Capítulo 1: Fundamentos de Entrada e Saída (E/S)

A Entrada e Saída (E/S), comumente referida como I/O (Input/Output), é fundamental para a funcionalidade de um computador, pois permite a interação entre o sistema operacional e os diversos dispositivos periféricos [1].

1.1 O que é E/S e a Função do SO

Os Sistemas Operacionais (SOs) controlam os dispositivos de E/S como parte de seus serviços [2]. Para isso, eles realizam as seguintes operações essenciais:

•

Emitem comandos para os dispositivos [2].

•

Transferem dados entre os dispositivos e a memória [2].

•

Tratam interrupções e erros [2].

É também responsabilidade do SO prover uma interface uniforme entre os dispositivos e o restante do sistema [1-3]. Essa uniformidade é crucial para que os processos possam realizar E/S através de chamadas de sistema padronizadas, como read e write [1, 3].

1.2 Categorização de Dispositivos de E/S

De forma simplificada, os dispositivos de E/S são categorizados com base em suas características de transferência de dados e tipo de acesso [2, 4]:

•

Dispositivos de Bloco: Caracterizados por transferências de tamanhos maiores (tipicamente de 512 bytes a 64 KB) e suporte a acesso aleatório [2]. Exemplos podem incluir discos rígidos e SSDs [3].

•

Dispositivos de Caractere: Envolvem transferências de dados de menor tamanho e acesso sequencial [2]. Exemplos incluem teclados, mouses, impressoras e interfaces de rede [1].

1.3 Componentes Físicos de E/S: Dispositivos e Controladores

As unidades de E/S são geralmente compostas por dois componentes principais [5]:

•

Um componente mecânico: A parte física do dispositivo [5].

•

Um componente eletrônico: Conhecido como controlador do dispositivo ou adaptador [5].

Esses controladores são responsáveis por traduzir os comandos recebidos do processador em instruções específicas para os dispositivos e vice-versa [6]. Isso significa que o SO não precisa conhecer os detalhes específicos de cada dispositivo, pois o controlador lida com essa complexidade [6]. As funcionalidades do controlador são comumente providas por circuitos na placa-mãe (on-board) ou em placas adaptadoras conectadas a barramentos do computador [5, 7].

Os dispositivos físicos são conectados aos seus controladores apropriados através de interfaces como SATA, SCSI, USB, Thunderbolt, FireWire, Serial ou Paralela [5]. O SO pode descobrir quais dispositivos e controladores estão presentes no sistema consultando informações padronizadas fornecidas pela BIOS (SMBIOS) ou pelo ACPI (Advanced Configuration and Power Interface) [7].

Capítulo 2: Arquiteturas de Interconexão de Hardware

Para entender como o Sistema Operacional interage com o hardware, é essencial compreender as interconexões entre processadores, memória e controladores de dispositivos [8]. A arquitetura dessas interconexões tem evoluído significativamente ao longo do tempo [8].

2.1 Evolução das Arquiteturas Intel: Northbridge/Southbridge

Historicamente, as placas-mãe utilizavam uma arquitetura de chipset de duas peças, composta por um northbridge e um southbridge [9-11].

•

O Northbridge (ou host bridge) era conectado diretamente à CPU via o front-side bus (FSB) e era responsável por tarefas que exigiam o mais alto desempenho [10]. Ele também era conhecido como Memory Controller Hub (MCH) [10]. Funções como o controle de memória, as pistas PCI Express para placas de expansão e outras funções de alto desempenho eram gerenciadas pelo northbridge [10, 12].

•

O Southbridge implementava as funcionalidades mais lentas da placa-mãe [11]. Em chipsets Intel, era chamado de I/O Controller Hub (ICH) [11]. Ele não era conectado diretamente à CPU, mas sim ao northbridge, que por sua vez o ligava à CPU [13].

Essa arquitetura mais antiga, onde todas as portas de baixa velocidade eram conectadas ao barramento PCI, foi superada pela necessidade de maior velocidade [14, 15].

2.2 Intel Hub Architecture (IHA)

A Intel Hub Architecture (IHA), também conhecida como Accelerated Hub Architecture (AHA), foi a arquitetura da Intel para a família de chipsets 8xx, introduzida por volta de 1999 com o Intel 810 [16, 17].

•

Utilizava um Memory Controller Hub (MCH) que se conectava a um I/O Controller Hub (ICH) através de um barramento de 266 MB/s [16, 17].

•

O chip MCH suportava memória e AGP (posteriormente substituído por PCI Express na série 9xx de chipsets) [16, 17].

•

O chip ICH fornecia conectividade para PCI (revisão 2.2 antes da série ICH5 e 2.3 a partir da ICH5), USB (versão 1.1 antes da série ICH4 e 2.0 a partir da ICH4), som (AC'97, Azalia adicionado na série ICH6), discos rígidos IDE (suplementado por Serial ATA a partir da série ICH5, totalmente substituído IDE a partir da série ICH8 para desktops e ICH9 para notebooks) e LAN [16, 17].

•

A Intel alegou que, devido ao canal de alta velocidade entre as seções, a IHA era mais rápida do que o design anterior northbridge/southbridge [14, 15]. A IHA também otimizava a transferência de dados com base no tipo de dado [14, 15].

•

Uma revisão posterior, Intel Hub Interface 2.0, foi usada em chipsets de servidor da linha E7xxx, permitindo caminhos de dados dedicados para transferir mais de 1.0 GB/s [14, 15].

A IHA é atualmente considerada obsoleta e não é mais utilizada [10, 18].

2.3 Platform Controller Hub (PCH) e Direct Media Interface (DMI)

A arquitetura Platform Controller Hub (PCH), introduzida em 2009 com os chipsets Intel 5 Series, supersede a Intel Hub Architecture [12, 18, 19]. Seu design visava resolver o gargalo de desempenho entre o processador e a placa-mãe que existia na arquitetura anterior [9, 20].

•

Como solução para o gargalo, várias funções pertencentes aos chipsets northbridge e southbridge foram reorganizadas [12, 21].

•

O northbridge e suas funções foram eliminados por completo: O controlador de memória, as pistas PCI Express para placas de expansão e outras funções do northbridge foram incorporadas ao die da CPU (como um system agent na Intel ou empacotadas no processador em um I/O die na AMD Zen 2) [12, 21]. Isso liberou grande parte da largura de banda necessária para os chipsets [22].

•

O PCH incorporou algumas das funções restantes do northbridge (como o clocking) além de todas as funções do southbridge, substituindo-o [21].

•

A conexão entre o PCH e a CPU é feita através da Direct Media Interface (DMI) [19, 21, 23]. A DMI é uma ligação proprietária da Intel que foi introduzida pela primeira vez em 2004 entre os chipsets 9xx e o ICH6 [23].

◦

A implementação original da DMI fornecia 10 Gbit/s (1 GB/s) em cada direção usando uma ligação ×4 [23].

◦

A DMI 2.0 (2011) dobrou a taxa de transferência de dados para 2 GB/s com uma ligação ×4 [23].

◦

A DMI 3.0 (2015) permite uma taxa de transferência de 8 GT/s por pista, totalizando 3.93 GB/s para a ligação CPU-PCH [23].

•

O PCH fornece recursos e interfaces de alto valor, incluindo controladores para DRAM, múltiplos displays gráficos, várias interfaces USB, controladores SATA, portas Ethernet LAN e Intel® High Definition Audio (Intel® HD Audio) [24].

Em sistemas mais recentes, especialmente em modelos de baixo consumo, a Intel começou a incorporar os controladores de E/S do southbridge diretamente no pacote da CPU, eliminando o PCH e resultando em um design system in package (SOP) [22]. CPUs de servidor e laptop AMD também adotam um design system on chip (SoC) autônomo que não requer um chipset [25].

Capítulo 3: Métodos de Interação entre CPU e Controladores de E/S

A interação entre a CPU (executando código do SO) e os controladores de dispositivos é fundamental para as operações de E/S [26-28]. Há duas formas básicas para essa interação ocorrer [26].

3.1 Visão Geral dos Métodos de Interação

A comunicação entre processadores e controladores acontece através de um barramento [27]. Para enviar comandos e dados, verificar o status das operações e receber dados dos dispositivos, é necessário ler e escrever nesse barramento [27]. Como todos os controladores conectados a um barramento têm acesso às informações ali transmitidas, um mecanismo de endereçamento é usado para selecionar o controlador desejado [29].

3.2 E/S Mapeada em Portas (Port-Mapped I/O)

A E/S mapeada em portas (Port-Mapped I/O), também conhecida como E/S Isolada, utiliza um espaço de endereçamento separado da memória principal [30].

3.2.1 Uso de Registradores de Controle e Buffers

Os controladores possuem registradores de controle internos que servem para a passagem de comandos e para verificações de status de suas operações [26]. Alguns controladores também têm buffers de dados para comunicação [26].

3.2.2 Instruções IN e OUT

A interação com os controladores via E/S mapeada em portas é feita usando instruções específicas do processador, como IN para leitura e OUT para escrita no barramento [26, 30, 31]. Essas são instruções privilegiadas [31]. Em arquiteturas x86 e x86-64, existem variações dessas instruções (outb, outw, outl) para copiar 1, 2 ou 4 bytes entre um registrador da CPU (como EAX ou suas subdivisões) e uma porta de E/S especificada, que é atribuída a um dispositivo de E/S [30].

3.2.3 Endereçamento de Portas e /proc/ioports

Os endereços de portas de E/S (IO ports) são pré-definidos pela indústria em faixas para cada tipo de dispositivo [29]. Dentro dessas faixas, há alguma liberdade de configuração para cada controlador [29]. Para controladores PCI, há um protocolo para negociação de endereços [29]. Em sistemas Linux, é possível verificar a relação das portas de E/S usadas pelos controladores através do arquivo /proc/ioports [31]. Exemplos de entradas nesse arquivo incluem 0060-0060 : keyboard e 03f8-03ff : serial [32-35].

3.3 E/S Mapeada em Memória (Memory-Mapped I/O)

A E/S mapeada em memória (Memory-Mapped I/O) é outra forma de interação, considerada mais eficiente e simplificada para o SO [36].

3.3.1 Conceito e Vantagens

Nessa abordagem, o mesmo espaço de endereçamento é usado para endereçar tanto a memória principal quanto os dispositivos de E/S [36, 37]. Os registradores e a memória dos dispositivos de E/S são mapeados para valores de endereço específicos [37]. Assim, quando a CPU acessa um endereço, ele pode se referir à RAM física ou à memória de um dispositivo de E/S [37]. Isso permite que as instruções da CPU usadas para acessar a memória também sejam usadas para acessar os dispositivos [37].

3.3.2 Filtragem e Desvio de Acessos

Para acomodar os dispositivos de E/S, áreas de endereços usadas pela CPU devem ser reservadas para E/S e não podem estar disponíveis para a memória física normal [37]. Essas faixas de endereço são filtradas no chipset de controle de acesso do processador à memória, desviando os acessos para os dispositivos responsáveis pelas faixas de endereço específicas [38]. Isso é negociado via protocolos de controle dos dispositivos conectados ao barramento PCI, por exemplo [36].

3.3.3 O Arquivo /proc/iomem e o Mapeamento de Memória

Em sistemas Linux, a relação de endereços de memória mapeados para a comunicação com controladores de dispositivos pode ser consultada no arquivo /proc/iomem [37, 39]. O Linux usa a chamada ioremap para realizar esses mapeamentos, utilizando instruções específicas de cada processador [40-42]. Exemplos de entradas em /proc/iomem incluem 000a0000-000bffff : PCI Bus 0000:00 (que pode incluir memória de vídeo) e d0000000-dfffffff : 0000:00:02.0 (que representa um dispositivo PCI) [35, 43-47].

3.3.4 O Mapa de Memória do x86

O mapa de memória física do sistema é crucial para o SO [39, 48]. Ele define para qual dispositivo (RAM, placa de vídeo, placa PCI, BIOS) cada região de endereços físicos deve ser roteada [39, 49].

•

Espaço de Endereçamento em Modo Real (< 1 MiB): No momento da inicialização do PC (BIOS), o CPU opera em Modo Real, onde a área de memória é limitada [50].

◦

IVT (Interrupt Vector Table): 0x00000000 - 0x000003FF (1 KiB) [51].

◦

BDA (BIOS Data Area): 0x00000400 - 0x000004FF (256 bytes) [51, 52].

◦

EBDA (Extended BIOS Data Area): Área de tamanho variável, geralmente abaixo de 0xA0000 [51, 53, 54].

◦

Memória de Vídeo: 0x000A0000 - 0x000BFFFF (128 KiB), mapeada para hardware [51].

◦

BIOS de Vídeo e Expansões de BIOS: Regiões como 0x000C0000 - 0x000EFFFF [51].

◦

Motherboard BIOS: 0x000F0000 - 0x000FFFFF (64 KiB) [51].

•

Memória Estendida (> 1 MiB): A região da RAM acima de 1 MiB não é padronizada, e pode conter regiões de hardware mapeado em memória ou tabelas ACPI [54, 55].

◦

O clássico "buraco" na memória do PC entre 640KB e 1MB e um buraco maior para placas de vídeo e dispositivos PCI são resultado do mapeamento de endereços de memória para esses dispositivos, não para RAM [39].

◦

Sistemas de 32 bits frequentemente têm problemas para usar 4 GB de RAM devido a endereços físicos reservados para dispositivos [56].

◦

Em modo de 64 bits, é possível "reclamar" memória usando endereços físicos acima da RAM total do sistema para acessar regiões de RAM que foram "roubadas" por dispositivos da placa-mãe, com o auxílio do chipset [56].

Capítulo 4: Estratégias de Operações de E/S

Compreendendo como o processador e os controladores interagem, as operações de E/S são realizadas comumente em variações de três estratégias principais [57].

4.1 Introdução às Estratégias de E/S

O SO gerencia a E/S buscando eficiência. As estratégias visam minimizar o tempo de espera do processador por operações de E/S, que são inerentemente mais lentas que a CPU [58].

4.2 E/S Programada (Polling / Busy Waiting)

Nessa estratégia, o processador envia um comando ao controlador e fica em espera pela conclusão da operação [57, 59].

•

O processador entra em um loop de escrita (ou leitura) de dados e espera pela conclusão do controlador, consultando uma variável de estado do controlador até que o dado enviado tenha sido processado ou um novo dado esteja disponível [59].

•

Como a velocidade do processador é muito superior à do controlador, ele acaba em uma "espera ocupada" (busy waiting), que não é produtiva [58]. A vantagem é que o dado é lido/escrito assim que possível [58].

4.3 E/S Orientada a Interrupção

Para evitar a espera ocupada, o SO instrui o controlador a realizar a transferência e, em vez de ficar testando constantemente a condição de conclusão, direciona o processador para realizar outra atividade [57, 58].

•

Quando o controlador está pronto para receber ou fornecer um novo dado, ele gera uma interrupção [60].

•

O processador, ao receber a interrupção, salva o endereço de sua próxima instrução e se direciona para a execução da rotina de tratamento apropriada [60].

•

Nesse momento, o SO lê ou escreve o novo dado no controlador, e a operação se repete [60].

4.4 Acesso Direto à Memória (DMA)

Essa estratégia também é baseada no uso de interrupções, mas explora a capacidade dos controladores de realizar transferências diretamente entre o dispositivo de E/S e a memória, sem a intervenção direta da CPU para cada byte [57, 60, 61].

4.4.1 Mecanismo e Vantagens

•

O SO reserva previamente espaços na memória para uso como buffers nas transferências de E/S [61].

•

Em uma operação de leitura, por exemplo, o SO informa ao controlador qual volume de dados deve ser lido do dispositivo e onde esses dados devem ser colocados na memória (os buffers pré-alocados) [61]. O mesmo se aplica à escrita [61].

•

Usando a capacidade de requisitar acesso à memória via DMA, o controlador transfere os dados para/da memória [61].

•

Ao concluir a transferência completa do bloco de dados, o controlador gera uma única interrupção, ativando o SO para tratar da conclusão da operação [61, 62].

•

A principal vantagem é que o SO só é ativado ao final da transferência completa do bloco, liberando o processador para outras tarefas durante a transferência dos dados [62].

4.4.2 Unidade de Gerenciamento de Memória de E/S (IOMMU)

Uma Input-Output Memory Management Unit (IOMMU) é uma unidade de gerenciamento de memória (MMU) que conecta um barramento de E/S com capacidade de DMA à memória principal [63].

•

Similar a uma MMU tradicional (que traduz endereços virtuais da CPU para físicos), a IOMMU mapeia endereços virtuais visíveis pelo dispositivo (endereços de dispositivo ou E/S) para endereços físicos [63-65].

•

A IOMMU também pode proteger a memória contra dispositivos defeituosos ou maliciosos, pois intercepta o acesso de um dispositivo de E/S à memória do sistema, determina o domínio ao qual o dispositivo foi atribuído e usa as entradas de TLB (ou tabelas de páginas de E/S) para verificar se o acesso é permitido e qual é a localização real na memória [64-66].

•

Outras funcionalidades da IOMMU incluem [67]:

◦

Substituir mecanismos como o Graphics Address Remapping Table (GART).

◦

Remapear endereços acima de 4GB para dispositivos de E/S que não suportam endereçamento de 64 bits.

◦

Permitir que um SO convidado em uma máquina virtual tenha controle direto de um dispositivo.

◦

Fornecer controle de acesso de dispositivos à memória com granularidade de página.

◦

Permitir acesso direto de um dispositivo ao espaço de E/S do usuário.

◦

Permitir a entrega direta de interrupções a um sistema operacional convidado.

◦

Filtrar e remapear interrupções.

◦

Compartilhar o espaço de endereço virtual do processo com dispositivos periféricos selecionados.

◦

Isolar/colocar dispositivos em sandbox para prevenir DMA malicioso acessando dados sensíveis de segurança do SO e do usuário na memória.

◦

Impor políticas de segurança do SO para acesso a dados.

•

Na arquitetura x86, antes da divisão de funcionalidade entre northbridge e southbridge para a CPU e PCH, a virtualização de E/S era realizada pelo chipset e não pela CPU [63].

Capítulo 5: Interrupções: Mecanismo Essencial para E/S

Interrupções são um mecanismo fundamental para a interação entre hardware e software, permitindo que eventos assíncronos sejam tratados pela CPU de forma eficiente [68].

5.1 O que são Interrupções?

Uma interrupção é um sinal enviado ao processador para indicar que há um evento que precisa de tratamento [68, 69]. O ciclo de busca e execução de instruções do processador só é alterado na ocorrência de uma interrupção [68, 69].

5.2 Tipos de Interrupções: Exceções e Interrupções Externas

As interrupções podem ser classificadas em [68]:

•

Interrupções Internas (Exceções): São síncronas e ocorrem em decorrência de erros ou eventos durante a execução de instruções [68]. Exemplos incluem divisão por zero, tentativa de acesso a uma posição inválida da memória, instrução inexistente ou falta de página [68]. Exceções são classificadas como faults, traps e aborts [70].

•

Interrupções Externas (Hardware): São assíncronas e geradas por um controlador de dispositivo que requer a atenção da CPU [68, 71]. Exemplos incluem interrupções geradas por controladores de disco, teclado, interface de rede, ou qualquer dispositivo conectado às linhas de controle do barramento do computador [71]. Interrupções de hardware que podem ser entregues ao processador via pino INTR ou através do APIC local são chamadas de interrupções de hardware mascaráveis [72].

5.3 Interrupções Geradas por Software (Instrução INT n)

É possível gerar uma interrupção programaticamente através da instrução assembly INT n [70, 73]. Esta instrução é um mnemônico para uma chamada gerada por software para um tratador de interrupção, usando um número de vetor de interrupção como operando (por exemplo, INT 35 força uma chamada implícita ao tratador de interrupção 35) [70, 71, 73]. A interrupção 0x80 (128) tem sido usada para chamadas de sistema [73, 74].

5.4 Processamento de Interrupções pelo Processador

Na ocorrência de uma interrupção ou exceção, o processador deve passar à execução de uma rotina de tratamento apropriada [72].

5.4.1 Salvamento de Estado e Mudança para Modo Kernel

•

Uma interrupção (externa ou exceção) causa uma mudança do processador para o modo kernel, onde o sistema operacional realiza o tratamento [72, 75].

•

Para suspender a execução atual e permitir o retorno posterior, o processador salva automaticamente o estado essencial: os valores dos registradores de Flags, CS (Code Segment) e EIP (Instruction Pointer) são copiados para a pilha do modo kernel da thread atual (push EFLAGS, push CS, push EIP) [69, 76-78].

•

Para cada thread, o SO deve reservar uma área de memória para a pilha em cada um dos anéis de execução (tipicamente anéis 0 e 3 são usados pelos SOs) [76].

5.4.2 O Vetor de Interrupções (IDT)

•

Após salvar o endereço da próxima instrução da thread atual, o processador é desviado para a execução da rotina de tratamento [77].

•

Existe uma posição na memória chamada Vetor de Interrupções (IDT - Interrupt Descriptor Table), que contém os endereços de cada uma das rotinas de tratamento de interrupção [77, 79-81].

•

A localização (endereço e tamanho) do IDT é definida por um registrador na CPU (IDTR) [77, 81].

•

O conteúdo do IDT é preenchido pelo SO, geralmente durante a fase de boot, quando as rotinas de tratamento são carregadas para a memória [75, 79].

•

O processador usa o número da interrupção ou exceção como um índice para o IDT, de onde obtém os novos valores para CS e EIP, retomando então seu ciclo de busca e execução, mas agora executando a rotina de tratamento [79, 80].

•

O IDT é um array de descritores de 8 bytes (em modo protegido) e pode conter até 256 descritores, correspondendo aos vetores de interrupção de 0 a 255 [79, 80, 82].

5.4.3 Retorno de uma Interrupção (Instrução IRET)

•

A rotina de tratamento deve terminar seu trabalho executando a instrução IRET (ou IRETD), que restaura da pilha os valores de EIP, CS e EFLAGS, possibilitando a continuação da execução da thread interrompida [81, 83].

•

Uma interrupção faz com que o processador seja desviado para a execução do SO, ou seja, o SO reassume o controle da CPU sempre que acontece uma interrupção [83].

5.5 Advanced Programmable Interrupt Controller (APIC)

O Advanced Programmable Interrupt Controller (APIC), introduzido com o processador Pentium, é um componente chave no gerenciamento de interrupções em sistemas modernos, especialmente em sistemas multiprocessador [84].

5.5.1 Local APIC e I/O APIC

•

Cada processador possui um APIC local, que recebe interrupções de pinos do processador, de fontes internas e de um I/O APIC externo (ou outro controlador de interrupção externo), enviando-as ao núcleo do processador para tratamento [84, 85].

•

Em sistemas com múltiplos processadores (MP), o APIC local também envia e recebe mensagens de interrupção interprocessador (IPI) para/de outros processadores lógicos no barramento do sistema, permitindo distribuir interrupções entre os processadores ou executar funções em todo o sistema [85, 86].

•

O I/O APIC externo é parte do chipset do sistema da Intel [87]. Sua função principal é receber eventos de interrupção externos dos dispositivos de E/S e retransmiti-los aos APICs locais como mensagens de interrupção [87]. Em sistemas MP, ele distribui interrupções externas aos APICs locais de processadores selecionados [87].

•

Os registradores do APIC são mapeados em memória e podem ser lidos e escritos usando a instrução MOV [87].

5.5.2 Fontes de Interrupção para Local APIC

Os APICs locais podem receber interrupções de [86, 88, 89]:

•

Dispositivos de E/S conectados localmente: Interrupções originadas de dispositivos conectados diretamente aos pinos de interrupção locais do processador (LINT0 e LINT1), ou através de um controlador de interrupção tipo 8259 [88].

•

Dispositivos de E/S conectados externamente: Interrupções de dispositivos conectados aos pinos de entrada de interrupção de um I/O APIC, enviadas como mensagens de interrupção de E/S [88].

•

Interrupções Interprocessador (IPIs): Usadas por um processador Intel 64 ou IA-32 para interromper outro processador ou grupo de processadores no barramento do sistema, para auto-interrupções de software, encaminhamento de interrupções ou escalonamento preemptivo [86].

•

Interrupções geradas pelo temporizador APIC: O temporizador APIC local pode ser programado para enviar uma interrupção local ao seu processador associado quando uma contagem programada é atingida [86].

•

Interrupções de contador de monitoramento de desempenho: Quando um contador de monitoramento de desempenho overflows [86].

•

Interrupções do sensor térmico: Quando o sensor térmico interno é acionado [89].

5.6 Números de Vetores de Interrupção e /proc/interrupts

Os números das interrupções variam de 0 a 255 [74, 82]:

•

A faixa de 0 a 31 é reservada pelos desenvolvedores do processador para exceções e interrupções arquiteturalmente definidas [74, 82].

•

As posições de 32 a 255 são designadas para interrupções definidas pelo usuário (user-defined) e são geralmente atribuídas a interrupções geradas por dispositivos de E/S externos [74, 82]. Em um sistema Linux, é possível examinar a atribuição e a ocorrência de interrupções exibindo o conteúdo do arquivo /proc/interrupts [74].

Capítulo 6: Drivers de Dispositivos e Estrutura do Software de E/S

A complexidade da interação com dispositivos de E/S é gerenciada pelo sistema operacional de forma modular, sendo os drivers de dispositivos um componente chave nessa estrutura [90].

6.1 O Papel dos Drivers de Dispositivos

O device driver é a parte do software do SO responsável por interagir com um controlador físico de dispositivo [27].

•

Contém códigos específicos para tratar os detalhes da interação com um dispositivo (ou classe de dispositivos) [4].

•

Esses códigos são tipicamente escritos pelo fabricante do dispositivo, seguindo as especificações do SO [27].

•

Inclui a parte do tratamento de interrupção do dispositivo [4].

•

Pode ser organizado em partes: uma para operar na ativação do dispositivo e outra após a ocorrência da interrupção [4].

•

Um device driver comumente se auto-bloqueia após instruir o controlador, à espera de uma interrupção [4].

•

O código pode ser reentrante para tratar diversas solicitações simultâneas [4].

6.2 Implementação: Kernel Space vs. User Space

A implementação dos device drivers é comumente feita no espaço do kernel (kernel space) [91]. No entanto, existem projetos que os implementam no espaço do usuário (user space) [91].

6.3 Características Comuns do Software de E/S

Independentemente da implementação, algumas características são comuns ao software de E/S [3]:

•

Independência de dispositivos: As mesmas operações podem ser aplicadas sobre diferentes tipos de dispositivos de armazenamento (HD, SSD, memory stick, CD/DVD, etc.) [3].

•

Nomenclatura uniforme dos arquivos: Consistência na forma como os arquivos são nomeados nos diferentes sistemas de arquivos [92].

•

Tratamento de erros: Erros são comumente tratados o mais próximo possível de onde são identificados [92].

•

Suporte para transferências síncronas (bloqueantes) e assíncronas (orientadas a interrupção) [92].

•

Utilização de buffers: Para compatibilizar requisições e transferências dos dispositivos em termos de tamanhos e velocidade [92]. O SO comumente organiza seu código para E/S de maneira modular, em camadas, que vão da interação com o controlador físico até a oferta de chamadas de sistema uniformizadas para E/S pelos processos [90].

Capítulo 7: Relógios e Temporizadores de Sistema

Os relógios e temporizadores de sistema são componentes de hardware e software cruciais para a operação do sistema operacional, indo além da simples manutenção da hora [93, 94].

7.1 Função dos Temporizadores

Os sistemas operacionais mantêm um relógio com o horário do sistema, com a ajuda de algum dispositivo temporizador presente na placa-mãe [93]. Além de manter a hora precisa, os dispositivos temporizadores servem para diversas outras finalidades [94-96]:

•

Manutenção da hora do dia: Útil para ajustar parâmetros de arquivos (data de criação e modificação) e associar timestamps a eventos do sistema [93, 94].

•

Prevenção de processos que rodam por mais tempo que o permitido: Essencial para o escalonamento do uso do processador em fatias de tempo (time slicing) [94, 95]. O SO consegue direcionar o processador para a execução de um processo e retomar o uso do processador após um tempo específico [95].

•

Contabilidade do uso da CPU: Para fins de gerenciamento de recursos [96].

•

Fornecimento de watchdog timers para partes do próprio sistema [96].

•

Coleta de perfil, monitoramento e estatísticas [96].

•

Um contador de "ticks" é usado para indicar o instante atual [94]. Sistemas geralmente contêm mais de um contador, associados à geração de interrupções programadas e periódicas [94].

7.2 High Precision Event Timer (HPET) e Outros

O High Precision Event Timer (HPET) é um temporizador de hardware desenvolvido em conjunto pela Intel e Microsoft, incorporado em chipsets de PC desde cerca de 2005 [96, 97].

•

Foi projetado para substituir temporizadores mais antigos, como o Programmable Interval Timer (PIT) e o Real-Time Clock (RTC) [96, 97].

•

Consiste em um contador principal (geralmente de 64 bits) que conta progressivamente, e de 3 a 32 comparadores de 32 ou 64 bits [96, 98, 99].

•

É programado usando E/S mapeada em memória, e seu endereço base pode ser encontrado via ACPI [96, 99].

•

Cada temporizador HPET inclui um registrador de correspondência (match register) e um comparador, e pode gerar uma interrupção quando o valor do registrador de correspondência iguala o valor do contador em execução [99]. Alguns temporizadores podem ser configurados para gerar interrupções periódicas [99].

•

Sistemas operacionais mais antigos que não suportam HPET usam as facilidades de temporização mais antigas [97].

7.3 Kernel Tickless (Sem Pulso)

Tradicionalmente, o kernel Linux interrompia periodicamente cada CPU a uma frequência predeterminada (ex: 100Hz, 250 Hz, 1000 Hz) para contabilidade de processos e balanceamento de carga, independentemente do estado de energia da CPU [100]. Esse "pulso do temporizador" (timer tick) impedia que CPUs ociosas permanecessem inativas o suficiente para se beneficiar de medidas de economia de energia [100].

Um kernel tickless é um kernel de sistema operacional no qual as interrupções do temporizador não ocorrem em intervalos regulares, mas são entregues apenas quando necessário [101].

•

No Linux, a partir da versão 3.10 com CONFIG_NO_HZ_FULL, o kernel pode ser configurado para desativar o timer tick para CPUs ociosas e até mesmo para processadores não ociosos [101].

•

Essa abordagem permite que CPUs ociosas permaneçam ociosas até que uma nova tarefa seja enfileirada, e CPUs que entraram em estados de menor consumo de energia podem permanecer nesses estados por mais tempo, resultando em economia de energia [102].

•

Outros kernels como XNU (Mac OS X 10.4+), NT (Windows 8+), Solaris 8 e FreeBSD 9+ também implementam ou introduziram modos tickless [101].