Capítulo 1

Introdução aos sistemas operacionais

Não viva mais de maneira fragmentada. Conecte-se...

Edward Morgan Forster

Eficiência é fazer o trabalho corretamente. Eficácia é fazer o trabalho que deve ser feito.

Zig Ziglar

Nada permanece, senão a mudança.

Heráclito

Abre-te, sésamo!

A História de Ali Babá

Objetivos

Este capítulo apresenta:

- *O que é um sistema operacional.*
- *Um breve histórico sobre os sistemas operacionais.*
- *Um breve histórico sobre a Internet e a World Wide Web.*
- *Os componentes centrais do sistema operacional.*
- *As metas dos sistemas operacionais.*
- *Arquiteturas do sistema operacional.*

1.1 Introdução

Bem-vindo ao mundo dos sistemas operacionais. Durante as últimas décadas a computação se desenvolveu a uma velocidade sem precedentes. A capacidade do computador continua a aumentar a taxas fenomenais, enquanto o custo declina extraordinariamente. Hoje, os usuários de computadores têm estações de trabalho de mesa que executam bilhões de instruções por segundo (BIPS), e supercomputadores que executam mais de um trilhão de instruções por segundo já foram construídos,[1, 2] números que há alguns anos eram inconcebíveis.

Processadores estão ficando tão baratos e poderosos que os computadores podem ser empregados em quase todas as situações de nossas vidas. Com computadores pessoais, podemos editar documentos, jogar, ouvir música, assistir a vídeos e gerenciar nossas finanças. Equipamentos portáteis, incluindo laptops, assistentes pessoais digitais (PDAs), telefones celulares e aparelhos MP3, em todos eles os computadores são os componentes principais. Arquiteturas de rede com e sem fio estão aumentando nossa interconectividade, permitindo que usuários se comuniquem instantaneamente a distâncias imensas. A Internet e a World Wide Web revolucionaram os negócios criando demanda para redes de computadores de grande porte e capacidade que processam números enormes de transações por segundo. As redes de computadores se tornaram tão poderosas que são usadas para executar pesquisas complexas e projetos de simulação, como modelar o clima da Terra, emular a inteligência humana e construir animações em 3D que parecem reais. Essa capacidade de computação poderosa e onipresente está modificando os papéis e responsabilidades dos sistemas operacionais.

Neste livro revemos os princípios dos sistemas operacionais, discutimos os últimos avanços na computação que estão redefinindo os sistemas operacionais e investigamos a estrutura e as responsabilidades dos sistemas operacionais. Considerações de projeto como desempenho, tolerância a falhas, segurança, modularidade e custo são exploradas detalhadamente. Também abordamos questões mais recentes do projeto de sistemas operacionais que surgem do rápido crescimento da computação distribuída possibilitada pela Internet e pela World Wide Web.

Dedicamos muito esforço para criar o que esperamos ser uma experiência informativa, interessante e desafiadora para você. Ao ler este livro, você talvez queira consultar o nosso site em www.deitel.com para atualizações e informações adicionais sobre cada tópico. Se quiser falar conosco, envie um e-mail para deitel@deitel.com.

1.2 O que é um sistema operacional?

Na década de 1960 a definição de um sistema operacional como *o software que controla o hardware* estava de acordo com aquela realidade, contudo, desde então, o panorama dos sistemas de computador evoluiu significativamente, exigindo uma descrição mais rica.

Hoje o hardware executa uma grande variedade de aplicações de software. Para aumentar a utilização do hardware, as aplicações são projetadas para ser executadas concorrentemente. Se elas não forem cuidadosamente programadas poderão interferir umas nas outras. Isso resultou na existência de uma camada de software, denominada **sistema operacional,** que separa as aplicações do hardware que elas acessam e fornece serviços que permitem que cada aplicação seja executada com segurança e efetivamente.

Sistema operacional é um software que habilita as aplicações a interagir com o hardware de um computador. O software que contém os componentes centrais do sistema operacional é denominado **núcleo**. Sistemas operacionais podem ser encontrados em dispositivos que vão de telefones celulares e automóveis a computadores pessoais e computadores de grande porte (mainframes). Na maioria dos sistemas operacionais um usuário requisita ao computador que realize uma ação (por exemplo, executar uma aplicação ou imprimir um documento), e o sistema operacional gerencia o software e o hardware para produzir o resultado desejado.

Para grande parte dos usuários, o sistema operacional é uma 'caixa-preta' entre as aplicações e o hardware sobre o qual funcionam e que assegura o resultado correto dadas as entradas adequadas. Sistemas operacionais são, primordialmente, gerenciadores de recursos — gerenciam hardware como processadores, memória, dispositivos de entrada/saída e dispositivos de comunicação. Também devem gerenciar aplicações e outras abstrações de software que, diferentemente do hardware, não são objetos físicos.

Nas seções seguintes apresentamos uma breve história dos sistemas operacionais desde o sistema simples, em lote (batch), de um único usuário da década de 1950, até as plataformas complexas de hoje: multiprocessadoras, distribuídas e multiusuário.

Revisão

1. (V/F) Sistemas operacionais somente gerenciam hardware.

2. Quais são as finalidades primárias de um sistema operacional?

Respostas:

1) Falso. Sistemas operacionais gerenciam aplicações e outras abstrações de software, como máquinas virtuais. **2)** As finalidades primárias de um sistema operacional são habilitar aplicações a interagir com um hardware de computador e gerenciar os recursos de hardware e software de um sistema.

1.3 O começo da história: décadas de 1940 e 1950

Sistemas operacionais evoluíram nos últimos 60 anos, passando por diversas fases ou gerações distintas que correspondem aproximadamente às décadas (veja o quadro "Reflexões sobre Sistemas Operacionais, Inovação").[3] Na década de 1940, os primeiros computadores digitais eletrônicos não tinham sistemas operacionais.[4, 5, 6] As máquinas daquela época eram tão primitivas que os programadores muitas vezes submetiam seus programas a linguagem de máquina, um bit por vez, em filas de chaves mecânicas. Com a introdução das perfuradoras de cartão, esses mesmos programas foram submetidos a cartões perfurados. Assim, linguagens de montagem (assembly) — que usavam abreviaturas parecidas com palavras em inglês para representar as operações básicas do computador — foram desenvolvidas para acelerar o processo de programação.

Os Laboratórios de Pesquisa da General Motors implementaram o primeiro sistema operacional no início da década de 50 para seu computador IBM 701.[7] Sistemas da década de 1950 geralmente executavam somente um job por vez, por meio de técnicas que tornavam mais suave a transição entre jobs para obter uma utilização máxima do sistema do computador.[8] O **job** (serviço) era composto de um conjunto de instruções de programas correspondentes a uma tarefa computacional particular, como folha de pagamento ou estoque. Jobs normalmente eram executados sem entradas do usuário durante minutos, horas ou dias. Esses primeiros computadores eram denominados **sistemas de processamento em lote de fluxo único** porque programas e dados eram submetidos em grupos ou lotes carregando-os consecutivamente em uma fita ou disco. Um processador de fluxo de job lia os comandos em linguagem de controle (que definiam cada job) e facilitava a configuração do job seguinte. Quando o job em questão terminava, a leitora de fluxo de job lia os comandos em linguagem de controle para o próximo job e executava as tarefas de preparo apropriadas para facilitar a transição para o job seguinte. Embora os sistemas operacionais da década de 1950 reduzissem os tempos de transição entre jobs, era muito comum solicitar aos programadores que controlassem diretamente recursos do sistema como memória e dispositivos de entrada/saída, o que se mostrava um trabalho lento, difícil e tedioso. E mais, os primeiros sistemas exigiam que um programa inteiro fosse carregado na memória para poder ser executado, limitando os programadores a criar programas pequenos com capacidades reduzidas.[9]

Revisão

1. Por que foram desenvolvidas linguagens de montagem (assembly)?
2. O que limitava tamanho e capacidades dos programas da década de 50?

Respostas:

1) Linguagens de montagem foram desenvolvidas para acelerar o processo de programação. Habilitavam os programadores a especificar comandos sob a forma de abreviações parecidas com palavras em inglês utilizadas pelos seres humanos para trabalhar com maior facilidade do que com instruções em linguagem de máquina. **2)** O programa inteiro tinha de ser carregado na memória para ser executado. Como era relativamente cara, a quantidade de memória disponível naqueles computadores era pequena.

Reflexões sobre sistemas operacionais

Inovação

Inovação é um desafio fundamental para projetistas de sistemas operacionais. Caso queiramos fazer os investimentos maciços exigidos para produzir novos sistemas operacionais ou novas versões de sistemas operacionais existentes, devemos avaliar constantemente novas tecnologias, novas aplicações de computação e comunicações e novos modos de pensar sobre como os sistemas devem ser construídos. Fornecemos milhares de referências e centenas de recursos da Web para que você possa ler mais sobre os tópicos de seu interesse. Você deve pensar em se associar a organizações profissionais como a ACM (**www.acm.org**), o IEEE (**www.ieee.org**) e o USENIX (**www.usenix.org**), que publicam periódicos sobre as últimas pesquisas e esforços de desenvolvimento no campo da computação. Deve também acessar a Web freqüentemente para se colocar a par de desenvolvimentos importantes na área. Há sempre um alto grau de risco na inovação, mas as recompensas podem ser substanciais.

1.4 A década de 1960

Os sistemas da década de 1960 também eram sistemas de processamento em lote, mas usavam os recursos do computador mais eficientemente executando diversos jobs ao mesmo tempo. Entre esses sistemas estavam muitos dispositivos periféricos como leitoras de cartão, perfuradoras de cartão, impressoras, unidades de fita e unidades de disco. Um job qualquer raramente utilizava todos os recursos do sistema eficientemente. Um job típico usaria o processador durante um certo período de tempo antes de executar uma operação de entrada/saída (E/S) em um dos dispositivos periféricos do sistema. Nesse ponto o processador ficava ocioso enquanto o job esperava a operação de E/S terminar.

Os sistemas da década de 1960 melhoraram a utilização de recursos permitindo que um job usasse o processador, enquanto outros utilizavam os dispositivos periféricos. Na verdade, executar uma mistura de diferentes jobs — alguns que utilizavam principalmente o processador (denominados **jobs orientados a processador** ou **jobs orientados à computação**) e outros que usavam principalmente dispositivos periféricos (denominados **jobs orientados a E/S**)— parecia ser a melhor maneira de otimizar a utilização de recursos. Com essas observações em mente, projetistas de sistemas operacionais desenvolveram sistemas de **multiprogramação** que gerenciavam diversos jobs ao mesmo tempo.[10, 11, 12] Em um ambiente de multiprogramação, o sistema operacional comuta rapidamente o processador de job em job mantendo vários deles em andamento e, ao mesmo tempo, os dispositivos periféricos também em uso. O **grau de multiprogramação** de um sistema (também chamado de **nível de multiprogramação**) indica quantos jobs podem ser gerenciados ao mesmo tempo. Assim, os sistemas operacionais evoluíram do gerenciamento de um job para o gerenciamento de diversos jobs ao mesmo tempo.

Em sistemas de computação multiprogramados, o compartilhamento de recursos é uma das metas principais. Quando recursos são compartilhados por um conjunto de processos, e cada processo mantém controle exclusivo sobre certos recursos a ele alocados, um determinado processo pode ficar esperando por um recurso que nunca se tornará disponível. Se isso ocorrer, esse processo não conseguirá terminar sua tarefa e o usuário talvez tenha de reiniciá-lo, perdendo todo o trabalho que o processo tinha executado até aquele ponto. No Capítulo 7, "Deadlock e adiamento indefinido", discutiremos como os sistemas operacionais podem lidar com tais problemas.

Normalmente os usuários da década de 1960 não estavam presentes no local onde o computador estava instalado quando seus jobs eram executados. Esses eram submetidos em cartões perfurados ou fitas de computador e permaneciam em mesas de entrada até que o operador do sistema (um ser humano) pudesse carregá-los no computador para execução. Muitas vezes o job de um usuário ficava parado durante horas ou até dias antes de poder ser processado. O menor erro em um programa, até mesmo uma vírgula ou um ponto que faltasse, 'bombardeava' o job, quando então o usuário (muitas vezes frustrado) corrigiria o erro, submeteria novamente o job e mais uma vez esperaria por horas ou dias pela próxima tentativa de execução. O desenvolvimento de softwares naquele ambiente era penosamente lento.

Em 1964, a IBM anunciou sua família de computadores System/360 ('360' refere-se a todos os pontos existentes em uma bússola para exprimir sua aplicabilidade universal).[13, 14, 15, 16] Os vários modelos de computadores 360 foram projetados para ser compatíveis com o hardware, para usar o sistema operacional OS/360 e para oferecer maior capacidade de computação à medida que o usuário ia subindo na série.[17] Com o passar dos anos, a IBM desenvolveu sua arquitetura 360 passando para a série 370[18, 19] e, mais recentemente, para a série 390[20] e para a zSeries.[21]

Foram desenvolvidos sistemas operacionais mais avançados para atender a diversos **usuários interativos** ao mesmo tempo. Usuários interativos comunicam-se com seus jobs durante a execução. Na década de 1960, os usuários interagiam com o computador via 'terminais burros' (dispositivos que forneciam uma interface de usuário, mas nenhuma capacidade de processamento) **on-line** (ligados diretamente ao computador via conexão ativa). Como o usuário estava presente e interagindo com o sistema do computador, esse precisava responder rapidamente às solicitações daquele; não fosse assim, sua produtividade poderia ser prejudicada. Como discutiremos no quadro "Reflexões sobre sistemas operacionais, Valor relativo de recursos humanos e de computador", a melhoria da produtividade tornou-se um objetivo importante para os computadores, porque recursos humanos são extremamente caros em comparação aos de computador. Foram desenvolvidos sistemas de **tempo compartilhado** para apoiar usuários interativos simultâneos.[22]

Muitos dos sistemas de tempo compartilhado da década de 1960 eram multimodais que suportavam processamento em lote, bem como aplicações de tempo real (sistemas de controle de processo industrial).[23] **Sistemas de tempo real** tentam dar uma resposta dentro de um certo prazo limitado. Por exemplo, uma medição realizada em uma refinaria de petróleo indicando que as temperaturas estão muito altas exige atenção imediata para evitar uma explosão. É comum que os recursos de um sistema de tempo real sejam extremamente subutilizados — é mais importante que tais sistemas dêem uma resposta rápida do que usem seus recursos eficientemente. Executar tanto jobs em lote quanto de tempo real significava que os sistemas operacionais tinham de distinguir entre tipos de usuários e fornecer a cada um deles um nível apropriado de serviço. Jobs de processamento em lote podiam sofrer atrasos razoáveis, enquanto aplicações interativas demandavam um nível mais alto de serviço e sistemas de tempo real exigiam níveis de serviços extremamente altos.

Entre os mais importantes esforços de desenvolvimento de sistemas de tempo compartilhado desse período estão o **CTTS** (*Compatible Time-Sharing System*, sistema compatível de tempo compartilhado)[24, 25] desenvolvido pelo MIT, o **TSS** (*Time Sharing System*, sistema de tempo compartilhado)[26] desenvolvido pela IBM, o sistema **Multics**[27] desenvolvido

pelo MIT, GE e Bell Laboratories para ser o sucessor do CTTS, e o **CP/CMS** (*Control Program/Conversational Monitor System*, programa de controle/sistema monitor conversacional) que eventualmente evoluiu para o sistema operacional **VM** (*Virtual Machine*, máquina virtual) da IBM, desenvolvido pelo Cambridge Scientific Center da IBM.[28, 29] Esses sistemas foram projetados para executar tarefas básicas de computação interativa para indivíduos, mas seu real valor provou ser a maneira como compartilhavam programas e dados e demonstravam a importância da computação interativa em ambientes de desenvolvimento de programas.

Os projetistas do sistema Multics foram os primeiros a usar o termo **processo** para descrever um programa em execução no contexto dos sistemas operacionais. Em muitos casos, os usuários submetiam jobs contendo vários processos que podiam ser executados simultaneamente. No Capítulo 3, "Conceito de processos", discutiremos como sistemas operacionais de multiprogramação gerenciam vários processos ao mesmo tempo.

Em geral, processos concorrentes são executados independentemente, mas sistemas de multiprogramação habilitam vários processos a cooperar para executar uma tarefa comum. No Capítulo 5, "Execução assíncrona concorrente", e no Capítulo 6, "Programação concorrente", discutiremos como os processos coordenam e sincronizam atividades e como os sistemas operacionais suportam essa capacidade. Mostramos muitos exemplos de programas concorrentes, alguns expressos genericamente em pseudocódigo e outros na popular linguagem de programação Java[MR].

O **tempo de retorno** – o tempo entre a submissão de um job e o retorno de seus resultados – foi reduzido para minutos ou até segundos. O programador não precisava mais esperar horas ou dias para corrigir os erros mais simples. Podia entrar com o programa, fazer a compilação, receber uma lista de erros de sintaxe, corrigi-los imediatamente, recompilar e continuar esse ciclo até o programa estar livre de erros de sintaxe. Então ele podia ser executado, depurado, corrigido e concluído com economias de tempo semelhantes.

O valor dos sistemas de tempo compartilhado no suporte do desenvolvimento de programas foi demonstrado quando o MIT, a GE e o Bell Laboratories usaram o sistema CTSS para desenvolver um sucessor próprio, o Multics. Esse programa foi notável por ter sido o primeiro grande sistema operacional escrito primariamente em uma linguagem de alto nível (EPL – estruturada segundo o modelo PL/1 da IBM), e não em linguagem de montagem (assembly). Os projetistas do **UNIX** aprenderam com essa experiência; criaram a linguagem de alto nível **C** especificamente para implementar o UNIX. Uma família de sistemas operacionais baseados no UNIX, entre eles o Linux e o UNIX Berkeley Software Distribution (BSD), evoluiu do sistema original criado por Dennis Ritchie e Ken Thompson no Bell Laboratories no final da década de 1960 (veja no site deste livro: "Biografia — Ken Thompson e Dennis Ritchie").

TSS, Multics e CP/CMS, todos incorporavam **memória virtual**, que discutiremos detalhadamente no Capítulo 10, "Organização da memória virtual", e Capítulo 11, "Gerenciamento de memória virtual". Em sistemas com memória

Reflexões sobre sistemas operacionais

Valor relativo de recursos humanos e de computador

Em 1965, programadores razoavelmente experientes ganhavam cerca de US$ 4 por hora. A taxa de aluguel de tempo de computador em computadores de grande porte (cuja capacidade era muito menor do que a das máquinas de mesa de hoje) era normalmente de US$ 500 ou mais por hora — e isso em dólares de 1965 que, devido à inflação, seriam comparáveis a milhares de dólares em moeda atual! Hoje, você pode comprar um computador de mesa topo de linha, de enorme capacidade, pelo custo do aluguel de uma hora de computador de grande porte, de capacidade bem menor, 40 anos atrás! Enquanto o custo de computação despencou, o custo de homem-hora subiu até um ponto que, atualmente, os recursos humanos são muito, mas muito mais caros do que os recursos de computação.

Hardware, sistemas operacionais e programas de aplicação para computadores são todos projetados para alavancar o tempo das pessoas, para melhorar eficiência e produtividade. Um exemplo clássico disso foi o advento de sistemas de tempo compartilhado na década de 1960. Esses sistemas interativos (com tempos de resposta quase imediatos) muitas vezes habilitavam os programadores a se tornarem muito mais produtivos do que era possível com os tempos de resposta de sistemas de processamento em lotes que chegavam a horas e até dias. Um outro exemplo clássico foi a criação da interface gráfica com o usuário (GUI) originalmente desenvolvida pelo Palo Alto Research Center (PARC) da Xerox na década de 1970. Com recursos de computação mais baratos e poderosos, e com o custo relativo de homem-hora subindo rapidamente em comparação ao de computação, os projetistas de sistemas operacionais devem fornecer capacidades que favoreçam o ser humano sobre a máquina, exatamente o oposto do que faziam os primeiros sistemas operacionais.

virtual, os programas conseguem endereçar mais localizações de memória do que as realmente providas na memória principal, também chamada memória real ou memória física.[30, 31] (Memória real é discutida no Capítulo 9, "Organização e gerenciamento da memória real"). Sistemas de memória virtual ajudam a livrar os programadores de grande parte da carga do gerenciamento da memória, liberando-os para se concentrarem no desenvolvimento de aplicações.

Desde que carregados na memória principal, os programas podiam ser executados rapidamente; todavia, a memória principal era demasiadamente cara para conter grandes números de programas de uma vez. Antes da década de 1960, os jobs eram, em grande parte, carregados na memória usando cartões perfurados ou fitas, um trabalho tedioso e demorado durante o qual o sistema não podia ser utilizado para executar jobs. Os sistemas da década de 1960 incorporavam dispositivos que reduziam o tempo ocioso, armazenando grandes quantidades de dados regraváveis em meios magnéticos de armazenamento relativamente baratos como fitas, discos e tambores. Embora discos rígidos permitissem acesso relativamente rápido a programas e dados em comparação com fitas, eram significativamente mais lentos do que a memória principal. No Capítulo 12, "Otimização do desempenho de discos", discutimos como sistemas operacionais podem gerenciar requisições de entrada/saída do disco para melhorar o desempenho. No Capítulo 13, "Sistemas de arquivos e de bancos de dados", discutimos como os sistemas operacionais organizam dados em coleções chamadas arquivos e gerenciam espaço em dispositivos de armazenamento como discos. Discutimos também como os sistemas operacionais protegem dados contra o acesso por usuários não autorizados e evitam que se percam quando ocorrem falhas de sistema ou outros eventos catastróficos.

Revisão

1. Como a computação interativa e a melhoria que causou no tempo de retorno aumentaram a produtividade do programador?

2. Quais foram os novos conceitos incorporados pelo TTS, Multics e CP/CMS? Por que foram tão úteis para os programadores?

Respostas:

1) O tempo entre a submissão de um job e o retorno de seus resultados foi reduzido de horas ou dias para minutos ou até segundos, o que habilitava os programadores a entrar, editar e compilar programas interativamente até eliminar seus erros de sintaxe e, assim, usar um ciclo similar para testar e depurar seus programas. **2)** TSS, Multics e CP/CMS incorporavam memória virtual. A memória virtual permite às aplicações acesso a mais memória do que estaria fisicamente disponível no sistema, o que, por sua vez, habilita os programadores a desenvolver aplicações maiores, mais poderosas. Além disso, sistemas de memória virtual liberam o programador de grande parte da carga do gerenciamento da memória.

1.5 A década de 1970

Os sistemas da década de 1970 eram primordialmente multimodais de multiprogramação que suportavam processamento em lote, tempo compartilhado e aplicações de tempo real. A computação estava em seus estágios incipientes, favorecida por desenvolvimentos anteriores e contínuos da tecnologia de multiprocessadores.[32] Os sistemas experimentais de tempo compartilhado da década de 1960 evoluíram para produtos comerciais sólidos na década de 1970. Comunicações entre sistemas de computadores por todos os Estados Unidos aumentaram quando os padrões de comunicações TCP/IP do Departamento de Defesa passaram a ser amplamente usados — especialmente em ambientes de computação militares e universitários.[33, 34, 35] A comunicação em redes locais (LANs) ficou prática e econômica com o padrão Ethernet desenvolvido no Palo Alto Research Center (PARC) da Xerox.[36, 37] No Capítulo 16, "Introdução às redes", discutiremos TCP/IP, Ethernet e conceitos fundamentais de redes.

Os problemas de segurança aumentaram com o crescimento dos volumes de informação transmitidos por linhas de comunicações vulneráveis (veja no site deste livro: "Curiosidades — Abraham Lincoln e o Cuidado com a Tecnologia"). A criptografia foi alvo de muita atenção — tornou-se necessário criptografar dados proprietários ou privados de modo que, mesmo que corressem riscos, não teriam nenhum valor senão para aqueles a quem eram destinados. No Capítulo 19, "Segurança", discutiremos como os sistemas operacionais protegem informações vulneráveis contra acesso não autorizado. Durante a década de 1970 os sistemas operacionais passaram a incluir capacidades de redes e de segurança, e seu desempenho continuou a melhorar para atender às demandas comerciais.

A revolução da computação pessoal começou no final da década de 1970 com sistemas como o Apple II, e explodiu na década de 1980.

Revisão

1. Quais foram os desenvolvimentos da década de 1970 que melhoraram a comunicação entre sistemas de computadores?

2. Qual foi o novo problema introduzido pelo crescimento da comunicação entre computadores? Como esse problema foi abordado?

Respostas:

1) Os padrões TCP/IP do DoD (Departamento de Defesa dos Estados Unidos) passaram a ser amplamente usados em redes de comunicações — inicialmente em ambientes de computação militares e universitários. E, também, o PARC, da Xerox, desenvolveu o padrão Ethernet, que tornava práticas e econômicas as redes locais (LANs) de velocidade relativamente altas. **2)** A comunicação entre computadores introduziu problemas de segurança, porque os dados eram enviados por linhas de comunicação vulneráveis. Empregou-se a criptografia para que os dados ficassem ilegíveis para todos, exceto para quem deveria recebê-los.

1.6 A década de 1980

Os anos 80 representaram a década do computador pessoal e da estação de trabalho.[38] A tecnologia do microprocessador evoluiu até o ponto em que era possível construir computadores de mesa avançados (denominados estações de trabalho) tão poderosos quanto os de grande porte de uma década atrás. O Personal Computer da IBM lançado em 1982 e o computador pessoal Apple Macintosh, em 1984, possibilitaram que indivíduos e pequenas empresas tivessem seus próprios computadores dedicados. Podiam-se utilizar recursos de comunicação para transmitir dados rápida e economicamente entre sistemas. Em vez de se levar dados a um computador central, de grande porte, para processamento, a computação era distribuída aos lugares onde necessário. Softwares como programas de planilhas de cálculo, editores de texto, pacotes de bancos de dados e pacotes gráficos ajudavam a dar impulso à revolução da computação pessoal, criando demanda entre as empresas que podiam usar esses produtos para aumentar sua produtividade.

Aprender e usar computadores pessoais provou ser algo relativamente fácil, em parte por causa das **interfaces gráficas com o usuário (GUI)** que utilizavam símbolos gráficos como janelas, ícones e menus para facilitar a interação do usuário com os programas. O Palo Alto Research Center (PARC) da Xerox desenvolveu o mouse e a GUI (para mais informações sobre a origem do mouse, consulte o site deste livro: "Biografia — Doug Engelbart"); o lançamento do computador pessoal Macintosh da Apple em 1984 popularizou o seu uso. Nos computadores Macintosh, a GUI estava embutida no sistema operacional, de modo que o aspecto e a sensação de todas as aplicações seriam similares.[39] Uma vez familiarizado com a GUI do Macintosh, o usuário podia aprender a usar as aplicações mais rapidamente.

À medida que os custos da tecnologia declinavam, a transferência de informações entre computadores interconectados em rede tornou-se mais econômica e prática. Proliferavam aplicações de correio eletrônico, transferência de arquivos e acesso a bancos de dados remotos. A **Computação Distribuída** (ou seja, usar vários computadores independentes para desempenhar uma tarefa comum) proliferou com o modelo cliente/servidor. **Clientes** são computadores de usuários que requisitam vários serviços; **servidores** são computadores que executam os serviços requisitados. Muitas vezes os servidores são dedicados a um tipo de tarefa, como renderizar gráficos, gerenciar bancos de dados ou servir páginas Web.

O campo da engenharia de software continuou a evoluir, com um grande impulso partindo do governo dos Estados Unidos que visava ao controle mais rígido dos projetos de software do Departamento de Defesa.[40] Entre as metas da iniciativa estavam a reutilização de códigos e a construção de protótipos para que desenvolvedores e usuários pudessem sugerir modificações desde o início do processo de projeto de software.[41]

Revisão

1. Qual aspecto dos computadores pessoais, popularizado pelo Apple Macintosh, os tornava especialmente fáceis de aprender e usar?

2. (V/F) Um servidor não pode ser um cliente.

Respostas:

1) Interfaces Gráficas com o Usuário (GUIs) facilitavam o uso do computador pessoal proporcionando uma interface uniforme, fácil de usar, para cada aplicação, o que habilitava os usuários a aprender a nova aplicação mais rapidamente. **2)** Falso. Um computador pode ser cliente e servidor. Por exemplo, um servidor Web pode ser ambos, cliente e servidor. Quando usuários requisitam uma página Web, ele é um servidor; se então, como servidor, ele requisitar informações de um sistema de banco de dados, irá se tornar um cliente do sistema de banco de dados.

1.7 A história da Internet e da World Wide Web

No final da década de 60, a **ARPA – Advanced Research Projects Agency** do Departamento de Defesa dos Estados Unidos levou adiante os esquemas para interligar em rede os sistemas centrais de computadores de cerca de uma dúzia de

universidades e instituições de pesquisas apoiadas financeiramente por essa agência. Eles seriam conectados por linhas de comunicações que operavam a uma taxa, então impressionante, de 56 quilobits por segundo (Kbps) — 1 Kbps é igual a mil bits por segundo — em uma época em que a maioria das pessoas (das poucas que podiam) estavam se conectando a computadores por linhas telefônicas a uma taxa de 110 bits por segundo. Harvey M. Deitel (HMD) recorda vividamente a agitação daquela conferência. Pesquisadores de Harvard falavam em comunicar-se com o 'supercomputador' Univac 1108 da Universidade de Utah, do outro lado do país, para processar a massa de cálculos relacionada às suas pesquisas sobre computação gráfica. A pesquisa acadêmica estava prestes a dar um salto gigantesco para a frente. Logo após essa conferência, a ARPA partiu para implementar o que rapidamente ficou conhecido como **ARPAnet —** a avó da **Internet** de hoje.

Embora a ARPAnet habilitasse os pesquisadores a interligar seus computadores em rede, seu principal benefício provou ser a capacidade de comunicação fácil e rápida via o que hoje conhecemos como correio eletrônico (e-mail). E isso vale até mesmo para a Internet atual, com e-mail, mensagem instantânea (instant messaging) e transferência de arquivos, facilitando as comunicações entre centenas de milhões de pessoas no mundo inteiro e crescendo rapidamente.

A ARPAnet foi projetada para operar sem controle centralizado, o que significava que, se uma parte da rede falhasse, as partes que continuavam funcionando ainda poderiam encaminhar pacotes de dados de transmissores a receptores por caminhos alternativos.

Os protocolos (conjunto de regras) para comunicação pela ARPAnet ficaram conhecidos como **Transmission Control Protocol/Internet Protocol (TCP/IP).** O TCP/IP era usado para gerenciar comunicação entre aplicações. Os protocolos garantiam que mensagens fossem encaminhadas (roteadas) adequadamente entre transmissores e receptores e que essas mensagens chegassem intactas. O advento do TCP/IP promoveu o crescimento da computação em todo o mundo. Inicialmente a utilização da Internet limitava-se às universidades e instituições de pesquisa; mais tarde os militares adotaram a tecnologia.

Eventualmente o governo decidiu permitir o acesso à Internet para propósitos comerciais. Essa decisão causou uma certa preocupação entre as comunidades de pesquisa e militares — achava-se que os tempos de resposta seriam prejudicados quando 'a Net' ficasse saturada de usuários. Na verdade, ocorreu o contrário. A comunidade de negócios rapidamente compreendeu que podia usar a Internet para ajustar suas operações e oferecer novos e melhores serviços a seus clientes. Empresas gastaram enormes quantidades de dinheiro para desenvolver e promover sua presença na Internet, o que gerou intensa concorrência entre empresas de telecomunicações, fornecedores de hardware e fornecedores de software para atender à crescente demanda de infra-estrutura. O resultado é que a **largura de banda** (a capacidade de transmissão de informações das linhas de comunicações) da Internet aumentou tremendamente, e os custos de hardware e comunicações despencaram.

A **World Wide Web (WWW)** permite que usuários de computador localizem e vejam documentos multimídia (documentos com texto, gráficos, animação, áudio ou vídeo) sobre praticamente qualquer assunto. Embora a Internet tenha sido desenvolvida há mais de três décadas, a introdução da World Wide Web (WWW) é um evento relativamente recente. Em 1989, Tim Berners-Lee, do CERN (European Center for Nuclear Research), começou a desenvolver uma tecnologia de compartilhamento de informações via documentos de texto interconectados (hyperlinked) (veja no site deste livro: "Biografia — Tim Berners-Lee"). Para implementar essa nova tecnologia, Berners-Lee criou a **HyperText Markup Language (HTML)**. Também implementou o **HyperText Transfer Protocol (HTTP)** para formar a espinha dorsal das comunicações desse novo sistema de informações de hipertexto, que ele batizou de World Wide Web.

Certamente os historiadores colocarão a Internet e a World Wide Web na lista das criações mais importantes e profundas da humanidade. No passado, a maioria das aplicações era executada em computadores 'autônomos' (computadores que não estavam conectados uns aos outros). As aplicações de hoje podem ser escritas para se comunicarem entre as centenas de milhares de computadores do mundo inteiro. A Internet e a World Wide Web fundem tecnologias de computação e de comunicações acelerando e simplificando nosso trabalho. Tornam as informações instantânea e convenientemente acessíveis a um grande número de pessoas. Habilitam indivíduos e pequenas empresas a conseguir exposição mundial. Estão mudando a maneira como fazemos negócios e conduzimos nossas vidas pessoais. E estão mudando nosso modo de pensar sobre a construção de sistemas operacionais. Os sistemas operacionais de hoje oferecem GUIs que habilitam os usuários a 'acessar o mundo' pela Internet e pela Web de maneira transparente, como se estivessem acessando o sistema local. Os sistemas operacionais da década de 1980 preocupavam-se primariamente com gerenciar recursos no computador local. Os sistemas operacionais distribuídos de hoje podem utilizar recursos em computadores do mundo inteiro. Isso cria muitos desafios interessantes que discutiremos por todo o livro, especialmente nos capítulos 16 a 19, que examinam redes, computação distribuída e segurança.

Revisão

1. Qual a diferença entre ARPAnet e as redes de computadores tradicionais? Qual era o seu benefício primário?
2. Quais foram as criações desenvolvidas por Berners-Lee que facilitaram o compartilhamento de dados pela Internet?

Respostas:

1) A ARPAnet era descentralizada, portanto a rede continuava habilitada a passar informações mesmo que partes dela falhassem. O benefício primário da ARPAnet era sua capacidade de comunicação rápida e fácil via e-mail. **2)** Berners-Lee desenvolveu a HTML e o HTTP, que possibilitaram a World Wide Web.

1.8 A década de 1990

O desempenho do hardware continuou a melhorar exponencialmente nos anos 90.[42] No final da década, um computador pessoal típico podia executar diversas centenas de milhões de instruções por segundo (MIPS) e armazenar mais de um gigabyte de informações em um disco rígido; alguns supercomputadores podiam executar mais de um trilhão de operações por segundo.[43] Capacidades de processamento e de armazenamento baratas habilitavam os usuários a executar programas complexos em computadores pessoais e possibilitavam que empresas de pequeno a médio portes usassem essas máquinas econômicas para os serviços (jobs) extensivos de banco de dados e processamento, antes delegados a sistemas de computadores de grande porte. A queda dos custos da tecnologia também levou a um aumento no número de computadores domésticos usados tanto para trabalho quanto para entretenimento.

A criação da World Wide Web na década de 1990 levou a uma explosão na popularidade da computação distribuída. Originalmente, os sistemas operacionais executavam gerenciamento isolado de recursos em um único computador. Com a criação da World Wide Web e as conexões de Internet cada vez mais velozes, a computação distribuída tornou-se trivial nos computadores pessoais. Usuários podiam requisitar dados armazenados em lugares remotos ou requisitar a execução de programas em processadores distantes. Grandes organizações podiam usar multiprocessadores distribuídos (redes de computadores contendo mais de um processador) para aumentar a escala de recursos e melhorar a eficiência.[44] No entanto, aplicações distribuídas ainda estavam limitadas pelo fato de a comunicação pela rede ocorrer em velocidades relativamente lentas comparadas a velocidades de processamento interno de computadores individuais. Computação distribuída será discutida detalhadamente no Capítulo 17, "Introdução a sistemas distribuídos", e Capítulo 18, "Sistemas distribuídos e serviços Web".

À medida que crescia a demanda por conexões com a Internet, o suporte a sistemas operacionais para tarefas de rede tornava-se um padrão. Usuários domésticos e em empresas aumentavam sua produtividade acessando recursos de redes de computadores. Todavia, o aumento da conectividade levou a uma proliferação de ameaças à segurança dos computadores. Projetistas de sistemas operacionais desenvolveram técnicas para proteger computadores contra esses ataques danosos. A crescente sofisticação de tais ameaças à segurança continuou a desafiar a capacidade de contra-ataque da indústria de computadores.

A Microsoft Corporation tornou-se dominante na década de 1990. Em 1981, essa empresa lançou a primeira versão do seu sistema operacional DOS para o computador pessoal da IBM. Em meados da década de 1980, a Microsoft desenvolveu o sistema operacional Windows, uma interface gráfica com o usuário sobreposta ao sistema operacional DOS. A Microsoft lançou o Windows 3.0 em 1990; essa nova versão apresentava uma interface amigável com o usuário e grande funcionalidade. O sistema operacional Windows tornou-se incrivelmente popular após o lançamento em 1993 do Windows 3.1, cujos sucessores Windows 95 e Windows 98 praticamente dominaram o mercado de sistemas operacionais para computadores de mesa no final da década de 1990. Esses sistemas operacionais, que tomavam emprestado muitos conceitos popularizados pelos primeiros sistemas operacionais Macintosh (como ícones, menus e janelas), habilitavam os usuários a executar múltiplas aplicações concorrentes com facilidade. A Microsoft também entrou no mercado de sistemas operacionais corporativos com o lançamento do Windows NT em 1993, que rapidamente tornou-se o sistema operacional preferido para estações de trabalho corporativas.[45] O Windows XP, baseado no sistema operacional Windows NT, é discutido no Capítulo 21, "Estudo de caso: Windows XP".

Tecnologia de objeto

A tecnologia de objeto tornou-se popular em muitas áreas da computação à medida que aumentava constantemente o número de aplicações escritas em linguagens de programação orientadas a objeto como a C++ ou a Java. Conceitos de objeto também facilitavam novas abordagens da computação. Cada objeto de software encapsula um conjunto de atributos e um conjunto de ações, o que permite que as aplicações sejam construídas com componentes que possam ser reutilizados em muitas aplicações, reduzindo o tempo de desenvolvimento de software. Nos **sistemas operacionais orientados a objetos (SOOO)**, objetos representam componentes do sistema operacional e recursos do sistema.[46] Conceitos de orientação a objetos como herança e interfaces foram explorados para criar sistemas operacionais modulares mais fáceis de manter e ampliar do que os construídos com técnicas anteriores. A modularidade facilita o suporte dos sistemas operacionais para arquiteturas novas e diferentes. A demanda por integração de objetos por meio de várias plataformas e linguagens levou ao suporte a objetos em linguagens de programação como a Java da Sun e a .NET da Microsoft (por exemplo, Visual Basic.NET, Visual C++NET e C#).

Movimento do software livre

Um outro movimento da comunidade de computação (particularmente na área de sistemas operacionais) durante a década de 1990 foi o movimento em direção ao software de fonte aberto. A maioria dos softwares é criada escrevendo-se o código-fonte em uma linguagem de programação de alto nível. Contudo, a maioria dos softwares comerciais é vendida como código-objeto (também chamado código de máquina ou binários) – o código-fonte compilado que os computadores podem entender. O código-fonte não é incluído, o que permite aos fabricantes ocultarem informações proprietárias e técnicas de programação. Todavia, softwares de fonte aberto ficaram cada vez mais comuns na década de 1990. Esse tipo de software é distribuído com o código-fonte, permitindo que os indivíduos examinem e modifiquem o software antes de compilá-lo e executá-lo. Por exemplo, o sistema operacional Linux e o servidor Web Apache, ambos softwares de fonte aberto, foram descarregados e instalados por milhões de usuários durante a década de 1990, e o número dessas operações cresce rapidamente no novo milênio.[47] O Linux, criado por Linus Torvalds (veja no site deste livro: "Biografia — Linus Torvald"), é discutido no Capítulo 20, "Estudo de caso: Linux".

Na década de 1980, Richard Stallman, um desenvolvedor de software do MIT, lançou um projeto para recriar e ampliar a maioria das ferramentas do sistema operacional UNIX da AT&T e disponibilizar o código sem nenhum custo. Stallman (veja no site deste livro: "Biografia — Richard Stallman") fundou a Free Software Foundation e criou o projeto **GNU** — uma abreviatura que significa "*G*NU *N*ão é *U*NIX" — porque discordava do conceito de vender permissão para utilizar software.[48] Ele acreditava que dar aos usuários a liberdade de modificar e distribuir software levaria a melhores softwares, orientados pelas necessidades dos usuários, e não pelo lucro pessoal ou corporativo. Quando Linus Torvald criou a versão original do sistema operacional Linux, empregou muitas das ferramentas publicadas gratuitamente pelo GNU sob a **General Public License (GPL)**. A GPL, publicada on-line em www.gnu.org/licenses/gpl.html, especifica que qualquer um pode modificar e redistribuir software livremente sob sua licença, contanto que as modificações sejam claramente indicadas e que qualquer derivado do software também seja distribuído pela GPL.[49] Embora a maioria dos softwares licenciados pela GPL seja gratuita, a GPL requer apenas que seu software seja livre no sentido de que os usuários tenham liberdade para modificar e redistribuir o software livremente. Portanto, fabricantes podem cobrar uma taxa para fornecer software licenciado pela GPL e seu código-fonte, mas não podem impedir que usuários finais os modifiquem e redistribuam. No final da década de 1990 foi fundada a **Open Source Iniciative (OSI)** para proteger o software de fonte aberto e promover os benefícios da programação de código-fonte aberto (ver www.opensource.org).

O software de fonte aberto facilita o aperfeiçoamento de produtos de software por permitir que qualquer um da comunidade dos desenvolvedores teste, depure e aperfeiçoe aplicações, o que aumenta a chance de descobrir e corrigir problemas imperceptíveis que, caso contrário, poderiam caracterizar riscos de segurança ou erros lógicos. Além disso, indivíduos e corporações podem modificar o fonte e criar software personalizado que atenda às necessidades de um ambiente particular. Muitos fabricantes de software livre mantêm sua lucratividade cobrando suporte técnico de indivíduos e corporações e personalizando seus produtos.[50] Embora a maioria dos sistemas da década de 1990 ainda executem sistemas operacionais proprietários — como sistemas operacionais de computadores de grande porte da IBM, sistemas UNIX, Macintosh da Apple e Windows da Microsoft —, sistemas operacionais de fonte aberto como Linux, FreeBSD e OpenBSD tornaram-se concorrentes viáveis. No futuro, eles indubitavelmente continuarão a ganhar terreno sobre as soluções proprietárias como resultado do melhoramento do produto.

Na década de 1990 os sistemas operacionais ficaram cada vez mais amigáveis ao usuário. As capacidades de GUI que a Apple inseriu no seu sistema operacional Macintosh na década de 1980 tornaram-se mais sofisticadas na década de 1990. Capacidades do tipo plug-and-play' (ligar-e-usar) foram embutidas em sistemas operacionais, habilitando os usuários a adicionar e remover componentes de software dinamicamente sem ter de reconfigurar manualmente o sistema operacional. Sistemas operacionais também mantêm perfis de usuários — o que atende às necessidades de autenticação e habilita a personalização da interface do sistema operacional por usuário.

Revisão

1. Como a tecnologia orientada a objeto afetou os sistemas operacionais?
2. Cite alguns dos benefícios do desenvolvimento de software livre.

Respostas:

1) Projetistas de sistemas operacionais podiam reutilizar objetos ao desenvolverem novos componentes. O aumento da modularidade devido à tecnologia orientada a objeto facilitou o suporte do sistema operacional para arquiteturas novas e diferentes. **2)** Software livre pode ser visto e modificado por qualquer um que pertença à comunidade de desenvolvimento. Porque essas pessoas testam, depuram e usam constantemente o software, há maior chance de se descobrir e corrigir erros. Além disso, o software livre habilita usuários e organizações a modificar um programa para atender às suas necessidades particulares.

1.9 2000 e afora

Nesta década o **middleware**, um software que liga duas aplicações (muitas vezes por uma rede), tornou-se vital, à medida que aplicações são publicadas na World Wide Web e os consumidores as usam via conexões de Internet de alta velocidade e preços acessíveis por linhas de televisão a cabo e linhas digitais de assinantes (DSL — Digital Subscriber Lines). O middleware é comum em aplicações Web nas quais um servidor Web — a aplicação que envia dados ao navegador (browser) do usuário — deve gerar conteúdo que satisfaça os requisitos de um usuário com a ajuda de um banco de dados. O middleware age como um *courier* passando mensagens entre o servidor Web e o banco de dados, simplificando a comunicação entre várias arquiteturas diferentes. **Serviços Web** compreendem um conjunto de padrões relacionados que podem habilitar quaisquer duas aplicações de computador a se comunicarem e trocar dados via Internet. Um serviço Web comunica-se com uma rede para fornecer um conjunto específico de operações que outras aplicações podem invocar. Os dados são passados de um lado para outro por meio de protocolos padronizados como o HTTP, o mesmo protocolo usado para transferir páginas Web comuns. Serviços Web operam usando padrões abertos, baseados em texto que habilitam a comunicação entre componentes escritos em linguagens diferentes e sobre plataformas diferentes. São peças de software prontas para o uso na Internet.

Serviços Web ajudarão a dar impulso ao movimento em direção à verdadeira computação distribuída. Por exemplo, a Amazon.com permite que desenvolvedores estabeleçam lojas on-line que pesquisam os bancos de dados de produtos da empresa e mostram informações detalhadas de produto via Amazon.com Web Services (www.amazon.com/gp/aws/landing.html). A máquina de busca Google também pode ser integrada a outras aplicações por meio das APIs Web do Google (www.google.com/apis),habilitando assim o acesso aos bilhões de sites Web fornecidos pelo Google para essas aplicações. Discutiremos serviços Web mais detalhadamente no Capítulo 18, "Sistemas distribuídos e serviços Web".

Multiprocessadores e arquiteturas de rede estão criando numerosas oportunidades de pesquisa e desenvolvimento de novas técnicas de projeto de hardware e software. Linguagens de programação seqüencial que especificam uma computação por vez agora são complementadas por linguagens de programação concorrente como a Java, que habilitam a especificação de computações paralelas; na Java, as unidades de computação paralela são especificadas via threads. Discutiremos threads e a técnica de multithreads no Capítulo 4, "Conceitos de thread".

Um número cada vez maior de sistemas exibem **paralelismo maciço**, ou seja, possuem grandes quantidades de processadores de modo que muitas partes independentes das computações podem ser executadas em paralelo. Esse é um conceito de computação drasticamente diferente da computação seqüencial dos últimos 60 anos; há problemas significativos e desafiadores no desenvolvimento de software apropriado para lidar com tal paralelismo. Discutiremos arquiteturas de computação paralela no Capítulo 15, "Gerenciamento de multiprocessador".

Sistemas operacionais estão padronizando interfaces com o usuário e de aplicação para que se tornem mais fáceis de usar e suportem um número maior de programas. A Microsoft já fundiu as linhas de consumidor e profissional de seu sistema operacional Windows no Windows XP. No seu próximo sistema operacional (de codinome Longhorn) a empresa planeja integrar os formatos de diferentes tipos de arquivos permitindo, por exemplo, que os usuários pesquisem todos os arquivos dos seus sistemas (documentos, planilhas de cálculo, e-mails etc.) que contenham certas palavras-chave. O Longhorn incluirá também uma interface com o usuário aperfeiçoada, em 3D, melhor segurança e suporte para DVDs.[51, 52] Sistemas de fonte aberto como o Linux serão mais amplamente usados e empregarão APIs (*Application Programming Interfaces*), como a **POSIX** (*Portable Operating System Interface*) para melhorar a compatibilidade com outros sistemas operacionais baseados no UNIX.

A computação em dispositivos móveis, como telefones celulares e PDAs, irá se tornar mais comum à medida que for equipada com processadores cada vez mais poderosos. Hoje, tais dispositivos são usados para funções como e-mail, navegação na Web e imagem digital. Aplicações intensivas em recursos como vídeo *full-motion* proliferarão nesses dispositivos. Como os recursos de dispositivos móveis são limitados pelo seu pequeno tamanho, a computação distribuída desempenhará um papel ainda maior, à medida que PDAs e telefones celulares requisitarem quantidades cada vez maiores de dados e de capacidade de processamento de computadores remotos.

Revisão

1. Quais tecnologias podem ser usadas para transpor a lacuna entre sistemas operacionais diferentes? Como essas tecnologias possibilitariam a execução da mesma aplicação em várias plataformas?

2. Por que a computação distribuída é útil para computações executadas por dispositivos móveis?

Respostas:

1) Máquinas virtuais e emuladores de sistemas operacionais transpõem a lacuna entre sistemas operacionais diferentes. Aplicações podem ser escritas uma vez para utilizar a funcionalidade da máquina virtual ou emulador. Máquinas virtuais ou emuladores podem ser implementados para ocultar, das aplicações, a representação da plataforma básica. **2)** Computação

distribuída permite que um dispositivo móvel delegue tarefas a outras máquinas com mais recursos. O dispositivo móvel, que tem recursos e tempo de bateria limitados, pode requisitar dados e capacidade de processamento por uma rede.

1.10 Bases de aplicação

Quando o Personal Computer da IBM (quase sempre chamado de 'PC') surgiu em 1981, imediatamente deu origem a uma imensa indústria de software na qual **fornecedores independentes de software** (*Independent Software Vendors* – ISVs) conseguiam comercializar pacotes para o IBM PC que podiam ser executados no sistema operacional MS-DOS (a versão da IBM era chamada DOS). Sistemas operacionais liberam os desenvolvedores de softwares de aplicação da obrigação de lidar com os detalhes complicados da manipulação do hardware de computador para gerenciar memória, executar entrada/saída, lidar com linhas de comunicação e assim por diante. O sistema operacional oferece uma série de chamadas a **interfaces de programação de aplicativos** (API) que os programadores de aplicações usam para realizar manipulações detalhadas de hardware e outras operações. A API fornece **chamadas ao sistema** pelas quais um programa usuário instrui o sistema operacional a trabalhar; o desenvolvedor de aplicações tem de saber, simplesmente, que rotinas chamar para executar as tarefas específicas (Figura 1.1). Note que, na Figura 1.1, a área acima da linha tracejada, o espaço do usuário, indica componentes de software que não fazem parte do sistema operacional e não podem acessar diretamente os recursos físicos do sistema. A área abaixo da linha tracejada, o espaço do núcleo, indica componentes de software que fazem parte do sistema operacional e têm acesso irrestrito aos recursos do sistema. Usamos essa convenção freqüentemente em nossos diagramas para indicar o privilégio que esses componentes de software gozam na execução. Se uma aplicação tentar usar mal os recursos do sistema ou tentar usar recursos que não lhe são permitidos, o sistema operacional deve intervir para evitar que a aplicação danifique o sistema ou interfira em outras aplicações de usuário.

Se um sistema operacional apresentar um ambiente propício ao desenvolvimento rápido e fácil de aplicações, ele e o hardware terão mais probabilidade de sucesso no mercado. O ambiente de desenvolvimento de aplicações criado pelo MS-DOS incentivou o desenvolvimento de dezenas de milhares de pacotes de software de aplicação, o que, por sua vez, incentivou usuários a comprar IBM PCs e compatíveis. O Windows podia muito bem ter uma base de aplicações de cem mil aplicações.

Uma vez que uma **base de aplicação** (isto é, a combinação do hardware com o ambiente do sistema operacional no qual a aplicação é desenvolvida) seja amplamente estabelecida, torna-se extremamente difícil solicitar aos usuários e desenvolvedores de software que convertam para um ambiente de desenvolvimento de aplicações completamente novo proporcionado por um sistema operacional radicalmente diferente. Assim, é provável que as novas arquiteturas que evoluirão nos próximos anos irão colocar todos os esforços no suporte às principais bases de aplicação existentes.

1.11 Ambientes de sistemas operacionais

Este livro focaliza conceitos de sistemas operacionais relacionados a computadores de uso geral com uma gama de recursos, entre eles quantidades consideráveis de memória principal, altas velocidades de processamento, discos de alta capacidade, vários dispositivos periféricos e assim por diante. Tais equipamentos são usados normalmente como computadores pessoais ou estações de trabalho.

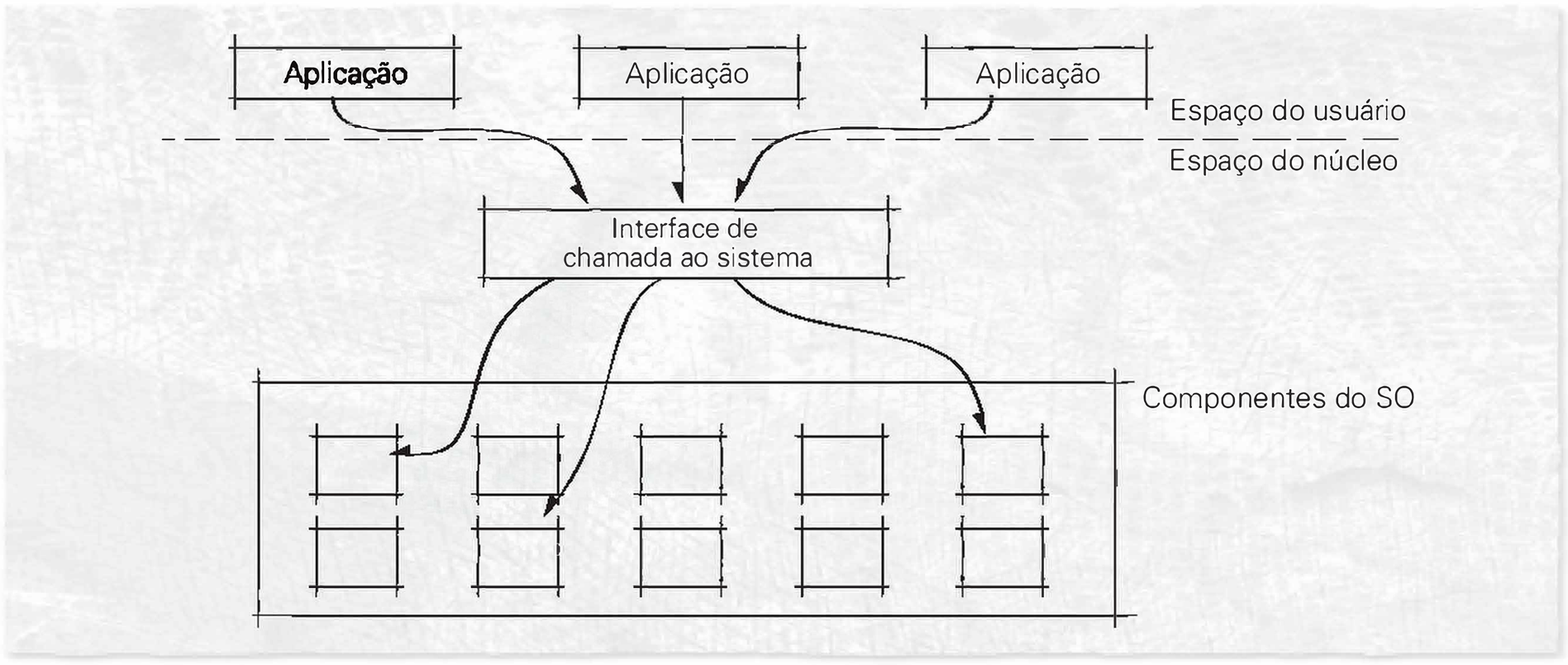

Figura 1.1 | *Interação entre aplicações e o sistema operacional.*

Muitos dos conceitos que se aplicam aos computadores de uso geral também se aplicam aos servidores Web e de bancos de dados mais avançados que contêm hardware de alto desempenho. Sistemas operacionais destinados a ambientes avançados devem ser projetados para suportar memórias principais de grande porte, hardware de uso específico e grandes números de processos. Discutiremos essas considerações no Capítulo 15, "Gerenciamento de multiprocessador".

Sistemas embarcados proporcionam um desafio diferente ao projeto de sistemas operacionais. São caracterizados por um pequeno conjunto de recursos especializados que fornecem funcionalidade a dispositivos como telefones celulares e PDAs. Em ambientes embarcados, o gerenciamento eficiente de recursos é a chave para a construção de um sistema operacional de sucesso. A capacidade de armazenamento freqüentemente é limitada, portanto o sistema operacional deve fornecer serviços usando uma quantidade mínima de código. Considerações como gerenciamento de energia (bateria) e a necessidade de interfaces amigáveis com o usuário criam outros desafios para o projeto de sistemas operacionais embarcados.

Sistemas de tempo real exigem que tarefas sejam realizadas em um período de tempo particular (quase sempre curto). Por exemplo, o recurso de piloto automático de uma aeronave deve ajustar constantemente velocidade, altitude e direção. Essas ações não podem esperar indefinidamente — e às vezes não podem esperar de jeito nenhum — que outras tarefas não essenciais sejam concluídas. Sistemas operacionais de tempo real devem assegurar que processos respondam imediatamente a eventos críticos. Sistemas operacionais de tempo real não crítico garantem que tarefas de tempo real sejam executadas com alta prioridade, mas não garantem quais dessas tarefas serão concluídas a tempo, e nem se serão concluídas. Sistemas de tempo real crítico garantem que todas as suas tarefas sejam concluídas a tempo. Discutiremos como o Linux e o Windows XP lidam com aplicações de tempo real nos capítulos 20 e 21, respectivamente. Esses sistemas são encontrados em muitos ambientes, incluindo robótica, aviação e outras aplicações de controle de sistema. Muitas vezes são usados em **sistemas de missão crítica** nos quais o sistema não cumpre seus objetivos (ou seja, a missão) se qualquer de suas tarefas não for concluída com sucesso e a tempo. Em sistemas de missão crítica como os de controle de tráfego aéreo, monitoração de reatores nucleares e comando e controle militar, vidas humanas podem correr riscos.

Sistemas críticos de negócios, como servidores Web e bancos de dados, devem atingir seus objetivos consistentemente. No e-business isso pode significar a garantia de tempos de resposta rápidos a usuários que estão comprando produtos pela Internet; em grandes corporações, pode significar habilitar funcionários a compartilhar informações eficientemente e garantir que informações importantes estejam protegidas de problemas como falta de energia elétrica e falhas de discos. Diferentemente dos sistemas de missão crítica, a empresa não fracassará necessariamente se um sistema crítico de negócios não atingir sempre seus objetivos.

Alguns sistemas operacionais devem gerenciar hardware que podem ou não existir fisicamente na máquina. Uma **máquina virtual (VM)** é uma abstração em software de um computador executado freqüentemente como uma aplicação de usuário sobre o sistema operacional nativo.[53] O sistema operacional de uma máquina virtual gerencia os recursos providos pela máquina virtual. Uma das aplicações das máquinas virtuais é permitir que várias instâncias de um sistema operacional sejam executadas simultaneamente. Outra é a emulação – usar software e hardware que imitam a funcionalidade de hardware ou software que não estão presentes no sistema.

Máquinas virtuais fazem interface com o hardware de um sistema via sistema operacional; outros programas de usuário podem interagir com elas. Uma VM pode criar componentes de software que representem componentes físicos — como processadores, memória, canais de comunicação, discos e relógios (Figura 1.2)[54] —, o que permite que vários usuários compartilhem hardware na ilusão de que estão sendo atendidos por uma máquina dedicada. Proporcionando essa ilusão, as máquinas virtuais promovem a **portabilidade**, a capacidade do software ser executado em várias plataformas.

A **JVM** (*Java Virtual Machine*, máquina virtual Java) é uma das mais amplamente usadas. A JVM é a fundação da plataforma Java e permite que as aplicações Java sejam executadas em qualquer JVM da versão correta, independentemente da plataforma na qual ela esteja instalada. A empresa VMware Software também fornece máquinas virtuais, particularmente para a arquitetura Intel, habilitando os proprietários de computadores baseados no Intel x-86 a executar sistemas operacionais como Linux e Windows simultaneamente em um só computador (cada máquina virtual aparece na sua própria janela).[55]

Máquinas virtuais tendem a ser menos eficientes do que máquinas reais porque acessam o hardware indiretamente (ou simulam hardware que, na verdade, não está conectado ao computador). Acesso indireto ou simulado ao hardware aumenta o número de instruções de software requeridas para realizar cada ação do hardware.[56]

Revisão

1. Um monitor de temperatura em uma usina nuclear seria provavelmente descrito como que tipo de sistema? Por quê?

2. Descreva as vantagens e desvantagens das máquinas virtuais.

Respostas:

1) Um sistema de tempo real crítico monitoraria a temperatura em uma usina nuclear para garantir que ela estivesse sempre na faixa apropriada e notificaria os operadores em tempo real (ou seja, instantaneamente), caso houvesse problemas.

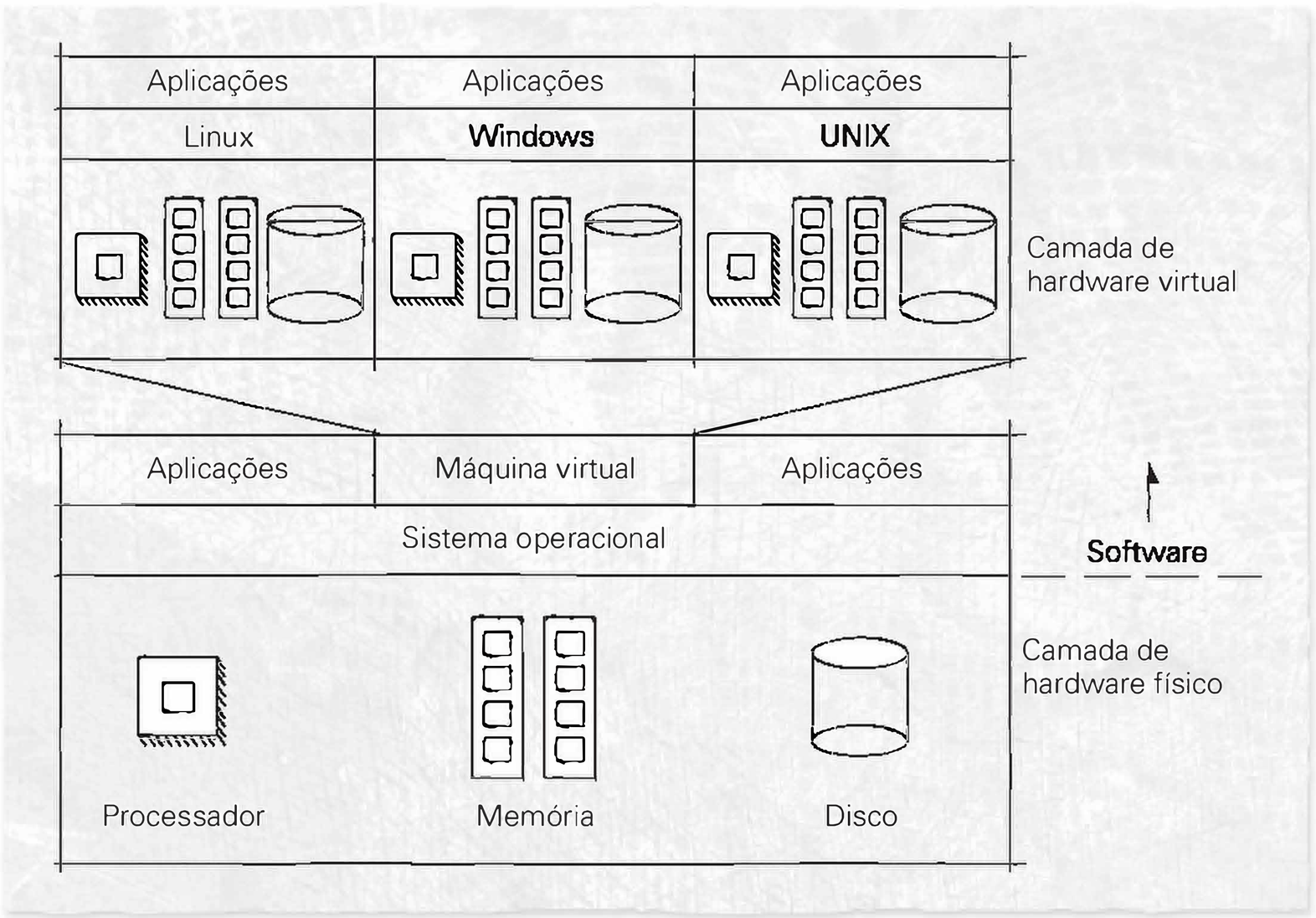

Figura 1.2 | *Diagrama de uma máquina virtual.*

2) Máquinas virtuais promovem a portabilidade habilitando o software a ser executado em várias plataformas, mas tendem a ser menos eficientes do que máquinas reais, porque devem executar instruções de software que simulem operações de hardware.

1.12 Componentes e objetivos do sistema operacional

Sistemas de computadores evoluíram dos primeiros que não continham nenhum sistema operacional para máquinas de multiprogramação, daí para máquinas de tempo compartilhado, depois para computadores pessoais e, por fim, para sistemas verdadeiramente distribuídos. À medida que a demanda por novas características e melhor eficiência cresciam e o hardware mudava, os sistemas operacionais evoluíram para cumprir novos papéis. Esta seção descreve vários componentes centrais e explica os diversos objetivos dos sistemas operacionais.

1.12.1 Componentes centrais do sistema operacional

Um usuário interage com o sistema operacional via uma ou mais aplicações de usuário e, muitas vezes, por meio de uma aplicação especial denominada **shell** ou interpretador de comandos.[57] A maioria dos interpretadores de comando atuais é implementada como interfaces de texto que habilitam o usuário a emitir comandos por meio de um teclado, ou como GUIs que permitem que o usuário 'aponte-e-clique' e 'arraste-e-solte' ícones para requisitar serviços do sistema operacional (por exemplo, para abrir uma aplicação). O Microsoft Windows XP fornece uma GUI por meio da qual os usuários podem dar comandos; o usuário pode abrir alternativamente uma janela de comandos que aceite comandos digitados.

O software que contém os componentes centrais do sistema operacional chama-se núcleo (kernel). Entre os componentes centrais do sistema operacional estão:

- **escalonador de processo**, que determina quando e por quanto tempo um processo é executado em um processador.
- **gerenciador de memória**, que determina quando e como a memória é alocada aos processos e o que fazer quando a memória principal estiver cheia.
- **gerenciador de E/S**, que atende às solicitações de entrada/saída de e para dispositivos de hardware, respectivamente.
- **gerenciador de comunicação interprocessos (IPC)**, que permite que os processos se comuniquem uns com os outros.
- **gerenciador de sistema de arquivos**, que organiza coleções nomeadas de dados em dispositivos de armazenamento e fornece uma interface para acessar os dados nesses dispositivos.

Quase todos os sistemas operacionais suportam um ambiente de multiprogramação no qual várias aplicações podem ser executadas concorrentemente. Uma das responsabilidades mais fundamentais de um sistema operacional é determinar qual processador executa um processo e durante quanto tempo esse processo é executado.

Um programa pode conter diversos elementos que compartilhem dados e que possam ser executados concorrentemente. Por exemplo, um navegador Web pode conter componentes isolados para ler a HTML de uma página Web, recuperar a mídia da página (ou seja, imagens, texto e vídeo) e exibir a página apresentando seu conteúdo na janela do navegador. Esses componentes de programa, executados independentemente, mas que realizam seu trabalho em um espaço de memória comum, são chamados **threads** (fluxos de execução). Tais componentes são discutidos no Capítulo 4, "Conceitos de thread".

Normalmente muitos processos competem para usar o processador. O escalonador de processos pode basear suas decisões em diversos critérios, como a importância de um processo, o tempo estimado de execução ou há quanto tempo está esperando para obter acesso ao processador. Discutiremos escalonamento do processador no Capítulo 8, "Escalonamento de processador".

O gerenciador de memória aloca memória para o sistema operacional e para os processos. Com o intuito de garantir que os processos não interfiram no sistema operacional ou uns nos outros, o gerenciador de memória impede que cada processador acesse memória que não lhe tenha sido alocada. Quase todos os sistemas operacionais de hoje suportam memória virtual, como discutido nos capítulos 10 e 11.

Uma outra função central do sistema operacional é gerenciar os dispositivos de entrada/saída (E/S) do computador. Dispositivos de entrada abrangem teclados, mouses, microfones e scanners; entre os dispositivos de saída estão monitores, impressoras e alto-falantes. Dispositivos de armazenamento (por exemplo, discos rígidos, discos óticos regraváveis e fitas) e placas de rede funcionam como dispositivos de entrada e saída. Quando um processo quer acessar um dispositivo de E/S, deve emitir uma chamada ao sistema operacional. Aquela chamada é subseqüentemente manuseada por um **driver de dispositivo**, que é um componente de software que interage diretamente com o hardware e em geral contém comandos e outras instruções específicas do dispositivo para realizar as operações de entrada/saída requisitadas.

A maioria dos sistemas de computador pode armazenar dados persistentemente (isto é, após o computador ter sido desligado). Como a memória principal geralmente é relativamente pequena e perde seus dados quando a fonte de energia é desligada, são usados dispositivos secundários de armazenamento persistente, mais comumente discos rígidos. E/S por disco — uma das formas mais comuns de E/S — ocorre quando um processo requisita acesso a informações que estão em um dispositivo de disco.

Entretanto, o armazenamento secundário é muito mais lento do que processadores e memória principal. O componente **escalonador de disco** de um sistema operacional é responsável pela reordenação das requisições de E/S por disco para maximizar o desempenho e minimizar a quantidade de tempo que um processo espera pelas E/S por disco. Sistemas de arranjo redundante de discos independentes (*Redundant Array of Independent Disks* – RAID) tentam reduzir o tempo que um processo espera pela E/S por disco, utilizando vários discos ao mesmo tempo para atender às requisições de E/S. Discutiremos algoritmos de escalonamento de discos e sistemas RAID no Capítulo 12, "Otimização do desempenho de discos".

Sistemas operacionais usam sistemas de arquivo para organizar e acessar eficientemente coleções nomeadas de dados, denominadas arquivos e localizadas em dispositivos de armazenamento. Conceitos de sistemas de arquivo são abordados no Capítulo 13, "Sistemas de arquivos e de bancos de dados".

Com freqüência, os processos (ou threads) cooperam para cumprir uma meta comum. Assim, muitos sistemas operacionais proporcionam comunicação entre processos (IPC) e mecanismos de sincronização para simplificar essas programações concorrentes. Comunicação entre processos habilita os processos a se comunicarem via mensagens enviadas entre eles (e entre threads); sincronização fornece estruturas que podem ser usadas para assegurar que processos (e threads) compartilhem dados adequadamente. Processos e threads são discutidos nos capítulos 3 a 8.

Revisão

1. Quais componentes de sistema operacional realizam cada uma das seguintes operações?

a. Escrever no disco.

b. Determinar qual processo será executado em seguida.

c. Determinar em que lugar da memória um novo processo deve ser colocado.

d. Organizar arquivos em um disco.

e. Habilitar um processo a enviar dados para outro.

2. Por que é arriscado permitir que usuários executem livremente operações de leitura e escrita para qualquer região do disco?

Respostas:

1) a) gerenciador de E/S; b) escalonador de processador; c) gerenciador de memória; d) gerenciador de sistema de arquivo; e) gerenciador de comunicação entre processos (IPC). **2**) É arriscado porque os usuários poderiam, acidentalmente ou com má intenção, sobrescrever dados críticos (por exemplo, arquivos do sistema operacional) ou ler informações vulneráveis (como documentos confidenciais) sem autorização.

1.12.2 Metas do sistema operacional

Usuários passaram a esperar certas características dos sistemas operacionais como:

- eficiência
- robustez
- escalabilidade
- extensibilidade
- portabilidade
- segurança
- interatividade
- usabilidade

Um **sistema operacional eficiente** alcança alto **rendimento** e baixo tempo de retorno. O rendimento mede a quantidade de trabalho que um processador pode concluir em um certo período de tempo. Lembre-se de que um dos papéis de um sistema operacional é fornecer serviços a muitas aplicações. Um sistema operacional eficiente minimiza o tempo gasto oferecendo esses serviços (veja o quadro "Reflexões sobre sistemas operacionais, Desempenho").

Um **sistema operacional robusto** é tolerante a falhas e confiável — o sistema não falhará devido a erros isolados de aplicações ou de hardware e, se falhar, ele o fará graciosamente (isto é, minimizando perda de trabalho e evitando danos ao hardware do sistema). Determinado sistema operacional fornecerá serviços a cada aplicação, a menos que o hardware no qual se confie falhe.

Um **sistema operacional escalável** é capaz de usar recursos à medida que são acrescentados. Se um sistema operacional não for escalável, rapidamente chegará a um ponto em que recursos adicionais não serão utilizados totalmente. Um sistema operacional escalável pode ajustar imediatamente seu grau de multiprogramação. Escalabilidade é um atributo particularmente importante dos sistemas multiprocessadores — à medida que são adicionados processadores ao sistema,

Reflexões sobre sistemas operacionais

Desempenho

Uma das metas mais importantes de um sistema operacional é maximizar o desempenho do sistema. Somos todos conscientes do desempenho em nossas vidas diárias. Medimos a quilometragem por litro de nosso carro, registramos vários recordes de velocidade, professores dão notas a estudantes, funcionários recebem avaliações de desempenho de seus empregadores, o desempenho de um executivo corporativo é medido pelos lucros da empresa, o desempenho dos políticos é medido por freqüentes pesquisas de opinião com seus eleitores e assim por diante.

Alto desempenho é essencial para sistemas operacionais. Todavia, o desempenho muitas vezes está 'nos olhos de quem o vê' — há muitos modos de classificar o desempenho de um sistema operacional. Para sistemas de processamento em lote, o rendimento é uma medida importante; para sistemas interativos de tempo compartilhado, os tempos de resposta rápidos são mais importantes.

Por todo este livro apresentamos muitas técnicas de melhoria de desempenho. Por exemplo, o Capítulo 8, "Escalonamento de processador", discute a alocação de tempo de processador a processos para melhorar o desempenho do sistema medido como interatividade e rendimento. O Capítulo 11, "Gerenciamento de memória virtual", examina alocação de memória a processos para reduzir seus tempos de execução. O Capítulo 12, "Otimização do desempenho do disco", focaliza a melhoria do desempenho do disco pela reordenação das requisições de E/S. No Capítulo 14, "Desempenho e projeto de processador", discutimos a avaliação de sistemas segundo diversos critérios importantes de desempenho. Os capítulos 20 e 21 abordam questões de desempenho nos sistemas operacionais Linux e Windows XP, respectivamente.

idealmente a capacidade de processamento deve crescer proporcionalmente ao número de processos, embora, na prática, isso não aconteça. Multiprocessamento é discutido no Capítulo 15, "Gerenciamento de multiprocessador".

Um **sistema operacional extensível** adapta-se bem a novas tecnologias e fornece capacidades de estender o sistema operacional para executar tarefas que vão além de seu projeto original.

Um **sistema operacional portável** é projetado de modo tal que possa operar em muitas configurações de hardware. Portabilidade de aplicações também é importante porque desenvolver aplicações custa caro; a mesma aplicação deve rodar em uma variedade de configurações de hardware para reduzir custos de desenvolvimento. O sistema operacional é crucial para conseguir esse tipo de portabilidade.

Um **sistema operacional seguro** impede que usuários e software acessem serviços e recursos sem autorização. **Proteção** refere-se aos mecanismos que implementam a política de segurança do sistema.

Um **sistema operacional interativo** permite que aplicações respondam rapidamente às ações do usuário ou a eventos. Um **sistema operacional utilizável** é aquele que tem o potencial de atender a uma base significativa de usuários. Esses sistemas operacionais geralmente fornecem uma interface com o usuário fácil de usar. Sistemas operacionais como Linux, Windows XP e MacOS X são caracterizados como utilizáveis porque cada um suporta um grande conjunto de aplicações e fornece as interfaces-padrão com o usuário. Muitos sistemas operacionais experimentais e acadêmicos não suportam um grande número de aplicações nem fornecem interfaces amigáveis com o usuário e, portanto, não são considerados utilizáveis.

Revisão

1. Quais metas dos sistemas operacionais correspondem a cada uma das seguintes características?
   a. Usuários não podem acessar serviços nem informações sem autorização adequada.
   b. O sistema operacional é executado sobre uma variedade de configurações de hardware.
   c. O desempenho do sistema aumenta continuamente quando acrescentados memória e processadores adicionais.
   d. O sistema operacional suporta dispositivos que não estavam disponíveis na época em que foi projetado.
   e. Falhas de hardware não causam necessariamente falha de sistema.
2. Como o suporte do driver de dispositivo contribui para a extensibilidade de um sistema operacional?

Respostas:

1) a) segurança; b) portabilidade; c) escalabilidade; d) extensibilidade; e) robustez. **2)** Drivers de dispositivo habilitam desenvolvedores a adicionar suporte para hardware que não existia na época em que o sistema foi projetado. A cada novo tipo de dispositivo adicionado a um sistema, deve ser instalado um driver de dispositivo correspondente.

1.13 Arquiteturas de sistemas operacionais

Os sistemas operacionais de hoje tendem a ser complexos, porque prestam muitos serviços e suportam uma variedade de recursos de hardware e software (veja os quadros "Reflexões sobre sistemas operacionais, Mantenha a simplicidade" e "curiosidades"). Arquiteturas de sistemas operacionais podem ajudar projetistas a gerenciar essa complexidade organizando componentes de sistema e especificando o privilégio com que cada componente é executado. No projeto monolítico, todos os componentes do sistema operacional estão no núcleo; no projeto de micronúcleo somente estão incluídos os componentes essenciais. Nas seções seguintes fazemos um apanhado de diversas arquiteturas importantes (veja o quadro "Reflexões sobre sistemas operacionais, Arquitetura").

1.13.1 Arquitetura monolítica

O **sistema operacional monolítico** é a arquitetura de sistema operacional mais antiga e mais comum. Cada componente do sistema operacional é contido no núcleo e pode comunicar-se diretamente com qualquer outro (simplesmente usando chamadas à função). O núcleo normalmente é executado com acesso irrestrito ao sistema de computador (Figura 1.3). OS/360, VMS e Linux são caracterizados, em sentido amplo, como sistemas operacionais monolíticos.[58] A intercomunicação direta entre componentes é que torna os sistemas operacionais monolíticos altamente eficientes. Entretanto, porque os núcleos monolíticos agrupam componentes todos juntos, é difícil isolar a fonte de problemas e de outros erros. Além disso, como todo o código é executado com acesso irrestrito ao sistema, sistemas de núcleo monolítico são particularmente suscetíveis a danos provocados por códigos sujeitos a erros ou mal-intencionados.

Revisão

1. Qual a característica que define um sistema operacional monolítico?

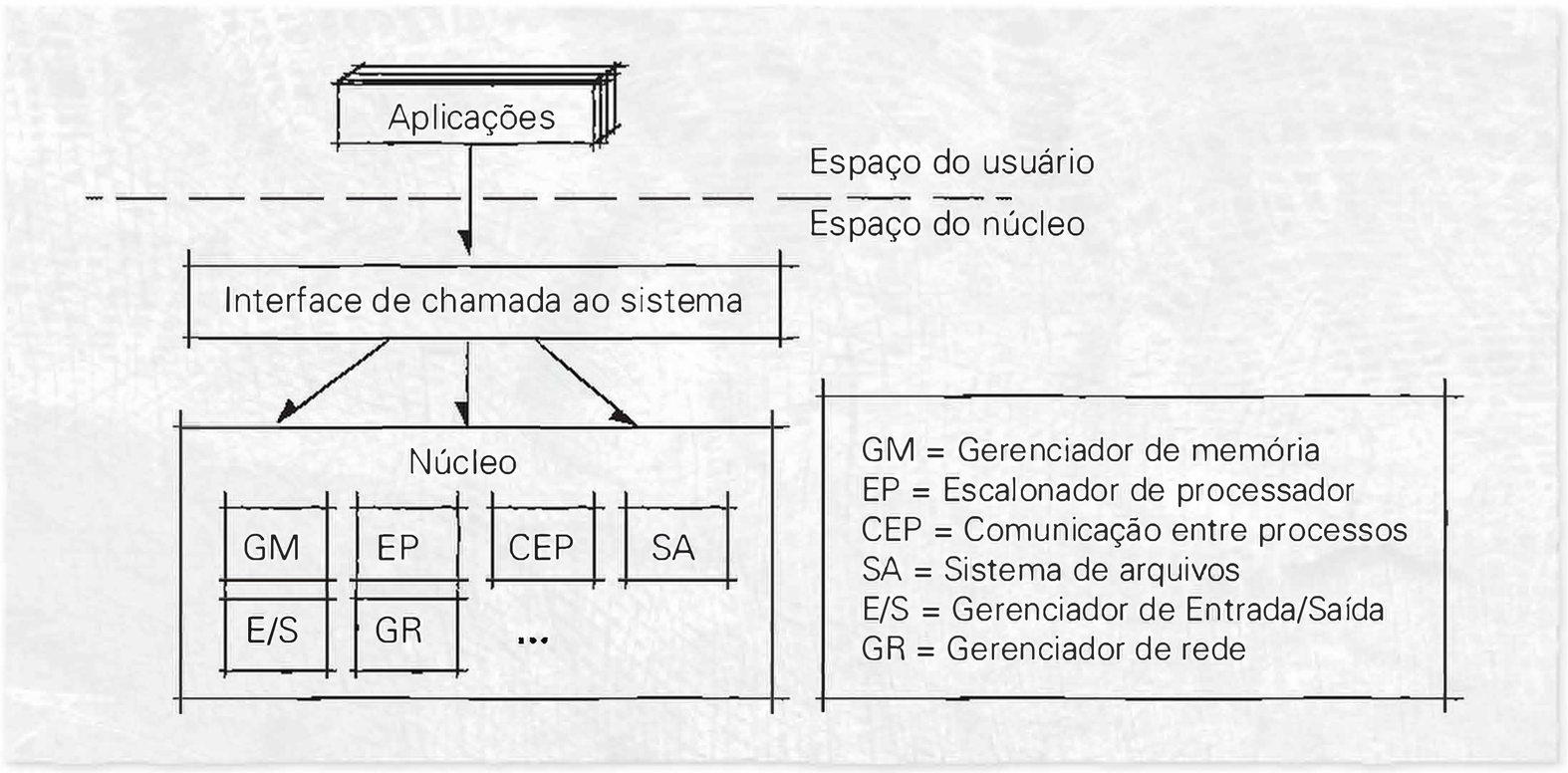

Figura 1.3 | Arquitetura de sistema operacional de núcleo monolítico.

2. Por que sistemas operacionais monolíticos tendem a ser eficientes? Qual a principal fragilidade dos núcleos monolíticos?

Respostas:

1) Em um sistema operacional monolítico todos os componentes do sistema operacional são contidos no núcleo. **2)** Núcleos monolíticos tendem a ser eficientes porque poucas chamadas atravessam do espaço do usuário para o espaço do núcleo. Como todos os códigos de sistemas operacionais em núcleos monolíticos operam com acesso irrestrito ao software e hardware do computador, esses sistemas são particularmente suscetíveis a danos provocados por códigos sujeitos a erros.

1.13.2 Arquitetura em camadas

À medida que os sistemas operacionais tornaram-se maiores e mais complexos, projetos puramente monolíticos mostraram-se intratáveis. A abordagem **em camadas** do sistema operacional tenta resolver essa questão agrupando em camadas componentes que realizam tarefas similares. Cada camada comunica-se exclusivamente com as camadas imediatamente acima e abaixo dela. Camadas de nível mais baixo prestam serviços às de nível mais alto usando uma interface que oculta sua implementação.

Reflexões sobre sistemas operacionais

Mantenha a simplicidade

Projetar, implementar, testar, depurar e manter sistemas complexos é caro. Freqüentemente projetistas de sistemas operacionais escolherão a mais simples de diversas abordagens para resolver determinado problema. Porém, às vezes, uma abordagem mais complexa pode render benefícios de desempenho ou outras melhorias que compensam. Compromissos entre simplicidade e desempenho são comuns na computação. Uma busca linear simples de um array é trivial para programar, mas é executada lentamente em comparação a uma busca binária mais elegante e complicada. Estruturas de árvores podem ser mais complexas para trabalhar do que arrays, mas facilitam e tornam mais velozes a execução de certos tipos de inserção e supressão. Nós sempre consideramos abordagens alternativas para a solução de problemas de sistemas operacionais e desenvolvimento de estratégias de gerenciamento de recursos. À medida que ler essas discussões, você verá os compromissos entre simplicidade e complexidade. Ao ler essas soluções, talvez você opte por certas alternativas. Os sistemas com os quais trabalhará no futuro poderão demandar abordagens diferentes. Nossa filosofia é apresentar os prós e os contras das abordagens populares para ajudá-lo a preparar-se para seus próprios julgamentos quando necessário em sua área de atuação.

Sistemas operacionais em camadas são mais modulares do que os monolíticos, porque a implementação de cada camada pode ser modificada sem exigir nenhuma modificação nas outras. Um sistema modular tem componentes autocontidos que podem ser reutilizados por todo o sistema. Cada componente oculta o modo como realiza sua tarefa e apresenta uma interface-padrão que os outros componentes podem usar para requisitar seus serviços. A modularidade impõe estrutura e consistência ao sistema operacional — muitas vezes simplificando validação, depuração e modificação. Entretanto, em uma abordagem de camadas, a requisição de um processo de usuário pode precisar passar por muitas camadas antes de ser atendida. Como é preciso invocar métodos adicionais para passar dados de uma camada para a seguinte, o desempenho se degrada em comparação ao do núcleo monolítico, que pode requerer apenas uma única chamada para atender a uma requisição similar. Além disso, como todas as camadas têm acesso irrestrito ao sistema, núcleos em camadas também são suscetíveis a danos causados por códigos sujeitos a erros ou mal-intencionados. O sistema operacional THE é um dos primeiros exemplos de um sistema operacional em camadas (Figura1.4).[59] Muitos dos sistemas operacionais atuais, incluindo o Windows XP e o Linux, implementam um certo nível de camadas.

Revisão

1. De que modo os sistemas operacionais em camadas são mais modulares do que os sistemas operacionais monolíticos?

2. Por que sistemas operacionais em camadas tendem a ser menos eficientes do que os monolíticos?

Respostas:

1) Em sistemas operacionais em camadas a implementação e a interface são separadas para cada camada, o que permite que cada uma seja testada e depurada separadamente. Também habilita os projetistas a mudar a implementação de cada camada sem precisar modificar as outras camadas. **2)** Em sistemas operacionais em camadas, podem ser necessárias diversas chamadas para se comunicar entre as camadas, ao passo que essa sobrecarga não existe em núcleos monolíticos.

1.13.3 Arquitetura de micronúcleo

Uma arquitetura de **sistema operacional de micronúcleo** fornece somente um número pequeno de serviços na tentativa de manter o núcleo pequeno e escalável. Entre esses serviços estão, normalmente, gerenciamento de memória de baixo nível, comunicação entre processos e sincronização básica de processos para habilitar a cooperação entre eles. Nos projetos de micronúcleo, a maioria dos componentes do sistema operacional — como gerenciamento de processo, rede, sistemas de arquivo e gerenciamento de dispositivos — é executada fora do núcleo com um nível de privilégio mais baixo (Figura 1.5).[60, 61, 62, 63]

Micronúcleos exibem um alto grau de modularidade, o que os torna extensíveis, portáveis e escaláveis.[64] E mais, como o micronúcleo não depende de cada componente para ser executado, um ou mais dos componentes podem falhar sem causar também a falha do sistema operacional. Entretanto, essa modularidade ocorre à custa de um maior nível de comunicação entre módulos, o que pode degradar o desempenho do sistema. Embora poucos dos sistemas operacionais populares atuais adotem o projeto de micronúcleo, o Linux e o Windows XP, por exemplo, contêm componentes modulares.[65]

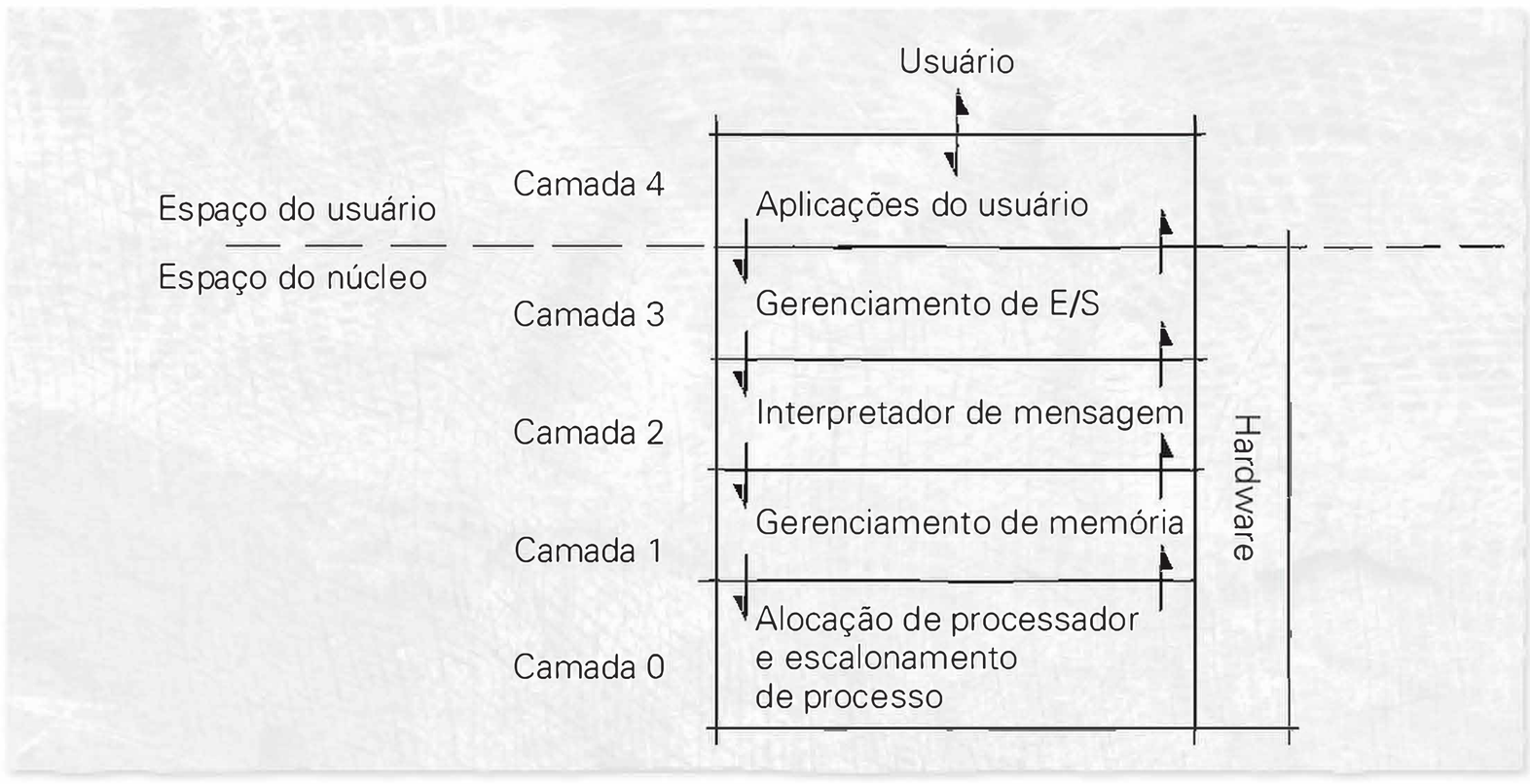

Figura 1.4 | Camadas do sistema operacional THE.

Reflexões sobre sistemas operacionais

Arquitetura

Exatamente como arquitetos usam abordagens diferentes para projetar edifícios, projetistas de sistemas operacionais empregam diferentes abordagens arquitetônicas para projetar sistemas operacionais. Às vezes essas abordagens são puras, no sentido de que somente uma abordagem arquitetônica é usada em todo o sistema. Em outras situações são utilizadas abordagens híbridas, que misturam as vantagens de diversos estilos arquitetônicos. A abordagem que o projetista escolher terá conseqüências monumentais sobre a implementação inicial e a evolução do sistema operacional. Quanto mais se avançar no projeto, mais difícil ficará mudar abordagens, por isso é importante escolher a arquitetura apropriada logo no início do desenvolvimento do sistema. De modo mais geral, é muito mais fácil construir o edifício corretamente da primeira vez do que modificá-lo depois de construído.

Uma das abordagens arquitetônicas mais comuns empregadas em softwares de sistemas como os de sistemas operacionais é chamada 'de camadas'. Esse tipo de software é dividido em módulos denominados camadas e cada uma delas desempenha certas tarefas. Cada camada invoca os serviços prestados pela camada abaixo dela, enquanto a implementação daquela camada fica oculta da camada acima dela. A arquitetura em camadas combina as virtudes das técnicas de engenharia de software de modularidade e de ocultação de informações para fornecer uma base sólida para a construção de sistemas de qualidade. Discutimos a abordagem de camadas do software por todo o livro, começando com uma menção histórica do sistema THE de Dijkstra (veja a Seção 1.13.2, "Arquitetura em camadas") e prosseguindo com explanações sobre como a técnica de camadas é usada no Linux e no Windows XP, nos capítulos 20 e 21, respectivamente.

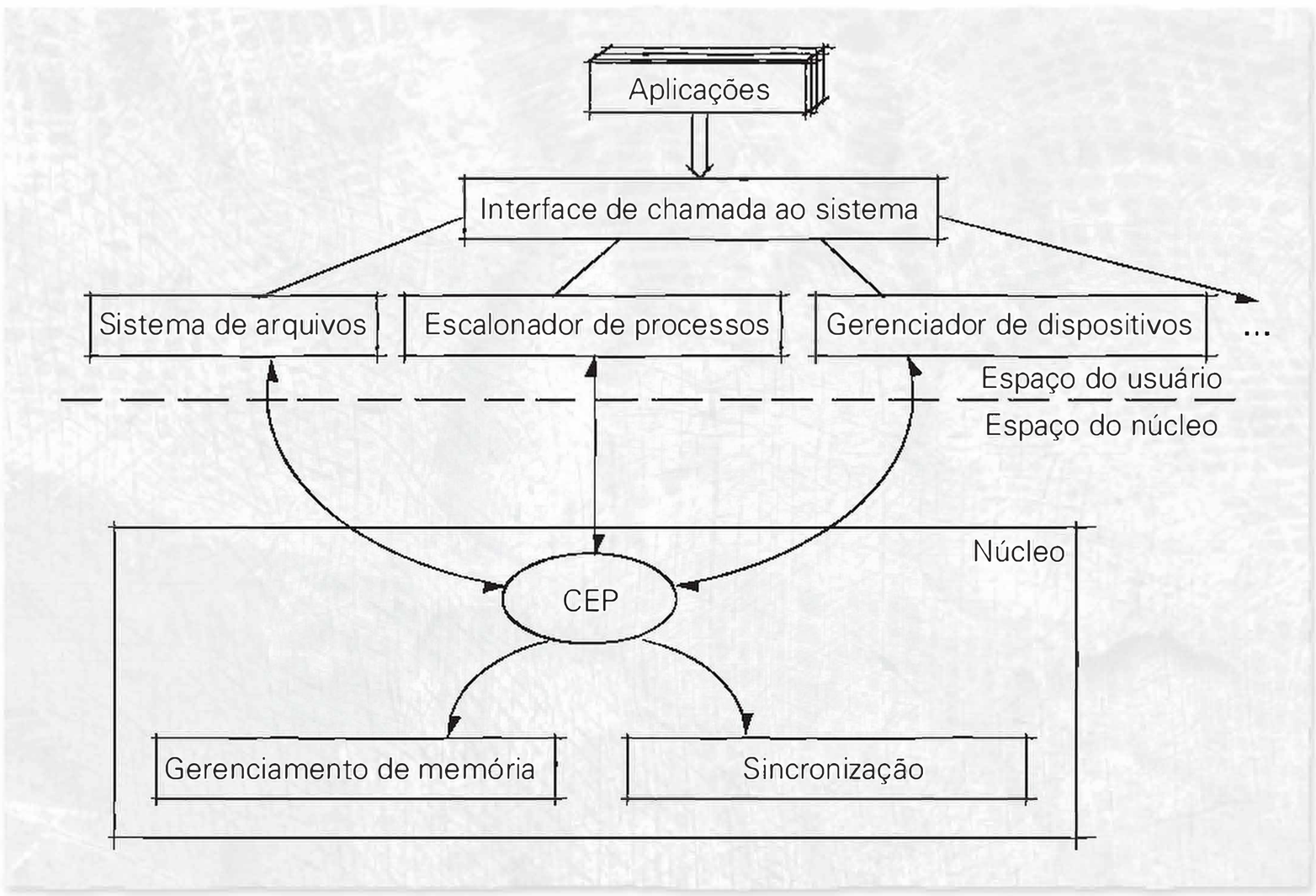

Figura 1.5 | Arquitetura de sistema operacional de micronúcleo.

Revisão

1. Qual a diferença entre uma arquitetura em camadas pura e uma arquitetura de micronúcleo?
2. Como os micronúcleos promovem a portabilidade?

Respostas:

1) Uma arquitetura em camadas habilita comunicação exclusivamente entre componentes de sistemas operacionais de camadas adjacentes. Uma arquitetura de micronúcleo habilita comunicação entre todos os componentes do sistema via micronúcleo. **2)** O micronúcleo não depende de uma plataforma de hardware particular; suporte para novo hardware pode ser fornecido carregando-se um novo módulo.

1.13.4 Sistemas operacionais de rede e distribuídos

Avanços na tecnologia de telecomunicações afetaram profundamente os sistemas operacionais. Um **sistema operacional de rede** habilita seus processos a acessar recursos (por exemplo, arquivos) que residem em outros computadores independentes de uma rede.[66] A estrutura de muitos sistemas operacionais de rede e distribuídos freqüentemente é baseada no modelo cliente/servidor (Figura 1.6). Os computadores-cliente dessa rede requisitam recursos — como arquivos e tempo de processador — via protocolo de rede apropriado. Os servidores respondem com os recursos apropriados. Nessas redes, projetistas de sistemas operacionais devem considerar cuidadosamente como gerenciar dados e comunicação entre computadores.

Alguns sistemas operacionais são mais 'de rede' do que outros. Em um ambiente de rede, um processo pode ser executado no computador no qual foi criado ou em um outro computador da rede. Em alguns sistemas operacionais de rede, usuários podem especificar exatamente onde seus processos são executados; em outros, o sistema operacional é que determina onde os processos são executados. Por exemplo, o sistema pode determinar que um processo pode ser executado mais eficientemente em um computador que esteja momentaneamente com uma carga baixa de processamento.[67]

Os sistemas de arquivos de rede são um componente importante dos sistemas operacionais de rede. No nível mais baixo, usuários adquirem recursos em outra máquina conectando-se explicitamente àquela máquina e recuperando arquivos. Sistemas de arquivos de rede de nível mais alto habilitam os usuários a acessar arquivos remotos como se estes estivessem no sistema local. Entre os exemplos de sistemas de arquivo de rede estão o Network File System (NFS) da Sun e os sistemas de arquivos Andrew e Coda da CMU. Sistemas de arquivos de rede são abordados detalhadamente no Capítulo 18, "Sistemas distribuídos e serviços Web".

UM **sistema operacional distribuído** é um sistema operacional único que gerencia recursos em mais de um sistema de computador. **Sistemas distribuídos** dão a ilusão de que vários computadores compõem um único computador de grande capacidade, de modo que um processo pode acessar todos os recursos do sistema independentemente da localização do processo dentro da rede de computadores do sistema distribuído.[68] Sistemas distribuídos muitas vezes são difíceis de implementar, porque requerem algoritmos complicados para habilitar os processos a se comunicarem e a compartilhar

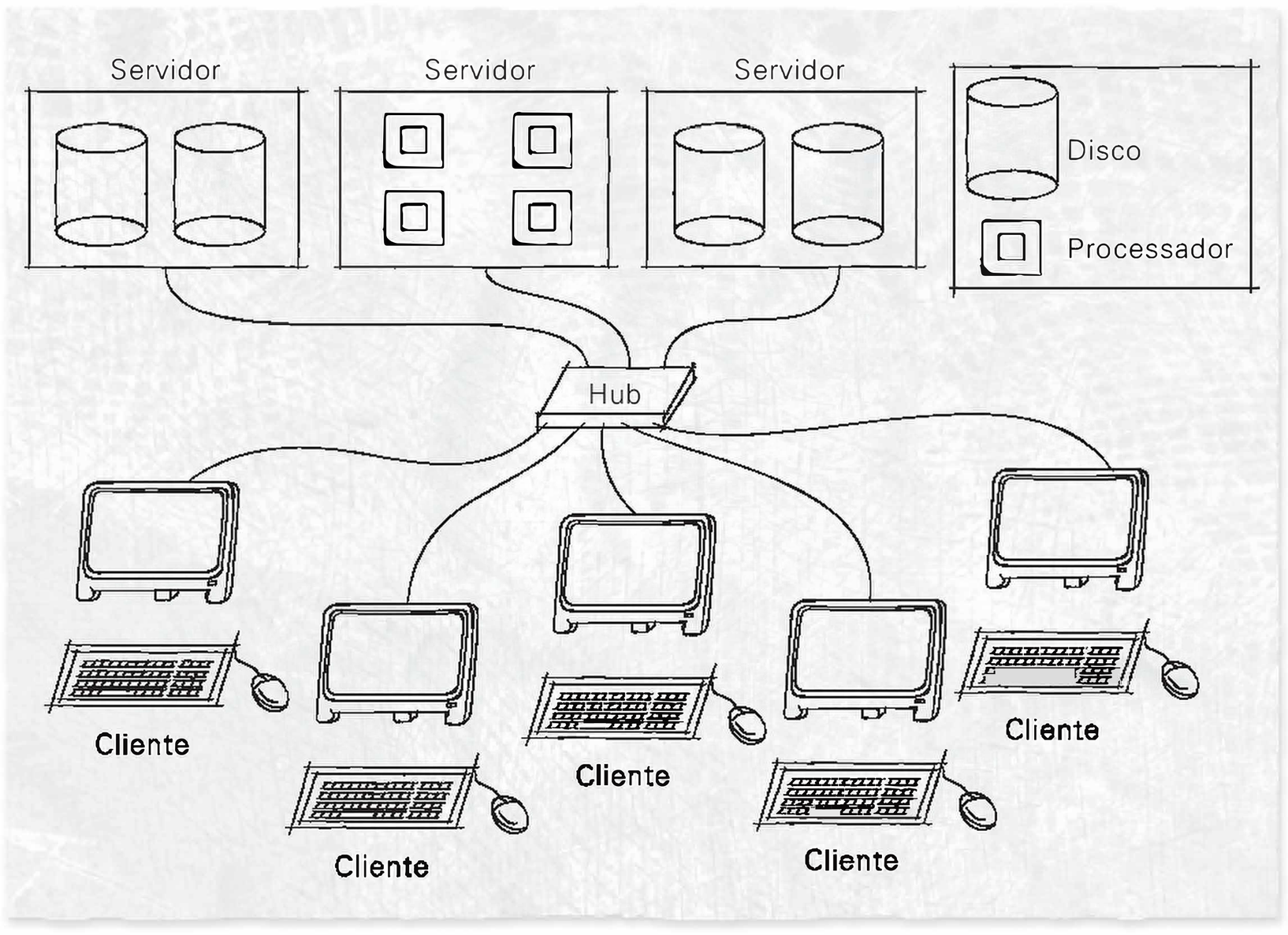

Figura 1.6 | Modelo de sistema operacional de rede cliente/servidor.

dados. Exemplos de sistemas operacionais distribuídos são o Chord do MIT e o Amoeba da Vrije Universiteit (VU) em Amsterdã.[69, 70] Discutiremos sistemas distribuídos no Capítulo 17, "Introdução a sistemas distribuídos".

Agora que apresentamos uma série aparentemente infindável de fatos, questões e acrônimos, passaremos à discussão dos princípios básicos de hardware e software de computadores no Capítulo 2, "Conceitos de hardware e software".

Revisão

1. Qual a principal diferença entre sistemas operacionais de rede e distribuídos?
2. Qual a vantagem primordial de um sistema operacional distribuído?

Respostas:

1) Um sistema operacional de rede controla um computador, mas coopera com outros computadores da rede. Em um sistema operacional distribuído, um sistema operacional controla muitos computadores da rede. **2)** A vantagem primordial é que os processos não precisam conhecer as localizações dos recursos que usam, o que simplifica a programação de aplicações. Isso ocorre à custa do programador de sistemas, que deve implementar algoritmos complicados para habilitar processos a se comunicarem e a compartilhar dados entre muitos computadores, criando a ilusão de que são um único computador de maior porte.

Resumo

Há alguns anos sistema operacional era definido como o software que controla o hardware, mas o panorama dos sistemas de computador evoluiu significativamente desde então, exigindo uma descrição mais complicada. Para aumentar a utilização do hardware, foram projetadas aplicações para ser executadas concorrentemente. Entretanto, se essas aplicações não forem cuidadosamente programadas, poderão interferir umas nas outras. O resultado é uma camada de software denominada sistema operacional que separa as aplicações (a camada de software) do hardware que acessam.

Quando um usuário requisita que o computador realize uma ação (ou seja, que execute uma aplicação ou imprima um documento), o sistema operacional gerencia o software e o hardware para produzirem o resultado desejado. Sistemas operacionais são primordialmente gerenciadores de recursos — gerenciam hardware, incluindo, processadores, memória, dispositivos de entrada/saída e dispositivos de comunicação. O sistema operacional também tem de gerenciar aplicações e outras abstrações de software que, diferentemente do hardware, não são objetos físicos.

Os sistemas operacionais evoluíram nos últimos 60 anos passando por diversas fases ou gerações distintas que correspondem aproximadamente às décadas. Na década de 40, os primeiros computadores eletrônicos digitais não possuíam sistemas operacionais. Os sistemas da década de 50 geralmente executavam apenas um job por vez, mas usavam técnicas que facilitavam a transição entre jobs para obter a máxima utilização do sistema do computador. Um job consistia em um conjunto de instruções que um programa executaria. Esses primeiros computadores eram chamados de sistemas de processamento em lote de fluxo único, porque programas e dados eram submetidos em grupos ou lotes, sendo carregados consecutivamente em fita ou disco.

Os sistemas da década de 1960 também eram de processamento em lote, mas utilizavam os recursos do computador mais eficientemente, executando vários jobs ao mesmo tempo. Os sistemas da década de 1960 melhoraram a utilização de recursos permitindo que um job usasse o processador enquanto outros utilizavam dispositivos periféricos. Com essas observações em mente, projetistas de sistemas operacionais desenvolveram sistemas de multiprogramação que gerenciavam vários jobs ao mesmo tempo, e esse número era indicado pelo grau de multiprogramação do sistema.

Em 1964, a IBM anunciou sua família de computadores System/360. Os vários modelos de computadores 360 foram projetados para ser compatíveis com hardware, usar o sistema operacional OS/360 e oferecer maior capacidade de computação à medida que o usuário progredia em seu uso quanto aos modelos de séries mais avançadas. Foram desenvolvidos sistemas operacionais mais avançados para atender a vários usuários interativos ao mesmo tempo. Desenvolveram-se sistemas de tempo compartilhado para suportar grandes números de usuários interativos simultâneos.

Sistemas de tempo real tentam fornecer uma resposta em um certo período de tempo limitado. Os recursos de um sistema de tempo real são freqüentemente muito subutilizados. Para esses sistemas é mais importante dar uma resposta rápida quando necessário do que usar seus recursos eficientemente.

O tempo de retorno — período entre a apresentação de um job e o retorno de seus resultados — foi reduzido a minutos ou até segundos. O valor dos sistemas de tempo compartilhado no suporte do desenvolvimento de programas foi demonstrado quando o MIT usou o sistema CTSS para desenvolver seu próprio sucessor, o Multics. TSS, Multics e CP/CMS incorporavam memória virtual, que capacita os

programas a acessar mais localizações de memória do que as fornecidas pela memória principal que também é chamada de memória real ou memória física.

Os sistemas da década de 1970 eram primordialmente multimodais de tempo compartilhado que suportavam processamento em lotes, tempo compartilhado e aplicações de tempo real. A computação pessoal estava em seu estágio incipiente, favorecida pelos primeiros desenvolvimentos na tecnologia do microprocessador. As comunicações entre sistemas de computadores por todos os Estados Unidos aumentaram à medida que os padrões de comunicação TCP/IP do Departamento de Defesa tornaram-se amplamente usados — em especial nos ambientes de computação militares e universitários. Problemas de segurança aumentaram à medida que volumes crescentes de informações eram transmitidos por linhas de comunicação vulneráveis.

Os anos 80 representaram a década do computador pessoal e da estação de trabalho. Em vez de os dados serem levados até um computador central de grande porte para processamento, a computação era distribuída aos lugares onde necessário. Computadores pessoais provaram ser relativamente fáceis de aprender e usar, em parte por causa das interfaces gráficas com o usuário (GUI) que usavam símbolos gráficos como janelas, ícones e menus para facilitar a interação do usuário com os programas. À medida que os custos da tecnologia decresciam, a transferência de informações entre computadores e redes de computadores tornava-se mais econômica e prática. O modelo de computação distribuída cliente/servidor disseminou-se amplamente. Clientes são computadores de usuários que requisitam vários serviços; servidores são computadores que executam os serviços requisitados.

A área de engenharia de software continuou a evoluir, com um grande impulso dado pelo governo dos Estados Unidos que visava especialmente a permitir um controle mais rígido dos projetos de software do Departamento de Defesa. Algumas metas da iniciativa eram pôr em prática a reutilização de códigos e um maior grau de abstração das linguagens de programação. Um outro desenvolvimento da engenharia de software foi a implementação de processos contendo múltiplos threads de instrução que seriam executados independentemente.

No final da década de 1960, a ARPA (Advanced Research Projects Agency) colocou em prática a interconexão em rede dos principais sistemas de computador de cerca de uma dúzia de universidades e instituições de pesquisa financiadas pela agência. A ARPA partiu para implementar o que foi apelidado de ARPAnet — a avó da Internet atual. O principal benefício da ARPAnet provou ser sua capacidade de comunicação rápida e fácil pelo que veio a ser conhecido como correio eletrônico (e-mail). Isso vale até mesmo para a Internet de hoje, com e-mail, mensagem instantânea e transferência de arquivos facilitando comunicações entre centenas de milhões de pessoas no mundo inteiro.

A ARPAnet foi projetada para operar sem controle centralizado. Os protocolos (conjunto de regras) para a comunicação pela ARPAnet ficaram conhecidos como Transmission Control Protocol/Internet Protocol (TCP/IP). O TCP/IP era usado para gerenciar a comunicação entre aplicações; protocolos garantiam que mensagens fossem encaminhadas (roteadas) adequadamente entre aplicações e chegassem intactas. Eventualmente o governo decidiu permitir o acesso à Internet para propósitos comerciais.

A World Wide Web (WWW) permite que usuários de computador localizem e vejam documentos multimídia (documentos com texto, gráficos, animações, áudio ou vídeo) sobre praticamente qualquer assunto. Embora a Internet tenha sido desenvolvida há mais de três décadas, a introdução da World Wide Web é um evento relativamente recente. Em 1989, Tim Berners-Lee, do CERN (European Center for Nuclear Research) começou a desenvolver uma tecnologia de compartilhamento de informações via documentos de texto interligados (hyperlinked). Para implementar essa nova tecnologia, ele criou a HyperText Markup Language (HTML). Também implementou o HyperText Transfer Protocol para formar a espinha dorsal das comunicações desse novo sistema de informações de hipertexto que batizou de World Wide Web.

O desempenho do hardware continuou a melhorar exponencialmente na década de 1990. Capacidade de processamento e armazenamento de baixo custo permitiam que usuários executassem programas complexos e grandes em computadores pessoais e habilitavam empresas de pequeno a médio portes a usar essas máquinas econômicas para as tarefas extensivas de banco de dados e processamento que anteriormente eram delegadas a sistemas de grande porte.

Na década de 1990, o movimento em direção à computação distribuída (ou seja, usar vários computadores independentes para executar uma tarefa comum) acelerou-se rapidamente. À medida que crescia a demanda por conexões com a Internet, o suporte de sistemas operacionais para as tarefas de rede tornaram-se padronizados. Usuários em suas casas e em grandes organizações aumentavam a produtividade acessando os recursos de redes de computadores.

A Microsoft Corporation tornou-se dominante na década de 1990. Seus sistemas operacionais Windows, que tomavam emprestado muitos conceitos popularizados pelos primeiros sistemas operacionais Macintosh (como ícones, menus e janelas), habilitavam os usuários a navegar entre múltiplas aplicações concorrentes com facilidade.

A tecnologia de objeto tornou-se popular em muitas áreas da computação. Um grande número de aplicações foi escrito em linguagens de programação orientadas a objeto como a C++ ou a Java. Nos sistemas operacionais orientados a objeto (SOOO), objetos representam componentes do sistema operacional. Foram explorados conceitos orientados a objeto, como herança e interfaces, para criar sistemas operacionais modulares mais fáceis de manter e ampliar do que sistemas operacionais construídos com técnicas anteriores.

A maioria dos softwares comerciais é vendida como código-objeto. O código-fonte não está incluído, habilitando os fornecedores a ocultar informações proprietárias e técnicas de programação. Software livre é distribuído com o código-fonte,

permitindo que os indivíduos tenham liberdade para examinar, executar, copiar, distribuir, estudar, modificar e melhorar o software. O sistema operacional Linux e o servidor Web Apache são ambos livres e de fonte aberto.

Na década de 1980, Richard Stallman, um desenvolvedor de software do MIT, lançou o projeto GNU para recriar e ampliar a maioria das ferramentas do sistema operacional UNIX da AT&T. Stallman criou o projeto GNU porque discordava do conceito de pagar pela permissão para utilizar software. A Open Source Iniciative (OSI) foi fundada para promover os benefícios da programação de fonte aberto. Software de fonte aberto facilita o aperfeiçoamento de produtos de software por permitir que qualquer um da comunidade dos desenvolvedores teste, depure e aperfeiçoe aplicações. Isso aumenta a chance de que sejam descobertos e corrigidos problemas imperceptíveis que, caso contrário, poderiam provocar riscos de segurança ou erros lógicos. E também indivíduos e corporações podem modificar o fonte e criar software personalizado que atenda às necessidades de um ambiente particular.

Na década de 1990 os sistemas operacionais tornaram-se cada vez mais amigáveis ao usuário. As capacidades de GUI que a Apple inseriu no seu sistema operacional Macintosh na década de 80 eram amplamente usadas em vários outros sistemas operacionais e tornaram-se mais sofisticadas. Capacidades *plug-and-play* ('ligar-e-usar') foram embutidas em sistemas operacionais, habilitando usuários a adicionar e remover componentes de software dinamicamente sem ter de reconfigurar manualmente o sistema operacional.

Middleware é um software que liga duas aplicações, muitas vezes por uma rede e freqüentemente entre máquinas incompatíveis. É particularmente importante para serviços Web, porque simplifica a comunicação entre várias arquiteturas. Serviços Web abrangem um conjunto de padrões relacionados que habilitam quaisquer duas aplicações de computador a se comunicarem e trocar dados via Internet. São peças de software prontas para o uso na Internet.

Quando o IBM PC surgiu, deu-se imediatamente origem a uma imensa indústria de software na qual fornecedores independentes de software (ISVs) conseguiam comercializar pacotes de software para o IBM PC que eram executados no sistema operacional MS-DOS. Se um sistema operacional apresentar um ambiente propício ao desenvolvimento rápido e fácil de aplicações, ele e o hardware terão mais probabilidade de sucesso no mercado. Uma vez que uma base de aplicações (a combinação do hardware com o ambiente do sistema operacional no qual a aplicação é desenvolvida) seja amplamente estabelecida, torna-se extremamente difícil solicitar aos usuários e desenvolvedores de software que a convertam a um ambiente de desenvolvimento de aplicações completamente novo, proporcionado por um sistema operacional radicalmente diferente.

Sistemas operacionais destinados a ambientes avançados devem ser projetados para suportar extensas memórias principais, hardware de uso específico e grandes números de processos. Sistemas embarcados são caracterizados por um pequeno conjunto de recursos especializados que fornecem funcionalidade a dispositivos como telefones celulares e PDAs. Nesses ambientes, o gerenciamento eficiente de recursos é a chave para a construção de um sistema operacional de sucesso.

Sistemas de tempo real exigem que as tarefas sejam realizadas em um período de tempo particular (muitas vezes curto). Por exemplo, o recurso de piloto automático de uma aeronave deve ajustar constantemente velocidade, altitude e direção. Essas ações não podem esperar indefinidamente — e às vezes não podem esperar de jeito nenhum — que outras tarefas não essenciais sejam concluídas.

Alguns sistemas operacionais devem gerenciar hardware que possa ou não existir fisicamente na máquina. Uma máquina virtual (VM) é uma abstração de software de um computador que freqüentemente é executada como uma aplicação de usuário sobreposta ao sistema operacional nativo. Um sistema operacional de máquina virtual gerencia os recursos providos pela máquina virtual. Uma aplicação de máquinas virtuais é permitir que várias instâncias de um sistema operacional sejam executadas concorrentemente. Outra utilização de máquinas virtuais é a emulação — capacidade de usar software ou hardware que representem funcionalidades de hardware ou de software que não estejam presentes no sistema. Proporcionando a ilusão de que as aplicações estão sendo executadas em hardware ou sistemas operacionais diferentes, as máquinas virtuais promovem a portabilidade — a capacidade de o software ser executado em várias plataformas — e muitos outros benefícios.

Um usuário interage com o sistema operacional via uma ou mais aplicações de usuário. Muitas vezes essa interação se dá por meio de um software especial denominado interpretador de comandos (shell). O software que contém os componentes centrais do sistema operacional é denominado núcleo (kernel). Entre os componentes típicos de sistemas operacionais, estão o escalonador de processo, o gerenciador de memória, o gerenciador de E/S, o gerenciador de comunicação entre processos (CEP) e o gerenciador de sistemas de arquivos.

Quase todos os sistemas operacionais modernos suportam um ambiente de multiprogramação no qual várias aplicações podem ser executadas concorrentemente. O núcleo gerencia a execução de processos. Componentes de programas, executados independentemente, mas que usam um único espaço de memória para compartilhar dados, são denominados threads.

Quando um processo quer acessar um dispositivo de E/S, deve emitir ao sistema operacional uma chamada ao sistema. Aquela chamada ao sistema é subseqüentemente tratada por um driver de dispositivo — um componente de software que interage diretamente com o hardware — e freqüentemente contém comandos e outras instruções específicos do dispositivo para realizar as operações de entrada/saída requisitadas.

Os usuários passaram a esperar certas características de sistemas operacionais como eficiência, robustez, escalabili-

dade, extensibilidade, portabilidade, segurança e proteção, interatividade e usabilidade.

Em um sistema operacional monolítico todos os componentes estão no núcleo. O resultado é que qualquer componente pode se comunicar diretamente com qualquer outro. Sistemas operacionais monolíticos tendem a ser altamente eficientes. Uma desvantagem dos projetos monolíticos é que é difícil determinar a fonte de erros imperceptíveis.

A abordagem de camadas de sistemas operacionais tenta abordar essa questão agrupando componentes que realizem tarefas similares em camadas. Cada camada comunica-se exclusivamente com as camadas imediatamente acima e abaixo dela. Em uma abordagem de camadas, a requisição de um processo de usuário pode precisar passar por muitas camadas antes de ser atendida. Como é preciso invocar métodos adicionais para passar dados e controle de uma camada para a seguinte, o rendimento do sistema decresce em comparação com o do núcleo monolítico que pode requerer apenas uma única chamada para atender a uma requisição similar.

Uma arquitetura de sistema operacional de micronúcleo fornece somente um número pequeno de serviços na tentativa de manter o núcleo pequeno e escalável. Micronúcleos exibem um alto grau de modularidade que os tornam extensíveis, portáveis e escaláveis. Todavia, essa modularidade é resultado de um maior grau de comunicação entre módulos que pode degradar o desempenho do sistema.

Um sistema operacional de rede é executado em um computador e habilita seus processos a acessar recursos, como arquivos e processadores, em um computador remoto. Um sistema operacional distribuído é um sistema operacional único que gerencia recursos em mais de um sistema de computador. Entre as metas de um sistema operacional distribuído, estão desempenho transparente, escalabilidade, tolerância a falhas e consistência.

Exercícios

1.1 Qual a diferença entre multiprogramação e multiprocessamento? Quais as principais motivações para o desenvolvimento de cada um?

1.2 Discuta brevemente a importância de cada um dos seguintes sistemas mencionados neste capítulo:

- **a.** MS-DOS
- **b.** CTSS
- **c.** Multics
- **d.** OS/360
- **e.** TSS
- **f.** UNIX
- **g.** Macintosh

1.3 Quais desenvolvimentos tornaram viável o computador pessoal?

1.4 Por que não é funcional usar uma máquina virtual para um sistema rígido de tempo real?

1.5 Que papel as interfaces gráficas com o usuário desempenharam na revolução do computador pessoal?

1.6 A GNU Public License (GPL) promove software livre, no sentido de 'liberdade'. Como a GPL oferece tal liberdade?

1.7 Como a computação distribuída afetou o projeto do sistema operacional?

1.8 Quais as vantagens e desvantagens da comunicação entre computadores?

1.9 Defina, compare e aponte as diferenças para cada um destes termos:

- **a.** on-line
- **b.** tempo real
- **c.** computação interativa
- **d.** tempo compartilhado

1.10 Como o middleware e os serviços Web promovem a interoperabilidade?

1.11 Avalie as arquiteturas monolítica, de camadas e de micronúcleo segundo

- **a.** eficiência
- **b.** robustez
- **c.** extensibilidade
- **d.** segurança

Projetos sugeridos

I.12 Elabore um trabalho de pesquisa sobre o sistema operacional Linux. De que modo ele suporta a doutrina do software 'livre' de Stallman? De que modo o Linux conflita com essa filosofia?

I.13 Elabore um trabalho de pesquisa sobre a Internet e como sua penetração afeta o projeto de sistemas operacionais.

I.14 Elabore um trabalho de pesquisa sobre o movimento do software de fonte aberto. Discuta se todo software de fonte aberto é livre, no sentido de 'liberdade' e 'preço'. Como a GPL e licenças similares promovem o software de fonte aberto?

I.15 Elabore um trabalho de pesquisa sobre a evolução dos sistemas operacionais. Não esqueça de mencionar as principais tecnologias de hardware, software e comunicação que favoreceram cada inovação nos sistemas operacionais.

I.16 Elabore um trabalho de pesquisa sobre o futuro dos sistemas operacionais.

I.17 Elabore um trabalho de pesquisa fornecendo uma taxonomia completa dos sistemas operacionais passados e presentes.

I.18 Elabore um trabalho de pesquisa sobre serviços Web. Discuta as principais tecnologias que fundamentam a infra-estrutura dos serviços Web. Como a disponibilidade de serviços Web afeta o desenvolvimento de aplicações?

I.19 Elabore um trabalho de pesquisa sobre aplicações para negócios críticos e para missão crítica. Discuta os atributos principais de hardware, software de comunicações e sistemas operacionais, essenciais para construir sistemas que suportem esses tipos de aplicações.

I.20 Elabore um trabalho de pesquisa sobre sistemas de máquinas virtuais. Não se esqueça de investigar o sistema operacional VM da IBM e a Java Virtual Machine da Sun (JVM).
I.21 Elabore um trabalho de pesquisa sobre sistemas operacionais e a lei. Pesquise a legislação relacionada a sistemas operacionais.
I.22 Elabore um trabalho de pesquisa sobre o impacto de sistemas operacionais nos negócios e na economia.
I.23 Elabore um trabalho de pesquisa sobre sistemas operacionais e segurança e privacidade. Não se esqueça de considerar as questões de vermes e vírus.
I.24 Elabore um trabalho de pesquisa sobre questões éticas com as quais os sistemas operacionais precisam se preocupar. Não se esqueça de tratar de questões como a utilização de sistemas de computador em situações de guerra e de ameaça à vida, vírus e vermes e outros tópicos importantes que você descobrir ao fazer a investigação para o seu trabalho.
I.25 Liste diversas tendências que estão na vanguarda dos futuros projetos de sistemas operacionais. De que maneira cada uma afetará a natureza dos futuros sistemas?
I.26 Elabore um trabalho de pesquisa discutindo o projeto de sistemas maciçamente paralelos. Não se esqueça de comparar sistemas de multiprocessamento de grande escala (por exemplo, o supercomputador Superdome da Hewlett-Packard, que contém até 64 processadores; www.hp.com/products1/servers/scalableservers/superdome/), com sistemas agrupados e server farms (parque de servidores), que contêm centenas de milhares de computadores pouco avançados que cooperam para desempenhar tarefas comuns (veja, por exemplo, www.beowulf.org). Use as informações em www.top500.org; você encontrará uma lista dos supercomputadores mais poderosos do mundo, para determinar o tipo de tarefas que cada um desses sistemas maciçamente paralelos desempenha.
I.27 Quais as tendências que estão na vanguarda dos impressionantes avanços na computação paralela? Quais desafios devem ser enfrentados por projetistas de hardware e de software antes que a computação paralela torne-se amplamente utilizada?
I.28 Elabore um trabalho de pesquisa comparando a pesquisa de sistemas operacionais Exokernel do MIT (www.pdos.lcs.mit.edu/exo.html) com micronúcleo Mach da CMU (www-2.cs.cmu.edu/afs/cs.cmu.edu/project/mach/public/www/mach.html).[71] Qual o foco primordial de cada sistema operacional? Não se esqueça de mencionar como os pesquisadores organizaram componentes como gerenciador de memória, escalonador de disco e gerenciador de processo. Ambos ou nenhum desses sistemas tornaram-se um sucesso comercial? Um desses sistemas, ou todos, influenciou os projetos de sistemas operacionais de sucesso comercial?
I.29 Por que os sistemas UNIX e baseados em UNIX continuaram populares nas últimas décadas? Qual o impacto do Linux nos futuros sistemas UNIX?

Notas

1. "Evolution of the Intel microprocessor: 1971–2007", www.berghell.com/whitepapers/Evolution%20of%20Intel%20Microprocessors%201971%202007.pdf.
2. "Top 500 list for november 2002", www.top500.org/list/2002/11/.
3. N. Weizer, "A history of operating systems", *Datamation*, jan. 1981, p. 119-126.
4. H. Goldstein, *The computer from Pascal to von Neumann*. Princeton: Princeton University Press, 1972.
5. N. Stern, *From Eniac to Univac: an appraisal of the Eckert-Mauchly computers*. Bedford: Digital Press, 1981.
6. C. Bashe et al., *IBM's early computers*. Cambridge: MIT Press, 1986.
7. N. Weizer, "A history of operating systems", *Datamation*, jan. 1981, p. 119-126.
8. H. Grosch, "The way it was in 1957", *Datamation*, set. 1977.
9. P. Denning, "Virtual memory", *ACM CSUR*, v. 2, nº 3, set. 1970, p. 153-189.
10. E. Codd, E. Lowry, E. McDonough e C. Scalzi, "Multiprogramming STRETCH: feasibility considerations", *Communications of the ACM*, v. 2, 1959, p. 13-17.
11. A. Critchlow, "Generalized multiprocessor and multiprogramming systems", *Proc. AFIPS, FJCC*, v. 24, 1963, p. 107-125.
12. L. Belady et al., "The IBM history of memory management technology", *IBM Journal of Research and Development*, v. 25, nº 5, set. 1981, p. 491-503.
13. "The evolution of S/390", www-ti.informatik.uni-tuebingen.de/os390/arch/history.pdf.
14. G. Amdahl, G. Blaauw e F. Brooks, "Architecture of the IBM system/360", *IBM Journal of Research and Development*, v. 8, nº 2, abr. 1964, p. 87-101.
15. N. Weizer, "A history of operating systems", *Datamation*, jan. 1981, p. 119-126.
16. B. Evans, "System/360: a retrospective view", *Annals of the History of Computing*, v. 8, nº 2, abr. 1986, p. 155-179.
17. G. Mealy, B. Witt e W. Clark, "The functional structure of OS/360", *IBM Systems Journal*, v. 5, nº 1, 1966, p. 3-51.
18. R. Case e A. Padeges, "Architecture of the IBM system/370", *Communications of the ACM*, v. 21, nº 1, jan. 1978, p. 73-96.
19. D. Gifford e A. Spector, "Case study: IBM's system/360–370 architecture", *Communications of the ACM*, v. 30, nº 4, abr. 1987, p. 291-307.
20. "The evolution of S/390", www-ti.informatik.uni-tuebingen.de/os390/arch/history.pdf.
21. D. Berlind, "Mainframe Linux advocates explain it all", *ZDNet*, 12 abr. 2002, techupdate.zdnet.com/techupdate/stories/main/0,14179,2860720,00.html.
22. K. Frenkel, "Allan L. Scherr: Big Blue's time-sharing pioneer", *Communications of the ACM*, v. 30, nº 10, out. 1987, p. 824-829.
23. T. Harrison et al., "Evolution of small real-time IBM computer systems", *IBM Journal of Research and Development*, v. 25, nº 5, set. 1981, p. 441-451.
24. F. Corbato et al., *The compatible time-sharing system, a programmer's guide*. Cambridge: MIT Press, 1964.
25. P. Crisman et al. (eds.), *The compatible time-sharing system*. Cambridge: MIT Press, 1964.
26. A. Lett e W. Konigsford, "TSS/360: a time-shared operating system", *Proceedings of the Fall Joint Computer Conference, AFIPS*, v. 33, parte 1, 1968, p. 15-28.
27. A. Bensoussan, C. Clingen e R. Daley, "The multics virtual memory: concepts and designs", *Communications of the ACM*, v. 15, nº 5, maio 1972, p. 308-318.
28. R. Creasy, "The origins of the VM/370 time-sharing system", *IBM Journal of Research and Development*, v. 25, nº 5, p. 483-490.

29. K. Conrow, "The CMS cookbook", *Computing and Networking Services*, Kansas State University, 29 jun. 1994, www.ksu.edu/cns/pubs/cms/cms-cook/cms-cook.pdf.

30. P. Denning, "Virtual memory", *ACM Computing Surveys*, v. 2, nº 3, set. 1970, p. 153-189.

31. R. Parmelee et al., "Virtual storage and virtual machine concepts", *IBM Systems Journal*, v. 11, nº 2, 1972.

32. G. Kildall, "CP/M: a family of 8- and 16-bit operating systems", *Byte*, v. 6, nº 6, jun. 1981, p. 216-232.

33. J. S. Quarterman e J. C. Hoskins, "Notable computer networks", *Communications of the ACM*, v. 29, nº 10, out. 1986, p. 932-971.

34. M. Stefik, "Strategic computing at DARPA: overview and assessment", *Communications of the ACM*, v. 28, nº 7, jul. 1985, p. 690-707.

35. D. Comer, *Internetworking with TCP/IP: principles, protocols, and architecture*. Englewood Cliffs, NJ: Prentice Hall, 1988.

36. J. Martin e K. K. Chapman, *Local area networks: architectures and implementations*. Englewood Cliffs, NJ: Prentice Hall, 1989.

37. R. Metcalfe e D. Boggs, "Ethernet: distributed packet switching for local computer networks", *Communications of the ACM*, v. 19, nº 7, jul. 1976.

38. E. Balkovich, S. Lerman e R. Parmelee, "Computing in higher education: the Athena experience", *Computer*, v. 18, nº 11, nov. 1985, p. 112-127.

39. C. Zmoelnig, "The graphical user interface. time for a paradigm shift?", 30 ago. 2001, www.sensomatic.com/chz/gui/history.html.

40. D. Engelbart, "Who we are. How we think. What we do", 24 jun. 2003, www.bootstrap.org/index.html.

41. E. Martin, "The context of STARS", *Computer*, v. 16, nº 11, nov. 1983, p. 14-20.

42. G. Moore, "Cramming more components onto integrated circuits", *Electronics*, v. 38, nº 8, 19 abr. 1965.

43. "One trillion-operations-per-second", *Intel Press Release*, 17 dez. 1996, www.intel.com/pressroom/archive/releases/cn121796.htm.

44. B. Mukherjee, K. Schwan e P. Gopinath, "A survey of multiprocessor operating systems", Georgia Institute of Technology, 5 nov. 1993, p. 2.

45. "Microsoft timeline", www.microsoft.com/museum/mustimeline.mspx.

46. R. Lea, P. Armaral e C. Jacquemot, "COOL-2: An object oriented support platform built above the CHORUS Microkernel", *Proceedings of the International Workshop on Object Orientation in Operating Systems 1991*, out. 1991.

47. A. Weiss, "The politics of free (software)", *netWorker*, set. 2001, p. 26.

48. "The GNU manifesto", www.delorie.com/gnu/docs/GNU/GNU>.

49. A. Weiss, "The politics of free (software)", *netWorker*, set. 2001, p. 27.

50. A. Weiss, "The politics of free (software)", *netWorker*, set. 2001, p. 27-28.

51. M. Ricciuti, "New Windows could solve age old format puzzle — at a price", *CNet*, 13 mar. 2002, news.com.com/2009-1017-857509.html.

52. P. Thurrott, "Windows 'longhorn' FAQ", *Paul Thurrott's Super-Site for Windows*, 6 out. 2003, www.winsupersite.com/faq/longhorn.asp.

53. M. D. Cannon et al., "A virtual machine emulator for performance evaluation", *Communications of the ACM*, v. 23, nº 2, fev. 1980, p. 72.

54. M. D. Cannon et al., "A virtual machine emulator for performance evaluation", *Communications of the ACM*, v. 23, nº 2, fev. 1980, p. 72.

55. "VMware: simplifying computer infrastructure and expanding possibilities", www.vmware.com/company/.

56. M. D. Cannon et al., "A virtual machine emulator for performance evaluation", *Communications of the ACM*, v. 23, nº 2, fev. 1980, p. 73.

57. "Shell", *whatis.com*, www.searchsolaris.techtarget.com/sDefinition/0,,sid12_gci212978,00.html.

58. B. Mukherjee, K. Schwan e P. Gopinath, "A survey of multiprocessor operating systems", *Georgia Institute of Technology* (*GIT-CC-92/0*), 5 nov. 1993, p. 4.

59. E. W. Dijkstra, "The structure of the 'THE'-multiprogramming system", *Communications of the ACM*, v. 11, nº 5, maio 1968, p. 341-346.

60. N. M. Karnik e A. R. Tripathi, "Trends in multiprocessor and distributed operating systems", *Journal of Supercomputing*, v. 9, nº 1/2, 1995, p. 4-5.

61. B. Mukherjee, K. Schwan e P. Gopinath, "A survey of multiprocessor operating system Kernels", *Georgia Institute of Technology* (*GIT-CC-92/0*), 5 nov. 1993, p. 10.

62. D. S. Miljocic, F. Douglis, Y. Paindaveine, R. Wheeler e S. Zhou, "Process migration", *ACM Computing Surveys*, v. 32, nº 3, set. 2000, p. 263.

63. J. Liedtke, "Toward real Microkernels", *Communications of the ACM*, v. 39, nº 9, set. 1996, p. 75, e T. Camp e G. Oberhsause, "Microkernels: a submodule for a traditional operating systems course", *Communications of the ACM*, 1995, p. 155.

64. J. Liedtke, "Toward real Microkernels", *Communications of the ACM*, v. 39, nº 9, set. 1996, p. 75, e T. Camp e G. Oberhsause, "Microkernels: a submodule for a traditional operating systems course", *Communications of the ACM*, 1995, p. 155.

65. D. S. Miljocic, F. Douglis, Y. Paindaveine, R. Wheeler e S. Zhou, "Process migration", *ACM Computing Surveys*, v. 32, nº 3, set. 2000, p. 263.

66. A. S. Tanenbaum e R. V. Renesse, "Distributed operating systems", *Computing Surveys*, v. 17, nº 4, dez. 1985, p. 424.

67. A. S. Tanenbaum e R. V. Renesse, "Distributed operating systems", *Computing Surveys*, v. 17, nº 4, dez. 1985, p. 424.

68. G. S. Blair, J. Malik, J. R. Nicol e J. Walpole, "Design issues for the COSMOS distributed operating system", *Proceedings from the 1988 ACM SIGOPS European Workshop*, 1988, p. 1-2.

69. "MIT LCS parallel and distributed operating systems", 2 jun. 2003, www.pdos.lcs.mit.edu.

70. "Amoeba WWW home page", abr. 1998, www.cs.vu.nl/pub/amoeba/.

71. D. R. Engler, M. F. Kaashoek e J. O'Toole Jr., "Exokernel: an operating system architecture for application-level resource management," *SIGOPS '95*, dez. 1995, p. 252.