Tópicos em Interação Humano – Computador

Aquisição de Vídeo em Linux

Thiago S. Barcelos
barcelos@ime.usp.br

Video4Linux

- API para interface com placas de captura de vídeo, sintonia de sinal de TV e rádio e teletexto
- Surgiu por volta de 1998 (Alan Cox), sendo incorporada ao Kernel 2.1.x
- Substituiu várias interfaces desenvolvidas independentemente pelos autores de *device drivers*
- Inicialmente, apenas uma interface de alto nível com o *device driver* bttv

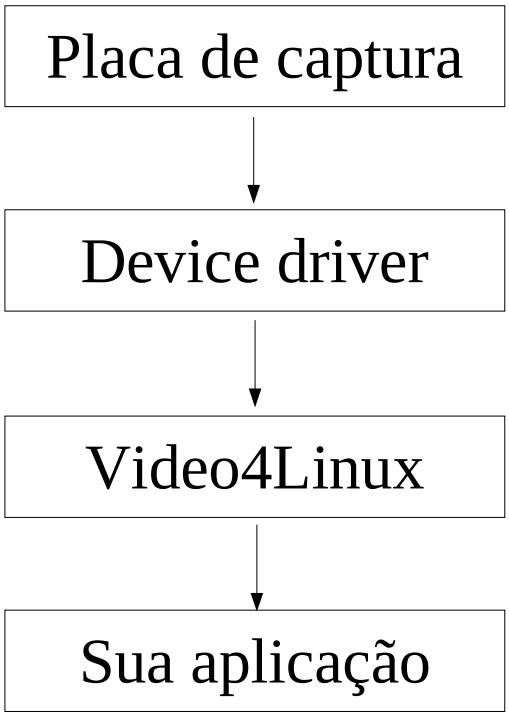
BTTV

- Primeiro *device driver* de suporte a placas de captura no Linux (~1997)
- Para placas baseadas no *chipset* bt848/bt878 da Conexant
- Usado nas placas mais largamente difundidas no mercado
- Exemplos: Osprey 100, Pinnacle miroVideo, Pixelview PlayTV...

Video4Linux x Video4Linux 2

- Uma nova especificação da Video4Linux surgiu por volta de 1999
- Objetivo: consertar algumas falhas de projeto da versão anterior
- Porém, ainda só disponível no Kernel 2.5.x
- Aplicações ainda usam a Video4Linux 1; uma camada de compatibilidade é prometida para a Video4Linux 2

Visão geral



Visão geral

- Instalação do *device driver* (insmod)
- Abertura do dispositivo conforme sua representação no sistema de arquivos
- Configuração de parâmetros de captura (chamadas `ioctl`)
- Leitura dos frames (memória compartilhada ou chamada `read()`)
- Exibição do vídeo

Device driver

- Várias distribuições fazem a instalação dos *drivers* automaticamente
- Os módulos necessários são o `videodev`, o driver específico da placa (em geral, `bttv`) e o `tuner` (no caso da placa possuir sintonia de TV ou rádio)
- Pode ser necessário especificar como parâmetro o modelo da placa e do sintonizador

Nomenclatura dos dispositivos

- `/dev/video*:` Interface de captura de vídeo
- `/dev/radio*:` Dispositivos de rádio AM/FM
- `/dev/vtx*:` Interface de teletexto
- O primeiro passo para lidar com o dispositivo é abri-lo com a chamada `open()` e obter um identificador numérico

Controle e configuração

- Uma vez aberto o dispositivo, qualquer operação sobre ele é feito através de chamadas à função `ioctl`(*descritor_arquivo*, *requisição*, ...)
- Vários parâmetros devem ser ajustados antes de começar a captura de vídeo (tamanho da imagem, quantidade de bits em cada pixel, parâmetros de *frame buffer*...)

Chamada `VIDIOCGCAP`

- A chamada `VIDIOCGCAP` obtem as características do dispositivo

`VID_TYPE_CAPTURE`	Can capture to memory
`VID_TYPE_TUNER`	Has a tuner of some form
`VID_TYPE_TELETEXT`	Has teletext capability
`VID_TYPE_OVERLAY`	Can overlay its image onto the frame buffer
`VID_TYPE_CHROMAKEY`	Overlay is Chromakeyed
`VID_TYPE_CLIPPING`	Overlay clipping is supported
`VID_TYPE_FRAMERAM`	Overlay overwrites frame buffer memory
`VID_TYPE_SCALES`	The hardware supports image scaling
`VID_TYPE_MONOCHROME`	Image capture is grey scale only
`VID_TYPE_SUBCAPTURE`	Capture can be of only part of the image

Recebendo os dados na memória

- Captura de vídeo lida com altas quantidades de informação
- É altamente desejável que a placa de captura consiga enviar os frames capturados por algum tipo de esquema de compartilhamento de memória
- Questão principal: *desempenho*. Evita o overhead de cópia de grandes quantidades de dados para o espaço de memória da aplicação

Recebendo os dados na memória

- Opções para receber os dados:
  - Alocar uma área de *memória compartilhada* entre processos;
  - Alocar um *frame buffer* fornecido pelo sistema de janelas
- Usando memória compartilhada, os dados que chegam ao programa podem ser repassados como entrada para outro algoritmo (p. ex., segmentação, processamento...)

Recebendo os dados na memória

- Frame buffer: alternativa eficiente para exibição dos frames tal e qual chegam da placa de captura
- Para exibição da imagem eventualmente processada, existem outras técnicas (mais tarde...)

Captura em memória compartilhada

- A chamada `VIDIOCGMBUF` permite obter o tamanho do buffer e quantos frames ele é capaz de armazenar
- Com essas informações, deve-se alocar uma área de memória compartilhada com a chamada `mmap()`

```
struct video_mbuf
{
   int   size;    /* Total memory to map */
   int   frames;     /* Frames */
   int   offsets[VIDEO_MAX_FRAME];
};
```

Captura em memória compartilhada

```
struct video_mbuf vbuf;
char *buffer;

ioctl (video_device, VIDIOCGMBUF, &vd.CapturedFrameBuffer);
buffer = (char*) mmap(0, vbuf.size,
                      PROT_READ | PROT_WRITE, MAP_SHARED,
                      video_device, 0);
```

- A quantidade de frames que o frame buffer pode armazenar é um parâmetro ajustável na instalação do módulo no Kernel (`gbuffers=n`)
- Mas *antes* disso ainda é necessário configurar alguns parâmetros da imagem

Propriedades da imagem

- A chamada `VIDIOCSWIN` define as dimensões do frame a ser capturado, dentro da faixa permitida informada por `VIDIOCGCAP`
- A chamada `VIDIOCSPICT` configura parâmetros como brilho, cor e contraste da imagem

Entradas e canais

- Placas de captura tem usualmente mais de uma entrada possível (*input channel)*
- Exemplos: S-Video, Television, Composite
- Cada entrada pode receber sinal em formatos diferentes (*channel norm*)
- Exemplos: PAL, NTSC, SECAM, PAL-M

Entradas e canais

- A chamada `VIDIOCSCHAN` recebe uma `struct video_channel` e muda a captura atual para a entrada indexada por `channel`
- Na inicialização, é desejável iterar por todos os canais com `VIDIOCGCHAN` obtendo as demais informações:

```
struct video_channel {
   int channel;      /* Número do canal */
   char name[32];    /* Nome do canal   */
   int tuners;       /* O canal tem um sintonizador ? */
   __u32  flags;
   __u16  type;
   __u16 norm;         /* Formato padrão do canal */
};
```

Capturando o vídeo

- Finalmente, podemos fazer a captura do vídeo em si
- Forma mais simples: fazer uma chamada `read()` no dispositivo, passando um buffer de tamanho adequado
- *Mas*, em geral:
  - Lento, apenas para snapshots
  - Não suportado por todos os dispositivos

Capturando o vídeo

- Forma mais eficiente: chamada `VIDIOCMCAPTURE`
- Alocamos uma `struct video_mmap` para cada buffer alocado anteriormente
- Os parâmetros da struct devem casar com as chamadas anteriores à `VIDIOCSWIN`

```
struct video_mmap {
   unsigned int frame;     /* Número do frame          */
   int      height,width;  /* Tamanho do frame         */
   unsigned int format;    /* Deve ser VIDEO_PALETTE_* */
};
```

Capturando o vídeo

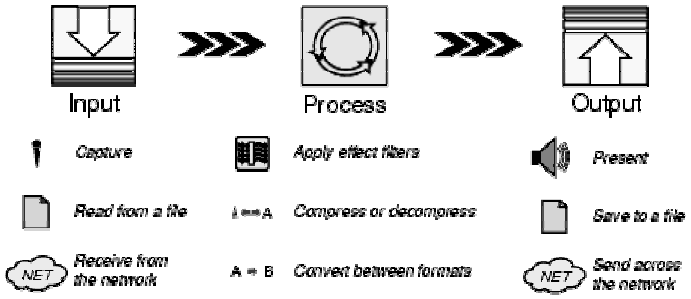
- A chamada `VIDIOCMCAPTURE` recebe a `struct video_mmap` como parâmetro
- Para captura contínua usando mais de um buffer, o único elemento que muda é o número do frame
- Logo depois deve ser feita a chamada `VIDIOCSYNC` para liberar o buffer e continuar

Formatos de pixel

- A `struct video_map` recebe o formato do pixel no campo `format`
- Dentre os principais formatos estão:
  - `VIDEO_PALETTE_GREY`: escala de cinza (8 bits)
  - `VIDEO_PALETTE_RGB24`
  - `VIDEO_PALETTE_RGB32`
  - `VIDEO_PALETTE_RGB555` (16 bits, último bit indefinido)
  - `VIDEO_PALETTE_RGB565`

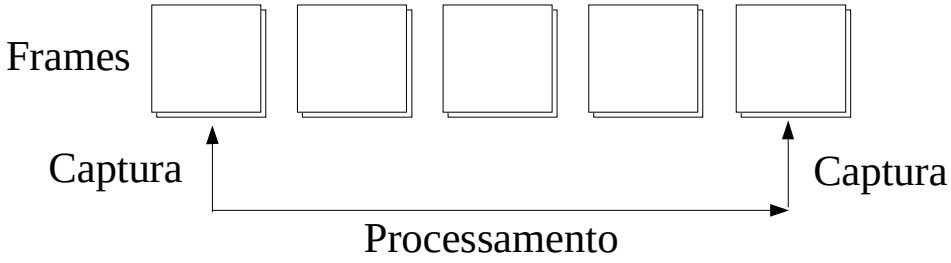
Algumas palavras sobre mídia baseada em tempo

- Vídeo é uma mídia contínua, no sentido que requer processamento e exibição em intervalos de tempo rígidos
- Atrasos no processamento e/ou exibição podem gerar resultados não aceitáveis, dependendo da aplicação



Video4Linux e sincronia de processamento

- Infelizmente, a forma de captura oferecida pelo Video4Linux *não* oferece garantia de recebimento de todos os frames capturados (i.e. captura em *streaming*)
- Qualquer atraso no ciclo na fase de processamento ou exibição gera perdas de frames



Video4Linux 2 e sincronia de processamento

- A Video4Linux 2 trará um modelo de captura em *streaming*
- Drivers compatíveis com Video4Linux 2 implementação uma fila de *buffers*
- Aplicações alocam *buffers* e executam operações de enfileiramento de *buffers*
- Passa a existir suporte à chamada `select()` - a aplicação “dorme” esperando por um buffer preenchido

Enquanto isso...

- Muitas aplicações tem contornado esse problema usando *multithreading*
- É possível simular o comportamento de fila aumentando o número de buffers (até o limite de 32) na instalação do módulo no Kernel.
- Uma nova thread fica responsável pelas chamadas à v4l e manutenção da fila

Exibição no vídeo

- Para exibir diretamente a entrada da placa, o melhor é usar o *frame buffer* oferecido pelo X
  - Os dados vão diretamente para a memória de vídeo
- Se os dados já sofreram processamento, uma alternativa para melhor desempenho é a extensão MIT-SHM do X (*Shared memory extension*)
  - Não é necessária cópia e passagem de dados no canal de comunicação da Xlib

A questão do formato do pixel

- Alguns formatos de pixel oferecidos pela Video4Linux casam com os formatos de pixel de configurações de vídeo do X
- Isso não é coincidência: o X é muito rigoroso quanto ao formato de pixel dos bitmaps que ele vai renderizar
- A sua aplicação talvez tenha que lidar com conversão do formato dos pixels da imagem antes de sua exibição

Documentação da Video4Linux

(www.bytesex.org/v4l)

driver hacking FAQ

Q: Should device drivers convert between different formats?

A: General policy is that they should not do that. Applications should handle the conversion if needed.

In some cases it is impossible not to convert the data somehow because the native format of the hardware doesn't match any of the standard v4l2 video image formats. In that case the driver should pick a standard format where the conversion is cheap, for example where the only thing you have to do is to reorder the bytes.

Outras alternativas?

- JMF: Java Media Framework (Sun)
  - Arquitetura para captura, exibição e codificação de vídeo
  - Provê uma modelagem de alto nível baseada em objetos e eventos
  - *Mas* usa a Video4Linux para o acesso ao hardware!

Aplicações existentes

- Xawtv (www.bytesex.org/xawtv)
  - Um dos primeiros programas de visualização de vídeo, até hoje muito usado

Aplicações existentes

- Avicap, um utilitário da biblioteca avifile (avifile.sourceforge.net)
  - Mesmo conceito do xawtv, interessante como exemplo de programação usando Qt

Aplicações existentes

- Gazetracker
  - Rastreador de olhar desenvolvido no IBM Almaden Research Center
  - Originalmente usava DirectX para captura

Referências

- Video4Linux - www.bytesex.org/v4l
- Especificação da Video4Linux 2 – www.thedirks.org/v4l2
- Esta apresentação ;-) www.ime.usp.br/~barcelos