade de processamento, em especial a conectividade da maioria dos modelos, está próxima à de desktops e com a vantagem da mobilidade. Segundo a *33ª pesquisa de uso de TI no Brasil* (Meirelles, 2022), em junho de 2022 havia 242 milhões de smartphones em uso no Brasil; dividindo pela população brasileira, há 1,13 smartphone por habitante.

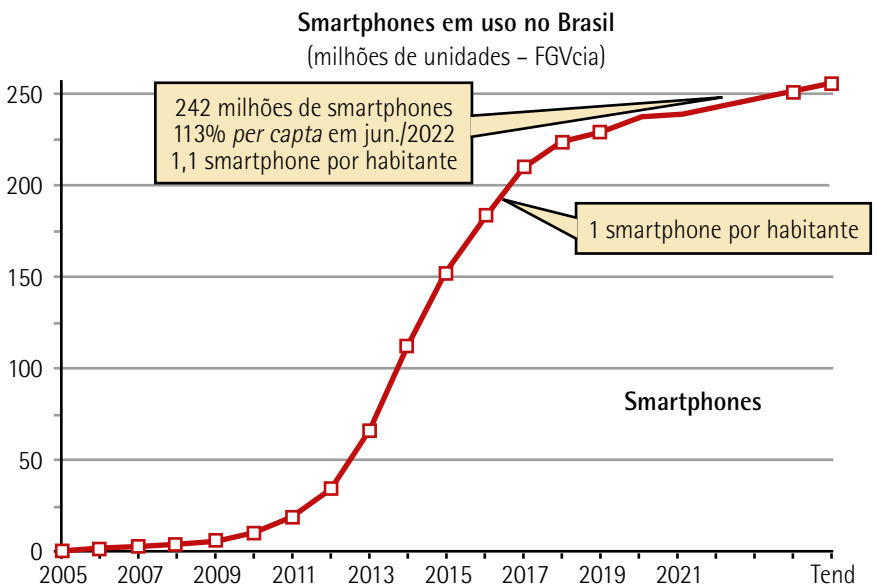

Figura 6 – Evolução da quantidade de smartphones no Brasil

Adaptada de: Meirelles (2022).

Analisando o SO que está instalado no dispositivo, o Android está presente em 81,1% dos smartphones no Brasil, o iOS em 18,6% e outros SOs representam 0,3% do total, de acordo com o site *Statcounter* (Mobile..., 2023b).

Saiba mais

Para observar o market share de sistemas mobile no Brasil, consulte:

MOBILE operating system market share worldwide: Jan 2015 – Feb 2023. *Statcounter*, 2023b. Disponível em: https://shre.ink/nYtZ. Acesso em: 25 set. 2023.

2.2 SO Android

Foi projetado pela Open Handset Alliance, um consórcio de empresas de tecnologia com o objetivo de popularizar e melhorar dispositivos móveis e serviços utilizando a plataforma Android. Esse consórcio é dirigido principalmente pela Google e é responsável por controlar importantes etapas do desenvolvimento do sistema.

Uma vantagem da utilização do Android é o fato de seu código-fonte ser aberto e gratuito. Com isso, é possível entender como recursos são implementados e até mesmo contribuir para o desenvolvimento do Android relatando erros (consulte http://source.android.com/source/report-bugs.html) ou participando nos grupos de discussão do Open Source Project (http://source.android.com/community/index.html).

Ter o código aberto proporciona uma vantagem ao desenvolvedor de aplicativos Android, que é a franqueza ou o grau de abertura da plataforma. Adicionalmente, outras empresas e fabricantes têm contribuído com a Google para desenvolver o código do sistema. Dessa forma, o Android pode tirar maior proveito de novas tecnologias e acessar recursos implementados no kernel do Linux.

Como adota uma variedade de fabricantes, o Android possui uma presença no mercado ainda mais rica em termos de hardware, pois é possível encontrar hardwares simples e com pequenas telas a hardwares de alta capacidade de processamento e com recursos inovadores. Como existem diferentes tipos de equipamentos com faixas de preço bem variadas, o Android é o SO móvel mais popular e mais utilizado.

2.2.1 Histórico do Android

Em 2003, foi fundada a empresa Android Inc. em Palo Alto, Califórnia, por Andy Rubin, Rich Miner, Nick Sears e Chris White. Inicialmente, ela desenvolveu um SO para câmeras digitais. Analisando o potencial de mercado para celulares, a meta da empresa foi desenvolver um dispositivo móvel mais inteligente e que considerasse a sua localização e as preferências do seu dono. Esses colaboradores trabalharam de maneira sigilosa e divulgaram publicamente apenas que eles desenvolviam um software para telefone móvel.

Em 2005, a Google adquiriu a Android Inc., tornando-a uma empresa pertencente ao grupo Google Inc.

Em outubro de 2008, ocorreu o lançamento do Android 1.0 para o público. Desde as primeiras versões do sistema, o kernel foi criado com base no Linux. O desenvolvimento em Linux implicou que o sistema fosse de código aberto e permitiu que o software pudesse ser alterado e personalizado em função das necessidades e dos desejos dos fabricantes de smartphones e inclusive da comunidade de usuário, desenvolvedores e programadores.

Em 2007, foi formada a Open Handset Alliance (OHA), uma aliança de empresa com o objetivo de criar padrões abertos para telefonia móvel. Nessa equipe, atuam operadoras de telefonia móvel, fabricantes de smartphones, fabricantes de circuitos integrados (CIs), empresas de software e de serviços. Das dezenas de entidades integrantes, pode-se destacar a Google, a fabricante de processadores Intel. A consultoria Accenture, a Samsung e a LG são exemplos de organizações que integram a OHA.

Saiba mais

A lista de todos os participantes da aliança pode ser consultada em:

Disponível em: https://shre.ink/nY2A. Acesso em: 25 set. 2023.

Com a junção de esforços dessas diversas empresas, foi possível desenvolver, manter e aprimorar o Android, alavancando a inovação para a tecnologia móvel, melhorando a experiência do usuário ou user experience (UX) e reduzindo os custos de desenvolvimento em relação aos esforços de uma empresa isolada.

O primeiro telefone celular com Android lançado comercialmente foi o T-Mobile G1 ou HTC Dream em 2008 nos EUA.

Até a versão 10 do Android, os nomes das versões eram de sobremesas em inglês e em ordem alfabética, o que pode ser observado no quadro a seguir:

Conforme a plataforma Android evoluiu e novas versões foram lançadas, cada versão do Android recebeu um identificador inteiro exclusivo, denominado nível de API. Assim, para cada versão do Android correspondente há um único nível de API do Android.

Tabela 1 – Versões do sistema Android

Número da versão	Nome da versão	Nível de API	Ano de lançamento
1.5	Cupcake	3	2009
1.6	Donut	4	2009
2.0	Eclair	5	2010

Número da versão	Nome da versão	Nível de API	Ano de lançamento
2.2	Froyo	8	2010
2.3	Gingerbread	9	2011
3.0	Honeycomb	11	2011
4.0	Ice Cream Sandwich	14	2011
4.1	Jelly Bean	16	2012
4.4	KitKat	19	2013
5.0	Lolipop	21	2015
6.0	Marshmallow	23	2015
7.0	Nougat	25	2016
8.0	Oreo	26	2017
9.0	Pie	28	2018

A partir da versão 10 do Android, foi abandonada a nomenclatura com nomes de doces organizados de forma alfabética.

No mundo real, os usuários instalam aplicativos em versões mais antigas e mais recentes do Android, assim os aplicativos Android devem ser projetados para trabalhar com vários níveis de API do Android.

2.2.2 Gestos

As telas sensíveis ao toque ou touchscreen dos smartphones possibilitam o controle o aparelho com gestos que envolvem toque rápido, arrastamento e pinça. O quadro a seguir apresenta uma lista com os principais gestos no Android.

Quadro 2 – Gestos comuns com Android

Nome do gesto	Ação física	Utilizado para
Toque rápido (touch)	Tocar rapidamente na tela de uma vez	Abrir um aplicativo, "pressionar" um botão ou um item de menu
Toque duplo rápido (double touch)	Tocar rapidamente na tela duas vezes	Ampliar e reduzir imagens, mapas do Google Maps e páginas web
Pressionamento longo (long press)	Tocar na tela e manter o dedo na posição	Selecionar itens em uma visualização – por exemplo, verificar um item em uma lista
Movimento rápido (swipe)	Tocar e mover rapidamente o dedo na tela na direção do movimento desejado	Mover item por item em uma série, como no caso de fotos. Um movimento do swipe para automaticamente no próximo item
Arrastamento (drag)	Tocar e arrastar o dedo pela tela	Mover objetos ou ícones, ou rolar precisamente uma página web ou lista
Zoom de pinça (pinch swipe)	Usando dois dedos, tocar na tela e juntá-los ou afastá-los	Ampliar e então reduzir a tela (por exemplo, ampliando texto e imagens)

Fonte: Deitel *et al.* (2015, p. 5).

Os smartphones Android englobam a funcionalidade de telefone celular, cliente de internet, MP3 player, console de jogos, câmera digital, entre outras, em um dispositivo móvel com telas multitouch coloridas. Nessas telas é possível realizar dois comandos simultaneamente, melhorando a experiência do usuário em jogos e aplicações gráficas.

Saiba mais

O código-fonte do Android está disponível para download no endereço indicado a seguir. Além do código-fonte, existem tutoriais para desenvolvedores, informações sobre segurança e arquitetura do sistema Android.

Disponível em: https://shre.ink/nYSj. Acesso em: 25 set. 2023.

2.2.3 Bugdroid

Para buscar uma experiência mais agradável ao usuário, a ideia de criar um mascote foi amadurecendo e a missão foi conferida a uma profissional da área. A ilustradora russa Irina Blok, também funcionária da Google, representou o pequeno robô de uma maneira mais agradável.

Figura 7 – Bugdroid

Disponível em: https://shre.ink/nYX1. Acesso em: 25 set. 2023.

2.3 Arquitetura do sistema Android

Trata-se de uma pilha de softwares baseados em Linux, isto é, um conjunto de componentes independentes que permitem a implementação do aplicativo de forma facilitada. Por ser escrito em Linux, o código aberto criado pode ser adaptado para diversos dispositivos e fatores de forma. A figura a seguir apresenta os principais componentes da plataforma Android.

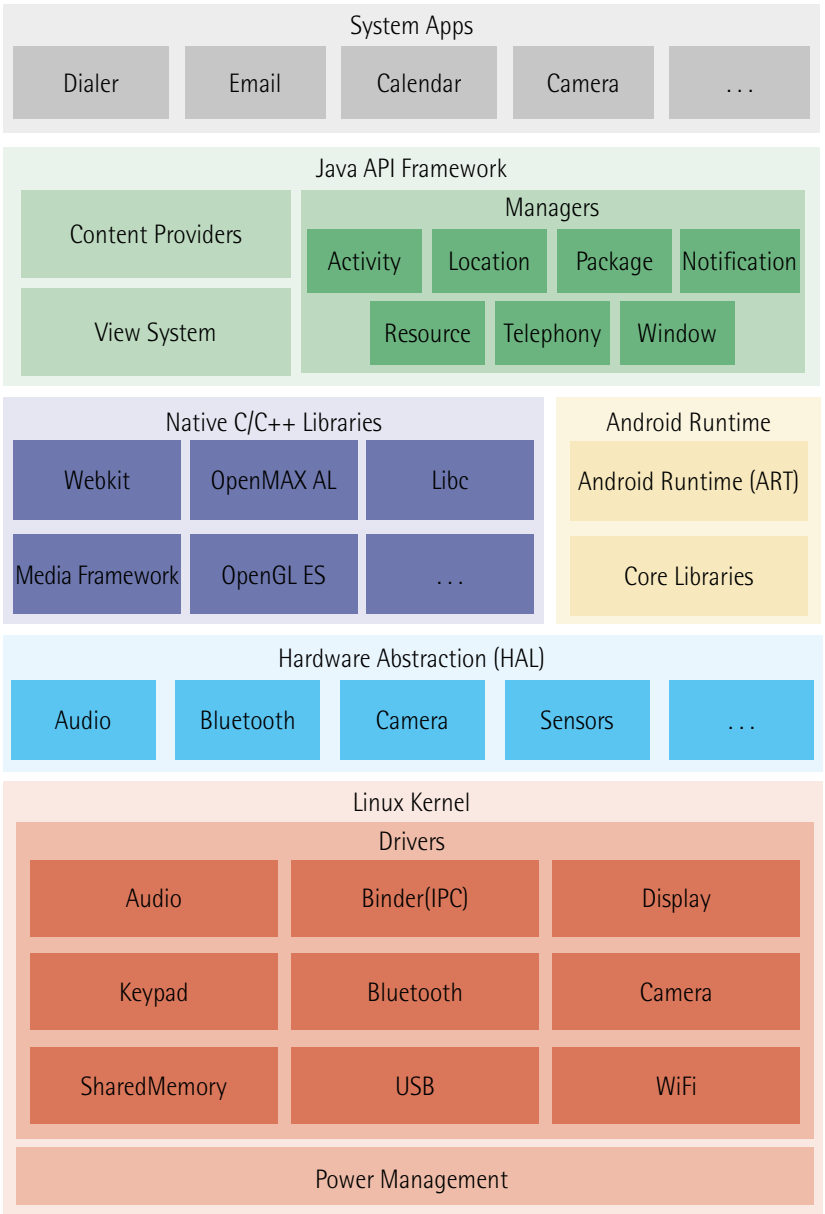

Figura 8 – Pilha de software do Android

Disponível em: https://shre.ink/nYfw. Acesso em: 25 set. 2023.

O Android é composto de uma pilha de software e utiliza o Linux de código aberto criado para diversos dispositivos e fatores de forma.

Também estarão inseridos no Android os drivers dos dispositivos, como câmera, áudio, tela, teclado, Bluetooth, comunicação via Wi-Fi, memória compartilhada, comunicação entre processos (IPC).

Ao utilizar o kernel do Linux, o Android teve a vantagem de usar os recursos de segurança já existentes no Linux, facilitando aos fabricantes de dispositivos desenvolverem drivers para o hardware de um kernel já consolidado.

Na camada de abstração de hardware ou hardware abstraction layer (HAL), são fornecidas as interfaces padrão que expõem as capacidades de hardware do dispositivo para a estrutura de API na linguagem Java de maior nível. A camada HAL é formada por módulos de biblioteca, que implementam uma interface para um tipo específico de componente de hardware, por exemplo, o módulo do Bluetooth ou da câmera. Quando uma framework API realiza uma chamada de sistema para acesso do hardware do dispositivo, o Android carrega o módulo da biblioteca para esse componente de hardware.

A partir da versão 5.0 do Android ou API nível 21, cada aplicativo executa o próprio processo com uma instância própria do Android Runtime (ART), o sistema de suporte de execução do Android. O ART permite que programas complexos sejam operados em aparelhos com configurações limitadas para memória, bateria e processamento. O ambiente de execução é composto da máquina virtual e das bibliotecas de programação para a máquina virtual.

O ART foi projetado para operar simultaneamente diversas máquinas virtuais em dispositivos de baixa memória executando arquivos DEX. Esse formato de arquivo foi desenvolvido especialmente para Android, promovendo o consumo mínimo de memória. A camada ART será detalhada mais adiante.

O Android também possui um conjunto das principais bibliotecas que permite o uso da maior parte das funções da linguagem de programação Java.

Por meio das bibliotecas nativas da linguagem C e C++, diversos componentes e serviços principais do Android, como ART e HAL, são implementados. A plataforma Android fornece as Java framework APIs, que possuem o objetivo de expor a funcionalidade de algumas dessas bibliotecas nativas aos aplicativos.

O conjunto completo de recursos do Android está disponível pelas APIs programadas na linguagem Java e são chamadas estruturas de API Java ou Java API framework. Elas compõem blocos de programação necessários para a criação dos aplicativos Android, simplificando a escrita de código com a reutilização de componentes e serviços de sistema modulares e principais, nos quais se destacam:

- Um sistema de visualização extensivo e útil para programar a interface do usuário de um aplicativo com diversos recursos gráficos, como botões, caixas de texto e grades.

- Um gerenciador de recursos, fornecendo acesso a recursos sem código, como strings localizadas, gráficos e arquivos de layout.

- Um gerenciador de notificação que permite que todos os aplicativos exibam alertas personalizados na barra de status.

- Um gerenciador de atividade que coordena o ciclo de vida dos aplicativos e fornece uma pilha de navegação inversa.

- Provedores de conteúdo que permitem que aplicativos acessem dados de outros aplicativos, como o aplicativo Contatos, ou compartilhem os próprios dados.

O Android já contém um conjunto de aplicativos principais instalados para e-mail, envio de SMS, calendários, navegador de internet, contatos que são conhecidos como aplicativos do sistema. Esses aplicativos inclusos na plataforma não têm status especial entre aqueles que o usuário opta por instalar. Portanto, um aplicativo terceirizado pode se tornar o navegador da web, o aplicativo de envio de SMS ou até mesmo o teclado padrão do usuário.

Os aplicativos do sistema funcionam como aplicativos para os usuários e fornecem capacidades principais que os desenvolvedores podem acessar pelos próprios aplicativos. Por exemplo: se o seu aplicativo quiser enviar uma mensagem SMS, não é necessário programar essa funcionalidade, é possível invocar o aplicativo de SMS que já está instalado para enviar uma mensagem ao destinatário que você especificar.

2.3.1 Interpretador *versus* compilador

Antes de explorar as máquinas virtuais do Android, serão apresentados os conceitos de interpretador e compilador.

Segundo Machado e Maia (2013), nos primeiros sistemas computacionais, era muito complexo para um programador realizar seu trabalho, pois deveria conhecer profundamente a arquitetura da máquina e programar em painéis por meio de cabos. Os programas eram desenvolvidos em linguagem de máquina e carregados diretamente na memória principal para execução. Naquela época, o programador tinha que determinar em qual região da memória o programa deveria ser carregado ou quais endereços de memória seriam reservados para as variáveis.

Posteriormente, surgiram as primeiras linguagens de montagem ou assembly e das linguagens de alto nível. De acordo com Silva (2017, p. 4), "uma linguagem de programação é um conjunto de notações formais para descrever ações ou operações a serem realizadas por um computador" e constituem ferramentas para o desenvolvimento de software.

Com isso, a principal preocupação do programador foi retirada de aspectos relacionados ao hardware, facilitando a construção de programas, documentação e manutenção.

Após escrever os programas nas linguagens de programação, isto é, escrever os programas-fonte, eles ainda não estão preparados para utilização. É necessária uma etapa de conversão, na qual toda representação simbólica das instruções da linguagem de programação é traduzida para o código de máquina, uma sequência de códigos binários e o insumo do hardware. O nome do arquivo gerado em linguagem de máquina é o programa objeto, e essa transformação pode ser realizada por diferentes formas.

Uma dessas ferramentas são os tradutores, que realizam a leitura de um programa-fonte em uma linguagem de programação de alto nível e a transformam para linguagem de máquina em um programa objeto.

Por sua vez, o interpretador é um dispositivo que realiza a interpretação, linha a linha, de cada comando e o executa. Não é feita a tradução de todo o programa para posterior execução e não é

gerado o programa objeto. Para executar um programa-fonte, primeiro é necessário interpretar. Caso exista algum erro nele, o interpretador deve acusá-lo imediatamente. Então, o programador pode editar o programa-fonte e corrigir o erro para depois interpretar novamente o comando que resultou no erro. São exemplos de linguagem interpretadas: Basic, Python e Java.

Um ponto negativo dessa ferramenta é que, se esse programa-fonte for executado posteriormente, ocorrerá uma nova tradução, comando por comando, pois os comandos em linguagem de máquina não ficam armazenados para futuras execuções. Isso representa um gasto maior na tradução dos códigos. Por outro lado, o interpretador possibilita a implementação de tipos de dados dinâmicos, isto é, que podem ser alterados durante a execução do programa, trazendo maior flexibilidade ao programa.

Prosseguindo, o compilador realiza a leitura e analisa todo o programa-fonte, que foi escrito em linguagem de alto nível, e realiza a tradução para a linguagem de máquina, com a criação de um programa objeto que corresponde às instruções em linguagem de máquina. O programa objeto é executado diretamente com a tradução de todo o código de uma única vez. Caso o compilador encontre algum erro, o programa precisará voltar ao programa-fonte, corrigir, recompilar e executar novamente o programa objeto. Se esse programa for executado uma segunda vez, não haverá necessidade de uma nova tradução, pois todos os comandos em linguagem binária foram memorizados em um novo programa completo. As linguagens Pascal, C e C++ são exemplos de linguagens compiladas.

A principal vantagem da compilação é a execução mais rápida em relação à interpretação, já que não se perde o tempo de tradução para cada execução. Entretanto, para cada modificação realizada no programa-fonte, é necessário fazer a tradução completar e gerar um programa objeto novo. Esse aspecto pode dificultar a fase de desenvolvimento do software, quando são realizadas inúmeras alterações.

O interpretador pode ser entendido como um tradutor que não gera módulo objeto. Com base no programa-fonte escrito em linguagem de alto nível, o interpretador, durante a execução do programa, faz a tradução de cada instrução separadamente e a executa imediatamente. Algumas linguagens tipicamente interpretadas são o Basic, o Perl e Python.

2.3.2 Máquina virtual Dalvik

Uma plataforma operacional representa a união entre hardware, SO e aplicações. Dessa forma, aplicações escritas para uma plataforma operacional não funcionam em outras. Para resolver essa questão de incompatibilidade, são utilizadas as máquinas virtuais.

A máquina virtual introduz uma camada intermediária entre hardware e o SO para compatibilizar diferentes plataformas. O processo para criação dessa camada adicional é denominado virtualização, como mostra a figura a seguir.

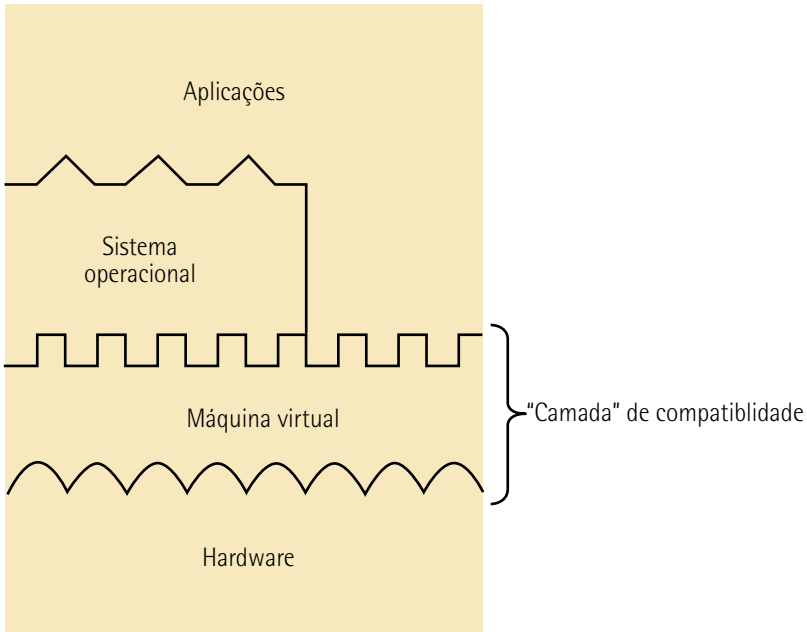

Figura 9 – Introdução da camada de virtualização

Fonte: Laureano (2006, p. 17).

Formalmente, uma máquina virtual ou virtual machine (VM) é definida como uma duplicata eficiente e isolada de uma máquina real (Popek; Goldberg, 1974). Por meio de máquinas virtuais, é possível executar um SO e de suas aplicações sobre outro SO, trabalhar com múltiplos sistemas operacionais e flexibilizar uma plataforma complexa de trabalho. Por exemplo, é possível testar o software em diferentes sistemas operacionais e em diversas versões sem a necessidade de hardware dedicado para o teste.

A máquina virtual Dalvik ou Dalvik virtual machine (DVM) é desenhada especificamente para o sistema Android e foi projetada por Dan Bonstein, com colaboração de engenheiros da Google. O nome homenageia a vila de pescadores Dalvik, na Islândia, onde antepassados de Dan viveram.

Em uma comparação de máquinas virtuais feita por Magenta e Nakamura (2014), uma máquina virtual Java é projetada para ser uma solução para diversos hardwares, sistemas e cenários, enquanto a Dalvik teve foco estritamente em dispositivos móveis. Por isso, a Dalvik considerou algumas limitações específicas do ambiente mobile que devem permanecer por um longo período, como o tempo de vida da bateria e dispositivos móveis cada vez menores. Como o hardware dos dispositivos Android geralmente é mais limitado em relação aos recursos disponíveis, a máquina virtual Java não atenderia às necessidades da plataforma, por isso foi preciso construir uma nova VM.

Essa máquina virtual requer pouca memória e foi projetada para permitir a execução de múltiplas VMs simultaneamente. Para endereçar a limitação da bateria, Dalvik foi projetada como uma máquina virtual baseada em registradores, que é apropriada para arquitetura de processadores ARM. A arquitetura ARM tende a operar com uma temperatura menor em relação à arquitetura Intel x86. Com isso, há um menor consumo da bateria. Por sua vez, a VM padrão Java tem uma estrutura com base em pilhas de dados.

Existem dois tipos de arquivos na máquina Dalvik: os de extensão .dex (Dalvik Executable file) e .odex (Optimized dex files). Os .dex contêm o código compilado da aplicação. Depois da compilação, são então zipados em um único arquivo .apk, isto é, um aplicativo Android.

Saiba mais

O site do projeto de código aberto Android detalha o layout dos arquivos .dex, que contém as definições das classes de objetos e dos dados associados a ela.

Disponível em: https://shre.ink/ngOj. Acesso em: 25 set. 2023.

Para reduzir o espaço necessário do Android e aumentar a velocidade de carregamento de um aplicativo Android, foram criados os arquivos .odex. Durante a instalação do aplicativo, o sistema aplica otimizações no arquivo .dex para acelerar sua execução.

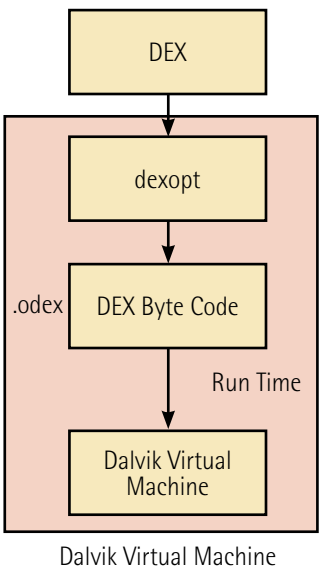

Figura 10 – Fluxo de arquivo na máquina virtual Dalvik

Entretanto, o aplicativo não é carregado de uma vez na memória para economizar espaço de armazenamento, uma restrição relevante, pois os primeiros smartphones tinham o tamanho da memória RAM inferior a 200 megabytes (Mb). Assim, em vez de compilar todo o código em linguagem de máquina antes de executar na máquina, ocorre a compilação Just in Time (JIT), em que o compilador atua também como um interpretador, pois há a complicação de pequenos pedaços de código em tempo de execução, gerando uma baixa utilização de memória.

Por outro lado, essa estratégia tem uma desvantagem: como a compilação ocorre em tempo de execução, há um impacto negativo no desempenho. Eventualmente, houve otimizações para reduzir esse impacto na máquina Dalvik.

Com a evolução dos dispositivos celulares e a expansão de memória, foi necessário desenvolver uma nova máquina virtual para melhorar o desempenho das aplicações.

A partir da versão 4.4 do Android, denominada KitKat, a máquina virtual Dalvik foi descontinuada no sistema Android e foi substituída pela ART, e a máquina ART é compatível com a Dalvik e executa os arquivos .dex. Dessa forma, os aplicativos desenvolvidos pela Dalvik podem ser executados nas ARTs. Entretanto, nem todos os aplicativos desenvolvidos pela ART vão funcionar na Dalvik.

2.3.3 Máquina virtual Android Runtime (ART)

Trata-se do tempo de execução utilizado pelas aplicações e alguns serviços de sistemas no Android. Na ART, os aplicativos são totalmente compilados quando são instalados no dispositivo. Com isso, há um desempenho melhor, já que não ocorre a conversão do código em bytecode para depois realizar a compilação. A estratégia é compilar o aplicativo antes da execução, denominada compilação antes do tempo ou ahead of time compilation (AOT). Houve uma melhoria de desempenho da ordem de dezenas de vezes em relação à compilação JIT.

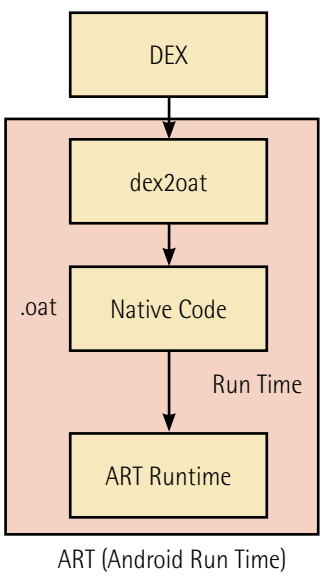

Figura 11 – Fluxo de arquivo na máquina virtual ART

Uma desvantagem da ART e da complicação AOT é a utilização de muito mais memória que Dalvik e JIT. Adicionalmente, há um tempo maior para instalação do aplicativo, que deve ser compilado inteiramente para completar a instalação. Outro inconveniente é o maior tempo necessário para a atualização do sistema, pois todos os aplicativos deveriam ser reotimizados e recompilados.

Para as partes de aplicativos que são frequentemente usadas, o ideal é que seu código já esteja pré-compilado. Contudo, a maioria das partes de um aplicativo é aberta muito raramente pelos usuários e são poucos os casos em que compensa ter o aplicativo totalmente pré-compilado.

Por exemplo, o TouchWiz, uma interface de toque para usuários desenvolvida pela empresa coreana Samsung em parceira com fornecedores, tem o intuito de apresentar uma interface de usuário completa. Diversos recursos do TouchWinz estão pré-compilados, portanto, eles são desindexados ou deodexed.

Assim, a partir a versão Android Nougat (7.0), a compilação JIT foi reintroduzida no tempo de execução, por meio da compilação guiada por perfil ou profile-guided compilation. Essa estratégia permite que o desempenho dos aplicativos Android seja melhorado conforme eles são executados. Por definição, o aplicativo é compilado usando a estratégia de compilação JIT. Todavia, quando a ART detecta que alguns métodos são muito usados pelo usuário, a ART pode pré-compilar e colocar esses métodos na memória para alcançar melhor desempenho.

Saiba mais

Para observar o funcionamento ART, acesse:

SADOWSKA, P. Android runtime – How Dalvik and ART work? *ProAndroidDev*, 2021. Disponível em: https://shre.ink/nYj0. Acesso em: 25 set. 2023.

Essa estratégia combina a redução de utilização da memória RAM com melhoria do desempenho durante a execução do aplicativo. Com essa mudança no tempo de execução, não há mais impactos no tempo de instalação dos aplicativos e na atualização do sistema. A pré-compilação das principais partes de um aplicativo ocorre apenas quando o dispositivo está em espera ou carregando para minimizar o impacto na bateria do dispositivo.

A desvantagem dessa estratégia é na obtenção dos dados de perfil e na pré-complicação dos métodos e classes que o usuário realmente aproveita. As primeiras execuções do aplicativo podem ser um pouco lentas, pois somente a compilação JIT está sendo utilizada. Visando aprimorar a experiência inicial do usuário, na versão Android Pie foi introduzido o perfil na nuvem.

A principal premissa do perfil na nuvem é que a maior parte das pessoas utilizam o aplicativo de forma bastante similar. Assim, para melhorar o desempenho após a instalação, é possível coletar dados de perfil de outros usuários do aplicativo. Esses dados de perfil agregado são utilizados para criar um arquivo denominado perfil principal comum para aplicação.

2.4 iOS

Em 29 de julho de 2007, a Apple lançou o primeiro iPhone nos EUA durante o evento MacWorld 2007, que foi apresentado ao mercado por Steve Jobs, CEO da empresa à época. Foi uma inovação revolucionária, pois o foco era a experiência do cliente, principalmente ao usar a tecnologia de múltiplos toques na tela ou multitouch.

Na origem do iPhone, ocorreu uma agregação de tecnologias, e muitas delas foram desenvolvidas incialmente para aplicações militares. A figura a seguir apresenta suas origens.

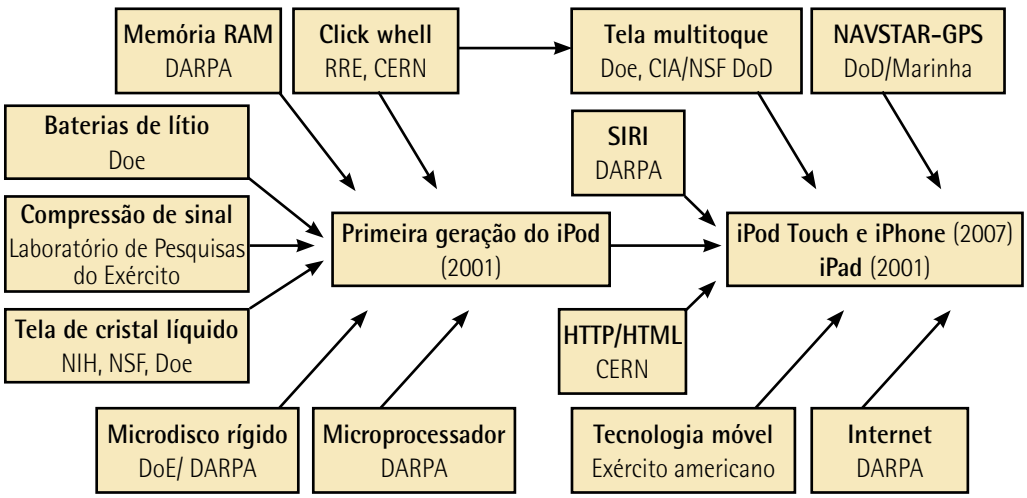

Figura 12 – Tecnologias que originaram o iPhone

Fonte: Mazzucato (2013, p. 153).

Para suportar essas funções, foi desenvolvido o iOS (iPhone Operating System), projetado como uma versão miniatura do MacOS. Como o MacOS era muito bem avaliado pelos seus usuários, o iOS manteve as qualidades em um dispositivo portátil. O iOS foi desenvolvido exclusivamente para dispositivos da Apple e nos dispositivos iPhone, iPod touch e iPad.

De acordo com Rocha e Finzi Neto (2011), a arquitetura do iOS é composta de quatro camadas, cada uma delas oferece um conjunto de frameworks que podem ser utilizados durante o desenvolvimento de aplicativos para os dispositivos móveis da Apple Inc. As camadas existentes são: CoreOS, Core Services, Media e Cocoa Touch.

A Cocoa Touch tem os principais frameworks utilizados para implementação de aplicações. Nessa camada são definidas a infraestrutura para tecnologia essenciais, como o serviço de notificação apple push, o gerenciamento de processos multitarefas e a interface com o usuário. Um recurso-chave dessa camada são as extensões de aplicativos ou app extensions, no qual um usuário pode compartilhar uma foto ou arquivo através dos app extensions sem necessidade de abrir o arquivo.

Por sua vez, na Media estão inseridas as tecnologias gráficas, de áudio, de vídeo e animações. O desenvolvedor de aplicativo pode utilizar o framework tecnologias gráficas ou graphics technologies para criar interfaces de usuário de alta qualidade. Um de seus recursos é o kit de interface do usuário ou user interface kit graphics (UIKit), que oferece diversas tecnologias de gráficos e animações. É considerada pela Apple uma das camadas mais importantes, pois é a filosofia da Apple a busca por oferecer a melhor experiência possível aos usuários em seus dispositivos.

Já a Core Services tem serviços essenciais do sistema que são usados por todos os aplicativos. Uma das tecnologias encontradas nela é denominada in-app dispatch e possibilita que os desenvolvedores vendam conteúdos e serviços dentro de suas aplicações. De acordo com o portal *DEVMEDIA* (Conheça..., 2015), um framework dessa camada é o Core Foundation Framework, que provê um conjunto de interfaces baseadas na linguagem C para gerenciamento de dados e serviços, por exemplo, coleções de dados, gerenciamento de cadeia de caracteres ou strings.

Por fim, a Core OS tem características de baixo nível, isto é, em linguagem de máquina. As estruturas criadas nessa camada permitem a implementação de outras tecnologias em camadas superiores. Além do kernel, nessa camada existem drivers e interfaces básicas do sistema, como formas de interagir com dispositivos externos por meio de bluetooth ou interfaces de gerenciamento de certificados, chaves públicas e privadas e gerenciamento de acesso. Ainda há o nível de sistema ou system, que engloba o kernel, os drivers e as interfaces de baixo nível Unix do SO. O kernel é responsável por threads, processos, arquivos de sistemas, rede e processo de comunicação. A biblioteca LibSystem tem recursos como concorrência, alocação de memória, recursos para cálculos matemáticos computacionais e concorrência entre processos.

Observação

Um framework é um conjunto de bibliotecas e estruturas para o desenvolvimento de aplicações. Dentro do framework existem recursos comuns que estão finalizados, validados e testados que podem ser reutilizados, permitindo maior eficiência para a resolução de problemas, otimização de recursos e detecção de erros.

De acordo com Anvaari e Jansen (2010), a arquitetura do iOS foi projetada com base na arquitetura básica encontrada no Mac OS X. Entretanto, existem recursos apenas disponíveis na plataforma de software do iPhone como a interface multitoque. Dessa forma, os usuários de desktops da Apple terão mais facilidade para manusear o smartphone.

A Apple disponibilizou APIs públicas para iPhone que permitem ao desenvolvedor integrar suas aplicações com o iPhone.

No entanto, o iOS e a loja de aplicativos App Store da Apple são licenciados e somente podem ser utilizados em produtos da empresa, o que limita o seu mercado aos produtos da marca.

Lembrete

Uma API é utilizada para enviar dados entre os aplicativos de software de maneira formalizada.

2.5 Outros sistemas operacionais móveis

Além do Android e do iOS, destaca-se o Symbian, que foi criado em 1998 pela Ericsson, Nokia, Motorola e Psion. Em 2008, a empresa norueguesa Nokia comprou as ações das outras partes e assumiu o controle total da Symbian. Até o fim da década de 2000, ele foi o sistema mais vendido no setor de smartphones não somente no Brasil, mas no mundo. Em 2013, a Nokia encerraria de vez a comercialização de telefones Symbian e, em 2014, todo o suporte de manutenção e desenvolvimento de software foi finalizado.

Outro SO móvel que surgiu foi o Windows Phone, lançado em 2010 com a proposta de ser um concorrente ao Android e iOS. A Microsoft realizou elevado investimento de dinheiro e tempo para desenvolvê-lo, apresentando versões com diversas melhorias. Depois, comprou a divisão mobile da Nokia.

2.6 Ecossistema de desenvolvimento de softwares móveis

Bosch e Bosch-Sijtsema (2010) definem um ecossistema de software como "um conjunto de soluções de software e uma comunidade especialista no domínio de serviços para a comunidade de usuários, que compõe soluções relevantes para as suas necessidades". Podemos apontar o Google, Android, Windows Kinect, SAP, Linux e iOS como exemplos de ecossistemas para diferentes domínios.

Analisando ecossistemas de software móveis, Android, iOS e Windows Phone concorrem para atrair desenvolvedores com o objetivo de aumentar a quantidade de aplicativos em disponíveis para seus dispositivos móveis.

Conforme Queiroz (2018), o modelo de loja de aplicativos da Apple, a App Store, promoveu a criação de uma indústria de desenvolvimento de softwares voltada para dispositivos móveis que não existia na época. Ela foi inspirada no iTunes, a loja de músicas para o MP3 player iPod. O iTunes é um reprodutor de áudio no qual o usuário poderia reproduzir e organizar música digital, arquivos de vídeo e permitia a compra de arquivos de mídia digital no formato gestão de direitos digitais. A loja ainda permitiu que qualquer desenvolvedor de aplicativos para dispositivos móveis os "inscrevesse" na loja da Apple. Após a aprovação da empresa, o app poderia ser baixado por qualquer usuário de iPhone.

Existem três elementos-chave em um ecossistema: um centralizador, uma plataforma e um conjunto de agentes de nicho. O centralizador ou keystone significa que o papel de governante de plataforma é da Apple. A plataforma central é o iOS, e os agentes de nicho, nesse contexto, são as operadoras de telefonia móvel, os desenvolvedores de aplicações, os usuários e os fabricantes de hardware, como apresentado na figura a seguir.

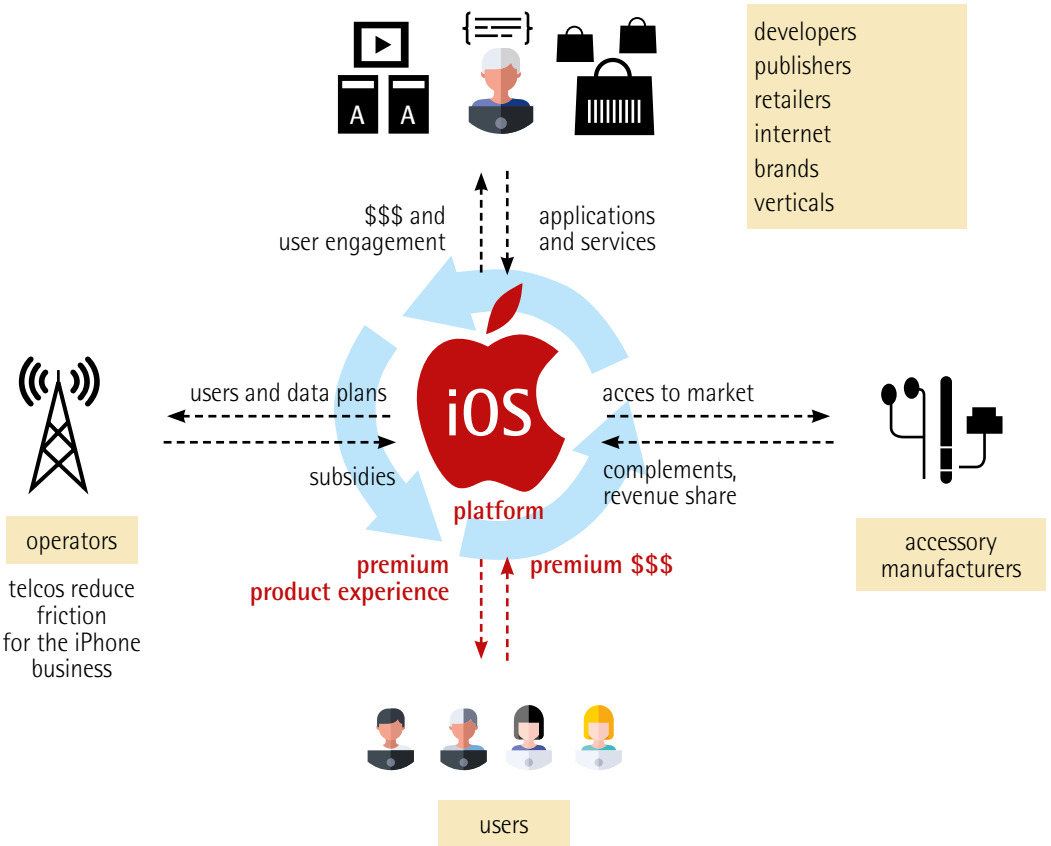

Figura 13 – Panorama dos atores e seus papéis no ecossistema de software móvel – caso iOS

Fonte: Miranda (2016, p. 29).

Foi a Apple que criou a plataforma iOS, o SO para iPhone. É por meio dela que os demais atores vão se relacionar. Pense em você com um usuário final da plataforma, isto é, uma das pessoas que utilizam o iPhone com seu iOS instalado. Considere também que usará outras aplicações que são criadas pelos desenvolvedores de software, como jogos, lojas de e-commerce, mídias sociais e mobile banking. Tais aplicações estão catalogadas em uma loja virtual, denominada App Store. Os usuários pagam para usufruir dessa plataforma, seja apenas comprando o smartphone, seja comprando produtos de software na loja virtual.

No caso dos desenvolvedores de software, eles são os responsáveis pela criação de novas aplicações e serviços para a plataforma, bem como por melhorias das aplicações existentes. O desenvolvedor pode disponibilizar sua aplicação na loja App Store de forma gratuita ou paga pelo usuário.

Os fabricantes de hardware fornecem componentes que serão acoplados ao smartphone da Apple e englobam fabricantes de processador utilizados diretamente na fabricação e montagem dos smartphones até empresas que produzem acessórios, como fones de ouvido, capas e películas protetoras.

Outro ator importante é a operadora de telefonia móvel. Dadas as capacidades de um smartphone, que vão muito além do que fazer e receber ligações, o usuário precisa contratar um plano de pacote de dados para acessar a internet. Com isso, as operadoras móveis obtêm uma nova fonte de receita, planos de dados feitos especialmente para smartphones.

Todos os atores estão interligados de alguma forma, caracterizando o cenário de ecossistema de software. No caso do ecossistema de software Apple, a iOS, compete diretamente com o ecossistema de software móvel da Google, o Android. Os concorrentes são responsáveis por retirar valor da keystone e suas parceiras.

Os programadores de software que pretendem desenvolver aplicativos para o Android podem fazer o download do Android software development kit (SDK) para uma versão específica. O SDK possui depurador, bibliotecas, emulador, documentação, código de exemplo e tutoriais. Visando ao desenvolvimento mais rápido, é possível usar ambientes de desenvolvimento gráfico integrado (IDEs), como o Eclipse, que permite criar aplicativos na linguagem Java.

Resumo

Nesta unidade, foram apresentados conceitos gerais que se aplicam a sistemas operacionais abertos, nos quais existe a liberdade de acesso e alteração do código-fonte, como no caso do Linux.

Posteriormente, foram estudados sistemas operacionais móveis, em especial, Android e iOS. Houve uma evolução expressiva da capacidade de tais sistemas operacionais desde o início dos anos 2000.

Com a inclusão de lojas de aplicativos, vimos que foi criado um ecossistema de desenvolvimento de software para dispositivos móveis, o que tem atraído milhares de profissionais da área.

Exercícios

Questão 1. (IBFC 2023, adaptada) Em um computador, o principal programa de sistema é o SO, que é responsável por arbitrar o acesso aos recursos de hardware disponíveis, como memória, processador e dispositivos de E/S. Em relação aos principais SOs, avalie as afirmativas a seguir.

I – O Linux é um software que opera em vários tipos de dispositivos.

II – O Android é um SO desenvolvido para smartphones e tablets.

III – O Windows é um SO desenvolvido pela Microsoft, cujo código-fonte não é aberto.

É correto o que se afirma em:

A) I, apenas.

B) III, apenas.

C) I e III, apenas.

D) II e III, apenas.

E) I, II e III.

Resposta correta: alternativa E.

Análise das afirmativas

I – Afirmativa correta.

Justificativa: Linux é um software (mais especificamente, um SO) que opera em diversos tipos de dispositivos, como servidores, roteadores, sistemas embarcados e computadores pessoais, entre outros.

II – Afirmativa correta.

Justificativa: Android é um SO desenhado especificamente para smartphones e tablets, sendo, desse modo, classificado como um SO móvel. Ele é baseado no núcleo Linux.

III – Afirmativa correta.

Justificativa: Windows é um SO desenvolvido pela Microsoft que é distribuído como software proprietário, ou seja, seu código-fonte não é disponibilizado publicamente nem é livremente modificável por terceiros.

Questão 2. (UFMT 2015, adaptada) O Android é um sistema operacional mobile de código aberto muito popular. Em relação às suas características, avalie as afirmativas a seguir.

I – Para que uma aplicação seja instalada, o usuário precisa aprovar as permissões solicitadas.

II – O Android já usou um tipo de máquina virtual Java denominada Dalvik.

III – O kernel do Android é baseado no kernel da família do sistema operacional BSD.

É correto o que se afirma em:

A) I, apenas.

B) III, apenas.

C) I e II, apenas.

D) II e III, apenas.

E) I, II e III.

Resposta correta: alternativa C.

Análise das afirmativas

I – Afirmativa correta.

Justificativa: no Android, as permissões são solicitadas ao usuário durante a instalação de um aplicativo. O usuário deve aprovar essas permissões antes que o aplicativo seja instalado.

II – Afirmativa correta.

Justificativa: o Android, originalmente, utilizava o Dalvik virtual machine (DVM) para executar aplicativos escritos em Java. A partir da versão 4.4 do Android, denominada KitKat, a Dalvik foi descontinuada no sistema Android e foi substituída pela Android Runtime (ART), e a máquina ART é compatível com a Dalvik. Dessa forma, os aplicativos desenvolvidos para Dalvik podem ser executados nas ART.

III – Afirmativa incorreta.

Justificativa: o kernel (núcleo) do Android é baseado no kernel do Linux, não no núcleo da família de sistemas operacionais Berkeley Software Distribution (BSD).