Descobrindo o Linux

Entenda o sistema operacional GNU/Linux

3ª Edição – Revisada e Ampliada

João Eriberto Mota Filho

Novatec

Copyright © 2006, 2010, 2012 da Novatec Editora Ltda.

Todos os direitos reservados e protegidos pela Lei 9.610 de 19/02/1998. É proibida a reprodução desta obra, mesmo parcial, por qualquer processo, sem prévia autorização, por escrito, do autor e da Editora.

Editor: Rubens Prates
Revisão gramatical: Denise Agostinetti
Editoração eletrônica: Carolina Kuwabata
Ilustração da capa: Renato Palet

Histórico de impressões:

Julho/2012	Terceira edição (ISBN: 978-85-7522-278-2)
Março/2007	Segunda edição (ISBN: 978-85-7522-120-4)
Abril/2006	Primeira edição (ISBN: 85-7522-090-X)

Novatec Editora Ltda.
Rua Luís Antônio dos Santos 110
02460-000 – São Paulo, SP – Brasil
Tel.: +55 11 2959-6529
Fax: +55 11 2950-8869
E-mail: novatec@novatec.com.br
Site: www.novatec.com.br
Twitter: twitter.com/novateceditora
Facebook: facebook.com/novatec
LinkedIn: linkedin.com/in/novatec

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Mota Filho, João Eriberto
Descobrindo o Linux : entenda o sistema operacional GNU/Linux / João Eriberto Mota Filho. -- 3. ed. rev. e ampl. -- São Paulo : Novatec Editora, 2012.

Bibliografia.
ISBN 978-85-7522-278-2

1. LINUX (Sistema operacional de computador) 2. Redes de computadores 3. UNIX (Sistema operacional de computador) I. Título.

12-06432 CDD-005.43

Índices para catálogo sistemático:

1. LINUX : Sistema operacional : Computadores : Processamento de dados 005.43

CAPÍTULO 1

História do GNU/Linux

Este é um capítulo totalmente voltado para a história do sistema operacional GNU/Linux. Nele será possível entender, com detalhes, como se deram os acontecimentos que contribuíram para a criação do mais famoso sistema operacional da atualidade.

1.1 Considerações iniciais

Todo computador precisa de um sistema operacional para funcionar. O sistema operacional é responsável por controlar a utilização dos recursos fornecidos pela máquina, como processador, memória e discos. Para entender a história do sistema operacional GNU/Linux, será necessário conhecer vários fatos anteriores à sua criação. Tais fatos serão explorados a partir de agora.

1.2 Antecedentes

Vários antecedentes históricos foram de vital importância para a criação do Kernel Linux. Os fatos básicos foram a invenção do telégrafo e do telefone. Depois disso, outros acontecimentos propiciaram as condições ideais para a concepção do sistema operacional GNU/Linux.

1.2.1 As comunicações com o telégrafo

Em 1835, Samuel Finley Breeze Morse (Figura 1.1), professor de artes plásticas e desenho da Universidade de Nova York, provou que sinais poderiam ser transmitidos por fios.

Para demonstrar sua teoria, utilizou pulsos elétricos gerados por um eletroímã. Depois, estabeleceu um código constituído de pulsos curtos, pulsos longos e períodos de silêncio. Nascia o Código Morse. A primeira mensagem entre pontos distantes (Washington e Baltimore) foi transmitida em 24 de maio de 1844 e dizia: "What hath God wrought?". A mensagem original poderá ser vista em `http://www.memory.loc.gov/mss/mcc/019/0001.gif`.

O telégrafo foi o primeiro meio elétrico de transmissão de mensagens a longa distância. Até hoje é utilizado por algumas pessoas e organizações como forma barata e rápida de transmissão. O Código Morse é amplamente utilizado no mundo, não

só com o telégrafo. Navios, por exemplo, utilizam piscadas de luz a partir de potentes holofotes para comunicarem-se entre si.

Figura 1.1 – Samuel F. B. Morse.
Fonte: The Samuel Morse Historic Site (http://www.morsehistoricsite.org).

1.2.2 Invenção do telefone

A ambição do homem sempre o levou a conquistar grandes sonhos. O telégrafo foi uma revolução nas comunicações mundiais. No entanto, ainda era pouco. A transmissão da voz humana a grandes distâncias era um desejo cada vez mais próximo.

Em 1876, Alexander Graham Bell (Figura 1.2a) ganhou a corrida pela invenção do telefone (Figuras 1.2b e 1.2c), vencendo o seu rival Elisha Gray.

Figura 1.2 – (a) Alexander Graham Bell demonstrando o telefone; (b) Primeiro telefone experimental utilizado por Graham Bell; (c) Telefone comercial, de 1877, elaborado por Graham Bell.
Fontes: American Memory (http://memory.loc.gov/ammem/bellhtml/004046.html); Smithsonian Institution (http://photo2.si.edu/infoage/infoage.html).

Pouco antes da invenção do telefone, em 1869, Elisha Gray (Figura 1.3) havia se juntado a Enos Barton, fundando a Gray and Barton, uma pequena empresa localizada em Cleveland, Ohio.

Figura 1.3 – Elisha Gray.
Fonte: 120 years of electronic music (http://www.obsolete.com/120_years/machines/telegraph).

Três anos depois, a Gray and Barton teve seu nome mudado para Western Electric Manufaturing Company.

1.2.3 AT&T

A tentativa da invenção do telefone por Graham Bell era um projeto ousado e que necessitava de um suporte financeiro. Assim, em 1875, pouco antes do surgimento do telefone, nasceu a American Telephone and Telegraph Corporation, mais conhecida como AT&T Corp. A AT&T Corp. foi formada a partir de um acordo de Alexander Graham Bell com Gardiner Hubbard e Thomas Sanders. Os dois últimos financiaram a criação da AT&T, que seria uma empresa de suporte ao projeto do telefone. A Figura 1.4 mostra uma propaganda utilizada na época.

Figura 1.4 – Propaganda da época.
Fonte: AT&T (http://www.att.com/history).

O telefone de Graham Bell foi inventado em 14 de fevereiro de 1876 e registrado em 03 de março de 1876, sob a patente número 174.465.

Com o invento do telefone, iniciou-se o projeto que possibilitaria as operações de telefonia nos EUA. Finalmente, em 1877, os três homens que criaram a AT&T fundaram a Bell Telephone Company, a primeira empresa de telefonia do mundo. Em 1878, em New Haven, cidade americana localizada no estado americano de Connecticut, foi inaugurada a primeira estação telefônica.

Em três anos, já existiam centrais telefônicas em várias cidades e vilarejos dos EUA. A Bell Telephone Company tornou-se, nesse espaço de tempo, a American Bell Telephone Company. A Figura 1.5 mostra o interior de uma estação telefônica da época.

Em 1880, a Western Electric Company, de Elisha Gray e Enos Barton, era a maior fábrica de produtos elétricos nos Estados Unidos da América. Produzia equipamentos diversos, como máquinas de escrever e telégrafos.

Figura 1.5 – Ambiente de trabalho em uma estação telefônica da época.
Fonte: AT&T – A brief history (`http://www.att.com/history/history1.html`).

Em 1882, a empresa American Bell Telephone Company (ou simplesmente American Bell, como era mais conhecida na época) conseguiu tornar-se acionista majoritária da Western Electric, passando a controlá-la. A partir daí, a Western Electric tornou-se uma unidade de produção da American Bell. Essa junção ficou conhecida como Bell System.

Em 03 de março de 1885, a AT&T foi totalmente incorporada à American Bell, como subsidiária desta, a fim de garantir a construção e a operação de redes telefônicas de longa distância.

Em 30 de dezembro de 1899, a AT&T tomou posse da American Bell e tornou-se a companhia mãe do Bell System.

O Bell System estava tão bem estruturado que possibilitou a invenção dos amplificadores elétricos pela AT&T. Com isso, em 1913, foram possíveis as ligações telefônicas em todo o continente.

Até 1894, por um problema de patente, apenas a American Bell podia operar a telefonia nos EUA. No entanto, resolvida essa questão, entre 1894 e 1904, mais de seis mil empresas de telefonia independentes passaram a operar nos EUA. Em 1904 já eram 3.317.000 aparelhos operando. Logo surgiram problemas diversos, principalmente de interconexão entre as empresas. Tais problemas foram resolvidos em 1913.

O Bell System progredia cada vez mais. Em 1914, a Western Electric Company tinha filiais em Londres, Antuérpia, Berlim, Milão, Paris, Viena, São Petersburgo, Budapeste, Tóquio, Montreal, Buenos Aires e Sidnei. Assim, estabeleceu-se a International Western Electric Company.

Em 1925, Walter Gifford, o novo presidente da AT&T, resolveu que a AT&T e o Bell System deveriam ser a base de uma "telefonia universal" nos EUA (entenda-se por telefonia universal a tentativa de estabelecer um padrão dentro do país naquela época). Assim sendo, Walter determinou a venda da International Western Electric Company para a recém-criada International Telephone and Telegraph Company (ITT). A transação se deu por US$ 33 milhões.

Em 1927 a AT&T inaugurou um serviço telefônico que atravessava o Atlântico, chegando a Londres. Para isso, foi utilizado um enlace rádio de dupla via. Inicialmente, cada minuto custava US$ 25. Tal serviço se estendeu, chegando ao Havaí em 1931 e a Tóquio em 1934. No entanto, o enlace rádio era ineficiente, pois sofria interferências e degradação de sinal, além da baixa capacidade de transmissão. Em virtude disso, em 1956, foi lançado o cabo submarino TAT-1, interligando os EUA à Europa, através do oceano Atlântico. O cabo que atravessava o oceano Pacífico foi lançado em 1964. A Figura 1.6 mostra o lançamento de um cabo submarino.

Figura 1.6 – Lançamento de cabo submarino.
Fonte: AT&T – A brief history (`http://www.att.com/history/history2.html`*).*

Os anos se passaram e a AT&T cresceu cada vez mais. A comunicação via satélite e a tecnologia celular foram algumas das conquistas. Atualmente, a AT&T é uma das maiores empresas de telefonia e materiais para redes de computadores do mundo. O seu site oficial é `http://www.att.com`.

1.2.4 Laboratórios Bell

Em 1907, Theodore Newton Vail (Figura 1.7), presidente da AT&T na época, uniu os departamentos de engenharia da AT&T (antiga American Bell) e da Western Electric em uma única organização. Essa união passou a chamar-se, em 1925, Bell Telephone Laboratories ou, simplesmente, Bell Labs. Ressalta-se que o Bell Labs era um braço tecnológico da AT&T.

Figura 1.7 – Theodore Vail.

Fonte: PBS (`http://www.pbs.org/transistor/album1/addlbios/vail.html`).

O Bell Telephone Laboratories (Bell Labs) tornou-se próspero e foi responsável por diversas invenções. Criou, por exemplo, o primeiro sistema comercial de adição de áudio a filmes. Em abril de 1927, realizou a primeira transmissão de televisão, que se deu entre Washington e Nova York. Em 1947 inventou o transistor e, em 1958, o raio laser. Com o tempo, outras importantes invenções vieram como, por exemplo, o LED e a tecnologia de celulares, ambos em 1962.

1.2.5 Lei antitruste americana

Em 1949, uma lei antitruste foi estabelecida nos EUA. Em 1956, a AT&T foi obrigada, pelo Departamento de Justiça dos EUA, a assinar um acordo antitruste. Tal acordo impôs limitações à Western Electric quanto à fabricação de equipamentos para o Bell System e à contratação dos seus serviços pelo governo. É evidente que o Bell Labs, como parte da AT&T, seria igualmente afetado pela lei antitruste.

Em 1974, a AT&T e o Departamento de Justiça discutiam um novo acordo antitruste, que permitiria várias mudanças em relação ao acordo de 1956. Paralelamente, como parte das imposições, a Western Electric foi absorvida por uma variante da AT&T que surgira na época: a AT&T Technologies. A AT&T Technologies tinha um outro foco de mercado, fabricando e vendendo produtos para o consumidor, além de sistemas de rede e tecnologia da informação. Em 1º de janeiro de 1984 a AT&T concordou em abrir mão de diversas companhias telefônicas locais. Essa atitude a livrou de outras restrições impostas ainda em 1956.

Na década de 1980, a AT&T continuou buscando a expansão global por intermédio de associações com outras empresas, o que deu origem a:

- AT&T Network Systems International;
- Goldstar Semiconductor;
- AT&T Taiwan;
- AT&T Microeletrônica (Espanha);
- Lycom;
- AT&T Ricoh;
- AT&T Network Systems (Espanha).

Em 1996, a AT&T foi dividida em três empresas, a saber:

- AT&T (comunicações e serviços);
- Lucent Technologies (sistemas e tecnologias); e
- NCR (computadores).

Como consequência imediata, o Bell Labs foi dividido em duas partes. A primeira foi agregada à AT&T e, a segunda, à Lucent Technologies. Com isso, o nome Bell Labs foi extinto.

Desde 2006, a Lucent Technologies fundiu-se com a empresa Alcatel, dando origem à Alcatel-Lucent.

1.2.6 MIT (Massachusetts Institute of Technology)

O MIT, Massachusetts Institute of Technology, foi fundado em 1861 por William Barton Rogers (Figura 1.8).

Figura 1.8 – William B. Rogers.

Fonte: MIT (http://libraries.mit.edu/archives/mithistory/biographies/rogers.html).

Não obstante à data de sua fundação, os primeiros alunos só foram admitidos em 1865, em virtude da Guerra Civil Americana. William Barton Rogers foi presidente do MIT de 1862 a 1870 e, depois, de 1879 a 1881.

O MIT (Figura 1.9) é um dos mais renomados estabelecimentos de ensino superior do mundo. Atualmente, oferece cursos nas áreas de ciência e tecnologia para um pouco mais de 10.000 alunos (graduação e pós-graduação). Está sediado na Cidade de Cambridge, Estado de Massachusetts, nos EUA.

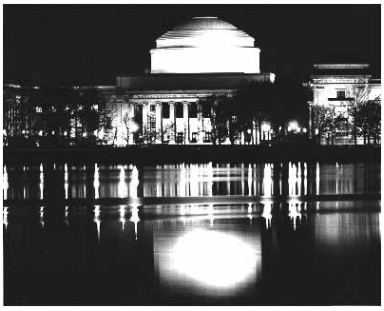

Figura 1.9 – Uma das fachadas do MIT.
Fonte: Wikipédia (`http://en.wikipedia.org/wiki/Image:Mitgreatdome.jpg`).

1.2.7 Criação do sistema operacional CTSS

O CTSS (Compatible Time-Sharing System) foi um dos primeiros sistemas operacionais a adotar a técnica de time sharing. Essa técnica, empregada até hoje, permite que vários usuários possam, simultaneamente, utilizar um ambiente para executar programas. Tudo isso ocorre sobre o mesmo sistema operacional, rodando em uma máquina. Esse tipo de sistema caracteriza o processo de compartilhamento de processador, memória e disco entre vários utilizadores. O conceito era simples. Fatias de tempo do processador, conhecidas como time slice, seriam destinadas aos programas carregados em memória. Assim, cada programa receberia a atenção individual da máquina por algumas frações de segundo, o que nos dá, até hoje, a ideia de que tudo está funcionando ao mesmo tempo.

O CTSS foi desenvolvido no Centro de Computação do MIT, por Fernando Jose Corbató. A primeira demonstração do CTSS ocorreu em 1961, rodando em um IBM 709, que pode ser visto na Figura 1.10.

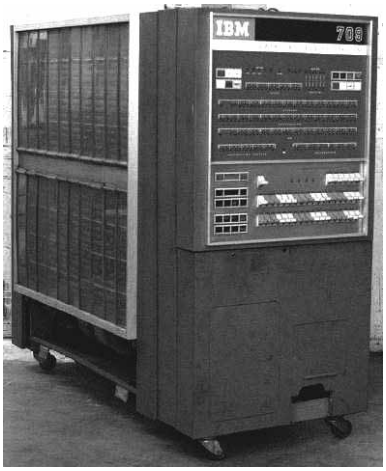

Figura 1.10 – IBM 709.
Fonte: Mark Bartelt (http://www.cacr.caltech.edu/~mark/IBM709.html).

Em novembro de 1962, o Centro de Computação do MIT passou a utilizar o IBM 7090 (Figura 1.11). Com isso, o CTSS foi portado para um novo hardware.

Figura 1.11 – IBM 7090.
Fonte: Stories of the Development of Large Scale Scientific Computing (http://www.computer-history.info/Page4.dir/pages/ibm_7090.html).

1.2.8 Projeto MAC (MIT Project MAC)

Ainda em novembro de 1962, Joseph Carl Robnett Licklider, integrante do MIT, propôs o Projeto MAC. Tal projeto foi aceito e o seu gerente passou a ser Robert M. Fano, professor do MIT.

O Projeto MAC foi criado para desenvolver dois produtos finais: um sistema operacional avançado e um laboratório de inteligência artificial. Em virtude disso, a sigla do projeto foi tratada com dois nomes diferentes: Multiple Access Computers e Man And Computers.

Em 1963, um estudo de verão sobre o Projeto MAC reuniu vários cientistas da computação em Cambridge. Os objetivos eram divulgar o CTSS e discutir o futuro da computação.

O subprojeto Multiple Access Computers tentaria desenvolver o sistema operacional Multics (MULTiplexed Information and Computing Service). O Multics deveria ser algo superior ao CTSS.

1.2.9 Sistema operacional Multics

O Projeto MAC começou a tomar proporções e teve apoio da Advanced Research Projects Agency (ARPA, agência subordinada ao Departamento de Defesa dos EUA), que disponibilizou dois milhões de dólares por ano, por oito anos, voltados exclusivamente para o desenvolvimento do Multics. Nesse mesmo período, o Bell Labs e a GE (General Electric), interessados no projeto, contribuíram com recursos de ordem semelhante ao ofertado pelo ARPA.

O objetivo final, em relação ao Multics, era um sistema operacional com suporte para memória virtual, utilizando recursos de paginação e segmentação de memória. Isso possibilitaria um sofisticado processo de transferência de dados entre discos e memória.

Ainda em 1963, as especificações de hardware para rodar o Multics foram enviadas para orçamento a algumas empresas. No momento de selecionar o fornecedor da máquina que seria utilizada com o Multics, a IBM ofereceu o IBM 360, lançado naquele ano. É interessante ressaltar que essa máquina não atendia às especificações do Multics. Notou-se, claramente, que a IBM não estava interessada nas ideias de paginação e segmentação desenvolvidas pela equipe do Projeto MAC. Nesse momento, Joseph Weizenbaum, professor do MIT, entrou para a equipe MAC com o objetivo de formar uma associação com a GE de Schenectady (cidade situada no Estado de New York, nos EUA, onde se encontra um dos laboratórios de pesquisas da General Electric), que estava receptiva a novas ideias. Assim, a GE propôs o mainframe GE-645. A DEC (Digital Equipment Corporation) também fez uma proposta. No fim, a proposta da GE foi escolhida e o contrato foi assinado em agosto de 1964.

O Bell Labs decidiu comprar um GE-645 no início de 1965 e juntou-se à equipe de desenvolvimento do Multics, no MIT. Logo depois, a GE também decidiu contribuir com o desenvolvimento.

A descrição do Multics foi apresentada em uma sessão especial na Fall Joint Computer Conference, em 1965. Nessa ocasião, algumas pessoas disseram que os objetivos da equipe de desenvolvimento eram muito ambiciosos. Muitos chegaram a citar que, à época, seria impossível fazer o Multics.

A linguagem PL/I foi escolhida para gerar o código do Multics. Com isso, iniciou-se efetivamente o desenvolvimento do Multics. O CTSS foi usado como sistema operacional para o trabalho dos desenvolvedores. Com o tempo, o próprio Multics foi utilizado para

o seu próprio desenvolvimento. No entanto, não foram obtidos resultados rápidos. Assim, diante de uma frustração inicial da equipe de desenvolvimento, o Bell Labs decidiu retirar-se do projeto em abril de 1969.

Em outubro de 1969 o Multics foi disponibilizado para a comercialização. Várias organizações importantes como, por exemplo, a Força Aérea Americana, a General Motors e a Ford utilizaram o Multics.

O desenvolvimento do Multics foi cancelado em julho de 1985. Depois disso, várias organizações começaram a suspender o uso do Multics. Há notícias de que o último Multics em produção foi desativado em outubro de 2000, no Quartel General do Comando Marítimo Canadense.

1.2.10 Sistema operacional Unix

Quando o Bell Labs resolveu integrar a equipe de desenvolvimento do Multics, emprestou alguns dos melhores programadores do mundo para o MIT. Pode-se citar, especialmente, Ken Thompson (cujo verdadeiro nome, quase nunca utilizado, é Kenneth Thompson), que escreveu um espetacular editor para o CTSS. Esse editor chamava-se QED e possuía recursos de busca e substituição utilizando expressões regulares. Tempos depois, o QED foi portado para o Multics por Ken Thompson e Dennis Ritchie (ambos vistos nas Figuras 1.12a e 1.12b, respectivamente).

Figura 1.12 – Os pais do Unix. (a) Ken Thompson; (b) Dennis Ritchie.
Fonte: Bell Labs (http://www.bell-labs.com/history/unix/thompsonbio.html e http://www.bell-labs.com/history/unix/ritchiebio.html).

Como já foi dito antes, em abril de 1969 o Bell Labs se desvinculou da equipe de desenvolvimento do Multics. No entanto, alguns integrantes do projeto Multics mantiveram um contato pessoal com profissionais do Bell Labs. Assim, motivados pelo Multics, Ken Thompson e Dennis Ritchie desenvolveram um projeto pessoal denominado Unics, que significava UNiplexed Information and Computing Service.

O projeto do Multics foi uma época de retrocesso para Ken Thompson e Dennis Ritchie. Muito se queria fazer; pouco foi feito. No entanto, eles sonhavam com a ideia de um sistema operacional avançado, utilizando time sharing. Assim, o Multics foi retrocesso e inspiração ao mesmo tempo.

O Unics era uma tentativa de fazer, com rapidez, um sistema operacional simples, versátil e moderno, mantendo-se as ideias de time sharing e de portabilidade entre computadores de todos os tamanhos. O nome surgiu como um trocadilho em relação ao Multics, uma vez que o Unics seria um Multics modesto. Algum tempo depois, em 1970, o nome foi mudado de Unics para Unix, conforme sugestão de Brian Kernighan.

Um esforço inicial se deu na tentativa de convencer o Bell Labs a adquirir um computador de médio porte. Thompson e Ritchie prometiam um sistema operacional em troca. Eles sugeriram as máquinas PDP-10 (da DEC) e Sigma 7 (da SDS – Scientific Data Systems, empresa criada em 1961, por Max Palevsky e que foi vendida para a Xerox Corporation em 1969). No entanto, a proposta da dupla de programadores foi rejeitada. Assim, eles se reuniram novamente e traçaram um projeto detalhado, que teria como produto final o sistema operacional e como necessidade um pequeno investimento financeiro. A máquina a ser utilizada poderia ser alugada ao invés de comprada. Mas, novamente, a proposta voltou a ser recusada. Eram claros os indícios de que o Bell Labs não estava interessado nesse tipo de atividade.

Notando a falta de incentivo, Thompson, Ritchie e Rudd Canaday, outro integrante do Bell Labs, começaram a desenvolver o projeto no papel e em quadro negro. Traçaram o desenho básico do sistema operacional, toda a teoria sobre o filesystem (necessário para que os sistemas operacionais possam utilizar corretamente os discos) e, depois, sobre o kernel (o coração de um sistema operacional, que faz a intermediação entre os processos e o hardware). Thompson, buscando uma máquina para o projeto, encontrou um computador usado, visivelmente velho, em outro departamento do Bell Labs. Era um PDP-7, também da DEC (Figura 1.13). Essa máquina estava sem uso e não foi difícil para Thompson conseguir a sua transferência.

Em meados de 1969, Ken Thompson começou a implementar o projeto do filesystem (chamado de "chalk filesystem" por Dennis Ritchie, uma vez que o mesmo havia sido projetado em um quadro de giz). Thompson dividiu o projeto em quatro partes e alocou uma semana para cada uma delas. Assim, ele trabalhou com os seguintes blocos: sistema operacional, ambiente shell, editor de texto e a compilação do sistema e dos programas. A linguagem de programação utilizada foi a Assembly.

Inicialmente, Ken Thompson trabalhou apenas com os requisitos primordiais de um sistema operacional. Primeiro, desenvolveu algumas aplicações em nível de usuário, todas voltadas para cópia, impressão, remoção e edição de arquivos. Depois, desenvolveu um ambiente shell (ambiente próprio para a entrada de linhas de comandos, como, por exemplo, o prompt do MS-DOS ou a "tela preta" do GNU/Linux). Com isso, o Unix começava a tomar forma.

Figura 1.13 – Unidade PDP-7.
Fonte: System photographs (http://simh.trailing-edge.com/photos.html).

Um novo problema surgia: o PDP-7 estava se tornando obsoleto rapidamente. Assim, em 1970, foi proposta a compra de uma nova máquina: o PDP-11. Doug McIlroy e Lee McMahon, líderes de dois departamentos de pesquisas do Bell Labs, perceberam os benefícios que o novo sistema operacional traria. Assim, ambos apoiaram a proposta. O PDP-11 (Figura 1.14) foi adquirido em meados de setembro do mesmo ano. No entanto, as unidades de disco não o acompanharam, tornando seu uso inviável. Em dezembro, após a chegada dos discos, começou um esforço para migrar o Unix.

Figura 1.14 – Thompson (de pé) e Ritchie programando no PDP-11 em um terminal teletipo.
Fonte: Bell Labs (http://www.bell-labs.com/about/history/unix/firstport.html).

1.2.11 Linguagem C

A primeira versão do Unix foi escrita em Assembly, uma complicada linguagem de baixo nível. Thompson tinha a intenção de passar o Unix para uma linguagem de alto nível.

A primeira tentativa foi utilizar a linguagem Fortran, em 1971. Essa tentativa se deu no PDP-7. No dia seguinte, a ideia de utilizar Fortran foi descartada. Assim, ele escreveu uma linguagem de programação simples, ainda no PDP-7, conhecida como B (uma simplificação do BCPL, o Basic Combined Programming Language, uma linguagem de alto nível criada em 1967). O nome B veio da primeira letra do BCPL. Dois problemas foram encontrados. O primeiro foi a lentidão da linguagem que, além de ser de alto nível, era interpretada. O segundo era que o PDP-7 tinha um processamento baseado em palavra (word-oriented) e o PDP-11 em bytes (byte-oriented). Ritchie usou o PDP-11 para adicionar funcionalidades ao B, que passou a se chamar NB (New B). A seguir, começou a fazer um compilador para o NB. Acabava de nascer a famosa linguagem C. O nome C vem da segunda letra do BCPL.

A primeira versão do Unix, uma versão ainda interna ao Bell Labs, foi lançada em novembro de 1971, escrito na linguagem B.

1.2.12 A nova fase do Unix

Com o surgimento da linguagem C, o Unix precisou ser reescrito e isso significava começar tudo de novo. Foi um processo lento, iniciado por Thompson em meados de 1972. Dois novos problemas surgiram. O primeiro era compreender como executar as rotinas auxiliares, o que exigiria a transferência do controle de um processo para outro. O segundo era a dificuldade de criar uma estrutura de dados, uma vez que a linguagem C não possuía esse tipo de estrutura. Por incrível que pareça, Thompson desistiu de fazer o Unix.

Ritchie não se deixou abater. Melhorou o C, já no PDP-11, e adicionou as estruturas necessárias, além de aprimorar bastante o compilador.

Em meados de 1973, Ritchie fechou um acordo de cooperação com Thompson e, novamente, refizeram todo o sistema.

Uma grande inovação do Unix, reprogramado em 1973, foram os pipes, que permitem aos programadores a possibilidade de amarrar vários processos e gerar uma única saída. Isso ocorre porque cada comando que emitimos em um sistema operacional é um processo. O pipe, que é representado pelo caractere barra vertical (|), permite encadear comandos. Exemplo:

```
comando1 | comando2 | comando3 | comando4
```

Resumindo, o resultado do `comando1` será processado pelo `comando2`. A seguir, esse resultado será processado pelo `comando3`. Na sequência, entra em ação o `comando4` e obtém-se, assim, um resultado final.

O conceito de pipe, implementado por Thompson, foi criado por Douglas McIlroy (Figura 1.15). McIlroy era chefe do Computing Science Research Center, um centro de pesquisas do Bell Labs, e trabalhava no conceito de pipes desde a década de 1950. Assim, voltando um pouco no tempo, por volta de 1969, ainda na época do quadro de

giz, enquanto Thompson e Ritchie esquematizavam o filesystem (sistema de arquivos), McIlroy buscava um modo de conectar processos de forma otimizada, obtendo bons resultados. Essa não foi uma tarefa simples. Segundo McIlroy, não era fácil dizer algo como "coloque o resultado de um `cat` dentro de um `grep`" ou "o resultado de um `who` dentro de um `cat` e, depois, dentro de um `grep`". Pior ainda, muitas vezes esses comandos poderiam ter chaves, opções e parâmetros. Algo como `ls -lh | grep -i issue`. Essa situação era realmente complicada e McIlroy não conseguira debelar o problema.

Figura 1.15 – Douglas McIlroy.
Fonte: Dartmouth College (`http://www.cs.dartmouth.edu/~doug`).

Mesmo frustrado, McIlroy não desistiu. Segundo o seu próprio relato, de 1970 a 1972 ele pensava em como fazer o pipe funcionar. Várias tentativas falharam. Um dia, finalmente propôs a Thompson uma associação entre eles para tentar criar uma sintaxe que permitisse implementar pipes no shell do Unix. Thompson respondeu: "eu estou me preparando para fazer isso".

Segundo McIlroy, Thompson já estava um pouco cansado de tanto ouvir besteiras. Ele não acatou exatamente o que foi proposto para a implementação dos pipes. Fez algo mais leve e que funcionou de forma parecida com o que temos hoje. Tudo isso em apenas uma noite. McIlroy narrou maravilhado: "ele colocou pipes no Unix; ele colocou esse recurso no shell, tudo em uma noite".

1.2.13 Unix nas universidades

Em 1976 Ken Thompson solicitou ao Bell Labs uma licença de seis meses e foi dar aulas na Universidade de Berkeley, no Estado americano da Califórnia. Lá ele ensinou o Unix e desenvolveu o que viria a ser depois a versão 6, uma versão voltada para universidades. O sistema foi um sucesso e se espalhou rapidamente pelas universidades norte-americanas. A maioria delas possuía o PDP. Além disso, a Digital estava praticando preços acessíveis às universidades, sendo que a máquina Digital VAX, uma das mais baratas e comum nas universidades, também aceitava o Unix.

Após a volta de Thompson para o Bell Labs, estudantes e professores continuaram a desenvolver o Unix, cujo código era aberto e permitia mudanças, mediante uma licença

universitária criada para tal fim. Surgia o Berkeley Software Distribution (BSD), um Unix totalmente adaptado ao ambiente acadêmico. Conforme disse Ken Thompson: "um sistema operacional feito por programadores para programadores".

1.2.14 Comercialização do Unix

O Unix se espalhou rapidamente pelo mundo acadêmico. Não havia dúvidas de que o mesmo poderia ser uma excepcional fonte de renda. A primeira ideia foi desenvolver programas para Unix para uso comercial. O principal diferencial do Unix era o sistema de time sharing, que permitia às pessoas compartilharem o mesmo computador ao mesmo tempo, utilizando os seus vários terminais. A portabilidade entre máquinas era grande. Os usuários da máquina poderiam trocar e-mails.

O sistema de e-mail criado era muito simples. Os usuários de uma determinada máquina deveriam estar cadastrados nela para poderem usá-la. Um usuário que trabalhasse no turno da manhã, por exemplo, poderia deixar uma mensagem para um usuário do turno da tarde. Bastaria enviá-la para `usuário@nome_da_máquina`. A partir daí, surgiu o nosso atual sistema de e-mail, sendo que "nome_da_máquina" foi substituído pelo domínio do provedor.

Em 1984, a AT&T, que era controlada pelo Bell Labs, criou uma subsidiária independente para si: a AT&T Computer Systems. Era subsidiária por ser controlada pela AT&T e independente por ter recursos próprios. No fim, era uma empresa nova e independente, apta a gerir qualquer tipo de negócio, inclusive a comercialização do Unix e derivados. Estava transposta uma barreira: a lei antitruste americana.

Várias versões de Unix foram produzidas. Muitas empresas passaram a vender máquinas projetadas para o uso com o Unix, dentre elas a Sun Microsystems, a SGI, a Hewlett-Packard, a NCR e a IBM. Em paralelo, na Universidade de Berkeley, um trabalho constante introduziu melhorias e versatilidade. Várias versões foram desenvolvidas pelo Bell Labs, inclusive a famosa System V, que criou um estilo seguido por vários sistemas atuais. Outra importante versão, também muito utilizada, é a BSD. BSD é a abreviatura de Berkeley Software Distribution, uma linha Unix desenvolvida em Berkeley que visa um produto final gratuito. Os BSD e derivados são famosos e muito utilizados. Os mais conhecidos são o MacOS X, o FreeBSD, o OpenBSD e o NetBSD.

Várias empresas desenvolveram os seus próprios Unix. Alguns deles:

- Solaris, da Sun Microsystems, atual Oracle America Inc.
- SunOS, da Sun Microsystems, atual Oracle America Inc.
- HP-UX, da Hewlett-Packard.
- AIX, da IBM.
- Tru64 Unix, da Compaq, atualmente fundida com a Hewlett-Packard.
- Xenix, da SCO, AT&T e Microsoft.
- OpenServer, da SCO.

A Figura 1.16 mostra a evolução do Unix entre 1969 e 1995. É lógico que o processo não parou em 1995.

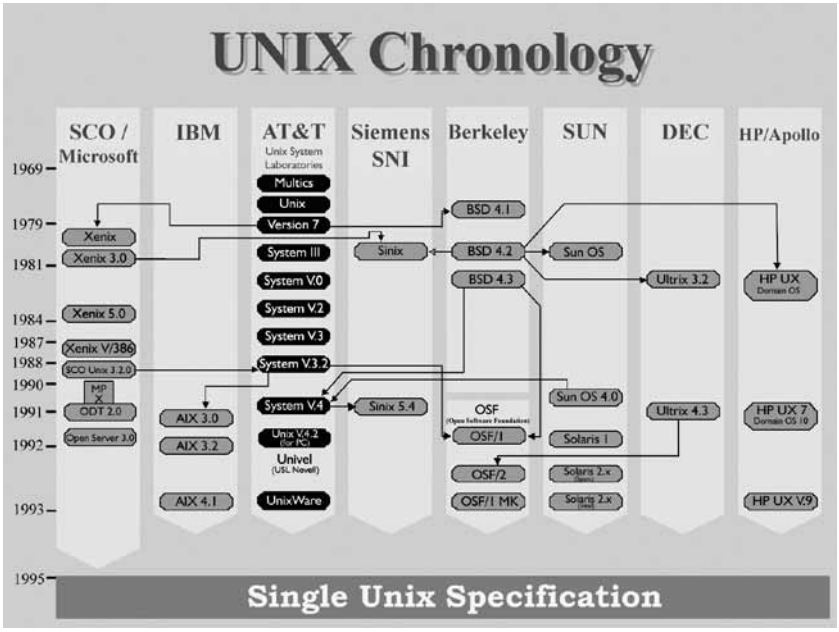

Figura 1.16 – Evolução do Unix entre 1969 e 1995.

Fonte: Eastern Mennonite University (`http://www.emu.edu/faculty/cooleycd/Spring2004/cs352/unix/overview.html`).

Em suma, como disse Ritchie: "Mais de trinta anos depois da criação e o Unix continua sendo um fenômeno".

1.2.15 Richard Stallman

Desde 1971, Richard Matthew Stallman trabalhava no Laboratório de Inteligência Artificial do MIT, desenvolvendo em uma máquina PDP-10 que rodava um sistema operacional chamado ITS (Incompatible Timesharing System, um trocadilho para o CTTS). Esse sistema operacional foi desenvolvido pelos próprios funcionários do laboratório. Ele não era comercial e os desenvolvedores do MIT o aperfeiçoavam cada vez mais. Stallman também fazia parte de uma comunidade voltada para o compartilhamento e a distribuição de software. Essa comunidade existiu por alguns anos. Naquela época ainda não havia o termo "free software" ou, como conhecemos, Software Livre. Mas, apesar do nome não existir ainda, o conceito de Software Livre já era aplicado. Segundo Stallman, "Quando alguém de outra universidade ou empresa precisava usar um programa do Laboratório de Inteligência, nós deixávamos com satisfação. E se você visse alguém usando um programa desconhecido e interessante, poderia pedir para ver o código dele também. Com isso, você poderia ler o código, alterá-lo e até aproveitar partes dele para gerar um novo programa". A Figura 1.17 mostra Richard Stallman.

Figura 1.17 – Richard Stallman.

Fonte: IrishEyes (http://irish.typepad.com/photos/mediatrips/stallmanthinks.jpg).

Toda essa situação de liberdade mudou drasticamente no início da década de 1980, quando a Digital descontinuou o PDP-10, que pode ser visto na Figura 1.18. A comunidade, liderada por desenvolvedores que trabalhavam com o PDP-10 no MIT, começou a desmanchar-se. O MIT resolveu comprar uma nova máquina, substituta do PDP-10, e um novo sistema operacional. O ITS, utilizado até então no PDP-10, não era baseado em time sharing e, portanto, era incompatível com os novos tempos. Assim, em 1982, a ideia dos administradores do MIT era comprar um software, não livre, baseado em time sharing. Os computadores modernos, como o VAX da Digital, tinham o seu próprio sistema operacional e nada mais era livre. Com isso, tornava-se necessário assinar um termo de confidencialidade para receber uma cópia do executável. Apenas o executável, nada de código-fonte. Nas próprias palavras de Stallman, "isso significava prometer não ajudar a quem precisasse; era uma proibição de uma comunidade colaborativa". As regras do contrato diziam: "se compartilhar o software com alguém, você será um pirata". Ainda: "se precisar de alguma alteração no software, peça-nos para fazê-la para você".

Figura 1.18 – Unidade PDP-10.

Fonte: Simulogics (http://www.simulogics.com/nostalgia/DEC/dec.htm).

Stallman já havia tido problemas com termos de confidencialidade em software proprietário, ainda na década de 1970. Naquela ocasião, o Laboratório de Inteligência Artificial havia recebido uma impressora laser, da marca Xerox. Ao que consta, era a única laser da Xerox, no mundo, que não se encontrava dentro da própria Xerox. Na

verdade, era uma adaptação de uma copiadora. Essa impressora estava acoplada a um PDP-10 e, constantemente, apresentava problemas com o papel, que prendia no rolo pressor. Esse problema só era detectado quando se estava diante da impressora, o que causava uma perda de tempo enorme porque, mediante uma falha, não se podia adotar uma ação imediata. A solução seria alterar o driver da impressora para que a mesma pudesse avisar sobre a ocorrência de falhas. Quase tudo naquela época era livre, exceto o driver daquela impressora, que envolvia um termo de confidencialidade, uma vez que a mesma era utilizada somente dentro da Xerox e no MIT. Após um contato, a Xerox se negou a fornecer a Stallman e ao MIT o código do driver. Em razão disso, o problema ficou sem solução. A partir desse fato, Stallman concluiu que seria ingenuidade pensar que a assinatura de um termo de confidencialidade garantiria ajuda em qualquer circunstância, caso precisasse.

Uma ideia nunca abandonara a mente de Stallman: como fazer para que a comunidade de programadores voltasse a existir novamente? A resposta parecia óbvia. Tendo em vista que um computador só funciona se tiver um sistema operacional, era necessário um sistema operacional livre. O Unix já não o era mais. A única saída seria fazer um sistema operacional.

Depois de analisar a situação, Stallman concluiu que o novo sistema, para ser bom e versátil, deveria ser compatível com o Unix. Além disso, usuários do Unix gostariam de ter o seu próprio Unix, sem precisar comprá-lo. Isso aumentaria as chances de um trabalho em comunidade gerar algo satisfatório.

Em janeiro de 1984, Richard Stallman demitiu-se do MIT e começou a escrever o código do novo sistema. Nas suas palavras, "Foi necessário sair do MIT para ele não interferir na característica de Software Livre do novo projeto. Se eu tivesse permanecido no MIT, alguém poderia querer reivindicar algo sobre o projeto. Isso iria gerar um software proprietário, pois termos de uso acabariam sendo impostos. E o objetivo final, que era criar uma nova comunidade para a troca de software, não seria atingido". Apesar de toda essa situação, Stallman foi convidado a continuar usando os recursos do Laboratório de Inteligência Artificial do MIT.

1.2.16 Definição de Software Livre

A expressão "Software Livre" gera uma enorme confusão na cabeça das pessoas. Muitos pensam que Software Livre (ou "free software") é algo gratuito. O termo "free" está ligado a livre e não a gratuito.

Software livre é um conceito especial. Esse conceito prevê que todo software será distribuído com seu código-fonte, podendo ser alterado e até mesmo redistribuído depois de alterado. Mas esse software não precisa ser gratuito. O seu pagamento pode se dar de várias formas. Por exemplo: você produz um banco de dados e o vende por uma determinada quantia. Isso irá custear a mídia, a embalagem etc. Quem quiser, poderá copiar livremente ou alterar o código e não terá de lhe pagar nada. No entanto, você

pode cobrar pelo suporte técnico. Pode ser um contrato mensal, por exemplo. É daí que vem o lucro. Outro exemplo: você faz um software e o vende dentro de uma caixa que também contém um manual com mil páginas. Qualquer pessoa pode adquirir uma cópia, gratuitamente, de outra pessoa que já tenha comprado o pacote. No entanto, não haverá o manual. Muitos irão preferir ter a bela caixa e o manual, comprando diretamente de você. Mas não se esqueça: em qualquer um desses casos, sempre haverá o código-fonte e, se uma pessoa passar uma cópia para outra, não será um caso de pirataria.

Segundo a definição de Richard Stallman, o Software Livre nos proporciona:

- a liberdade de executar um programa, seja qual for o propósito;
- a liberdade de modificar um programa para adaptá-lo às suas necessidades e, para que isso ocorra, você deve ter acesso ao código-fonte;
- a liberdade de redistribuir cópias, gratuitamente ou mediante uma taxa;
- a liberdade de distribuir versões modificadas do programa e, nesse caso, toda a comunidade poderá beneficiar-se dos aperfeiçoamentos.

Temos de saber diferenciar free software e freeware. O free software, na sua mais ampla concepção, traz consigo o código-fonte, pode ser vendido e ser livremente alterado, adaptado e redistribuído. O freeware é obrigatoriamente de graça, mas não traz consigo o código-fonte e, em consequência, não pode ser alterado.

1.2.17 Projeto GNU

O sistema operacional de Richard Stallman recebeu o nome de Projeto GNU ou sistema operacional GNU. A palavra gnu, originalmente, refere-se a um mamífero ruminante, semelhante a um búfalo, com chifres espiralados, que vive no continente africano. A Figura 1.19 mostra um gnu.

Figura 1.19 – Gnu.

Fonte: Karibu Southern Africa Safari (`http://home.vicnet.net.au/~neils/africa/gnu.htm`*).*

No caso do sistema operacional de Richard Stallman, GNU é um trocadilho que significa "Gnu's Not Unix", ou seja, o projeto GNU é uma concepção livre, ao contrário do Unix e de outros softwares, que eram livres e deixaram de sê-lo. Esse tipo de trocadilho, na época, era muito utilizado por programadores para nomearem seus projetos. Assim, o projeto GNU refere-se a uma série de aplicativos livres, desenvolvidos para os mais diversos fins, contendo editores de texto, planilhas de cálculo etc., tudo com o intuito de compor um sistema operacional livre. O projeto está hospedado em `http://www.gnu.org`. O símbolo do projeto é a caricatura da cabeça de um gnu, como pode ser visto na Figura 1.20.

Figura 1.20 – Símbolo do projeto GNU.
Fonte: The GNU Operating System (`http://www.gnu.org`).

Conta Stallman que o início do projeto foi um pouco conturbado. Ele ouvira falar de um tal Free University Compiler Kit, um compilador desenvolvido para múltiplas linguagens, incluindo C e Pascal. Richard escreveu para o autor perguntando se ele poderia inserir esse compilador no sistema operacional GNU. Obteve uma resposta debochada do autor, segundo o qual a universidade era free, mas o compilador não. Stallman, revoltado, iniciou o desenvolvimento do GNU pelo compilador. Assim, nasceu o compilador C chamado GCC (GNU C Compiler).

Entre o início e o fim do desenvolvimento do GCC, Stallman fez o GNU Emacs, um editor de textos muito utilizado até hoje. A sua elaboração começou em setembro de 1984. No início de 1985 o GNU Emacs já podia ser utilizado.

O GNU Emacs já rodava com perfeição sobre o Unix e muitas pessoas pediram para usá-lo. Assim, Stallman o disponibilizou em um servidor ftp público do MIT, o `prep.ai.mit.edu`. Esse servidor está no ar até hoje e hospeda parte do projeto GNU. É possível logar-se a ele como usuário `anonymous` e navegar nos seus diretórios. Lá estará o projeto GNU!

1.2.18 Free Software Foundation

O interesse, por parte das pessoas, em usar o Emacs crescia cada vez mais. Alguns começaram a ajudar no seu desenvolvimento, o que era um objetivo antigo de Stallman.

Assim sendo, o inevitável ocorreu: era necessário injetar capital no projeto, que crescia cada vez mais e necessitava de colaboradores.

A falta de capital para manter o projeto foi contornada com uma solução clássica: ainda em 1985, Richard Stallman fundou uma instituição chamada Free Software Foundation (FSF), criada para arrecadar fundos para a manutenção do Projeto GNU. A FSF existe até hoje e aceita doações. No entanto, a sua maior fonte de renda é originária da venda de CD-ROM com códigos-fonte e binários, além da venda de manuais impressos. Os empregados dessa instituição desenvolvem e mantêm vários programas e pacotes do sistema GNU, destacando-se as bibliotecas C e o Shell BASH, utilizado na maioria dos GNU/Linux.

É interessante dizer que todo sistema operacional possui um núcleo de controle, denominado kernel. O sistema operacional em si é constituído do kernel e de programas como editores de texto e utilitários de cópia de arquivos etc. O projeto GNU já possui vários programas, a maioria testados em Unix. No entanto, ainda não há um kernel maduro. Hurd é o nome do kernel que está em desenvolvimento e sem previsão para a primeira versão estável.

O endereço do site da Free Software Foundation é `http://www.fsf.org`. O Kernel Hurd pode ser encontrado em `http://hurd.gnu.org`.

1.2.19 Free software e open source

Dois termos muito utilizados atualmente são free software (ou Software Livre) e open source (ou código aberto). Muitos encontram dificuldades em explicar a diferença entre os termos. A grande verdade é que free software e open source são a mesma coisa. Software livre, já definido anteriormente, refere-se à liberdade de poder executar, estudar, modificar e redistribuir versões, originais ou modificadas, de um programa. Assim sendo, o Software Livre é uma filosofia, uma forma de pensar. Open source seria um modelo de desenvolvimento que, no fim, respeita os mesmos princípios do Software Livre.

A Open Source Initiative, cujo site é `http://www.opensource.org`, estabeleceu um conceito de open source baseado na Definição Debian de Software Livre (DFSG – Debian Free Software Guidelines), disponível em `http://www.debian.org/social_contract`. Segundo a Open Source Initiative, um open source deve seguir preceitos referentes aos seguintes tópicos:

- redistribuição livre;
- código-fonte;
- trabalhos derivados;
- integridade do código-fonte do autor;
- não-discriminação às pessoas ou grupos;
- não-discriminação às diversas intenções de utilização;
- a licença não deve ser específica para um produto;

- a licença não deve restringir outro software;
- a licença não pode ser calcada sobre qualquer tecnologia.

O maior defensor do conceito "Open Source" é Eric Reymond. Eric descreve o modelo de desenvolvimento open source no livro "The Cathedral and the Bazaar" (A Catedral e o Bazar).

Alguns fornecedores de software distorcem um pouco o conceito de código aberto, disponibilizando o código-fonte de um programa, total ou parcialmente, mediante uma licença que restringe o uso desse código de alguma forma. Recentemente, surgiu o conceito de FOSS (Free and Open Source Software). Esse conceito foi criado, principalmente, para mesclar comunidades em eventos, simpósios, seminários, trabalhos conjuntos etc.

1.2.20 Licença GNU GPL

A Licença GNU General Public License foi desenvolvida pela Free Software Foundation (FSF) para especificar se um software é livre ou não. Existem várias outras licenças, inclusive compatíveis com a GNU GPL, mas essa é a mais recomendada.

Numa avaliação geral, a GNU GPL se baseia nas quatro liberdades básicas: executar, estudar, modificar e redistribuir versões, originais ou modificadas, de um programa.

A Licença GNU GPL está disponível em `http://www.gnu.org/copyleft/gpl.html`. Em `http://www.gnu.org/licenses/license-list.html` estão disponíveis vários exemplos de licenças e os devidos comentários por parte da Free Software Foundation. A distribuição Debian também mantém um site, com licenças comentadas, em `http://wiki.debian.org/DFSGLicenses`.

> É importante ressaltar que uma licença é um acordo entre partes. Nesse caso, entre o desenvolvedor e os usuários finais. Assim sendo, geralmente, uma licença como a GNU GPL tem validade no Brasil e em outros países. Esse fato esgota dúvidas, tais como: a GPL tem validade no Brasil? A resposta é sim, mesmo que não na sua totalidade (uma licença poderá conter cláusulas consideradas abusivas, por exemplo, em um determinado país).

1.2.21 Minix

Até a versão 6 do Unix, lançada em abril de 1976, o seu código-fonte estava disponível para as universidades, sob uma licença da AT&T. John Lions, da Universidade de New South Wales, na Austrália, chegou a escrever um pequeno livro que descrevia o Unix, linha por linha. Esse livro era utilizado pelas universidades do mundo inteiro, sob uma licença da AT&T, nos cursos de sistemas operacionais.

Em dezembro de 1978, a AT&T lançou a versão 7 do Unix. A licença que acompanhava a nova versão estabelecia que o Unix não poderia mais ser estudado nas universidades. O Unix estava se tornando um líder de mercado, e as novas restrições faziam parte de uma estratégia de manutenção de sigilo sobre a sua tecnologia. Assim sendo, várias universidades acabaram com os seus cursos de sistema operacional ou os mantiveram apenas no campo teórico. Isso gerava um problema: algumas vezes, pela falta de com-

provação prática, alguns conceitos eram distorcidos; outros, abandonados. Muita coisa só poderia ser entendida na prática.

Para solucionar o problema da falta de um Unix para realizar estudos, Andrew Stuart Tanenbaum, conhecido também como Andy Tanenbaum (Andy é um apelido), resolveu fazer, em 1986, um sistema operacional similar ao Unix. A Figura 1.21 mostra Tanenbaum.

Figura 1.21 – Andrew Tanenbaum.
Fonte: Universidade de Vrije (`http://www.cs.vu.nl/peop/faculty-en.html`).

Na época, Tanenbaum trabalhava como professor de ciência da computação na Universidade de Vrije, sediada em Amsterdã, capital da Holanda. Na verdade, até hoje o professor Tanenbaum trabalha na mesma universidade. Sua ideia era fazer um sistema operacional igual ao Unix por fora, mas totalmente diferente por dentro. E, por não usar nenhuma linha do Unix da AT&T, ele poderia ser utilizado individualmente ou em sala de aula para ensinar o funcionamento de um sistema operacional. Assim, estudantes poderiam "dissecá-lo" para ver como um sistema operacional funciona por dentro.

Tanenbaum lançou a primeira versão do seu sistema operacional em novembro de 1986. O nome escolhido para o mesmo foi Minix, que significava mini-Unix. A ideia de mini relaciona-se à necessidade de ser pequeno o bastante para que qualquer pessoa possa entender o seu funcionamento. Ademais, há milhares de comentários no código, com o intuito de facilitar o aprendizado.

Além de resolver o problema de licença, Tanenbaum cita como outra grande vantagem do Minix o fato de ter sido escrito mais de uma década depois do Unix, o que o torna mais bem estruturado por utilizar teorias modernas.

O Minix, assim como os mais novos Unix, foi escrito em C. Foi projetado para operar em computadores padrão IBM PC. No entanto, já existem versões para as arquiteturas Atari, Amiga, Macintosh, e SPARC. Na arquitetura IBM PC, roda a partir do IBM PC XT, requerendo muito pouco espaço em disco. Pode ser instalado em um filesystem próprio ou sobre o MS-DOS. Também pode rodar em sistemas de virtualização como VMWare e VirtualBox.

O site oficial do Minix é `http://www.minix3.org`.

1.2.22 USENET

USENET é o nome de um sistema de grupos de discussão, também conhecido como newsgroup, que utiliza servidores do tipo news (protocolo de rede NNTP) para trocar informações sobre vários assuntos. Pode-se criar um grupo diferente para cada assunto. É um excelente sistema para a disseminação do conhecimento. Existem servidores news públicos, que permitem a conexão para a leitura das mensagens lá armazenadas. O sistema de newsgroup difere das listas de discussão convencionais em muitos aspectos. Muitas são as vantagens, sendo que a maior delas é a possibilidade de ler mensagens postadas há vários anos.

Para ler as mensagens de newsgroup é necessário utilizar um leitor de news e se cadastrar em um servidor. Uma excelente opção é o servidor `news.gmane.org`, também acessível pela web no site `http://news.gmane.org`.

1.2.23 Minix na USENET

Conta Andrew Tanenbaum que, após o lançamento da primeira versão do Minix, um grupo de discussão, até hoje existente, foi formado na USENET para discutir o sistema operacional. Trata-se do grupo `comp.os.minix`. Poucas semanas depois da sua criação, o grupo já tinha quarenta mil inscritos. Havia muitas propostas de melhorias para o Minix. As pessoas queriam ajudar e, inclusive, enviavam códigos prontos. No entanto, Tanenbaum resistiu, por anos, a quase todas as propostas, pois isso deixaria o sistema grande e pesado. O objetivo do projeto Minix era disponibilizar às pessoas um código leve e fácil de entender. O Minix não foi feito para ser realmente um sistema operacional mas sim para ensinar como era um sistema operacional.

1.3 Padrão POSIX

POSIX é uma sigla que significa Portable Operating System Interface e consiste em uma norma concebida pelo Institute of Electrical and Electronics Engineers (IEEE), instituição americana criada para definir padrões para equipamentos elétricos, que atualmente atua em todos os ramos da tecnologia. Assim, o Padrão POSIX, norma IEEE Std 1003.1-1988, é um conjunto de regras que define como um sistema operacional deve funcionar para ser um Unix. Também normatiza alguns aspectos dos programas desenvolvidos para Unix.

1.4 Kernel Linux

Em 03 de julho de 1991, um jovem estudante de ciência da computação da Universidade de Helsinque, na capital da Finlândia, postou uma mensagem no newsgroup `comp.os.minix`, que dizia (tradução apenas do trecho que interessa):

```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
```

```
Subject: Gcc-1.40 e uma questão sobre posix
Keywords: gcc, posix
Message-ID: <1991Jul3.100050.9886@klaava.Helsinki.FI>
Date: 3 Jul 91 10:00:50 GMT
Organization: University of Helsinki
Lines: 28

Olá internautas,

Em razão de um projeto no qual trabalho (baseado no Minix), estou interessado nas definições dadas
pelo Padrão Posix. Alguém pode, por favor, citar um endereço que contenha as últimas normas Posix?
Sites ftp serão bem-vindos.
--- corte ---
Linus Torvalds torvalds@kruuna.helsinki.fi
--- corte ---
```

Em 25 de agosto de 1991, ele postou outra mensagem (atente para a linha Summary no cabeçalho da mensagem):

```
From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: O que você gostaria de ver a mais no Minix?
Summary: pequena pesquisa para o meu novo sistema operacional
Keywords: 386, preferences
Message-ID: <1991Aug25.205708.9541@klaava.Helsinki.FI>
Date: 25 Aug 91 20:57:08 GMT
Organization: University of Helsinki
Lines: 20

Olá para todos que estão usando Minix -

Estou fazendo um sistema operacional independente (apenas um hobby, nada grande e profissional
como o GNU) para AT 386 (486) e similares. Iniciei em abril e, agora, está começando a dar
certo. Preciso de um retorno sobre as coisas que as pessoas gostam/não gostam no Minix, porque o
meu sistema se parece com ele (o mesmo layout de filesystem, por razões práticas, dentre outras
coisas).

Atualmente, estou portando o bash (1.08) e o gcc (1.40) e as coisas têm funcionado. Isso significa
que vou ter algo prático em poucos meses e gostaria de saber quais características as pessoas vão
querer. Qualquer sugestão será bem-vinda, apesar de não prometer que eu vá implementá-la :-)

Linus (torvalds@kruuna.helsinki.fi)

Obs: Sim - ele é independente de qualquer código Minix e tem um filesystem do tipo multi-threaded.
Ele NÃO é portável (usa características do 386 etc.) e provavelmente nunca irá suportar qualquer
outro tipo de HD que não seja AT, pois isso é tudo o que eu consegui.
```

Linus Benedict Torvalds foi o autor das duas mensagens. Na época, com 21 anos, ele falava de um sistema operacional que estaria criando. Ao que tudo indica, Linus não tinha ideia do que se transformaria a sua criação. Hoje em dia, o GNU/Linux é amplamente utilizado no mundo inteiro e em diversas arquiteturas de hardware. Na Figura 1.22, Linus Torvalds, mais de dez anos depois da criação do Kernel Linux, mostrando algo que ele nunca imaginou que pudesse acontecer.

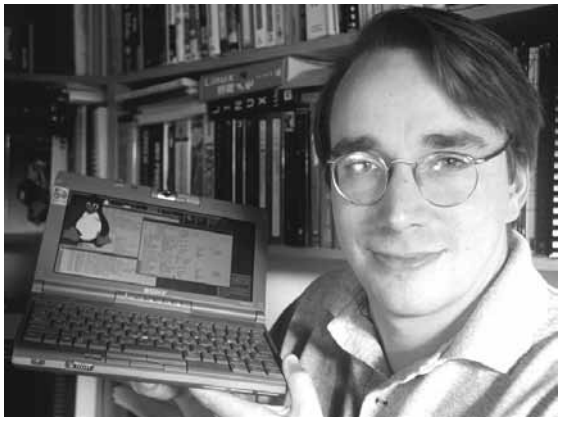

Figura 1.22 – Linus Torvalds, o criador do Kernel Linux.
Fonte: Valitut Palat (http://www-fi.valitutpalat.fi/lehti/tiedotteet/20002712.html).

Ainda em 1991, ele postou a seguinte mensagem:

From: torvalds@klaava.Helsinki.FI (Linus Benedict Torvalds)
Newsgroups: comp.os.minix
Subject: Código-fonte de kernel compatível com o Minix para AT 386
Keywords: 386, versão preliminar
Message-ID: <1991Oct5.054106.4647@klaava.Helsinki.FI>
Date: 5 Oct 91 05:41:06 GMT
Organization: University of Helsinki
Lines: 55

Você aspira pelos bons tempos do Minix 1.1, quando os homens serão independentes e escreverão os seus próprios drivers de dispositivos? Está sem um bom projeto e deseja dedicar-se a um sistema operacional que você possa tentar modificar de acordo com as suas necessidades? Está se sentindo isolado quando todo mundo trabalha no Minix? Perde uma noite inteira tentando fazer um programa funcionar? Então esta mensagem é exatamente para você :-)

Como mencionei há um mês (?) atrás, estou trabalhando em uma versão livre de um sistema similar ao Minix para computadores AT 386. Ele está finalmente atingindo o estágio de uso (pode ser que ainda não esteja do jeito que você quer), e vou disponibilizar o código para ampla divulgação. Ele está na versão 0.02 (+1 (muito pequeno) patch pronto), porém, estou rodando com sucesso bash/gcc/gnu-make/gnu-sed/compress etc. sobre ele.

Os fontes deste projeto podem ser achados em nic.funet.fi (128.214.6.100), no diretório /pub/OS/Linux. O diretório também contém alguns README e um par de binários para trabalhar sobre o Linux (bash, atualizado, e gcc - o que mais você pode querer?). :-) O fonte completo do kernel está disponível. Como no Minix, o código pode ser utilizado. Os fontes das bibliotecas são parcialmente livres, então eu não posso distribuí-los atualmente. O sistema, como está, pode ser compilado e sabe-se que ele funciona. Os fontes para os binários (bash e gcc) podem ser encontrados no mesmo servidor, em /pub/gnu.

--- corte ---

Estou interessado em ouvir alguém que tenha escrito qualquer utilitário/biblioteca para o Minix. Se seus produtos forem livremente distribuídos (sob licença ou domínio público), gostaria da sua autorização, para adicioná-lo ao sistema.

--- corte ---

Linus

--- corte ---

Esta última mensagem é considerada a mensagem oficial de lançamento do Kernel Linux. Por isso, o seu aniversário é comemorado em 05 de outubro. Note também a presença do nome do novo sistema: Linux. Linux é a junção dos nomes Linus e Unix. Cabe ressaltar que o nome escolhido por Linus para o seu sistema foi Freix (Free Unix). No entanto, o administrador do servidor FTP `nic.funet.fi` não aprovou o nome e se recusou a disponibilizar o código, caso Linus não o mudasse.

Gostaria de ouvir Linus Torvalds ensinando como se deve pronunciar a palavra Linux? Basta visitar o site `http://www.paul.sladen.org/pronunciation`.

1.5 O sistema operacional GNU/Linux

Mas o que é o Linux afinal? Essa pergunta também causa confusão a muitas pessoas. Linus, em seu e-mail inicial, disse o seguinte: "estou fazendo um sistema operacional". A Free Software Foundation começou o GNU pelos aplicativos e ainda não conseguiu terminar o kernel (Hurd). Linus, ao contrário da FSF, começou pelo kernel. No entanto, nunca chegou a desenvolver os aplicativos. Ou seja: o Linux é só um kernel, não um sistema operacional. E nos tempos atuais, não há mais a intenção de fazer um sistema operacional completo. Manter um kernel, mesmo trabalhando em comunidade, já é trabalhoso o suficiente. A Figura 1.23 mostra a evolução do kernel.

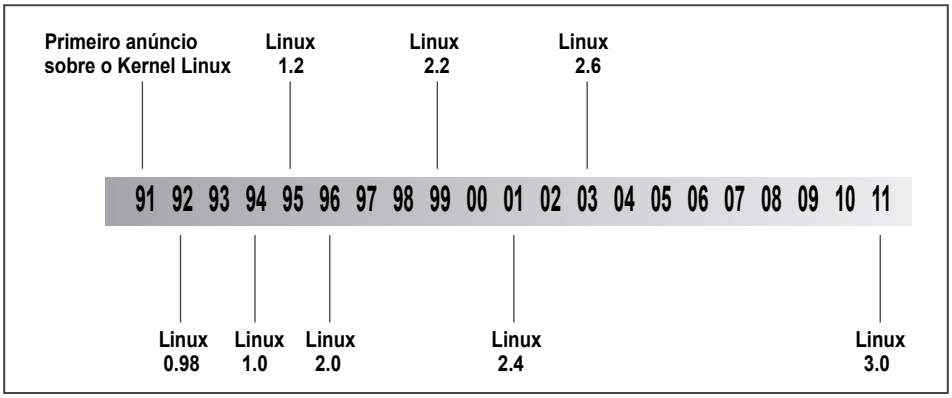

Figura 1.23 – Linha do tempo do Kernel Linux.

Linus utilizou exaustivamente os programas gerados pela FSF para o Projeto GNU. A lógica era simples: os programas da FSF funcionavam no Unix. Se funcionassem também no Linux, seria um sinal de que o kernel estaria ajustado e similar ao kernel do Unix. Desde o início, Linus distribuiu o seu kernel com programas da FSF, promovendo uma integração (um pouco unilateral, diga-se de passagem).

Por tudo isso, o nome Linux refere-se apenas ao kernel criado por Linus Torvalds. O sistema operacional que conhecemos por Linux chama-se, na verdade, sistema operacional GNU/Linux. Essa é a forma correta de se referir ao conjunto, ao sistema operacional.

Normalmente, falamos apenas Linux. Isso ocorre tanto pelo desconhecimento quanto pela comodidade. No entanto, neste livro, por uma questão didática, procuraremos sempre utilizar o nome GNU/Linux para o sistema operacional e somente Linux para o kernel.

O site oficial de distribuição do Kernel Linux é `http://www.kernel.org`.

A sistemática de designação do kernel passou por algumas fases. Na primeira fase, até o kernel 2.6.10, inclusive, foram utilizados por três números separados por pontos. Exemplo: 2.6.8. O primeiro número indicava a versão "master" do kernel. O segundo número indica modificações muito significativas, enquanto o terceiro indica pequenas correções ou a adição de recursos.

A partir do kernel 2.6.11, lançado em março de 2005, começou a ser utilizada uma designação com três ou quatro números. Exemplo: 2.6.19.2. Os dois primeiros números têm pouco uso atualmente, uma vez que levará anos para mudá-los. A versão 2.6, por exemplo, existiu por quase uma década. O terceiro número representa a versão atual. O quarto número representa correções de bugs e falhas de segurança. O objetivo principal desta última mudança, proposta por Linus Torvalds, seria facilitar o desenvolvimento e tornar o ciclo de lançamento mais rápido, pois seria possível disponibilizar um kernel considerado maduro mas, ao mesmo tempo, corrigir problemas que fossem surgindo.

Até o lançamento do kernel 2.6, quando ainda só eram utilizados três números, se o segundo número fosse ímpar, estaríamos lidando com um kernel em desenvolvimento. Exemplo: 2.5.8. Já um número par indicaria um kernel estável.

Atualmente, quando uma nova versão de kernel é lançada, ela só possui três números. A primeira correção dessa versão irá gerar o quarto número. Na eminência de lançamento, o kernel que está sendo trabalhado passa a ter três números novamente e é acrescido o sufixo `rc`, sigla de release candidate ou candidato a lançamento. Um exemplo de ciclo de desenvolvimento:

- A versão estável `2.6.50` é lançada.
- A nova versão recebe alterações e passa a se chamar `2.6.50.1`.
- Novas alterações e teremos `2.6.50.2`, `2.6.50.3` e `2.6.50.4`.
- A versão `2.6.50.4` está sendo trabalhada para ser a nova estável e começa a receber novos recursos e aprimoramentos. Neste momento ela passa a se chamar `2.6.51-rc1`.
- Se muito tempo se passar e muitos recursos forem adicionados, a versão `2.6.51-rc1` passará a se chamar `2.6.51-rc2`. Mas vamos considerar que isso não tenha ocorrido.
- A versão `2.6.51-rc1` estabiliza e é lançada, definitivamente, com o nome de `2.6.51`.

Cabe ainda ressaltar que o Kernel Linux, desde o início, contou com a ajuda de vários desenvolvedores para tornar-se o que é hoje. Alan Cox é um profundo conhecedor do kernel e ajuda Linus nas partes mais difíceis do desenvolvimento. Mas lembre-se: o Linux é desenvolvido por pessoas do mundo todo, da comunidade livre. Se você quiser colaborar,

bastará entrar no grupo de discussão oficial do kernel e propor as alterações que achar necessárias. Maiores informações poderão ser encontradas em `http://www.kernel.org`.

> Atualmente, o código-fonte do Kernel Linux possui mais de 11 milhões de linhas. Ainda, caso você deseje acompanhar os lançamentos de novas versões do kernel via Agenda do Google, siga as instruções contidas em `http://lwn.net/Articles/253955`.

Um ciclo histórico detalhado de desenvolvimento do Kernel Linux pode ser visto na Figura 1.24.

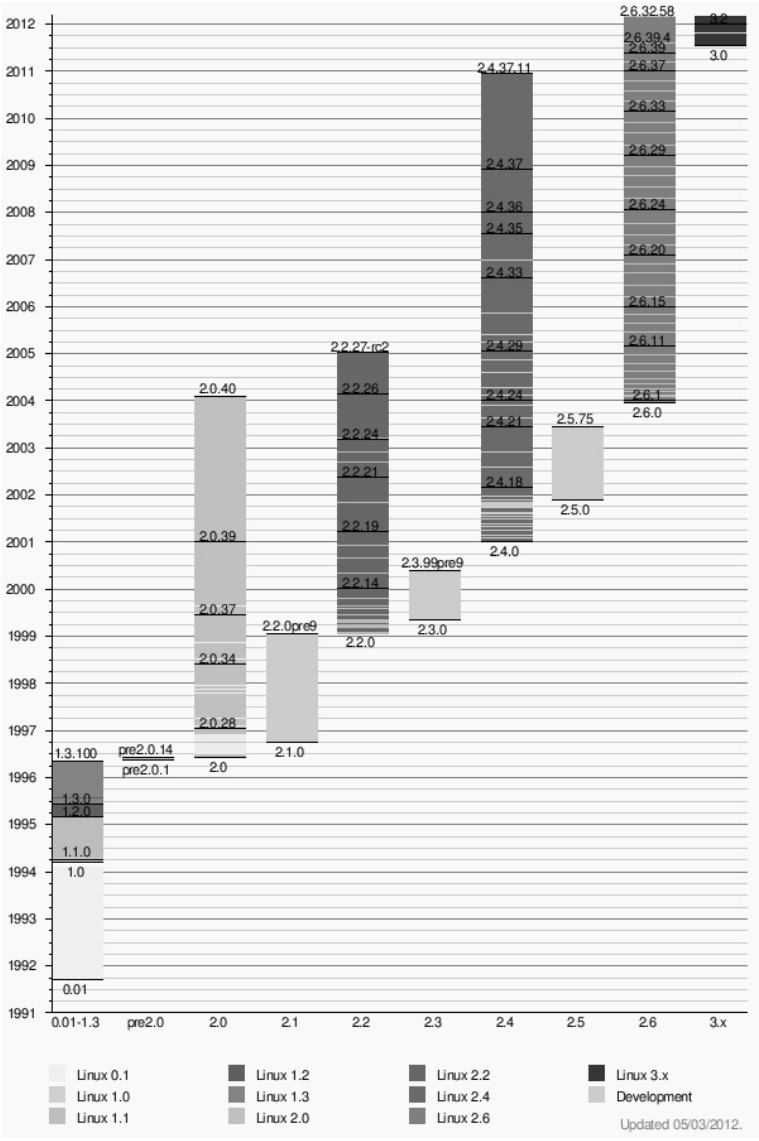

Figura 1.24 – Ciclo histórico detalhado de desenvolvimento do Kernel Linux. Atualizado em 05 mar. 2012.
Fonte: Wikipedia.Org (`http://en.wikipedia.org/wiki/Linux_kernel`), baseado nos dados contidos em Kernel.Org (`http://www.kernel.org/pub/linux/kernel`).

1.6 Motivos para criar o Linux

Em uma entrevista foi perguntado a Linus: "O que o levou a escrever o Linux?". A resposta foi a seguinte:

> "Bem, como eu disse, queria um determinado desempenho em casa e o DOS (e o Windows) não me ofereciam isso. Comecei tentando um pequeno clone do Unix, chamado Minix. Eu era capaz de entender algo sobre as coisas que pretendia com ele. Por outro lado, faltava-me a plena funcionalidade do Unix. A simplicidade do Minix (e os problemas de performance do Minix) levaram-me a desejar algo melhor. No entanto, o Unix custava muito e não seria fácil encontrar algo bom sem dinheiro (que eu definitivamente não tinha). Uma versão de Unix razoavelmente boa, com ferramentas de desenvolvimento etc., custava alguns milhares de dólares. Como eu era um estudante pobre e havia usado todo o meu dinheiro para comprar um computador, eu realmente não tinha opção... Mas, como eu conhecia computadores, comecei a fazer um sistema para mim mesmo, e o resto da história todos conhecem".

A referida entrevista encontra-se disponível, na íntegra, em `http://www.hio.hen.nl/~eniac/Commissies/CommIT/95_96/it4/09_linus.html`, com cópia em `http://www.softpanorama.org/People/Torvalds/Interviews/linus1995.shtml`.

1.7 Distribuições GNU/Linux

Quando juntamos um Kernel Linux, diversos aplicativos, compiladores etc., alguns da Free Software Foundation (projeto GNU) e outros não, temos uma distribuição GNU/Linux. Em síntese, distribuição é a união do kernel com vários programas compatíveis com ele. Como no início a esmagadora maioria dos aplicativos utilizados era proveniente do GNU, criou-se o sistema operacional GNU/Linux.

Atualmente, existem várias distribuições. O site DistroWatch.com (`http://distrowatch.com`) afirma que temos 318 distribuições ativas (dados de março de 2012). As maiores e mais antigas ainda em produção são, na ordem: Slackware, Debian, openSUSE e Red Hat. Muitas das outras distribuições são derivadas dessas.

As quatro primeiras distribuições que existiram na história foram as seguintes:

- **"Ted"**: não oficialmente, foi a primeira de todas. Um desenvolvedor Unix, chamado Theodore Y. Ts'o, enviou uma mensagem para um grupo de discussão oferecendo uma distribuição GNU/Linux, gerada por ele, a US$ 2.50. A mensagem foi enviada em 20 de janeiro de 1992 e dizia que a distribuição estaria pronta em 01 de fevereiro de 1992. A mensagem original está disponível em `http://www.kclug.org/old_archives/linux-activists/1992/jan/2/0152.shtml`. Não se pode avaliar, pela Internet, o que aconteceu com essa distribuição, pois ninguém respondeu ao autor da mensagem dentro do grupo. Num encontro casual, em 2005, Ts'o me disse que ninguém comprou a distribuição. Mais tarde, Ted, como prefere ser chamado, liderou o projeto Kerberos e participou da elaboração dos filesystems Ext2, Ext3 e Ext4.
- **MCC Interim Linux**: oficialmente foi a primeira de todas, sendo lançada em fevereiro de 1992 pelo Manchester Computing Centre. Foi desenvolvida por Owen Le Blanc. Não possuía ambiente gráfico. Essa distribuição teve vida curta.

- **SLS (Softlanding Linux System)**: lançada em agosto de 1992. Foi produzida por uma empresa chamada Softlanding. Deu origem à distribuição Slackware, existente até hoje. A SLS não existe mais.
- **TAMU (Texas A&M University):** A TAMU foi uma distribuição GNU/Linux, provavelmente criada em abril de 1992, mas com certeza em 1992. A Texas A&M University é uma universidade pública do Texas. Foi fundada em 1876. A Distribuição TAMU não existe mais.

A Figura 1.25 mostra uma linha do tempo com as datas de lançamento das primeiras versões das distribuições GNU/Linux mais relevantes da história. Os fundos mais escuros representam distribuições que não existem mais.

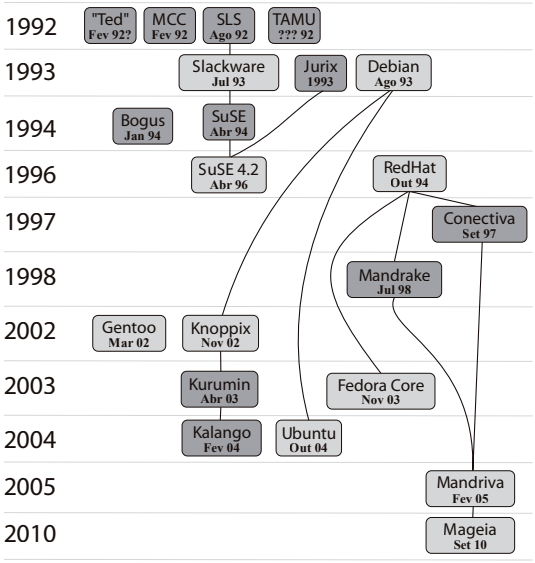

Figura 1.25 – Linha do tempo das principais distribuições GNU/Linux.

A Figura 1.25 mostra que há distribuições baseadas em outras. Por exemplo: a distribuição Fedora Core é baseada na Red Hat. Isso significa que ela herdou uma série de características inerentes à distribuição Red Hat. Já as distribuições Debian e Red Hat são chamadas de puras, por não serem baseadas em nenhuma outra. Também pode ocorrer de uma distribuição ser baseada em outra que, por sua vez, já é baseada em alguma outra. Por exemplo: a extinta distribuição Kurumin era baseada na distribuição Knoppix que, por sua vez, é baseada em Debian.

A Figura 1.26 mostra algumas grandes distribuições e quantas outras, existentes atualmente, baseiam-se nelas. O eixo horizontal representa as distribuições matrizes e o eixo vertical a quantidade de distribuições filhas (apenas as que estão ativas).

As distribuições são desenvolvidas em vários países. A Figura 1.27 mostra os países que mais desenvolvem distribuições GNU/Linux. Observe como o Brasil encontra-se bem posicionado.

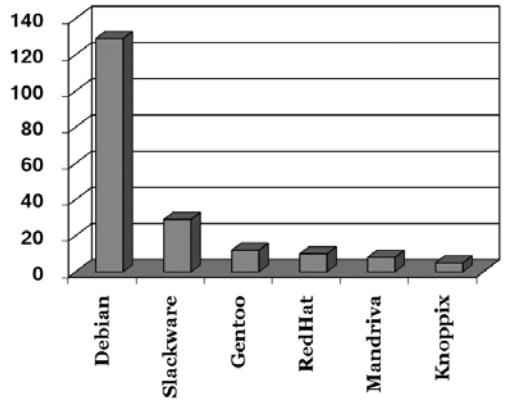

Figura 1.26 – Distribuições baseadas em outras distribuições. Pesquisa realizada em 05 abr. 2011. Fonte: DistroWatch (`http://distrowatch.com`).

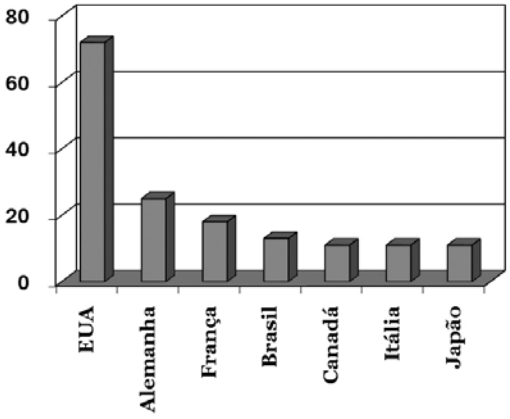

Figura 1.27 – Países que mais desenvolvem distribuições GNU/Linux. Pesquisa realizada em 05 abr. 2011. Fonte: DistroWatch (`http://distrowatch.com`).

Cada distribuição GNU/Linux possui um foco, um objetivo. Cabe a cada um escolher a que melhor se ajuste às suas necessidades.

1.8 Linux International

A Linux International é uma instituição sem fins lucrativos, sediada nos EUA, que visa à expansão do Linux e da comunidade Linux. A LI, como é chamada, reúne vários pequenos grupos e faz um trabalho, de nível mundial, que pode ser acompanhado em `http://www.li.org`. O seu presidente, desde 1996, é Jon "Maddog" Hall. Maddog, que pode ser visto na Figura 1.28, percorre o mundo ministrando palestras para divulgar o Linux e colher resultados da utilização do sistema. Maddog trabalhou nas seguintes organizações: Western Electric Corporation, Aetna Life and Casualty, Laboratórios Bell, Digital Equipment Corporation, VA Linux Systems e SGI.

Figura 1.28 – Jon "Maddog" Hall no FISL6.

A Linux International mantém um contador de usuários e máquinas em `http://counter.li.org`. Ao cadastrar-se, você receberá um número e um cartão virtual, que pode ser visto na Figura 1.29.

Figura 1.29 – Cartão virtual da counter.li.org.

1.9 Tux

À semelhança do gnu do Projeto GNU, o Kernel Linux possui um logotipo, cujo nome é Tux (fusão de Torvalds com Unix). O Tux pode ser visto na Figura 1.30.

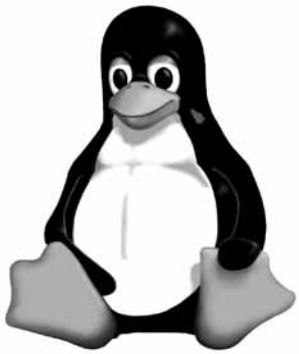

Figura 1.30 – Tux, o símbolo do Kernel Linux.
Fonte: LinuxTage (`http://www.linuxtage.at/presse`)

O Tux foi criado a partir de um concurso sugerido no grupo de discussão do kernel.

Em determinado momento, Linus Torvalds entrou na conversa e sugeriu que fosse um pinguim um pouco gordo, mas não extremamente obeso, que transmitisse uma imagem de tranquilidade e satisfação. Deveria estar sentado, sorridente e satisfeito, digerindo com felicidade um almoço, o que não o deixaria com vontade de estar em pé. A mensagem original, escrita por Linus Torvalds em 9 de maio de 1996, pode ser vista em `http://www.ussg.iu.edu/hypermail/linux/kernel/9605.1/0109.html`.

Com a intervenção de Torvalds, o concurso terminou perdendo sentido. Assim sendo, em meados de 1996, Larry Ewing (Figura 1.31), estudante de engenharia elétrica da Texas A&M University (TAMU), apresentou um logotipo da forma como Torvalds queria. Esse logotipo, conhecido como Tux, foi homologado e utilizado para simbolizar o Kernel Linux. Larry desenhou o Tux no GIMP, um famoso Software Livre para o tratamento de imagens.

O GIMP e o GNU/Linux já foram utilizados em muitos filmes famosos, como Titanic. Veja detalhes em `http://digitalcontentproducer.com/dcc/revfeat/video_linux_hollywood`.

Figura 1.31 – Larry Ewing, o criador do Tux.
Fonte: Linux Expo 5.0 (`http://www.crynwr.com/~nelson/linuxexpo`).

Cabe ressaltar mais uma vez que o Tux é o símbolo do Kernel Linux. O sistema operacional GNU/Linux não possui símbolo. Isso mostra, definitivamente, a diferença entre o Linux e o GNU/Linux.

1.10 Linux Foundation

O Linux Foundation é uma organização sem fins lucrativos, originário da união de duas antigas organizações, o Free Standards Group (FSG) e o Open Source Development Labs (OSDL). O seu objetivo maior é, ao mesmo tempo, promover, proteger, e padronizar o Kernel Linux. Possui nomes de peso no topo da sua organização, como Linux Torvalds e Theodore Ts'o.

Em março de 2012, o Linux Foundation possuía os seguintes grupos de trabalho: consumer eletronics, accessibility, carrier grade, Linux RAID, LSB (Linux Standard Base), networking, OpenPrinting, OSAPA (Open Source As Prior Art), SPDX (Software Package Data Exchange) e Workgroups FAQ.

As explicações sobre as atividades desenvolvidas por cada grupo podem ser obtidas por intermédio dos links existentes em `http://www.linuxfoundation.org/collaborate/workgroups`.

Várias empresas e desenvolvedores seguem as normas da Linux Foundation. Alguns nomes de peso: Adobe, AMD, Cisco, Dell, Epson, Fujitsu, Google, Hitachi, HP, Huawei, IBM, Intel, Lexmark, LG, Motorola, NEC, Nokia, Nvidia, Oracle, Panasonic, Samsung, Sony, Sun, Toyota e Yahoo.

Há uma relação completa em `http://www.linuxfoundation.org/about/members`. Esta é uma informação interessante e que vale ser observada.

1.11 Linux Standard Base (LSB)

O Linux Standard Base é um grupo do Linux Foundation que possui os seguintes objetivos:

- Desenvolver e divulgar padrões que permitam que qualquer programa feito para GNU/Linux funcione em qualquer distribuição GNU/Linux, diminuindo custos para a portabilidade de códigos.
- Ajudar a coordenar os esforços da Linux Foundation, no sentido de convencer fabricantes e desenvolvedores de software a escrever ou portar programas para GNU/Linux.

O LSB desenvolveu uma norma chamada LSB Specification, com a intenção de padronizar procedimentos. O site da LSB é `http://www.linuxfoundation.org/collaborate/workgroups/lsb`. No mesmo site é possível acessar as últimas edições da norma.

1.12 OpenI18n

Inicialmente, para evitar confusões, cabe esclarecer que o nome é "open i dezoito ene".

O OpenI18n ou Open Internationalization Initiative é uma tentativa de internacionalizar o Unix e seus derivados, como o GNU/Linux, e os seus diversos aplicativos. A internacionalização se dá pela preparação de um programa para poder receber adaptações referentes à língua, à localização, ao formato de data, à moeda etc. Geralmente, as referidas adaptações são efetivamente realizadas pelo processo denominado L10n.

O nome I18n é obtido da seguinte forma: a palavra Internationalization possui 20 caracteres. Retirando-se o primeiro caractere (I) e o último (n), sobram 18 caracteres.

O site do OpenI18n é `http://www.openi18n.org`.

1.13 L10n

O L10n é um esforço complementar ao I18n. O movimento L10n, que é a abreviatura de Localization, executa, efetivamente, alguma internacionalização ou regionalização. Apenas como exemplo, enquanto o I18n prepara um determinado programa para ser internacionalizado, o L10n faz com que todos os textos desse programa apareçam em português do Brasil e com que o fuso horário seja ajustado corretamente, dentre outras possibilidades possíveis.

1.14 Filesystem Hierarchy Standard (FHS)

O FHS é uma padronização da hierarquia de diretórios dos Unix e derivados. Pretende-se, com isso, criar um padrão para que todos os diretórios e arquivos importantes fiquem em locais fixos. Mais uma vez, o objetivo principal é a interoperabilidade entre os Unix e derivados.

A norma FHS está disponível em `http://www.pathname.com/fhs` e será analisada no Capítulo 6.

1.15 Conclusão

Um longo caminho histórico, repleto de antecedentes, foi traçado até chegarmos ao Kernel Linux. É muito importante entender todo esse caminho. Muitos fatos que conhecemos hoje em dia foram causados por episódios antigos. A maior lição que se extrai de tudo isso é a capacidade do homem. A vontade de fazer suplanta obstáculos. Ken Thompson aprendeu isso. Linus Torvalds também. Sem falar em Richard Stallman, que conseguiu difundir uma filosofia pelo mundo inteiro.

Uma consideração importante ao final deste primeiro capítulo é o fato de que o Linux é apenas um kernel. Ele foi criado por Linus Torvalds, mas os aplicativos utilizados, desde o começo, foram feitos pela FSF para o projeto GNU. Assim sendo, o nome correto do sistema operacional é GNU/Linux. Erroneamente, ou por falta de conhecimento ou por comodismo, as pessoas utilizam o termo Linux para referenciar algo bem mais amplo do que o Linux realmente é. Mas que fique bem claro: Linux é somente o kernel. E o Tux é o símbolo do kernel, somente.

Este capítulo foi quase todo baseado em fontes oficiais. Podemos citar os sites do Bell Labs e do Linux. Nada foi retirado de sites que não demonstravam integridade. Todas as informações extraídas de sites não oficiais foram checadas em, pelo menos, três fontes diferentes. Muitas pessoas envolvidas na história do Unix e do Linux foram contactadas e a maioria demonstrou interesse em ajudar. O trabalho de pesquisa durou cerca de seis meses (referência à 1ª edição deste livro, sem considerar as atualizações em edições subsequentes).

Apesar de já tê-lo feito no início deste livro, gostaria de registrar, novamente, os meus agradecimentos às pessoas, sem as quais este capítulo não teria passado de suposições e lendas: Dennis Ritchie, Theodore Ts'o, Andrew Tanenbaum e Richard Stallman. Todos ajudaram, pessoalmente ou por e-mail, nos anos de 2005 e 2006, a levantar os fatos como eles realmente ocorreram. Na 3ª edição do livro, uma homenagem especial a Dennis Ritchie, que faleceu em 12 de outubro de 2011, e a Ken Thompson. A dedicatória no início da edição havia sido escrita bem antes de Ritchie nos deixar. A página pessoal de Dennis pode ser encontrada em `http://cm.bell-labs.com/who/dmr`. A Figura 1.32 mostra uma das fotos públicas mais recentes de Dennis Ritchie, antes do seu falecimento aos 70 anos de idade.

Figura 1.32 – Dennis Ritchie em maio de 2011 ao receber um prêmio no Japão.
Fonte: theguardian (`http://www.guardian.co.uk/technology/2011/oct/13/dennis-ritchie`).