CAPÍTULO V

ENTRADAS E SAÍDAS

Uma das funções principais de um S.O. é controlar o acesso aos dispositivos de entrada e saída (abreviadamente: E/S ou I/O) e fornecer aos usuários uma interface simples com os mesmos. Na medida do possível, a interface oferecida para todos os dispositivos de E/S deve ser a mesma (o que garante a Independência de Dispositivo - ***device independency***).

1. Princípios de Hardware de E/S

Os dispositivos de E/S serão enfocados sob o ponto de vista de sua programação, que no entanto está muitas vezes ligada à sua operação interna.

1.1. Dispositivos de E/S

Divide-se os dispositivos de E/S em duas categorias:

- dispositivos de bloco: armazenam a informação em blocos de tamanho fixo, cada um dos quais com o seu próprio endereço, podendo ser lido independentemente de todos os outros. Ex: discos.
- dispositivos de caracteres: entrega ou aceita uma cadeia de caracteres, sem considerar qualquer estrutura de bloco. Ex: terminal e impressora.

Alguns dispositivos não se adaptam bem a nenhuma das duas categorias acima. Por exemplo o relógio, que nem apresenta estrutura de bloco nem realiza transmissão de caracteres, realizando apenas uma interrupção a intervalos predefinidos. Com isto, vemos que a classificação acima não é exaustiva, servindo apenas como orientação para uma melhor compreensão de diversos dispositivos.

1.2. Controladores de Dispositivos

Em geral, os dispositivos estão constituídos em duas partes:

- parte mecânica, que é a que efetivamente executa as ações requeridas;
- parte eletrônica, que controla o dispositivos, sendo por isso denominada controlador ou adaptador.

O S.O. trata com o controlador e não com o dispositivo em si. Os microcomputadores (e diversos minicomputadores) se utilizam, para comunicação com os dispositivos de E/S (isto é, com os seus controladores), do esquema de duto único.
Sistemas maiores em geral se utilizam de dutos múltiplos e computadores especializados de E/S, chamados canais de E/S, que retiram grande parte da carga da UCP com relação ao processamento de E/S.

A interface entre o controlador e o dispositivo é geralmente de baixo nível, cuidando de todos os detalhes do funcionamento do dispositivo, bem como da organização da informação de forma a ser reconhecida pela UCP.

Cada controlador possui alguns registradores, que são utilizados para comunicação com a UCP. Esta comunicação pode ser realizada de duas formas:

- registradores no espaço de memória: neste caso, os registradores dos controladores ocupam posições no espaço de endereçamento da memória;
- registradores em espaço de E/S: neste caso, os registradores dos controladores são incluídos num espaço separado de endereçamento, conhecido como espaço de E/S.

As operações de E/S são então realizadas através da escrita, pela UCP, de comandos nos registradores dos controladores e da leitura de resultados nesses mesmos registradores.

1.3. Acesso Direto à Memória

Vários controladores (especialmente de dispositivos de bloco) suportam uma técnica de transferência conhecida como acesso direto à memória (DMA). Para exemplificar a operação do DMA, vejamos como opera o acesso a disco com e sem a utilização dessa técnica.

Quando não existe DMA, o acesso a disco é feito da seguinte forma:

1. o controlador de disco lê, um a um, os bits de dados vindos do disco;
2. quando um bloco completo foi transferido, é feito um teste para verificar se houve erros;
3. se não houve erros, o controlador causa uma interrupção na UCP;
4. a UCP entra num "loop" de leitura, que lê um byte por vez do controlador.

Como pode ser visto, esse ciclo gasta um tempo considerável da UCP devido à necessidade desta permanecer num "loop" para a transferência de todos os dados provindos do disco. Para eliminar o tempo gasto pela UCP nesse "loop", é utilizada a técnica de DMA, que faz com que o ciclo de transferência do disco fique como abaixo:

1. a UCP dá ao controlador dados sobre o setor a ler, endereço de memória onde ela quer que a informação seja colocada, e número de bytes que devem ser transferidos;
2. após o controlador ler o bloco e verificar os erros, como indicado anteriormente, ele automaticamente copia todo o bloco na posição de memória especificada;
3. quando o S.O. quiser utilizar os dados, estes já se encontram na posição correta da memória, não sendo necessário utilizar tempo de UCP para a transferência.

2. Princípios de Software de E/S

Deve-se procurar organizar o software de E/S em camadas, de forma que as camadas mais baixas ocultem as particularidades do hardware às camadas mais altas e estas últimas se ocupem de apresentar ao usuário uma interface clara e agradável.

2.1. Objetivos do Software de E/S

Os conceitos importantes que devem ser considerados num projeto de software para E/S são os seguintes:

1. **independência de dispositivo**: isto é, deve ser possível escrever programas que possam ser utilizados, sem alteração, para dispositivos diversos, como discos

flexíveis, discos rígidos, e mesmo terminais de vídeo e linhas seriais. O S.O. é que deve cuidar dos problemas causados pelas diferenças entre os dispositivos, e não o usuário;

2. **denominação uniforme**: os nomes dados aos dispositivos devem ser independentes do tipo do mesmo;
3. **tratamento de erros**: deve-se tratar os erros nos níveis tão mais próximos do hardware quanto possível, de forma que os níveis superiores somente se ocupem com erros que não podem ser tratados pelos níveis inferiores;
4. **transmissão síncrona (bloqueante) X assíncrona (interrupções)**: na transmissão assíncrona, a UCP pede uma transferência e em seguida vai realizar outras operações. Quando a transferência está pronta, uma interrupção é gerada para informar a UCP de que a operação foi completada. Ao se escrever programas, é mais simples que as transmissões sejam consideradas bloqueantes, de forma que, ao pedir uma transferência, o programa para até que ela esteja pronta (por exemplo, num read). O S.O. deve cuidar de fazer com que uma transferência assíncrona apareça ao usuário como síncrona;
5. **dispositivos compartilhados X dedicados**: alguns dispositivos podem ser compartilhados entre diversos usuários sem problema (p.ex.: discos, podendo vários usuários possuir arquivos abertos no mesmo). O mesmo já não acontece com outros (p.ex.: impressora), que devem ser utilizados por um único usuário por vez. O S.O. deve ser capaz de cuidar tanto de dispositivos compartilhados como de dispositivos dedicados, de uma forma que evite problemas em quaisquer dos casos.

Os objetivos indicados acima podem ser atingidos de um modo abrangente e eficiente estruturando-se o software de E/S em quatro camadas:

1. tratadores de interrupções;
2. condutores de dispositivos (***device drivers***);
3. software de E/S independente de dispositivo;
4. software em nível de usuário;

As atribuições de cada uma dessas camadas serão apresentadas a seguir.

2.2. Tratadores de Interrupção

Devido a extrema dependência do hardware, as interrupções devem ser tratadas nos níveis mais baixos do S.O., de forma que a menor parte possível do sistema tome conhecimento da existência de interrupções.

O método tradicional de trabalho é fazer com que os condutores de dispositivos (***device drivers***) sejam bloqueados quando executam um comando de E/S. Quando a interrupção correspondente ocorre, a rotina de interrupção desbloqueia esse processo (através de um V em semáforo, ou um signal em variável de condição ou da transmissão de uma mensagem, dependendo do tipo de sincronização utilizado pelo S.O.).

2.3. Condutores de Dispositivos (*device drivers*)

O condutor de dispositivo contém todo o código dependente do dispositivo. Cada condutor de dispositivo maneja um tipo de dispositivo ou, no máximo, uma classe de dispositivos correlacionados.

O condutor de dispositivo envia os comandos aos controladores (através dos registradores corretos) e verifica que esses comandos sejam corretamente executados, sendo a única parte do S.O. que tem conhecimento do controlador utilizado. Em termos gerais, a tarefa do condutor de dispositivo é aceitar comandos do software independente de dispositivo e fazer com que sejam executados, devendo transformar o pedido dos termos abstratos em que é realizado para termos concretos. Por exemplo, no condutor de disco, verificar onde fisicamente o bloco requisitado se encontra, verificar o estado do motor (ligado ou desligado), posição da cabeça de leitura, etc.

Ele então determina os comandos que devem ser enviados ao controlador do dispositivo e os envia. A partir desse ponto temos duas situações possíveis:

1. o controlador leva algum tempo até terminar a execução do comando, caso em que o condutor de dispositivo se bloqueia;
2. o controlador executa o comando automaticamente, não sendo necessário o bloqueamento.

No primeiro caso, o condutor bloqueado será acordado por um interrupção proveniente do controlador. Os dados obtidos com a operação são então passados ao software independente de dispositivo.

2.4. Software Independente de Dispositivo

A separação exata entre os condutores e o software independente de dispositivo é variável com o S.O. As funções geralmente realizadas pela parte do software independente de dispositivo são as seguintes:

- **interfaceamento uniforme para os condutores de dispositivo**: é a função básica, compreendendo a execução de funções comuns a todos os dispositivos de E/S;
- **denominação dos dispositivos**: consiste no mapeamento dos nomes simbólicos nos condutores corretos;
- **proteção dos dispositivos**: em microprocessadores, em geral o usuário pode fazer o que desejar com os dispositivos de E/S, não existindo nenhuma proteção implícita. Já em computadores grandes, o acesso a dispositivos de E/S é proibido aos usuários. No UNIX, o acesso a dispositivos de E/S é protegido pelos bits rwx, como qualquer arquivo.
- **fornecimento de um tamanho de bloco independente do dispositivo**: para dispositivos de bloco, devemos fazer com que todos os presentes no sistema apresentem ao usuário o mesmo número de blocos. Para dispositivos de caracter, alguns apresentam um caracter por vez (p.ex.: terminais), enquanto que outros apresentam-nos em conjuntos de vários caracteres (p.ex.: leitora de cartões). Essas diferenças devem ser ocultadas;
- **bufferização**: deve existir bufferização para se poder lidar com situações como as seguintes:

a. dispositivos de bloco requerem o acesso a blocos completos, enquanto que o usuário pode requerer acesso a unidades menores;
b. em dispositivos de caracteres, o usuário pode tentar escrever mais rápido do que o dispositivo pode ler, ou o dispositivo pode enviar antes que o usuário esteja pronto para ler.

- **alocação de memória em dispositivos de bloco**: a descoberta de quais o blocos livres e quais os ocupados em um dispositivo de bloco não é dependente do dispositivo e pode, portanto, ser realizada neste nível do S.O.
- **alocação e liberação de dispositivos dedicados**: pode ser tratada também nesta parte do S.O. Normalmente isto é feito com chamadas de sistema do tipo open (para pedir acesso) e close (para liberar o acesso);
- **informe de erros**: a maioria dos erros é altamente dependente do hardware e deve, portanto, ser executada nos condutores de dispositivos. No entanto, após um condutor informar um erro, o tratamento desse erro é independente do dispositivo, sendo em geral o erro informado ao processo que pediu o acesso. No caso de um erro que possa comprometer completamente o funcionamento do sistema, entretanto, uma mensagem deve ser enviada e o sistema terminado.

2.5. Software de E/S no Espaço de Usuário

Uma parte do software de E/S consiste em rotinas de biblioteca ligadas com programas de usuário e mesmo programas inteiros rodando fora do modo ***kernel***.

Chamadas de sistema, incluindo-se as de E/S, são feitas geralmente por rotinas de biblioteca. Muitas dessas rotinas não fazem nada mais do que colocar os parâmetros no local apropriado para a chamada de sistema e realizar a chamada. Outras entretanto fazem um trabalho real, em geral de formatação de dados (p.ex.: printf).

Temos também os sistemas de ***spooling***, que consistem em uma forma de tratar dispositivos de E/S dedicados em sistemas de multiprogramação. O exemplo típico de dispositivo que utiliza ***spool*** é a impressora. A impressora poderia ser tratada através da utilização de open e close. No entanto, considere o que ocorreria caso um usuário realizasse open e não utilizasse a impressora por horas.

Dessa forma, é criado um processo especial, chamado ***daemon***, que é o único a ter permissão de acesso à impressora, e um diretório de ***spool***. Quando um processo quer imprimir um arquivo ele o coloca no diretório de ***spool*** e, então, o ***daemon*** o imprime quando possível.