... 5.2.5 - Software no Nível do Usuário

★ uma outra importante categoria é o sistema *spooling*, que não consiste de procedimentos da biblioteca; e sim o modo de negociação com os dispositivos dedicados de entrada/saída em um sistema com multiprogramação.

- ... embora seja fácil permitir que algum processo do usuário abra o arquivo especial associado à impressora, suponha que o processo permaneça com o dispositivo requisitado por várias horas — nenhum outro processo poderá imprimir.
- ... ao invés disso, o que é feito é criar um processo especial, chamado *daemon*, e um diretório especial, chamado *spooling directory*;
- ... para imprimir um arquivo, o aplicativo recebe o arquivo inteiro para ser impresso e o coloca no *spooling directory* para depois o *daemon* transferí-lo do *spooling directory* para a impressora;
- ... protegendo-se os arquivos especiais contra o uso direto por usuários, o problema de se ter alguém monopolizando-os é eliminado.

... 5.2.5 - Software no Nível do Usuário

★ *spooling* não é somente usado para impressoras, mas também em outras situações;

- ... por exemplo, transferência de arquivo na rede frequentemente usa um *daemon* de rede capaz de ler um arquivo do diretório de *spooling* da rede e transmití-lo;

Camada	Funcionalidades
Processos do Usuário	Executa Operação de Entrada/Saída
Software Independente do Dispositivo	Identificação, Proteção, Bloqueio Bufferização
Drivers de Dispositivos	Inicia Registradores do Dispositivo verifica status da operação
Gerenciadores de Interrupção	Desbloqueia o Driver quando a operação de entrada/saída se completa.
Dispositivos	Executa fisicamente a operação de E/S

... 5.2.4 - Software Independente do Dispositivo

- ✶ **alocação de blocos nos dispositivos** - uando um arquivo é criado e preenchido com dados, novos blocos de disco têm que ser alocados para o arquivo;

- ... para realizar esta alocação, o sistema operacional precisa de uma lista ou mapa de bits dos blocos livres no disco, mas o algoritmo para localizar um bloco livre é independente do dispositivo e pode ser implementado acima do nível do *driver*.

- ✶ **dispositivos dedicados** - é o sistema operacional que examina a requisição para usar o dispositivo, aceitando dependendo da sua disponibilidade.

- ✶ **gerenciamento de erros** - um erro típico é a não leitura de um bloco ruim;

- ... após o *driver* tentar ler o bloco várias vezes, ele informa ao *software* independente do dispositivo a razão, permitindo o tratamento do erro;‘

- ... se o erro é crítico, o sistema operacional deve apresentar uma mensagem e, eventualmente, terminar sua execução.

5.2.5 - Software no Nível do Usuário

- ✶ embora muito do *software* de E/S esteja embutido no sistema operacional, uma pequena porção dele consiste de bibliotecas ligadas juntamente com programas do usuário, e até mesmo com programas inteiros executando fora do núcleo;

- ... chamadas de sistema, incluindo chamadas do subsistema de entrada/saída, são normalmente feitas por procedimentos da biblioteca;

- ... por exemplo, um programa em C que contém a chamada abaixo descrita, terá a biblioteca **fread** ligada com o arquivo objeto gerado na compilacção;

```
bytes_lidos = fread(buffer, tam_item, n_itens, arquivo);
```

- ... a coleção de todos estes procedimentos da biblioteca é parte desse sistema.

... 5.2.4 - Software Independente do Dispositivo

✶ a função básica do *software* independente do dispositivo é realizar as funções de entrada/saída que são comuns a todos os dispositivos, e prover uma **interface uniforme** para o *software* do usuário.

✶ **identificação dos dispositivos** - encarregar do mapeamento simbólico dos nomes dos dispositivos para os seus *drivers* apropriados;

- ... no UNIX, por exemplo, cada *device driver* tem a ele associado um *inode*, o qual armazena informações necessárias à localização do *driver* e do dispositivo associados;

- ... estes dispositivos estão dispostos no diretório */dev* e o usuário tem permissão limitada para operá-los.

✶ **proteção dos dispositivos** - em muitos microcomputadores não há nenhuma proteção, enquanto que em alguns *mainframes* e superminis, acessos a dispositivos de entrada/saída é completamente proibido a usuários.

... 5.2.4 - Software Independente do Dispositivo

✶ **software independente do dispositivo** - deve encobrir este fato e prover um tamanho de bloco uniforme para camadas superiores, por exemplo, pelo tratamento de vários setores como um simples bloco lógico;

- ... deste modo, os níveis superiores somente negociam com dispositivos abstratos que usam o mesmo tamanho de bloco, independente do tamanho físico do setor.

✶ **bufferização** - tanto para dispositivos de blocos como para de caracter;

- ... para dispositivos de bloco, o *hardware* executa escrita e leitura de blocos inteiros, mas o processo do usuário está livre para ler ou escrever unidades arbitrárias;

- ... para dispositivos de caracter, usuários podem escrever dados no sistema mais rápido do que a taxa com que eles são transferidos para o dispositivo físico, necessitando assim de "buferização" (p. ex., teclado).

... 5.2.3 - Drivers de Dispositivos

❷ em outros casos, entretanto, as operações são rápidas, situação esta em que o *driver* não precisa ser bloqueado (requisições não bloqueantes);

- ... por exemplo, o deslocamento da tela em terminais requer apenas escrita de uns poucos bytes nos registradores do controlador; assim, nenhum movimento mecânico é necessário e a operação toda pode se completar em alguns poucos μs.

★ Neste ponto, após a operação ter sido completada, o *driver* deve verificar a ocorrência de erros. Se tudo estiver correto, ele passa os dados (o bloco lido, por exemplo) para a próxima camada do *software* de entrada/saída.

★ Finalmente, ele retorna alguma informação de *status* de erros. Se alguma requisição está na fila, uma delas pode agora ser selecionada e iniciada. Se nenhuma está na fila, o *driver* fica aguardando a próxima requisição.

5.2.4 - Software Independente do Dispositivo

★ Embora alguns dos *softwares* de entrada/saída sejam específicos do dispositivo, uma grande fração deles é independente do dispositivo.

★ O limite exato entre os *drivers* e o *software* independente dos dispositivos é função do sistema, uma vez que algumas funções que são independentes do dispositivo podem atualmente estarem nos *drivers*, por eficiência e por outras razões;

★ ... as funções listadas abaixo são tipicamente implementadas na parte do *software* que é independente do dispositivo:

- interface uniforme para com os *drivers* de dispositivos;
- identificação simbólica dos dispositivos;
- proteção dos dispositivos;
- manipulação de blocos independente dos dispositivos.
- bufferização e alocação de espaço nos dispositivos;
- alocação e liberação de dispositivos dedicados;
- gerenciamento de erros.

... 5.2.3 - Drivers de Dispositivos

★ em termos gerais, o trabalho de um *driver* é aceitar requisições abstratas de um *software* de mais alto nível, e providenciar para que o pedido seja atendido;

- ... uma típica requisição é ler um bloco – se o *driver* está desocupado no momento, a requisição é aceita, sendo processada imediatamente, caso contrário, esta normalmente entra numa fila de requisições pendentes.

★ ... por outro lado, os termos abstratos da requisição precisam ser transcritos para ações concretas, por exemplo, para um *disk driver*, isto significa:

- ... informar onde o bloco se encontra no disco;
- ... verificar se o motor do *driver* está girando;
- ... determinar se o braço está no cilindro apropriado, e assim por diante.

★ resumidamentes, o *driver* deve decidir quais operações do controlador são requeridas e em que sequência estas devem ser disparadas.

... 5.2.3 - Drivers de Dispositivos

★ uma vez determinado quais comandos emitir ao controlador, este inicia a emissão escrevendo nos registradores do controlador do dispositivo;

- ... alguns controladores podem manusear somente um comando por vez, outros aceitam uma lista de comandos que são carregadas sem a ajuda do sist. operacional.

★ após o comando ou comandos terem sido emitidos, podem ocorrer duas situações:

❶ em muitos casos o *device driver* deve esperar até que o controlador execute as operações requisitadas, ou seja, a requisição é bloqueante;

- ... se estas operações forem lentas (envolvendo movimentos mecânicos, por exemplo), o *driver* bloqueia até que as operações se completem.

5.2.2 - Manipulação de Interrupções

★ interrupções são eventos complicados de se tratar e devem ser isoladas de modo que apenas uma pequena parte do sistema operacional as manipule;

- ... um meio para isolá-las é bloquear os processos aguardando operações de entrada/saída até que uma interrupção anuncie que a operação se completou.

★ quando a interrupção acontece, a rotina de tratamento correspondente libera o processo bloqueado, o que é conseguido retirando-o da fila referentes àquele evento;

- ... em alguns sistemas isto é conseguido fazendo-se um UP sobre um semáforo, em outros ele fará um SIGNAL sobre a variável de condição no monitor; e ainda em outros, uma mensagem é enviada ao processo bloqueado;

- ... em todos os casos o efeito da interrupção é que o processo que estava previamente bloqueado deverá agora estar habilitado para execução.

5.2.3 - Drivers de Dispositivos

- **driver do dispositivo** - código dependente do dispositivo responsável pela manipulação do mesmo ou uma classe de dispositivos intimamente relacionados.

- ... desta maneira, o *driver* é a parte do sistema operacional que conhece quantos registradores o dispositivo oferece, como enviar comandos escrevendo nesses registradores e como testá-los no tocante às operações;

- ... por exemplo, é de responsabilidade do *driver* do controlador de disco saber para que serve cada um dos registradores;

- ... adicionalmente, é de sua responsabilidade o reconhecimento dos setores, trilhas, cilindros, cabeças de leitura/escrita, motor, fator de entrelaçamento e todos os mecanismos que fazem um disco trabalhar propriamente.

... 5.2.1 - Objetivos do Software de Entrada/Saída

✯ outra característica chave são as transferências por blocos (síncronas) e as manipuladas por interrupção (assíncronas);

- ... muitos dispositivos de entrada/saída são assíncronos: a CPU inicia a transferência e se ocupa de outras atividades até que chegue uma interrupção;
- ... o sist. oper. realiza as operações de forma assíncrona (programação bem mais simples), mas para o usuário tudo se apresentam como uma transferência de blocos.

✯ dispositivos podem ser dedicados ou compartilhados, permitindo que sejam utilizados por muitos usuários ao mesmo tempo;

- ... outros dispositivos, como impressoras, devem ser dedicados a um único usuário até que a operação requisitada seja finalizada; por outro lado estes dispositivos introduzem uma variedade de problemas, como o *deadlock* .

... 5.2.1 - Objetivos do Software de Entrada/Saída

- ... sistemas operacionais devem manipular ambos os dispositivos compartilhados e dedicados de maneira a evitar os problemas introduzidos por eles;
- ... estes objetivos podem ser organizados de maneira clara e eficiente pela estruturação do *software* em quatro camadas:

❶ manipulação de interrupções;

❷ *drivers* de dispositivos;

❸ *software* do sistema operacional independente do dispositivo;

❹ *software* do nível do usuário.

5.2 - Princípios do Software

- **idéia básica** - organizar o *software* em camadas, com as mais baixas escondendo peculiaridades do *hardware* e as mais altas apresentando-se simples para o usuários;
- **conceito chave no projeto do software de entrada/saída** — independência do dispositivo *(device independence)*;
- ... deve ser possível escrever programas que usem arquivos gravados em dispositivos de armazenamento (p. ex., disquete ou em disco rígido), sem a necessidade de modificar o programa para cada tipo de dispositivo;
- ... de preferência, deve ser possível utilizar o programa sem recompilá-lo.

5.2.1 - Objetivos do Software de Entrada/Saída

✯ ... relacionado com a indepêndencia do dispositivo está a uniformidade de nome, que deve ser simplesmente uma *string* ou um inteiro que independa do dispositivo;

✯ outra característica importante é a manipulação de erros, que em geral devem ser manipulados o mais próximo possível do *hardware*;

- ... se o controlador encontra um erro, ele deve tentar corrigí-lo se possível, caso contrário, o *driver* do dispositivo deve fazê-lo, talvez apenas tentando ler novamente;
- ... muitos erros são transientes e desaparecem se a operação for repetida, por conseguinte, somente se as camadas mais baixas não conseguirem resolver o problema é que este deve ser apresentado às camadas superiores.