... 5.2 - Streams

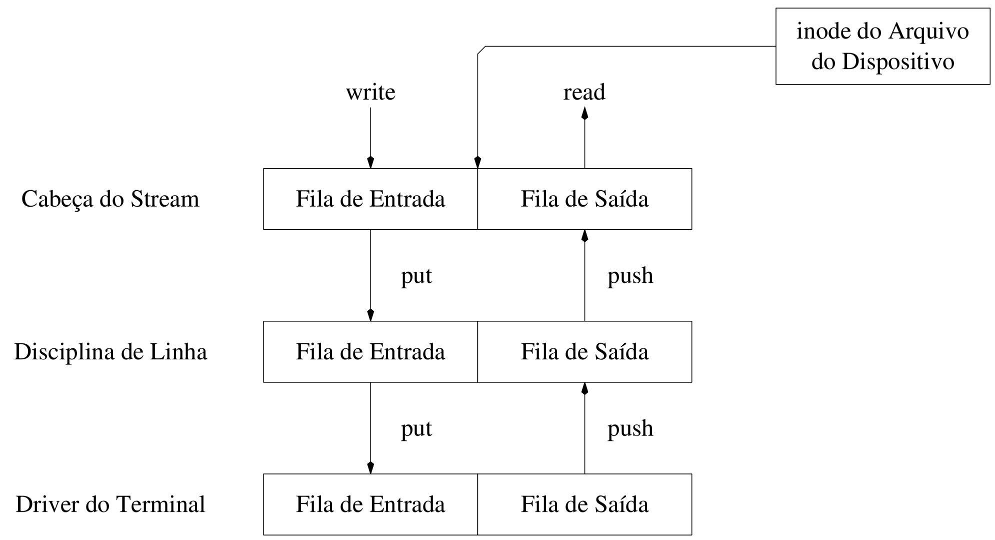
- ★ onúcleo descobre se o *driver* é comum ou baseado em *streams* quando da consulta à tabela de chaveamento de dispositivo.
- ★ o fluxo de dados entre listas se dá por estruturas denominadas *mensagens*;
- ... uma mensagem consiste de uma lista de blocos contendo dados ou controle;
- ... mensagens de controle são originadas durante o processamento de uma chamada **ioctl** executada sobre o descritor do dispositivo;
- ... a chamada de sistema **open** causa a criação do *stream*, com um conjunto de blocos para o fluxo de mensagens.

... 5.2 - Streams

- ★ quando um processo escreve num *stream*, o núcleo copia os dados do espaço de endereçamento do processo para os blocos alocados ao *stream*;
- ... o *stream* transfere estes blocos para a próxima lista envocando sua rotina de *put*. Caso tal operação falhe (por não haver blocos disponíveis nesta lista, por exemplo), estes blocos são marcados para processamento futuro;
- ... a leitura de um *stream* se dá com a chamada **ioctl** com a opção PUSH. Streams são destruidos por ocasião da chamada **close** para seu dispositivo.
- ★ *streams* possibilitam a implementação de *drivers* onde as chamadas **open**, **close**, **read** e **write** possuem código dependente do dispositivo físico, o que se verifica apenas nos níveis mais baixos.

... 5.2 - Streams

- *driver* de terminal composto de um *stream* com 03 pares de listas.



... 5.2 - Streams

★ quando um processo escreve dados num *stream*, o núcleo envia os dados para a próxima lista, até chegar ao dispositivo;

- ... quando o *driver* recebe dados do dispositivo, ocorre o processo inverso: os dados são enviados até chegar à lista de nível mais alto, permanecendo a disposição do processo que requisitou a operação.

★ cada lista do *stream* consiste numa estrutura de dados contendo:

  - um procedimento de abertura invocado durante uma chamada **open**;
  - um procedimento de fechamento invocado durante uma chamada **close**;
  - um procedimento de *put* para adicionar dados à lista;
  - um procedimento de padrão (serviço) chamado quando a lista é processada;
  - um ponteiro para a próxima lista no *stream*;
  - um ponteiro para a fila de itens aguardando passagem para a para a próxima lista;
  - campos utilizados para o controle de fluxo, escalonamento de serviços. etc.

... 5.1 - Drivers Convencionais

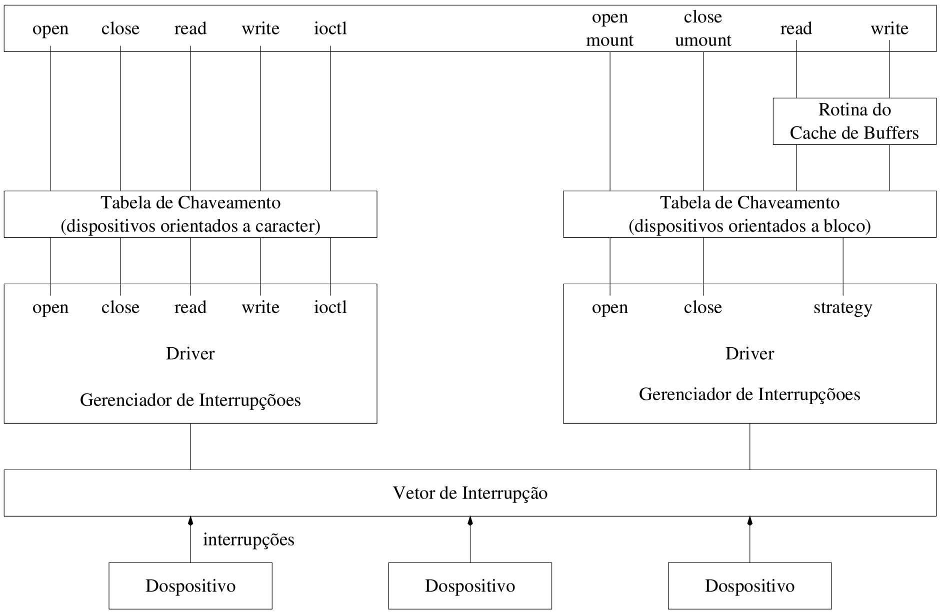
- ★ **dispositivos tipo bloco** - **read** e **write** seguem o mesmo procedimento;
- ★ já para dispositivos nos quais o **driver** tem que iteragir com as rotinas de *buffer cache*, o procedimento é outro. Uma rotina denominada *strategy* perfaz as operações de leitura e escrita em tais dispositivos.
- ... quando uma operação de leitura ou escrita é requisitada, o núcleo identifica a rotina *strategy* definida para o dispositivo;
- ... e passa à mesma o endereço do cabeçalho do buffer para onde os dados devem ser copiados, caso leitura, ou contendo os dados para escrita.

5.2 - Streams

- ★ **stream** - conceito que provê maior modularidade na implementação de *drivers* para dispositivos do tipo caracter (principalmente *drivers* de rede que são estruturados em múltiplas camadas);
- ... um *stream* é um conjunto de pares de listas ligadas. Uma lista armazena requisições de escrita e na outra estão as leituras que o dispositivo já efetuou;
- ... cada par de listas implementa um determinado nível de abstração, desde os mais altos (como as chamadas de sistema) até os mais baixos (como controle direto do dispositivo), passando por níveis intermediários (como protocolos).

... 5.1 - Drivers Convencionais

★ a figura abaixo mostra o esquema básico de entrada/saída no UNIX.



... 5.1 - Drivers Convencionais

★ uma chamada **close** faz com que o núcleo acesse o respectivo procedimento para o dispositivo, cuja identificação é obtida na tabela de arquivos.

★ chamadas **ioctl** permitem o usuário operar tanto arquivos regulares quanto dispositivos do tipo caracter. Operações típicas são:

- bloquear um arquivo;
- desligar o eco do terminal;
- ajustar taxa de transferência de modens;
- reenrolar uma fita.

... 5.1 - Drivers Convencionais

★ cada dispositivo tem um nome idêntico a nomes de arquivos (p. ex., **/dev/tty**, **/dev/console**, etc.) e um respectivo *inode* associado a um arquivo especial;

★ cada dispositivo tem seu *device driver* que implementa as chamadas de sistema **open**, **close**, **read**, **write** e **ioctl** para o seu dispositivo, bem como provê tratamento para as interrupções oriundas deste.

★ em dispositivos que operam em bloco, a transferência de dados entre o espaço de endereçamento de um processo e o dispositivo de dá via *buffer cache*;

- ... a razão para tal é que tais dispositivos (discos, por exemplo) são lentos, sendo portanto a "bufferização" um meio de aumentar a taxa de transferência de dados.

- ... dispositivos do tipo caracter (terminais, por exemplo) são inerentemente rápidos e não necessitam deste recurso.

... 5.1 - Drivers Convencionais

★ esquema básico de entrada/saída no UNIX:

- ... quando um processo executa uma chamada de sistema **open**, por exemplo, o núcleo acessa o *inode* do arquivo passado à chamada e descobre que é um arquivo associado a um dispositivo de E/S;

- ... através de uma *tabela de chaveamento de dispositivo*, o núcleo obtém o endereço da rotina de **open** para este dispositivo;

- ... uma vez iniciado o dispositivo, um campo na tabela de arquivos é adicionado, sendo o índice deste campo (descritor) retornado ao processo.

Cap. 5 - Entrada/Saída no UNIX

- **objetivo:** com base nos conceitos de entrada/saída anteriormente apresentados, detalhamos como é o sistema de entrada/saída no UNIX.

★ ... o sistema de entrada/saída será coberto apresentando-se os dispositivos no UNIX:

- **drivers convencionais** - *drivers* são programas que manipulam o controlador do dispositivo e no UNIX um *driver* implementa as chamadas de sistema **open**, **close**, **read**, **write** e **ioctl** para o seu dispositivo.

- **streams** - conceito que provê maior modularidade na implementação de *drivers* para dispositivos do tipo caracter.

5.1 - Drivers Convencionais

★ dispositivos no UNIX podem ser do tipo bloco ou caracter e sua interface com os dispositivos se dá via subsistema de arquivos.