Unidade III

5 GERÊNCIA DE DISPOSITIVOS DE ENTRADA E SAÍDA – HARDWARE DE E/S

Os primeiros sistemas computacionais eram destinados a cálculos matemáticos e possuíam dispositivos de E/S rudimentares, que apenas permitiam inserir e buscar dados e programas diretamente na memória principal. Posteriormente, foram construídos terminais compostos de teclado e monitor de vídeo, para as operações de leitura e escrita de dados, e os discos rígidos, como meio de armazenamento não volátil de dados e programas.

Nos sistemas computacionais atuais, existe uma grande diversidade de dispositivos de E/S dos mais variados tipos que podem estar conectados ao computador. Esta diversidade de periféricos traz um relevante desafio para construir e manter um SO, já que cada um desses dispositivos possui especificidades, as quais exigem mecanismos de acesso específicos.

Este tópico apresentará uma visão geral das estruturas de hardware associadas aos dispositivos de entrada/saída existentes em computadores convencionais para a interação com o usuário, armazenamento de dados e comunicação.

Cada tipo de dispositivo tem seu próprio conjunto de recursos, definições de bits de controle e protocolos para interação com o controlador, os quais são diferentes entre si. Como projetar o SO de modo a podermos anexar novos dispositivos ao computador sem ter que reescrever o sistema operacional? E, quando os dispositivos variam tanto, como o SO pode fornecer uma interface de E/S conveniente e uniforme para as aplicações?

5.1 Dispositivos de E/S

Além de controlar os processos, a memória principal e o sistema de arquivos, o SO administra todos os dispositivos de E/S, também conhecidos como periféricos. Esses dispositivos possibilitam a interação do computador com o mundo externo.

Como existe uma interdependência entre as gerências, o subsistema de E/S oferece suporte para o sistema de arquivos. Adicionalmente, devem ser compartilhados adequadamente entre os diferentes processos.

Por exemplo, a interação com usuário pode ocorrer através de teclado, mouse, tela de toque ou touch screen, joystick. Outra interação se dá por meio da escrita e leitura de dados em dispositivos de armazenamento como discos rígidos, DVD-ROMs, pendrives. Também há interação através da captura e reprodução de áudios e vídeos, microfone e alto-falantes. Por meio de impressoras e plotadoras, acontece a impressão de informações. Adicionalmente, existe a troca de informações com outros computadores

através de diversos tipos de redes como rede locais (LAN), Wi-Fi, Bluetooth, radiofrequência, telefonia celular e Internet das Coisas (IoT). A figura a seguir apresenta dispositivos de E/S em um smartphone.

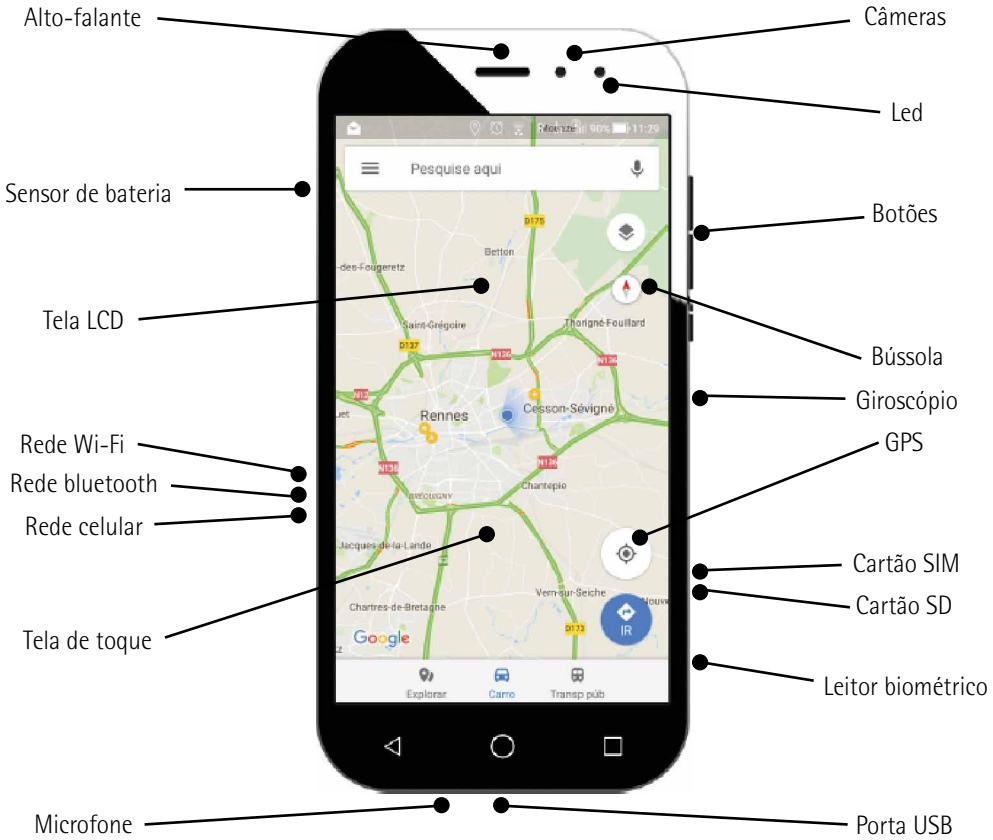

Figura 31 – Dispositivos de E/S de um smartphone

Adaptada de: Maziero (2019, p. 235).

Existem periféricos específicos para algumas aplicações, por exemplo, em ambientes industriais, é frequente encontrar aqueles que realizam a monitoração e o controle de máquinas e processos de produção, como braços robotizados, tornos de comando numérico e controladores de processos químicos. Outro exemplo é o de veículos com computadores embarcados que possuem dispositivos de entrada para coletar dados de combustível e do funcionamento do motor e de saída para controlar injeção eletrônica, tração e freios ABS, por exemplo.

Em função da diversidade dos dispositivos de E/S, o SO deve implementar uma camada denominada subsistema de E/S, com o objetivo de isolar a complexidade dos dispositivos físicos. Com isso, ele pode se tornar flexível, de modo a permitir a comunicação das diversas aplicações com qualquer equipamento de E/S. Características técnicas como velocidade de operação, unidade de transferência, forma de representar os dados e operações que podem ser realizadas são abordadas pela camada de device driver, visando oferecer uma interface uniforme e padronizada entre o subsistema de E/S e todos os periféricos.

A gerência de dispositivos de E/S é uma das mais complexas funções do SO, sendo responsável por gerar a abstração, uma das funções essenciais do SO. Por meio da abstração de recursos, a interação do programador com a máquina se torna facilitada e permite que os programas e os hardwares evoluam de forma independente. A implementação de gerência de dispositivos é estruturada através de camadas, e uma delas visualiza os diversos tipos de dispositivos do sistema de modo único. A outra camada, denominada subsistema de E/S, é específica para cada dispositivo.

O SO envia comandos para periféricos a fim de emitir instruções (read, write, entre outras), interceptar interrupções e tratar erros de leitura de dados.

Para controlar a E/S de dados, não é interessante que a CPU tenha que ficar continuamente monitorando o status de dispositivos como discos ou teclados. O mecanismo de interrupções permite que o hardware "chame a atenção" da CPU e priorize a execução da rotina de atendimento à interrupção quando há necessidade de capturar ou enviar dados para periféricos.

Observação

Atualmente, existe uma variedade de dispositivos de E/S que podem estar conectados a um computador. Essa diversidade de periféricos é um dos maiores desafios presentes na construção e manutenção de um SO, porque cada um deles possui especificidades e exige mecanismos de acesso específicos.

Um dispositivo de E/S comunica-se com um sistema computacional através do envio de sinais por um cabo ou ondas eletromagnéticas pelo ar. O dispositivo se comunica com a máquina por um ponto de conexão, ou porta. Quando os dispositivos compartilham um conjunto de fios, a conexão é denominada barramento ou bus. Um bus é um conjunto de fios e um protocolo rigidamente definido que especifica um conjunto de mensagens que podem ser enviadas nos fios. Do ponto de vista eletrônico, as mensagens são transmitidas por padrões de tensões elétricas aplicados aos fios em intervalos definidos. Quando o dispositivo A tem um cabo que se conecta ao dispositivo B, o dispositivo B tem um cabo que se conecta ao dispositivo C, e o dispositivo C se conecta a uma porta no computador, essa configuração é chamada de cadeia margarida. Uma cadeia margarida opera usualmente como um bus.

Os buses são amplamente usados na arquitetura de computadores e variam em seus métodos de sinalização, em velocidade, no throughput e nos métodos de conexão. Uma estrutura típica de bus de PC será mostrada na figura a seguir. Nela, um bus PCI (o bus comum em um sistema PC) conecta o subsistema processador-memória aos dispositivos rápidos, e um bus de expansão conecta dispositivos relativamente lentos, como o teclado e as portas serial e USB. Na parte superior direita da figura, quatro discos estão conectados, juntos, em um bus small computer system interface (SCSI) conectado a um controlador SCSI. Outros barramentos comuns usados para interconectar as partes principais de um computador incluem o PCI express (PCIe).

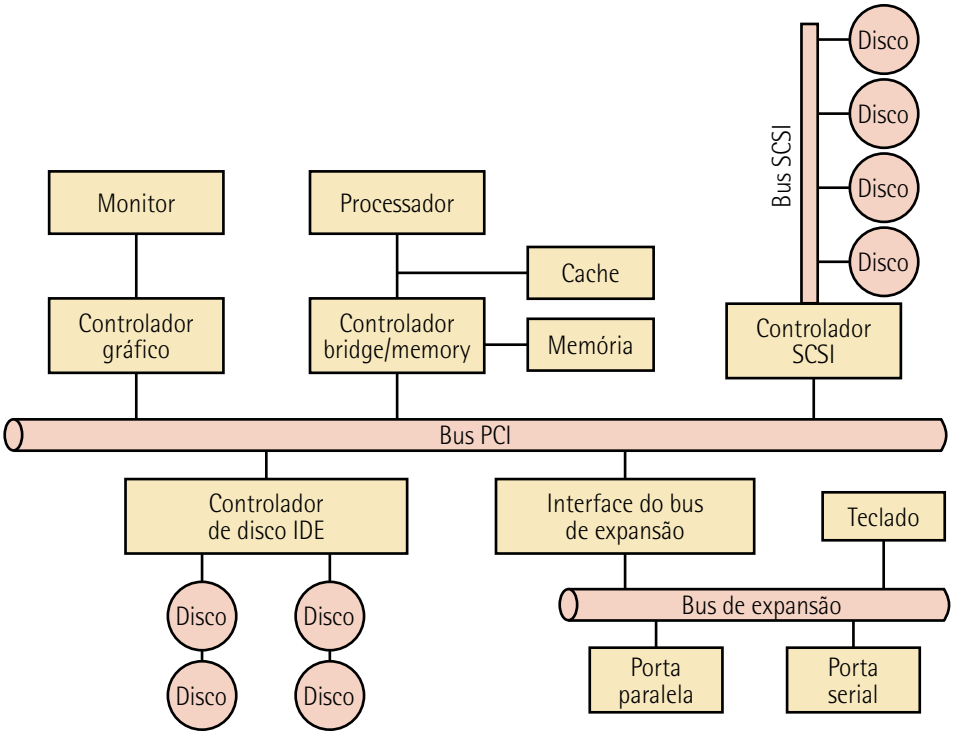

Figura 32 – Estrutura típica de barramento de PC

Fonte: Silberschatz, Galvin e Gagne (2015, p. 323).

O controlador é um conjunto de componentes eletrônicos que podem operar uma porta, um bus ou um dispositivo. Um controlador de porta serial é um controlador de dispositivo simples. Trata-se de um chip individual (ou parte dele) no computador que controla os sinais nos fios de uma porta serial. Por outro lado, um controlador de bus SCSI não é simples. Como o protocolo SCSI é complexo, o controlador de bus SCSI costuma ser implementado como uma placa de circuito separada (ou um adaptador hospedeiro) que é conectada ao computador. Ele contém, tipicamente, um processador, microcódigo e alguma memória privada para habilitá-lo a processar as mensagens do protocolo SCSI. Alguns dispositivos têm seus próprios controladores embutidos. Se você examinar um drive de disco, verá uma placa de circuito anexada a um dos lados. Essa placa é o controlador de disco, que implementa a parte do protocolo referente ao disco para algum tipo de conexão – SCSI ou serial advanced technology attachment (SATA), por exemplo. Possui microcódigo e um processador para executar várias tarefas, como o mapeamento de setores danificados, a pré-busca, o armazenamento em buffer e o armazenamento em cache.

5.2 Subsistema de E/S

Um dos objetivos do SO é tornar as operações de E/S mais simples para o usuário e suas aplicações. Para isso, foi criado o subsistema de E/S, que é responsável por isolar a complexidade de operações específicas para cada tipo de dispositivo da camada de sistema de arquivo, do sistema gerenciador de banco de dados (SGBD) ou diretamente da aplicação. Com isso, torna-se possível para as aplicações a manipulação de qualquer tipo de dispositivo com mais simplicidade.

O subsistema de E/S é formado por um conjunto de rotinas que possibilita a comunicação com qualquer dispositivo que possa ser interligado ao computador. As rotinas de E/S representam as chamadas de sistema ou system calls e permitem ao usuário realizar operações de E/S sem se preocupar com especificidades do dispositivo com o qual está trabalhando. Nesse caso, quando um usuário salva um arquivo em disco, ele não está interessado em conhecer detalhes como a sua formatação ou em que trilha/setor dele o arquivo será armazenado. A figura a seguir apresenta a estrutura de camada da gerência de dispositivos.

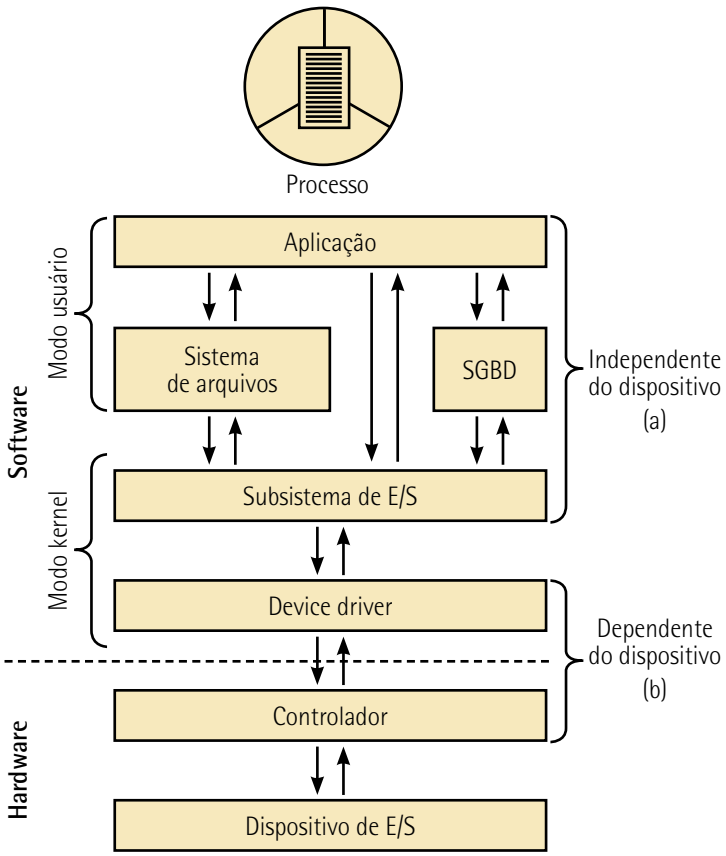

Figura 33 – Arquitetura de camada da gerência de dispositivos

Fonte: Machado e Maia (2013, p. 209).

5.3 Característica de dispositivos de E/S

Para se adequar às muitas possibilidades de interação do computador com o mundo externo, os dispositivos de E/S possuem características diversas de velocidade de transferência, forma de transferência dos dados e método de acesso. A velocidade de transferência de dados de um dispositivo pode variar de alguns bytes por segundo, no caso de dispositivos simples como teclados e mouses, a gigabytes por segundo, para algumas placas de interface gráfica ou de acesso a discos de alto desempenho. A tabela a seguir apresenta alguns exemplos de dispositivos de E/S com suas velocidades típicas de transferência de dados.

Tabela 3 – Velocidades típicas de dispositivos de E/S

Dispositivo	Velocidade
Teclado	10 KB/s
Mouse ótico	100 B/s
Interface infravermelho (IrDA-SIR)	14 KB/s
Interface paralela padrão	125 KB/s
Interface de áudio digital S/PDIF	384 KB/s
Interface de rede Fast Ethernet	11.6 MB/s
Pendrive ou disco USB 2.0	60 MB/s
Interface de rede Gigabit Ethernet	116 MB/s
Disco rígido SATA 2	300 MB/s
Interface gráfica high-end	4.2 GB/s

Fonte: Maziero (2019, p. 238).

Os dispositivos de E/S podem ser classificados de diferentes formas, dependendo de seu fluxo de dados para o computador. São eles:

- **Entrada de dados**: CD-ROM, teclado etc.
- **Saída de dados**: impressoras e projetores.
- **E/S de dados**: modens, discos rígidos e telas touch screen.

Adicionalmente, os periféricos podem ser separados em função da forma com que os dados são armazenados. Nesse caso, os dispositivos podem ser classificados como dispositivos estruturados ou de bloco e dispositivos não estruturados ou de caractere. Os dispositivos de bloco armazenam os dados em blocos que possuem tamanho fixo e endereço próprio. Neles, cada bloco pode ter escrita ou leitura independentemente dos outros. O disco rígido, o pendrive, o DVD-ROM e demais dispositivos de armazenamento são exemplos dessa categoria.

Por outro lado, nos dispositivos não estruturados ou de caractere há o envio e recebimento de caracteres. Entretanto, neles não há endereçamento e não possuem funcionalidades de posicionamento. A impressora, o mouse e a placa de rede são exemplos de dispositivos de caractere.

Outra classificação possível está relacionada com o acesso a posições de memória no dispositivo de E/S. Um dispositivo é de acesso direto quando um bloco pode ser recuperado através de seu endereço. Em contrapartida, é de acesso sequencial quando para se acessar um bloco o dispositivo percorre sequencialmente os demais blocos.

5.4 Interface de acesso

Analisando do ponto de vista do SO, a característica mais relevante dos dispositivos de E/S é a interface de acesso, isto é, a abordagem a ser utilizada para o acesso ao periférico, para a configuração, o envio e seu recebimento. Geralmente, cada dispositivo disponibiliza um conjunto de registradores denominados portas de E/S, que podem ser acessados pelo barramento e são utilizados para comunicação com o processador.

Essas portas podem ser divididas em quatro grupos: portas de entrada, de saída, de status e portas de controle. Nas portas de entrada ou data-in ports, são enviados dados para o processador, enquanto nas de saída ou data-out ports são recebidas informações do processador. Através das portas de status, o processador consulta o estado interno do dispositivo, verificando se as operações foram executadas sem erro. Por fim, as portas de controle são utilizadas pelo processador para o envio de comandos aos dispositivos e modificação dos parâmetros de configuração.

A figura a seguir representa a interface de acesso do dispositivo de E/S, e a quantidade de portas e as suas funções dependem do dispositivo selecionado.

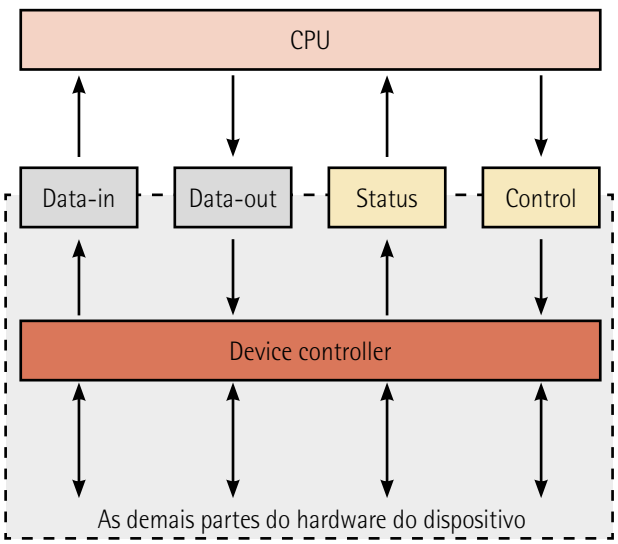

Figura 34 – Interfaces de acesso de dispositivos de E/S

Fonte: Maziero (2019, p. 239).

5.5 Padronização de hardware

As unidades de E/S possuem componentes mecânicos e a controladora do dispositivo. O componente mecânico é o próprio dispositivo e é aparente para o usuário final. Já a controladora do dispositivo é um componente eletrônico do dispositivo e pode ser programável. No caso dos PCs, normalmente é um um circuito integrado.

Em geral, os fabricantes de controladores de dispositivos desenvolvem as interfaces seguindo um padrão oficial, que é fornecido por organizações reconhecidas internacionalmente como ANSI, IEEE, ISO e outras. Os fabricantes seguirão os padrões definidos como os tipos de conectores, as tensões elétricas e as propriedades mecânicas. Esse padrão contribui para a geração de interoperabilidade entre os dispositivos.

Saiba mais

Para mais informações em relação aos órgãos de padronização, consulte:

(ANSI) American National Standards Institute – Instituto Nacional Americano de Padronização. Disponível em: http://www.ansi.org. Acesso em: 26 set. 2023.

(IEEE) Institute of Electrical and Electronics Engineers – Instituto de Engenheiros Eletricistas e Eletrônicos. Disponível em: www.ieee.org. Acesso em: 26 set. 2023.

(ISO) International Organization for Standardization – Organização Internacional de Normalização. Disponível em: www.iso.org. Acesso em: 26 set. 2023.

5.6 Disco rígido

O armazenamento de dados em discos denominados discos rígidos ou hard disks (HDs) é frequentemente usado em computadores para salvar desde arquivos pessoais até dados de mais baixo nível utilizados para o funcionamento do SO. A gravação dos dados no disco ocorre através do campo magnético criado entre os pratos (discos).

Conforme abordado, o disco rígido é um tipo de memória secundária e não volátil, isto é, mesmo na ausência de energia elétrica os dados permanecerão no disco. Os arquivos são apagados definitivamente quando ocorre a formatação lógica do dispositivo.

Analisando a estrutura interna, o HD é composto de pratos circulares denominados discos, onde os dados são armazenados. Com a utilização de um braço atuador, que fica entre os pratos que compõem o disco, os dados são escritos e lidos. Os discos ficam em rotação de alta velocidade, normalmente entre 4.200 e 15.000 rotações por minuto (RPM), enquanto estiverem energizados por um motor elétrico. A figura a seguir apresenta a estrutura interna de um HD.

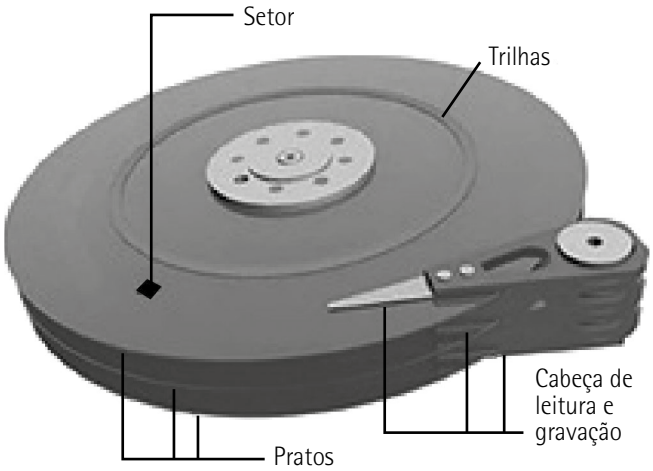

Figura 35 – Estrutura interna de um HD

Fonte: Córdova Jr., Ledur e Morais (2018, p. 204).

Cada disco é dividido logicamente em trilhas (ou cilindros) e setores; a interseção de uma trilha e um setor em uma face define um bloco físico, que é a unidade básica de armazenamento e transferência de dados no disco.

Para realizar a comunicação entre o disco rígido e os demais componentes do sistema computacional, é necessária uma interface de comunicação, e as mais comuns para HD são IDE, SCSI e SATA.

A interface IDE, sigla de integrated drive electronics, surgiu nos anos 1980 e durante muito tempo foi o padrão dos discos mais usados em computadores pessoais e unidades de CD ou DVD. Seu funcionamento baseia-se no padrão de barramento de 16 bits IBM PC Industry Standard Architecture (ISA). O IDE foi adotado como padrão pelo American National Standards Institute (ANSI) em novembro de 1990. Nos computadores atuais, o controlador IDE é geralmente embutido na placa-mãe. No passado, os controladores eram dispositivos externos e separados, o que reduzia os problemas relacionados aos controladores integrados. Este padrão suporta velocidades de até 1 Gbit/s, através de cabos paralelos de 40 ou 80 vias.

As placas-mãe mais antigas possuíam apenas uma interface IDE. Posteriormente, tornou-se padrão o uso de duas delas, chamadas de IDE0 e IDE1. Cada interface IDE pode gerenciar dois dispositivos, e para diferenciar aqueles conectados à mesma interface, um dos dispositivos era configurado como mestre ou master e o outro como escravo ou slave.

O padrão SCSI, Small Computer System Interface, foi muito usado em servidores e estações de trabalho de alto desempenho. Ao longo do tempo, a taxa de transferência das interfaces SCSI aumentou, permitindo modelos de discos mais velozes que os IDE. As placas de interfaces SCSI possuem um conector externo e dois conectores internos. Um barramento SCSI suporta até 16 dispositivos e atinge taxas de transferência de até 2,5 Gigabit/s (divididos entre os dispositivos conectados no mesmo barramento).

O padrão SATA, abreviatura de Serial Advanced Technology Attachment, é o modelo de interface de discos em desktops e notebooks atuais. A transmissão dos dados entre o disco e a controladora é serial, atingindo taxas de transferência de 6 Gbit/s através de cabos com sete vias.

Uma vantagem relevante do SATA em relação ao IDE é a diminuição do tamanho dos cabos, que eram enormes nos dispositivos IDE, já que a transferência de dados ocorria de forma paralela. Como no SATA a transmissão é serial, um pequeno cabo é capaz de realizar a tarefa. A transferência em um dispositivo SATA também supera os antigos padrões IDE. No novo formato, temos dois tipos de velocidade, que são 150 MB/s, no padrão SATA normal, chegando a 300 MB/s nos novos discos denominados SATA II.

Outra tecnologia utilizada para armazenamento de dados em massa, que vem se consolidando no mercado, é a unidade dos estados sólidos ou discos SSD, sigla em inglês para solid state drive. Diferentemente de um disco rígido, os discos SSDs não possuem discos físicos ou agulhas mecânicas, pois os dados são armazenados em chips de memória flash. Essa tecnologia proporciona mais rapidez no acesso aos dados, é menos propensa a falhas pelo fato de não possuir peças que se movimentam e permite que o computador tenha mais agilidade para executar tarefas ou abrir software. Como os discos SSDs são uma tecnologia recente, o valor dos discos é mais elevado em comparação com os HDs quando contrapomos dispositivos com a mesma capacidade de armazenamento.

Observação

Como os discos rígidos são dispositivos eletromecânicos, eles são extremamente lentos se comparados à velocidade da memória ou do processador. Para cada leitura ou escrita, a cabeça de leitura deve se posicionar na trilha desejada e aguardar o disco girar até encontrar o setor desejado. O atraso dessa operação pode ter forte impacto no desempenho do acesso a disco.

5.7 Discos e partições

De forma simplificada, um disco é enxergado pelo SO como um grande vetor de blocos de dados de tamanho fixo, numerados sequencialmente. Tanto as operações de leitura como as de escrita de dados nesses dispositivos são realizadas bloco a bloco. Por esse motivo, tais dispositivos são denominados dispositivos de blocos, em inglês, block devices ou block-oriented devices.

No espaço de armazenamento de cada dispositivo, há uma pequena área de configuração reservada no início do disco e pode existir uma ou mais partições, fato que representa espaços independentes. A área de configuração contém uma tabela de partições com informações sobre o particionamento do dispositivo, tais como quantidade de blocos, número do bloco inicial, tamanho do bloco, entre outras sobre cada partição. Adicionalmente, nesta área está armazenado um pequeno código executável utilizado no processo de inicialização do SO ou boot. Assim, essa área é conhecida como boot sector ou master boot record (MBR) nos PCs.

No início de cada partição existente do disco há um ou mais blocos reservados, empregados para a descrição do conteúdo daquela partição e para armazenar o código de lançamento do SO, se aquela for uma partição inicializável ou bootable partition. Essa área reservada é denominada bloco de inicialização ou volume boot record (VBR). O restante dos blocos da partição está disponível para o armazenamento de arquivos. A figura a seguir apresenta a organização básica do espaço de armazenamento em um disco rígido típico, com três partições.

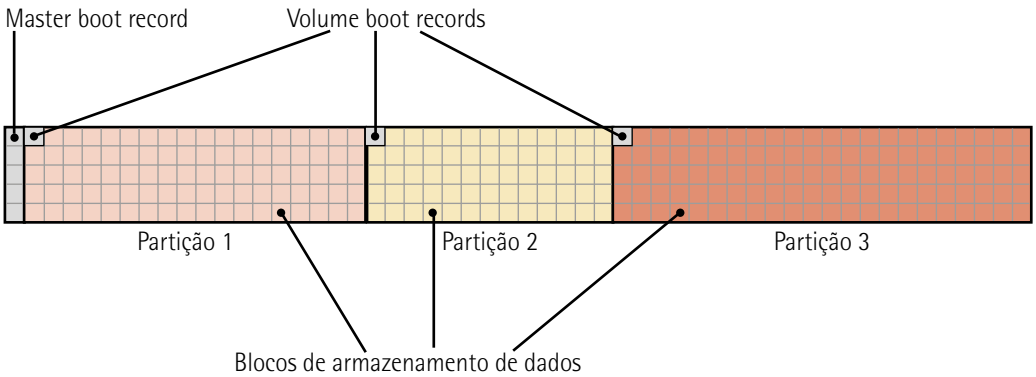

Figura 36 – Organização em partições de um disco rígido

Fonte: Maziero (2019, p. 304).

Um termo frequentemente utilizado em sistemas de arquivos é o volume, o qual representa um espaço de armazenamento de dados, do ponto de vista do SO. Em sua forma mais simples, cada volume corresponde a uma partição, porém configurações mais complexas são possíveis. Por exemplo, o subsistema LVM (logical volume manager) do Linux permite construir volumes lógicos combinando vários discos físicos, como nos sistemas RAID.

Antes de ser utilizado, cada volume ou partição deve ser formatado, isto é, preenchido com as estruturas de dados necessárias para armazenar arquivos, diretórios, atalhos e outras entradas. Cada volume pode ser formatado de forma independente e receber um sistema de arquivos distinto dos demais.

5.7.1 Arranjos RAID

Os dispositivos de armazenamento em massa são responsáveis por armazenar os dados permanentes, de forma que eles devem estar sempre disponíveis no computador. De acordo com Córdova Jr., Ledur e Morais (2018), há várias tecnologias que possibilitam a comunicação entre os discos e a placa-mãe, bem como seu gerenciamento. Buscando soluções eficientes para os problemas de desempenho e confiabilidade dos discos rígidos, uma técnica utilizada é a construção de discos virtuais compostos de conjuntos de discos físicos, que são chamados de conjunto redundante de discos econômicos ou RAID, abreviatura de redundant array of inexpensive disks.

Um sistema RAID é constituído de dois ou mais discos rígidos que são enxergados pelo SO e pelas aplicações como um único disco lógico, ou seja, um grande espaço contíguo de armazenamento de dados e que podem trabalhar paralelamente. Seu objetivo principal é proporcionar mais desempenho

nas operações de transferência de dados, através do paralelismo no acesso aos vários discos, e oferecer mais confiabilidade no armazenamento, usando mecanismos de redundância dos dados armazenados nos discos, como cópias de dados ou códigos corretores de erros. Esses discos podem ser discos rígidos, mas há uma tendência de também usar a tecnologia para SSDs.

Existem diversas formas de organização de um conjunto de discos rígidos em RAID, e cada uma delas possui suas próprias características de desempenho e confiabilidade. Tais formas são geralmente denominadas níveis RAID. Os níveis RAID foram padronizados pela Storage Networking Industry Association (SNIA) em 2009.

No RAID nível 0 (linear), os discos físicos ou partições estão simplesmente concatenados em sequência para construir um disco lógico. Teoricamente, essa estratégia possibilita maior velocidade de leitura e de escrita, já que os acessos a blocos em discos físicos diferentes podem ser realizados paralelamente. A figura a seguir apresenta a estrutura do RAID nível 0 (linear).

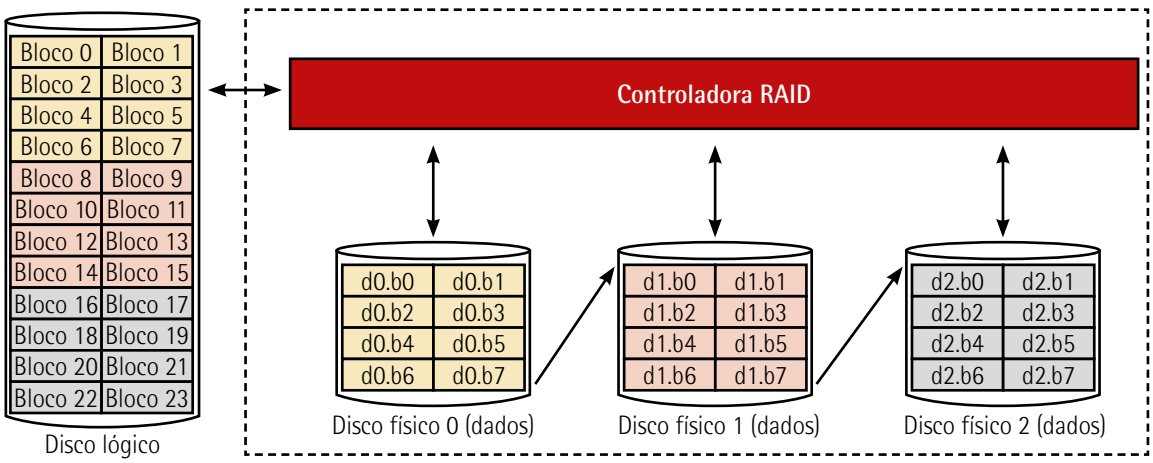

Figura 37 – RAID nível 0 (linear)

Fonte: Maziero (2019, p. 271).

Entretanto, essa vantagem pode ser bastante reduzida quando há concentração de acessos em uma área específica do disco lógico, que provavelmente estará mapeada em um mesmo disco. Com isso, alguns discos tendem a ser mais usados.

Um problema dessa abordagem é que não há nenhuma redundância de dados, tornando o sistema suscetível a erros de disco, pois quando um disco falhar, todos os blocos armazenados nele serão perdidos. De acordo com Maziero (2019), como a probabilidade de falhas aumenta com o número de discos, essa solução gera redução da confiabilidade do sistema de discos.

No RAID nível 0 (striping), os dados são divididos em áreas de tamanho fixo denominadas fatias ou faixas (stripes). Cada faixa de disco físico armazena um ou mais blocos do disco lógico e, geralmente, são utilizadas faixas de 32, 64 ou 128 KB. Tais fatias são agrupadas para construir um único disco lógico, que armazena as unidades da matriz, conforme apresentado na figura a seguir.

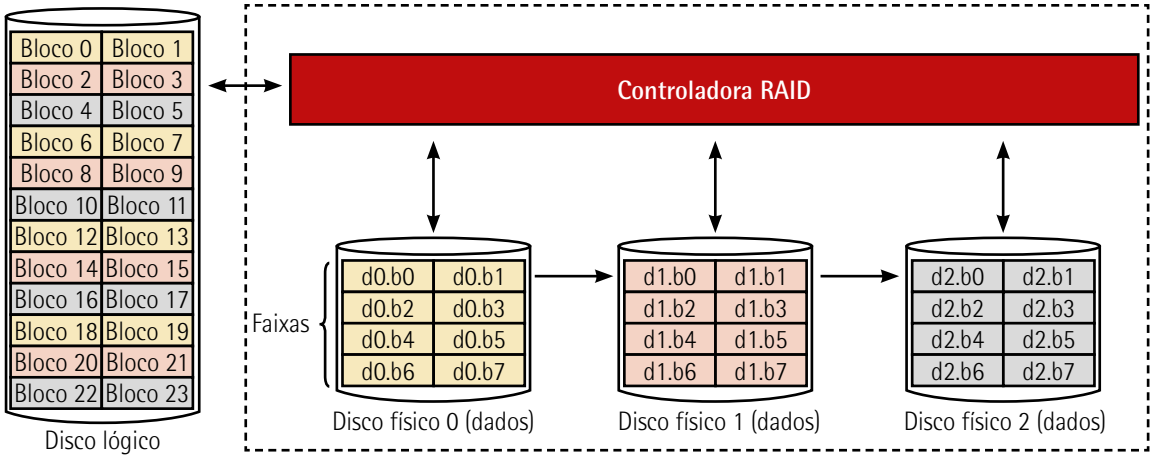

Figura 38 – RAID nível 0 (striping)

Fonte: Maziero (2019, p. 272).

A vantagem da utilização de múltiplos discos simultaneamente é a melhoria do desempenho elevado nas operações de E/S, pois com o maior espalhamento dos blocos sobre os discos físicos contribui-se para a distribuição mais igualitária da carga de acessos entre eles. Considerando suas características de suporte a grande volume de dados e alto desempenho em escrita ou leitura, o RAID 0 (striping) é recomendado para armazenamento não crítico de dados temporários que precisam ser lidos/gravados em alta velocidade, por exemplo, em uma estação de edição de vídeo ou processamento de dados científicos.

No RAID 1, ocorre a replicação do conteúdo em dois ou mais discos e esse processo é conhecido como espelhamento de discos ou disk mirroring. É possível associar RAID 1 a conjuntos de discos em RAID 0, levando a configurações híbridas como RAID 0+1, RAID 1+0 ou RAID 1E. A figura a seguir apresenta uma estrutura de RAID 1 com dois discos rígidos, fato considerado o mais comum.

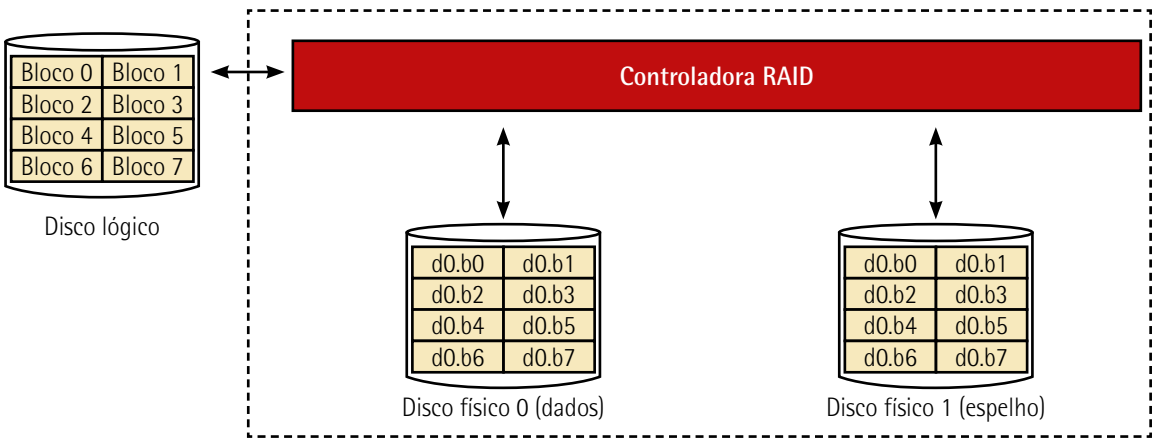

Figura 39 – RAID nível 1 (espelhamento)

Fonte: Maziero (2019, p. 273).

Com a utilização do RAID 1, a confiabilidade do sistema de disco é aumentada, já que cada bloco lógico fica armazenado em dois ou mais discos diferentes. Mesmo que ocorram falhas em um deles, os demais continuam acessíveis. Adicionalmente, o desempenho em leituras também é impactado positivamente, porque a controladora pode distribuir as leituras entre as cópias dos dados.

Entretanto, não ocorre melhoria de desempenho na operação de escrita, pois cada operação deve ser replicada em todos os discos. Outra desvantagem dessa solução é o custo de implantação mais elevado, pois os diversos discos físicos são vistos como apenas um disco lógico do mesmo tamanho.

Por sua vez, no RAID nível 2, os dados são separados em bits individuais que são escritos nos discos físicos em sequência; discos adicionais são usados para armazenar códigos corretores de erros, em um arranjo similar ao usado nas memórias RAM. Esses códigos corretores de erros permitem resgatar dados no caso de falha em blocos ou discos de dados. Devido à eficiência de redução e à complexidade de implementação, o RAID 2 caiu em desuso porque os novos HDs já são fabricados com mecanismos similares a ele, impedindo certas falhas.

Similarmente ao RAID 2, o RAID 3 fatia os dados em bytes escritos nos discos em sequência. É inserido um disco adicional para armazenamento de um byte com os bits de paridade dos bytes correspondentes em cada disco, sendo utilizado para a recuperação de erros nos demais discos. Para cada leitura ou escrita, os dados de paridade são atualizados, transformando o disco de paridade em um gargalo de desempenho. O RAID 4 é muito similar ao RAID 3, com a exceção de que o fatiamento é feito em blocos em vez de bits.

O RAID nível 5, conhecido como acesso independente de paridade distribuída, também guarda os dados de paridade para tolerar falhas em blocos ou discos, como o RAID 4. Entretanto, os bits de paridade não ficam concentrados em um único disco físico, sendo distribuídos uniformemente entre eles. A figura a seguir ilustra a possibilidade de distribuição das informações de paridade.

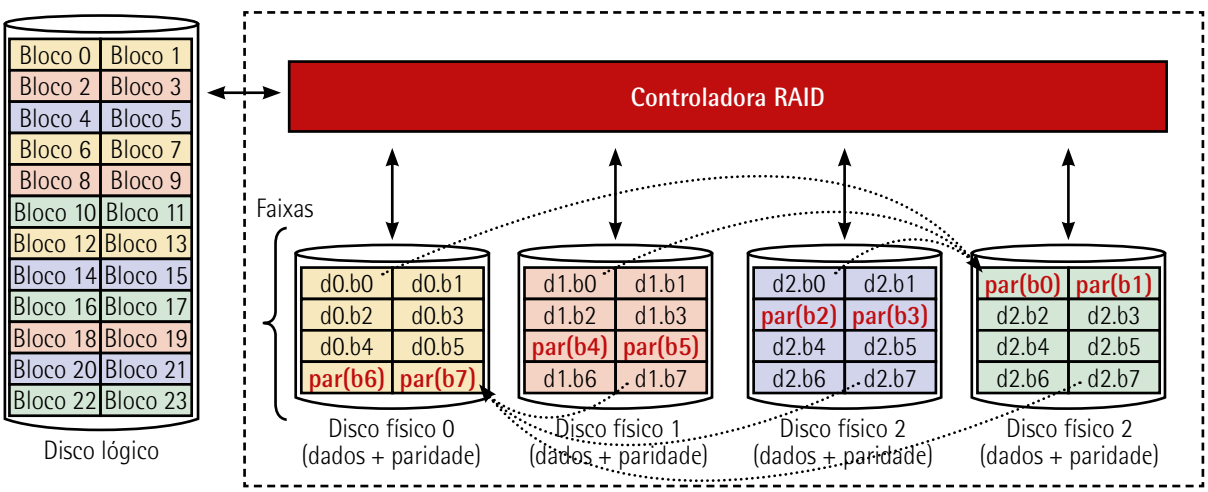

Figura 40 – RAID nível 5 (paridade distribuída)

Fonte: Maziero (2019, p. 274).

Com essa solução, elimina-se o gargalo de desempenho no acesso aos dados de paridade visto no RAID 4. O RAID 5 é uma abordagem de RAID popular por oferecer bom desempenho e redundância de dados, desperdiçando menos espaço que o espelhamento (RAID 1).

6 GERÊNCIA DE DISPOSITIVOS DE ENTRADA E SAÍDA – SOFTWARE DE E/S

Neste tópico serão detalhados os elementos de software para os sistemas de E/S de um SO, como os drivers e a interação entre os dispositivos de E/S, os seus controladores e o processador.

6.1 Device drivers

O device driver, ou somente driver, é um programa ou rotina utilizado para criar uma interface e gerenciar um dispositivo de E/S ou outros periféricos, e essa comunicação entre eles é realizada por intermédio dos controladores de E/S. Os subsistemas de E/S tratam de funções ligadas a todos os dispositivos, enquanto os drivers lidam somente com seus aspectos específicos.

Os drivers atuam recebendo comandos gerais sobre acessos aos dispositivos e traduzindo-os para comandos específicos a um dispositivo, que serão executados pelos controladores de dispositivo. Cada device driver manipula apenas um tipo de dispositivo ou grupo de dispositivos semelhantes. Por exemplo, o driver RTL8110SC(L), desenvolvido pela empresa Realtek Corp., é de uso exclusivo das interfaces de rede RTL8110S, RTL8110SB(L), RTL8169SB(L), RTL8169S(L) e RTL8169 desse fabricante.

Eles são parte integrante do núcleo do SO ou kernel, em modo privilegiado, tendo assim acesso total ao hardware. Os códigos dos drivers constituem um dos maiores riscos à estabilidade e segurança do SO, pois são desenvolvidos pelas fabricantes do hardware e seus próprios desenvolvedores. Drivers mal construídos ou mal configurados dão origem a problemas como travamentos ou reinicializações inesperadas. Normalmente, eles são escritos em linguagem de programação assembly, uma linguagem de máquina e de baixo nível, isto é, mais próxima do hardware. Como existe dependência entre os drivers e o núcleo do sistema, os fabricantes de dispositivos desenvolvem device drivers diferentes para cada SO para um mesmo dispositivo.

De forma geral, as funções implementadas por um driver podem ser separadas em três grupos: funções de E/S, de gerência e funções de tratamento de evento. As funções de E/S são as responsáveis pela transferência de dados entre o SO e os periféricos, elas recebem e enviam dados de acordo com a classe do dispositivo: caracteres (bytes), blocos de tamanho fixo (discos), blocos de tamanho variável (pacotes de rede) ou áreas de memória compartilhadas entre o dispositivo e a CPU (imagens/vídeo e outros).

As funções de gerência realizam a gestão do dispositivo e do próprio driver. Além de funções para coordenar a inicialização e finalização do driver e do dispositivo, geralmente são fornecidas funcionalidades para configurar o dispositivo, desligá-lo ou colocá-lo em espera quando este não for usado, e para tratar os erros dele. Existem chamadas de sistema ou system calls que disponibilizam essas funções aos processos no espaço de usuário, por meio de chamadas de sistema específicas como ioctl, em Unix e Linux e DeviceIOControl, em Windows.

Por sua vez, as funções de tratamento de eventos são ativadas quando uma requisição de interrupção é originada pelo dispositivo. Toda requisição de interrupção gerada pelo dispositivo é encaminhada ao controlador de interrupções do hardware, que a entrega ao núcleo do SO. No núcleo, um tratador de interrupções ou IRq handler reconhece e identifica a interrupção junto ao controlador e em seguida envia uma notificação de evento a uma função do driver para o devido tratamento.

A figura a seguir apresenta a visão geral de um driver de dispositivo.

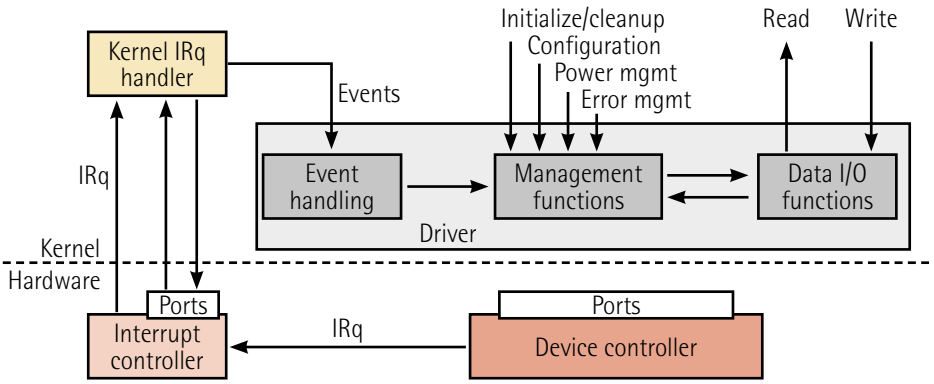

Figura 41 – Visão geral de um driver de dispositivo

Fonte: Maziero (2019, p. 251).

Adicionalmente, um driver mantém estruturas de dados locais, para armazenar informações sobre o dispositivo e as operações em andamento.

6.2 Controlador de E/S

Os controladores de E/S são elementos de hardware que manipulam diretamente os dispositivos de E/S. A comunicação entre o SO, mais exatamente o device driver, com os dispositivos ocorre através dos controladores. A figura a seguir apresenta um computador com CPU, memória e dispositivos de E/S, e a interligação entre o processador, a memória principal e os controladores de E/S ocorre através de barramentos.

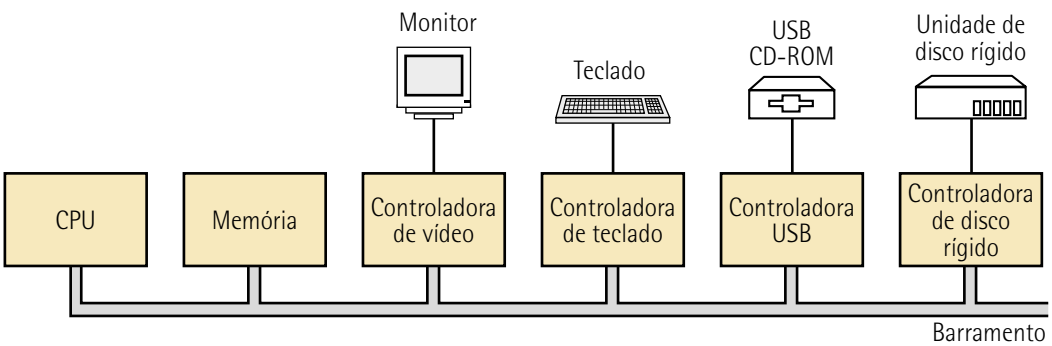

Figura 42 – Diferentes controladores de dispositivos

Fonte: Tanenbaum e Woodhull (2008, p. 215).

O controlador possui memória e registradores exclusivos utilizados na execução de instruções enviadas pelo device driver. Essas instruções de baixo nível são responsáveis pela comunicação entre o controlador e o dispositivo de E/S. Em operações de leitura, o controlador deve armazenar em seu buffer interno uma sequência de bits provenientes do dispositivo até formar um bloco. Após verificar a ocorrência de erros, o bloco pode ser transferido para um buffer de E/S na memória principal.

Os controladores são utilizados na execução de instruções enviadas pelo driver. Em operações de leitura, armazenam em seu buffer interno uma sequência de bits provenientes até formar um bloco. Após verificar se no bloco existem erros, são transferidos para um buffer de E/S na memória principal.

A transferência do bloco para o buffer pode ser realizada por um controlador de acesso direto na memória ou direct memory access (DMA).

Lembrete

A utilização de barramentos com controladores específicos para os dispositivos de E/S é bastante frequente e visa prioritariamente otimizar o processador (CPU).

6.3 Interação entre dispositivos de E/S e controladores

As system calls responsáveis por essa comunicação são chamadas de system calls de E/S, e um de seus objetivos é simplificar a interface entre as aplicações e os dispositivos.

As operações de E/S podem ser classificadas em síncrona ou assíncrona. A síncrona ocorre quando o processo que realizou a operação fica aguardando em estado de espera por seu término. Na assíncrona, o processo que efetuou a operação não aguarda pelo seu término e continua pronta para ser executada. Nesse caso, deve existir uma sinalização indicando que a operação foi finalizada.

O driver comunica-se com os dispositivos através dos controladores. Ele pode ser uma placa independente conectada a um slot do computador ou implementada na mesma placa do processador. O controlador possui memória e registradores próprios e existem três estratégias mais frequentes pelos drivers para essa interação, são elas: E/S controlada por programa, controlada por eventos e acesso direto à memória.

Na estratégia de E/S, a interação do dispositivo é controlada por programa, sendo denominada varredura, polling ou entrada/saída programada, em inglês programmed I/O (PIO). Nesta solução, o driver realiza uma solicitação de operação do dispositivo, utilizando as portas control, data-out ou data-in de sua interface e aguarda a finalização da operação requisitada, monitorando os bits da respectiva porta de status de forma contínua.

Uma maneira mais eficiente de interação com dispositivos de E/S, denominada interação controlada por evento, consiste em requisitar a operação almejada e interromper a execução do processo corrente, de forma a liberar o processador para realização de outras tarefas. Ao finalizar o processamento da operação solicitada, o controlador originará uma requisição de interrupção ou interrupt request (IRq) para notificar o driver correspondente, que voltará a executar o fluxo de instruções.

Nessa estratégia, podemos dividir uma operação de E/S em duas partes: um bloco que inicia a operação, ativado pelo driver por requisição de um processo ou thread, e uma rotina de tratamento de interrupção ou interrupt handler, ativada a cada interrupção, para relatar a conclusão da última operação solicitada.

Normalmente, o tratamento de operações de E/S é uma operação lenta, já que a velocidade de dispositivos periféricos é muito inferior à do processador. Além disso, utilizar o processador para buscar dados dos periféricos é muito ineficiente, pois resulta em transferências adicionais desnecessárias de dados. Para resolver esse problema, foram desenvolvidos mecanismos de acesso direto à memória ou direct memory access (DMA) que permitem transferência entre a memória principal e os controladores de E/S.

Exemplo de aplicação

Considere uma operação de escrita de dados de um buffer na memória RAM para o controlador em um disco rígido. A sequência de passos para realizar a operação seria:

1. o processador acessa as portas do controlador de DMA associado ao dispositivo desejado, para informar o endereço inicial e o tamanho da área de memória RAM contendo os dados a serem escritos no disco. O tamanho da área de memória deve ser um múltiplo do tamanho dos blocos físicos do disco rígido (512 ou 4096 bytes);

2. o controlador de DMA solicita ao controlador do disco a transferência de dados da RAM para o disco e aguarda a conclusão da operação;

3. o controlador do disco rígido recebe os dados da memória; essa operação pode ser repetida várias vezes, caso o volume de dados seja maior que o tamanho máximo de cada transferência;

4. finalizada a transferência de dados, o controlador de DMA notifica o processador sobre a conclusão da operação, usando uma requisição de interrupção (IRq).

A dinâmica dessas operações é ilustrada de forma simplificada na figura a seguir.

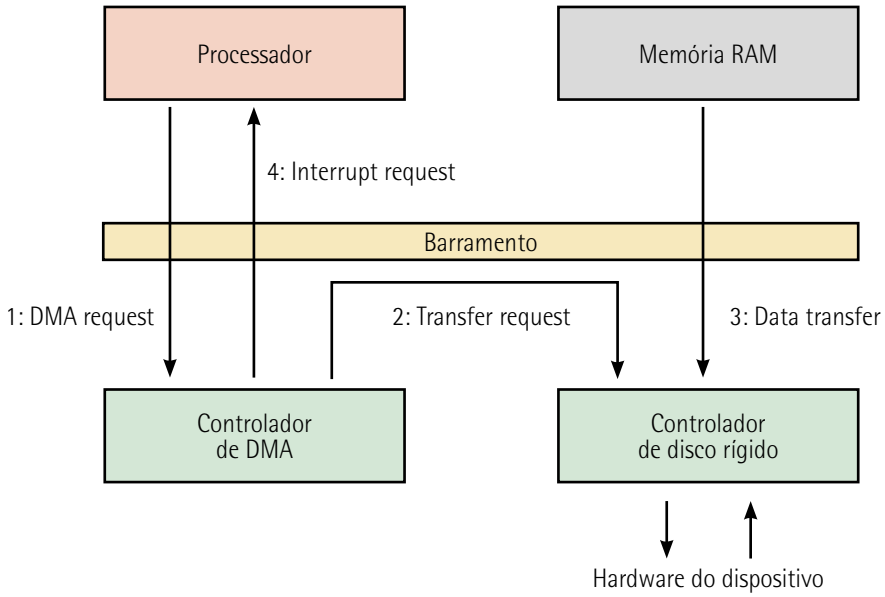

Figura 43 – Funcionamento do acesso direto à memória

Fonte: Maziero (2019, p. 257).

6.3.1 Interrupções

O hardware da CPU tem um fio denominado linha de solicitação de interrupção que a CPU verifica após a execução de cada instrução. Quando a CPU detecta que um controlador confirmou um sinal na linha de solicitação de interrupção, ela executa um armazenamento do estado atual e alterna a execução para a rotina de manipulação de interrupções, que está em um endereço fixo na memória.

O manipulador de interrupções determina a causa da interrupção, executa o processamento necessário, realiza uma recuperação do estado anterior e executa a instrução de retorno de interrupção, ou return from interrupt, para retornar a CPU ao estado de execução antes da interrupção. Pode-se dizer que controlador do dispositivo envia uma interrupção ao confirmar um sinal na linha de solicitação de interrupção, a CPU recebe a interrupção e a encaminha para o manipulador de interrupções, e o manipulador desativa a interrupção ao atender ao dispositivo. A figura a seguir resume o ciclo de I/O dirigido por interrupções.

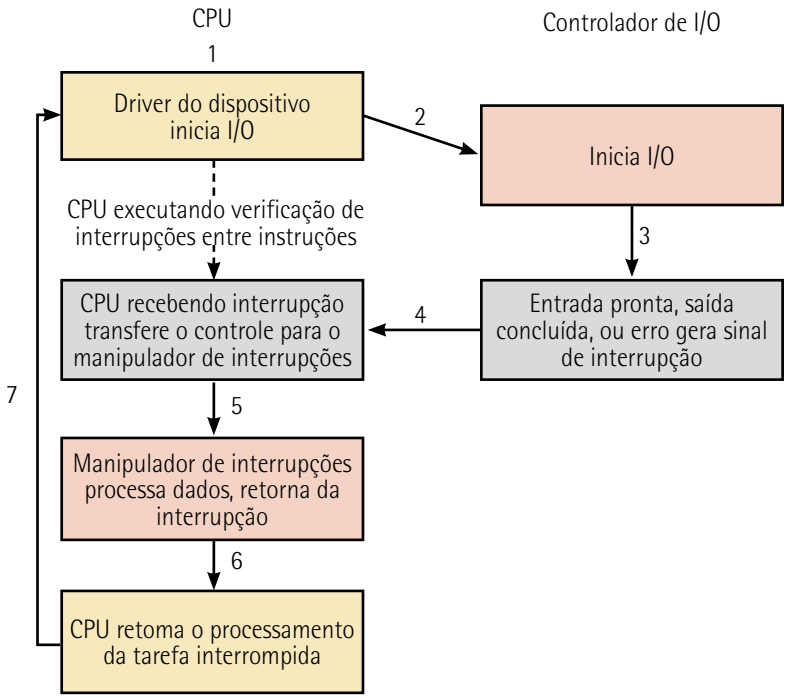

Figura 44 – Ciclo de E/S por interrupção

Fonte: Silberschatz, Galvin e Gagne (2015, p. 323).

O mecanismo básico de interrupção que acabamos de descrever habilita a CPU a responder a um evento assíncrono, como quando um controlador de dispositivo está pronto para o serviço. A maioria das CPUs tem duas linhas de solicitação de interrupção: um tipo não mascarável, reservado para eventos tais como erros de memória irrecuperáveis, e outro tipo mascarável, que pode ser desativado pela CPU antes da execução de sequências de instruções críticas que não devem ser interrompidas. A interrupção mascarável é usada pelos controladores de dispositivos para solicitar serviço.

O mecanismo de interrupção aceita um endereço que corresponde ao número de seleção da rotina de manipulação de interrupções específica em um pequeno conjunto. Na maioria das arquiteturas, esse endereço é um deslocamento em uma tabela chamada de vetor de interrupções. Entretanto, os computadores possuem mais dispositivos e, consequentemente, mais manipuladores de interrupções do que elementos de endereço no vetor de interrupções.

No mecanismo de interrupção, também é implementado um sistema de níveis de prioridades de interrupções. Esses níveis habilitam a CPU a atrasar a manipulação de interrupções de baixa prioridade sem mascarar todas as interrupções e possibilita que uma interrupção de alta prioridade intercepte a execução de uma interrupção de baixa prioridade.

Um SO moderno interage com o mecanismo de interrupção de várias maneiras. Em tempo de inicialização, ele sonda os barramentos de hardware para determinar quais dispositivos estão presentes e instala os manipuladores de interrupção correspondentes no vetor de interrupções. Durante operações de E/S, os diversos controladores de dispositivos lançam interrupções quando

estão prontos para o serviço. Essas interrupções significam que a saída foi concluída, que dados de entrada estão disponíveis, ou que uma falha foi detectada. O mecanismo de interrupção também é usado para manipular uma grande variedade de exceções, tais como a divisão por zero, o acesso a um endereço de memória protegido ou inexistente ou a tentativa de executar uma instrução privilegiada em modalidade de usuário. Os eventos que disparam interrupções possuem uma propriedade em comum: são ocorrências que induzem o SO a executar uma rotina autocontida urgente.

Outro exemplo de interrupção é encontrado na implementação de chamadas de sistema. Normalmente, um programa usa chamadas de bibliotecas para emiti-las. As rotinas de biblioteca verificam os argumentos fornecidos pela aplicação, constroem uma estrutura de dados para transportar os argumentos para o kernel e, então, executam uma instrução especial chamada interrupção de software, ou interceptação. Essa instrução identifica o serviço de kernel desejado. Quando um processo executa a instrução de interceptação, o hardware de interrupções armazena o estado do código do usuário, alterna o processador para a modalidade de kernel e despacha para a rotina do kernel que implementa o serviço solicitado. A interceptação recebe uma prioridade de interrupção relativamente baixa comparada às atribuídas a interrupções de dispositivos. Dessa forma, a execução de uma chamada de sistema em nome de uma aplicação é menos urgente do que o atendimento de um controlador de dispositivos antes que sua memória interna estoure e ocorra perda de dados.

Lembrete

A interrupção é um evento que requer a atenção do SO e pode ser originada em diversas situações, tais como uma interrupção externa, quando um dispositivo periférico requer uma operação de E/S ou uma interrupção interna, gerada pela execução de um comando.

Além de controlar os processos, a memória, o SO possui o controle de todos os dispositivos de E/S. Dessa forma, ele envia comandos para os periféricos tanto para leitura e escrita quanto para interceptar interrupções e tratar erros.

6.4 Classes de dispositivos

Visando à simplificação para construção de aplicações e camadas mais elevadas do próprio SO, agrupam-se os dispositivos de E/S em classes ou famílias com características similares, para os quais uma interface genérica é definida. Por exemplo, discos rígidos IDE, SATA, discos SSD e DVD-ROMs realizam basicamente o mesmo propósito: armazenar arquivos. Entretanto, possuem características mecânicas e elétricas distintas.

Uma das classes é a dos dispositivos orientados a caracteres, cujas transferências de dados são sempre feitas byte por byte, em sequência. Ela pode ser entendida como um fluxo contínuo de entrada ou de saída de bytes que não realiza operações de posicionamento, limitando-se a enviar ou receber uma sequência de caracteres. Não é possível alterar o valor de um byte que já foi enviado, dada a característica sequencial do fluxo de dados. Os exemplos mais comuns de dispositivos orientados

a caracteres são ligados às interfaces paralelas e seriais do computador, como mouse e teclado. Os terminais de texto e modems de transmissão de dados por linhas seriais, como em linhas telefônicas, também fazem parte da classe.

Outra classe é a dos dispositivos orientados a blocos, nos quais as operações de E/S de dados são feitas usando blocos de bytes de tamanho fixo. Esses blocos são lidos ou escritos em posições específicas do dispositivo, isto é, são endereçáveis. Com isso, as operações de leitura ou escrita podem ser realizadas de maneira independente entre os blocos. Cada bloco possui seu próprio endereço e um tamanho que pode variar entre 512 e 32.768 bytes. Discos rígidos, CD-ROMs, DVDs, pendrives, fitas magnéticas e outros dispositivos de armazenamento são exemplos típicos dessa família. A figura a seguir apresenta o funcionamento de um dispositivo orientado a bloco.

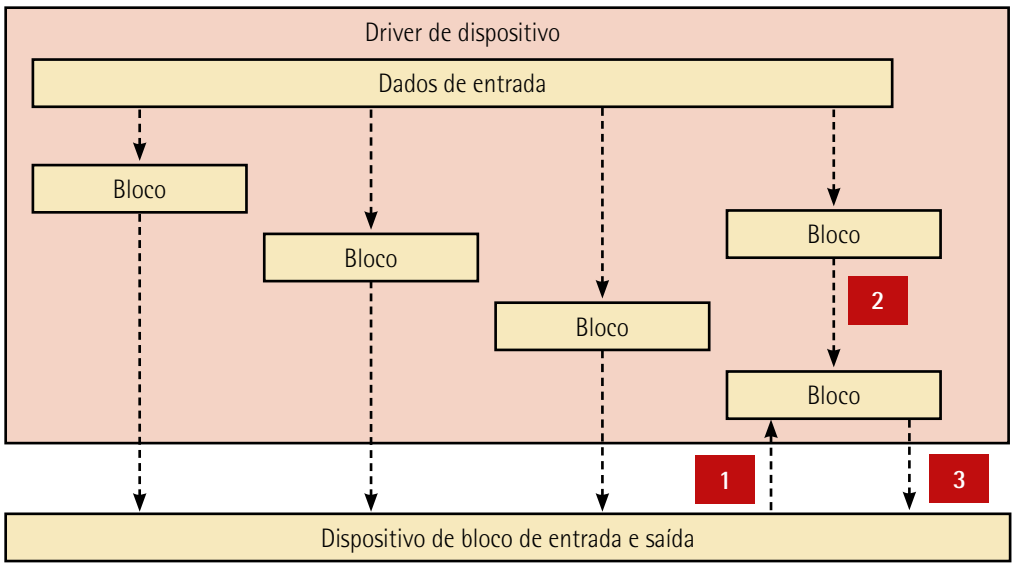

Figura 45 – Dispositivo orientado a bloco

Fonte: Córdova Jr., Ledur e Morais (2018, p. 90).

Na classe dos dispositivos de rede, estão aqueles que permitem enviar e receber mensagens entre processos e computadores distintos. As mensagens são blocos de dados de tamanho variável, com envio e recepção feitas de forma sequencial, sendo que não é possível alterar o conteúdo de uma mensagem que já foi enviada. As interfaces Ethernet para redes cabeadas, Wi-Fi ou wireless fidelity para redes sem fio, Bluetooth para redes de curto alcance e GPRS são bons exemplos.

Temos ainda os dispositivos gráficos, que permitem a renderização de texto e gráficos em terminais de vídeo, tornando permanente o trabalho de processamento digital e salvando o resultado em um arquivo final. Considerando os requisitos de desempenho, que são mais rigorosos para jogos online e filmes, esses dispositivos exigem uma elevada taxa de dados na transferência. Para isso, sua interface genérica é formada por funções para consultar e configurar o dispositivo gráfico e uma área de memória compartilhada entre o processador e o dispositivo, usualmente denominada frame buffer, o qual permite acesso direto à memória de vídeo. Programas ou bibliotecas com interação direta com o dispositivo

gráfico possuem acesso a essa área de memória por meio de bibliotecas específicas, como DirectX em ambientes Windows ou DRI, direct rendering engine, no Linux.

Existem dispositivos que não se enquadram diretamente nessas categorias, como interfaces de áudio, receptores de GPS ou sensores de temperatura. Para essas situações, alguns SOs optam por criar classes adicionais, como o Windows, enquanto outros buscam enquadrá-los em uma das famílias já existentes, como os sistemas em Unix. No Linux, por exemplo, as aplicações obtêm acessos aos dispositivos como se fossem orientados a caracteres, isto é, uma sequência de bytes enviada ao dispositivo é tratada como um fluxo de áudio a ser reproduzido. É função do driver do dispositivo e da camada de interface genérica transformar essa interface orientada a caracteres nas operações de baixo nível necessárias para reproduzir o fluxo de áudio desejado.

6.5 Dispositivos de hardware no Linux

Para aproveitar melhor o SO, é extremamente útil a um programador identificar os arquivos que compõem o hardware do computador, para que se tenha um bom resultado ao utilizar o Linux. Por exemplo, os periféricos são identificados através de arquivos especiais no diretório /proc.

Cada periférico precisa de um canal de comunicação específico para o intercâmbio de dados com o processador e pode conter mais de um endereço I/O. As portas estão localizadas no arquivo /proc/ioports.

Segundo Negus (2014), cada fluxo de comunicação entre os periféricos e o processador deve ser sinalizado, de modo que o processador detecte que existem bits a serem lidos. Tais mensagens de ativação poderão ser verificadas através do comando dmesg. As IRQs relacionadas aos dispositivos de E/S estão localizadas no arquivo /proc/interrupts.

O Linux possui a tecnologia plug and play, responsável pelo reconhecimento automático de dispositivos, de forma a facilitar a expansão segura dos computadores e suprimir a configuração manual desses dispositivos. Para consultar os periféricos plug and play, pode-se utilizar o comando lspnp ou consultá-los no diretório /proc/bus/pnp.

O Linux também identifica os discos rígidos existentes no computador e as partições existentes. O arquivo que informa os HDs e suas devidas partições reconhecidas pelo sistema estão em /proc/partitions. Caso o computador possua um HD com duas partições, eles serão encontrados nos arquivos no diretório /dev.

A seguir, descreveremos a manipulação de hardwares e dispositivos em Linux, já que ele possui uma sequência de operações própria.

O processo de boot no Linux é realizado em duas etapas. Na primeira delas, são realizados testes de POST (power-on self-tests) com o objetivo de verificar a integridade física do equipamento e detectar componentes básicos, como: memória, discos, placa de vídeo etc.

Na segunda etapa, identifica-se o dispositivo de inicialização e também é verificado, no registro mestre de inicialização (MBR), o boot loader, caso este seja o sistema a ser carregado. O gerenciador de boot indica ao BIOS os arquivos que deverão ser carregados para a memória e aqueles que devem ser executados para inicialização do equipamento. Então, o arquivo /boot/vmlinuz (kernel) é executado e o então arquivo/boot/initrd (módulos) é carregado na memória para o acesso aos periféricos.

Lembrete

O registro mestre de inicialização ou master boot record (MBR) é um tipo especial de setor de inicialização no começo dos discos rígidos particionados de computadores.

Um conflito de hardware ocorre quando um ou mais dispositivos utilizam a mesma IRQ, I/O ou DMA. Um sistema com configurações de hardware em conflito funciona de forma instável, ocasionando travamentos constantes, mal funcionamento de um ou mais dispositivos e até mesmo, em casos mais graves, a perda de dados. Para resolução de seus conflitos, é necessário o conhecimento da configuração de cada dispositivo no sistema.

Saiba mais

A fim de obter uma visão mais ampla a respeito do boot no Linux, bem como de outros assuntos referentes ao sistema, leia a obra a seguir:

NEGUS, C. *Linux, a Bíblia*: o mais abrangente e definitivo guia sobre Linux. Rio de Janeiro: Alta Books, 2014.

Resumo

Nesta unidade, vimos que embora os aspectos do hardware de I/O sejam complexos quando considerados no nível de detalhe do projeto de componentes eletrônicos, os conceitos que acabamos de descrever são suficientes para nos habilitar a entender muitos recursos de I/O dos SOs.

Consequentemente, acentuamos que tais recursos incluem: barramento ou bus; controlador; porta de I/O e seus registradores; transferência de dados entre os dispositivos de E/S, o controlador e as interfaces do sistema computacional; e delegação desse trabalho a um controlador de DMA para grandes transferências.

Por fim, demonstramos ainda que os dispositivos de E/S podem ser classificados por tipo de conexão, tipo de transferência de dados e tipo de compartilhamento de dados.

Exercícios

Questão 1. (FAFIPA 2020, adaptada) Considere uma situação hipotética em que uma analista de Tecnologia da Informação foi solicitada para configurar um servidor de arquivos implementando técnicas RAID (redundant array of inexpensive disks). Para realizar a configuração do RAID, ela pretende utilizar o GNU/Linux Debian pelo fato de esse SO suportar diversos tipos de RAID via software. Durante o processo de configuração do RAID, ela optou pelo RAID do tipo 0. Qual foi a motivação dessa escolha?

A) Implementar redundância.

B) Replicar o conteúdo do disco principal.

C) Otimizar o desempenho.

D) Implementar um subsistema JBOD.

E) Implementar a redundância baseada em paridade.

Resposta correta: alternativa C.

Análise da questão

Um sistema RAID é constituído por dois ou mais discos rígidos, que são enxergados pelo SO e pelas aplicações como um único disco lógico. Existem várias formas de organização de um conjunto de discos rígidos em RAID, e cada uma apresenta suas próprias características de desempenho e confiabilidade. Elas são denominadas "níveis RAID".

No RAID nível 0, os discos físicos, ou as partições, estão simplesmente concatenados em sequência para construir um disco lógico. Teoricamente, essa estratégia possibilita maior velocidade de leitura e de escrita, já que os acessos a blocos em discos físicos diferentes podem ser realizados paralelamente. Nessa abordagem, não há redundância de dados e, desse modo, se um disco falhar, todos os blocos armazenados nele serão perdidos.

Considerando sua característica de alta velocidade de escrita e de leitura, o RAID 0 foi escolhido com a motivação de otimizar o desempenho do sistema.

Questão 2. (Instituto AOCP 2018, adaptada) A unidade central de processamento (CPU) e a memória não são os únicos recursos que o SO tem de gerenciar. Os dispositivos de E/S também interagem intensivamente com o SO. Os dispositivos são constituídos, geralmente, por duas partes: o controlador e o dispositivo propriamente dito. O controlador é um chip, ou um conjunto de chips em uma placa, que controla fisicamente o dispositivo. Uma vez que cada tipo de controlador é diferente, diferentes programas são necessários para controlá-los. O programa que se comunica com um controlador, emitindo comandos a ele e aceitando respostas, é denominado:

A) Driver de saída.

B) Driver de E/S.

C) Driver de dispositivo.

D) Driver de configuração.

E) Driver de entrada.

Resposta correta: alternativa C.

Análise da questão

O device driver, ou driver de dispositivo, é um programa utilizado para criar uma interface entre o SO e um dispositivo de E/S. Os drivers de dispositivo atuam recebendo comandos gerais sobre acessos aos dispositivos e traduzindo-os para comandos específicos para determinado dispositivo, que serão executados pelo controlador.