Capítulo 1

Conceitos básicos

Um sistema de computação é constituído basicamente por hardware e software. O hardware é composto por circuitos eletrônicos (processador, memória, portas de entrada/saída, etc.) e periféricos eletro-óptico-mecânicos (teclados, mouses, discos rígidos, unidades de disquete, CD ou DVD, dispositivos USB, etc.). Por sua vez, o software de aplicação é representado por programas destinados ao usuário do sistema, que constituem a razão final de seu uso, como editores de texto, navegadores Internet ou jogos. Entre os aplicativos e o hardware reside uma camada de software multi-facetada e complexa, denominada genericamente de Sistema Operacional. *Neste capítulo veremos quais os objetivos básicos do sistema operacional, quais desafios ele deve resolver e como ele é estruturado para alcançar seus objetivos.*

1.1 Objetivos

Existe uma grande distância entre os circuitos eletrônicos e dispositivos de hardware e os programas aplicativos em software. Os circuitos são complexos, acessados através de interfaces de baixo nível (geralmente usando as portas de entrada/saída do processador) e muitas vezes suas características e seu comportamento dependem da tecnologia usada em sua construção. Por exemplo, a forma de acesso de baixo nível a discos rígidos IDE difere da forma de acesso a discos SCSI ou leitores de CD. Essa grande diversidade pode ser uma fonte de dores de cabeça para o desenvolvedor de aplicativos. Portanto, torna-se desejável oferecer aos programas aplicativos uma forma de acesso homogênea aos dispositivos físicos, que permita abstrair as diferenças tecnológicas entre eles.

O sistema operacional é uma camada de software que opera entre o hardware e os programas aplicativos voltados ao usuário final. O sistema operacional é uma estrutura de software ampla, muitas vezes complexa, que incorpora aspectos de baixo nível (como drivers de dispositivos e gerência de memória física) e de alto nível (como programas utilitários e a própria interface gráfica).

A Figura 1.1 ilustra a arquitetura geral de um sistema de computação típico. Nela, podemos observar elementos de hardware, o sistema operacional e alguns programas aplicativos.

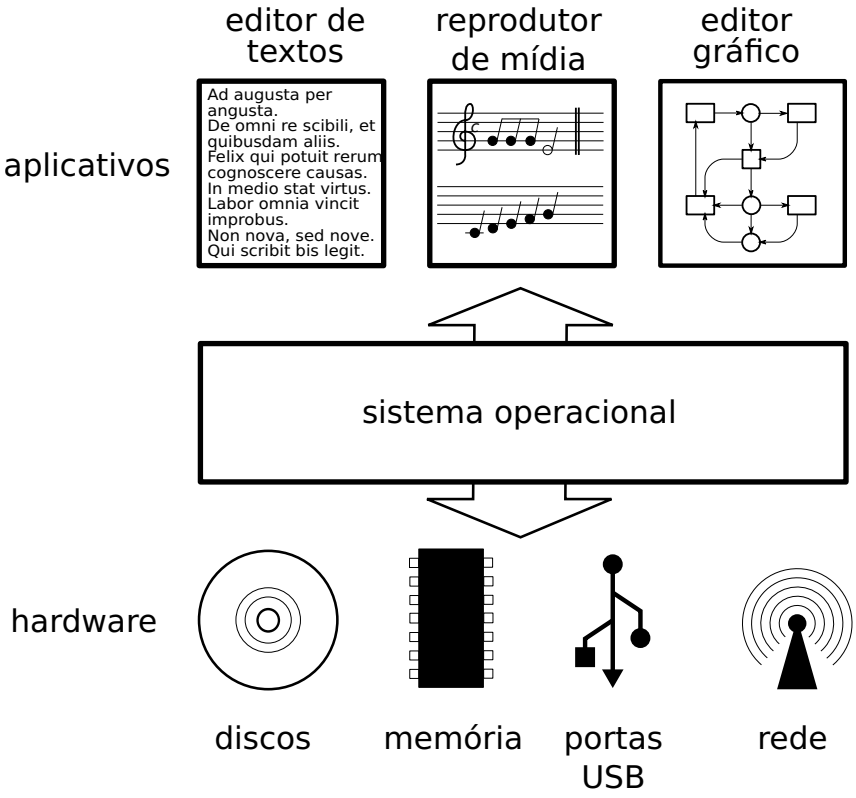

Figura 1.1: Estrutura de um sistema de computação típico

Os objetivos básicos de um sistema operacional podem ser sintetizados em duas palavras-chave: "abstração" e "gerência", cujos principais aspectos são detalhados a seguir.

1.1.1 Abstração de recursos

Acessar os recursos de hardware de um sistema de computação pode ser uma tarefa complexa, devido às características específicas de cada dispositivo físico e a complexidade de suas interfaces. Por exemplo, a sequência a seguir apresenta os principais passos envolvidos na abertura de um arquivo (operação `open`) em um leitor de disquete:

1. verificar se os parâmetros informados estão corretos (nome do arquivo, identificador do leitor de disquete, buffer de leitura, etc.);
2. verificar se o leitor de disquetes está disponível;
3. verificar se o leitor contém um disquete;
4. ligar o motor do leitor e aguardar atingir a velocidade de rotação correta;
5. posicionar a cabeça de leitura sobre a trilha onde está a tabela de diretório;
6. ler a tabela de diretório e localizar o arquivo ou subdiretório desejado;

7. mover a cabeça de leitura para a posição do bloco inicial do arquivo;

8. ler o bloco inicial do arquivo e depositá-lo em um buffer de memória.

Assim, o sistema operacional deve definir interfaces abstratas para os recursos do hardware, visando atender os seguintes objetivos:

- *Prover interfaces de acesso aos dispositivos, mais simples de usar que as interfaces de baixo nível*, para simplificar a construção de programas aplicativos. Por exemplo: para ler dados de um disco rígido, uma aplicação usa um conceito chamado *arquivo*, que implementa uma visão abstrata do disco rígido, acessível através de operações como `open`, `read` e `close`. Caso tivesse de acessar o disco diretamente, teria de manipular portas de entrada/saída e registradores com comandos para o controlador de disco (sem falar na dificuldade de localizar os dados desejados dentro do disco).

- *Tornar os aplicativos independentes do hardware*. Ao definir uma interface abstrata de acesso a um dispositivo de hardware, o sistema operacional desacopla o hardware dos aplicativos e permite que ambos evoluam de forma mais autônoma. Por exemplo, o código de um editor de textos não deve ser dependente da tecnologia de discos rígidos utilizada no sistema.

- *Definir interfaces de acesso homogêneas para dispositivos com tecnologias distintas*. Através de suas abstrações, o sistema operacional permite aos aplicativos usar a mesma interface para dispositivos diversos. Por exemplo, um aplicativo acessa dados em disco através de arquivos e diretórios, sem precisar se preocupar com a estrutura real de armazenamento dos dados, que podem estar em um disquete, um disco IDE, uma máquina fotográfica digital conectada à porta USB, um CD ou mesmo um disco remoto, compartilhado através da rede.

1.1.2 Gerência de recursos

Os programas aplicativos usam o hardware para atingir seus objetivos: ler e armazenar dados, editar e imprimir documentos, navegar na Internet, tocar música, etc. Em um sistema com várias atividades simultâneas, podem surgir conflitos no uso do hardware, quando dois ou mais aplicativos precisam dos mesmos recursos para poder executar. Cabe ao sistema operacional definir *políticas* para gerenciar o uso dos recursos de hardware pelos aplicativos, e resolver eventuais disputas e conflitos. Vejamos algumas situações onde a gerência de recursos do hardware se faz necessária:

- Cada computador normalmente possui menos processadores que o número de tarefas em execução. Por isso, o uso desses processadores deve ser distribuído entre os aplicativos presentes no sistema, de forma que cada um deles possa executar na velocidade adequada para cumprir suas funções sem prejudicar os demais. O mesmo ocorre com a memória RAM, que deve ser distribuída de forma justa entre as aplicações.

- A impressora é um recurso cujo acesso deve ser efetuado de forma mutuamente exclusiva (apenas um aplicativo por vez), para não ocorrer mistura de conteúdo nos documentos impressos. O sistema operacional resolve essa questão definindo uma fila de trabalhos a imprimir (*print jobs*) normalmente atendidos de forma sequencial (FIFO).

- Ataques de negação de serviço (*DoS – Denial of Service*) são comuns na Internet. Eles consistem em usar diversas técnicas para forçar um servidor de rede a dedicar seus recursos a atender um determinado usuário, em detrimento dos demais. Por exemplo, ao abrir milhares de conexões simultâneas em um servidor de e-mail, um atacante pode reservar para si todos os recursos do servidor (processos, conexões de rede, memória e processador), fazendo com que os demais usuários não sejam mais atendidos. É responsabilidade do sistema operacional do servidor detectar tais situações e impedir que todos os recursos do sistema sejam monopolizados por um só usuário (ou um pequeno grupo).

Assim, um sistema operacional visa abstrair o acesso e gerenciar os recursos de hardware, provendo aos aplicativos um ambiente de execução abstrato, no qual o acesso aos recursos se faz através de interfaces simples, independentes das características e detalhes de baixo nível, e no qual os conflitos no uso do hardware são minimizados.

1.2 Tipos de sistemas operacionais

Os sistemas operacionais podem ser classificados segundo diversos parâmetros e perspectivas, como tamanho, velocidade, suporte a recursos específicos, acesso à rede, etc. A seguir são apresentados alguns tipos de sistemas operacionais usuais (muitos sistemas operacionais se encaixam bem em mais de uma das categorias apresentadas):

Batch (de lote) : os sistemas operacionais mais antigos trabalhavam "por lote", ou seja, todos os programas a executar eram colocados em uma fila, com seus dados e demais informações para a execução. O processador recebia os programas e os processava sem interagir com os usuários, o que permitia um alto grau de utilização do sistema. Atualmente, este conceito se aplica a sistemas que processam tarefas sem interação direta com os usuários, como os sistemas de processamento de transações em bancos de dados. Além disso, o termo "em lote" também é usado para designar um conjunto de comandos que deve ser executado em sequência, sem interferência do usuário. Exemplos desses sistemas incluem o OS/360 e VMS, entre outros.

De rede : um sistema operacional de rede deve possuir suporte à operação em rede, ou seja, a capacidade de oferecer às aplicações locais recursos que estejam localizados em outros computadores da rede, como arquivos e impressoras. Ele também deve disponibilizar seus recursos locais aos demais computadores, de forma controlada. A maioria dos sistemas operacionais atuais oferece esse tipo de funcionalidade.

Distribuído : em um sistema operacional distribuído, os recursos de cada máquina estão disponíveis globalmente, de forma transparente aos usuários. Ao lançar uma aplicação, o usuário interage com sua janela, mas não sabe onde ela está executando ou armazenando seus arquivos: o sistema é quem decide, de forma transparente. Os sistemas operacionais distribuídos já existem há tempos (Amoeba [Tanenbaum et al., 1991] e Clouds [Dasgupta et al., 1991], por exemplo), mas ainda não são uma realidade de mercado.

Multi-usuário : um sistema operacional multi-usuário deve suportar a identificação do "dono" de cada recurso dentro do sistema (arquivos, processos, áreas de memória, conexões de rede) e impor regras de controle de acesso para impedir o uso desses recursos por usuários não autorizados. Essa funcionalidade é fundamental para a segurança dos sistemas operacionais de rede e distribuídos. Grande parte dos sistemas atuais são multi-usuários.

Desktop : um sistema operacional "de mesa" é voltado ao atendimento do usuário doméstico e corporativo para a realização de atividades corriqueiras, como edição de textos e gráficos, navegação na Internet e reprodução de mídias simples. Suas principais características são a interface gráfica, o suporte à interatividade e a operação em rede. Exemplos de sistemas *desktop* são os vários sistemas Windows (XP, Vista, 7, etc.), o MacOS X e Linux.

Servidor : um sistema operacional servidor deve permitir a gestão eficiente de grandes quantidades de recursos (disco, memória, processadores), impondo prioridades e limites sobre o uso dos recursos pelos usuários e seus aplicativos. Normalmente um sistema operacional servidor também tem suporte a rede e multi-usuários.

Embarcado : um sistema operacional é dito embarcado (embutido ou *embedded*) quando é construído para operar sobre um hardware com poucos recursos de processamento, armazenamento e energia. Aplicações típicas desse tipo de sistema aparecem em telefones celulares, sistemas de automação industrial e controladores automotivos, equipamentos eletrônicos de uso doméstico (leitores de DVD, TVs, fornos-micro-ondas, centrais de alarme, etc.). Muitas vezes um sistema operacional embarcado se apresenta na forma de uma biblioteca a ser ligada ao programa da aplicação (que é fixa). LynxOS, μC/OS, Xylinx e VxWorks são exemplos de sistemas operacionais embarcados para controle e automação. Sistemas operacionais para telefones celulares inteligentes (*smartphones*) incluem o Symbian e o Android, entre outros.

Tempo real : ao contrário da concepção usual, um sistema operacional de tempo real não precisa ser necessariamente ultra-rápido; sua característica essencial é ter um comportamento temporal previsível (ou seja, seu tempo de resposta deve ser conhecido no melhor e pior caso de operação). A estrutura interna de um sistema operacional de tempo real deve ser construída de forma a minimizar esperas e latências imprevisíveis, como tempos de acesso a disco e sincronizações excessivas.

Existem duas classificações de sistemas de tempo real: *soft real-time systems*, nos quais a perda de prazos implica na degradação do serviço prestado. Um exemplo

seria o suporte à gravação de CDs ou à reprodução de músicas. Caso o sistema se atrase, pode ocorrer a perda da mídia em gravação ou falhas na música que está sendo tocada. Por outro lado, nos *hard real-time systems* a perda de prazos pelo sistema pode perturbar o objeto controlado, com graves consequências humanas, econômicas ou ambientais. Exemplos desse tipo de sistema seriam o controle de funcionamento de uma turbina de avião a jato ou de uma caldeira industrial.

Exemplos de sistemas de tempo real incluem o QNX, RT-Linux e VxWorks. Muitos sistemas embarcados têm características de tempo real, e vice-versa.

1.3 Funcionalidades

Para cumprir seus objetivos de abstração e gerência, o sistema operacional deve atuar em várias frentes. Cada um dos recursos do sistema possui suas particularidades, o que impõe exigências específicas para gerenciar e abstrair os mesmos. Sob esta perspectiva, as principais funcionalidades implementadas por um sistema operacional típico são:

Gerência do processador : também conhecida como gerência de processos ou de atividades, esta funcionalidade visa distribuir a capacidade de processamento de forma justa[1] entre as aplicações, evitando que uma aplicação monopolize esse recurso e respeitando as prioridades dos usuários. O sistema operacional provê a ilusão de que existe um processador independente para cada tarefa, o que facilita o trabalho dos programadores de aplicações e permite a construção de sistemas mais interativos. Também faz parte da gerência de atividades fornecer abstrações para sincronizar atividades inter-dependentes e prover formas de comunicação entre elas.

Gerência de memória : tem como objetivo fornecer a cada aplicação uma área de memória própria, independente e isolada das demais aplicações e inclusive do núcleo do sistema. O isolamento das áreas de memória das aplicações melhora a estabilidade e segurança do sistema como um todo, pois impede aplicações com erros (ou aplicações maliciosas) de interferir no funcionamento das demais aplicações. Além disso, caso a memória RAM existente seja insuficiente para as aplicações, o sistema operacional pode aumentá-la de forma transparente às aplicações, usando o espaço disponível em um meio de armazenamento secundário (como um disco rígido). Uma importante abstração construída pela gerência de memória é a noção de *memória virtual*, que desvincula os endereços de memória vistos por cada aplicação dos endereços acessados pelo processador na memória RAM. Com isso, uma aplicação pode ser carregada em qualquer posição livre da memória, sem que seu programador tenha de se preocupar com os endereços de memória onde ela irá executar.

[1] Distribuir de forma justa, mas não necessariamente igual, pois as aplicações têm demandas de processamento distintas; por exemplo, um navegador de Internet demanda menos o processador que um aplicativo de edição de vídeo, e por isso o navegador pode receber menos tempo de processador.

Gerência de dispositivos : cada periférico do computador possui suas peculiaridades; assim, o procedimento de interação com uma placa de rede é completamente diferente da interação com um disco rígido SCSI. Todavia, existem muitos problemas e abordagens em comum para o acesso aos periféricos. Por exemplo, é possível criar uma abstração única para a maioria dos dispositivos de armazenamento como *pen-drives*, discos SCSI ou IDE, disquetes, etc., na forma de um vetor de blocos de dados. A função da gerência de dispositivos (também conhecida como *gerência de entrada/saída*) é implementar a interação com cada dispositivo por meio de *drivers* e criar modelos abstratos que permitam agrupar vários dispositivos distintos sob a mesma interface de acesso.

Gerência de arquivos : esta funcionalidade é construída sobre a gerência de dispositivos e visa criar arquivos e diretórios, definindo sua interface de acesso e as regras para seu uso. É importante observar que os conceitos abstratos de arquivo e diretório são tão importantes e difundidos que muitos sistemas operacionais os usam para permitir o acesso a recursos que nada tem a ver com armazenamento. Exemplos disso são as conexões de rede (nos sistemas UNIX e Windows, cada socket TCP é visto como um descritor de arquivo no qual pode-se ler ou escrever dados) e as informações do núcleo do sistema (como o diretório `/proc` do UNIX). No sistema operacional experimental *Plan 9* [Pike et al., 1993], todos os recursos do sistema operacional são vistos como arquivos.

Gerência de proteção : com computadores conectados em rede e compartilhados por vários usuários, é importante definir claramente os recursos que cada usuário pode acessar, as formas de acesso permitidas (leitura, escrita, etc.) e garantir que essas definições sejam cumpridas. Para proteger os recursos do sistema contra acessos indevidos, é necessário: a) definir usuários e grupos de usuários; b) identificar os usuários que se conectam ao sistema, através de procedimentos de autenticação; c) definir e aplicar regras de controle de acesso aos recursos, relacionando todos os usuários, recursos e formas de acesso e aplicando essas regras através de procedimentos de autorização; e finalmente d) registrar o uso dos recursos pelos usuários, para fins de auditoria e contabilização.

Além dessas funcionalidades básicas oferecidas pela maioria dos sistemas operacionais, várias outras vêm se agregar aos sistemas modernos, para cobrir aspectos complementares, como a interface gráfica, suporte de rede, fluxos multimídia, gerência de energia, etc.

As funcionalidades do sistema operacional geralmente são inter-dependentes: por exemplo, a gerência do processador depende de aspectos da gerência de memória, assim como a gerência de memória depende da gerência de dispositivos e da gerência de proteção. Alguns autores [Silberschatz et al., 2001, Tanenbaum, 2003] representam a estrutura do sistema operacional conforme indicado na Figura 1.2. Nela, o núcleo central implementa o acesso de baixo nível ao hardware, enquanto os módulos externos representam as várias funcionalidades do sistema.

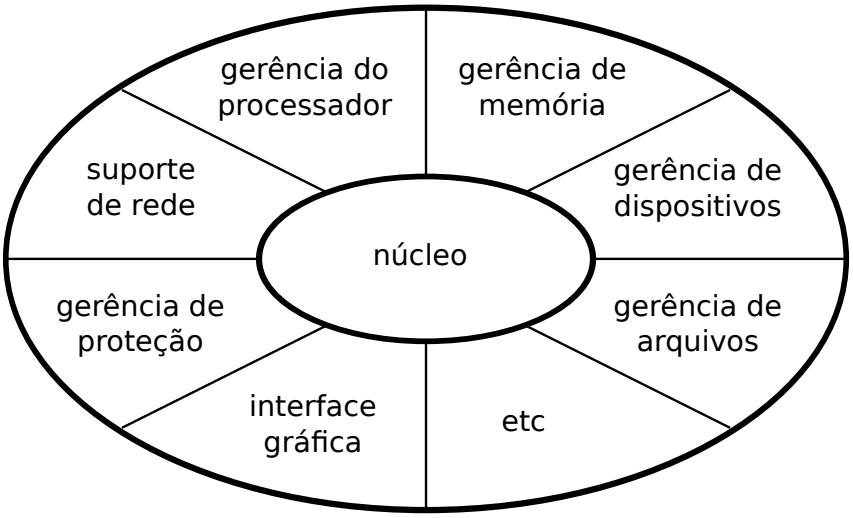

Figura 1.2: Funcionalidades do sistema operacional

Uma regra importante a ser observada na construção de um sistema operacional é a separação entre os conceitos de política e mecanismo[2]. Como *política* consideram-se os aspectos de decisão mais abstratos, que podem ser resolvidos por algoritmos de nível mais alto, como por exemplo decidir a quantidade de memória que cada aplicação ativa deve receber, ou qual o próximo pacote de rede a enviar para satisfazer determinadas especificações de qualidade de serviço.

Por outro lado, como *mecanismo* consideram-se os procedimentos de baixo nível usados para implementar as políticas, ou seja, atribuir ou retirar memória de uma aplicação, enviar ou receber um pacote de rede, etc. Os mecanismos devem ser suficientemente genéricos para suportar mudanças de política sem necessidade de modificações. Essa separação entre os conceitos de política e mecanismo traz uma grande flexibilidade aos sistemas operacionais, permitindo alterar sua personalidade (sistemas mais interativos ou mais eficientes) sem ter de alterar o código que interage diretamente com o hardware. Alguns sistemas, como o InfoKernel [Arpaci-Dusseau et al., 2003], permitem que as aplicações escolham as políticas do sistema mais adequadas às suas necessidades.

1.4 Estrutura de um sistema operacional

Um sistema operacional não é um bloco único e fechado de software executando sobre o hardware. Na verdade, ele é composto de diversos componentes com objetivos e funcionalidades complementares. Alguns dos componentes mais relevantes de um sistema operacional típico são:

[2]Na verdade essa regra é tão importante que deveria ser levada em conta na construção de qualquer sistema computacional complexo.

Núcleo : é o coração do sistema operacional, responsável pela gerência dos recursos do hardware usados pelas aplicações. Ele também implementa as principais abstrações utilizadas pelos programas aplicativos.

Drivers : módulos de código específicos para acessar os dispositivos físicos. Existe um driver para cada tipo de dispositivo, como discos rígidos IDE, SCSI, portas USB, placas de vídeo, etc. Muitas vezes o driver é construído pelo próprio fabricante do hardware e fornecido em forma compilada (em linguagem de máquina) para ser acoplado ao restante do sistema operacional.

Código de inicialização : a inicialização do hardware requer uma série de tarefas complexas, como reconhecer os dispositivos instalados, testá-los e configurá-los adequadamente para seu uso posterior. Outra tarefa importante é carregar o núcleo do sistema operacional em memória e iniciar sua execução.

Programas utilitários : são programas que facilitam o uso do sistema computacional, fornecendo funcionalidades complementares ao núcleo, como formatação de discos e mídias, configuração de dispositivos, manipulação de arquivos (mover, copiar, apagar), interpretador de comandos, terminal, interface gráfica, gerência de janelas, etc.

As diversas partes do sistema operacional estão relacionadas entre si conforme apresentado na Figura 1.3. A forma como esses diversos componentes são interligados e se relacionam varia de sistema para sistema; algumas possibilidades são discutidas na Seção 1.6.

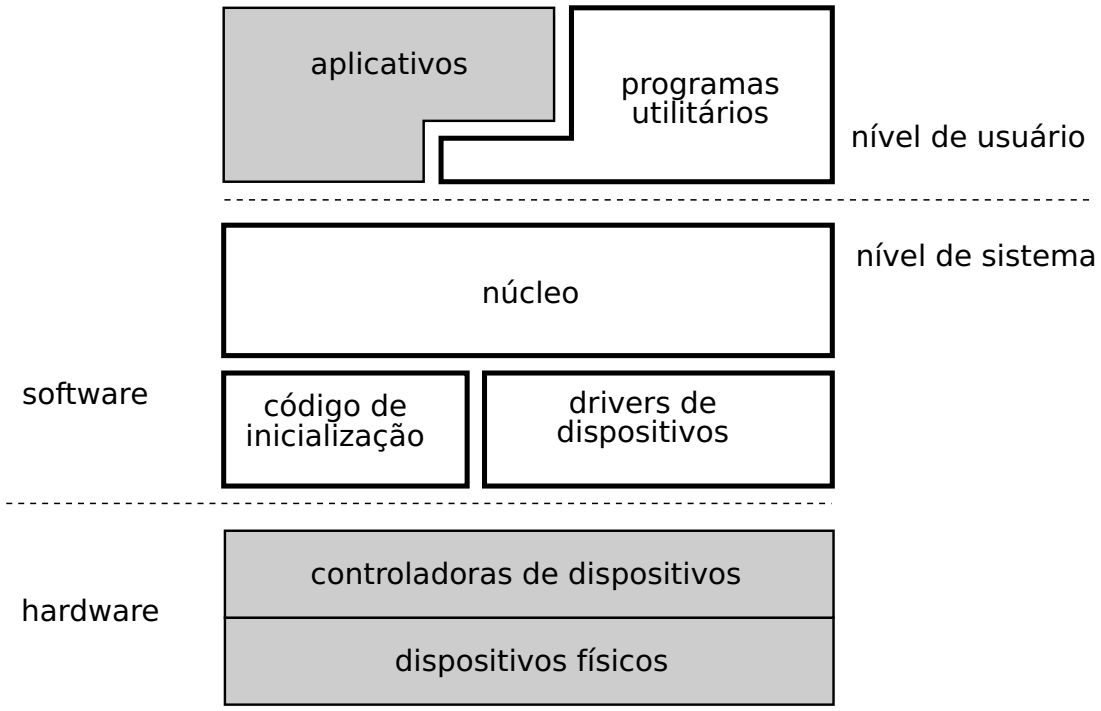

Figura 1.3: Estrutura de um sistema operacional