OBJETIVO: Exercitar a manipulação de arquivos/dados binários através da decodificação de um arquivo de vídeo MPEG.

QUESTÃO ÚNICA

Um vídeo MPEG é formado por uma sequência de imagens (*frames* ou *pictures*). Cada imagem pode ser do tipo "`I`" (*intra*), que contém uma imagem completa, ou dos tipos "`B`" ou "`P`", que possuem apenas informações do que mudou em relação a uma imagem anterior (Figura 1).

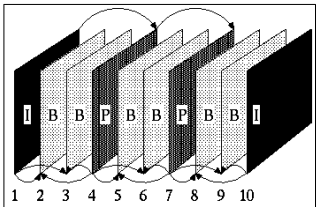

Figura 1. Referência: [1]

Por ser um padrão de vídeo voltado para transmissão, o MPEG possui um formato diferenciado dos estudados até o momento. Mais especificamente, este padrão é formado por uma série de *streams* (fluxos) independentes, permitindo a visualização contínua do vídeo mesmo diante de erros de transmissão em partes anteriores.

Cada *stream* contém um cabeçalho e pode conter outros *streams* dentro dele. Por exemplo, um *stream* do tipo "`Sequence`" pode ter um ou mais *streams* do tipo "`Group of Pictures`" que poderá ter um ou mais *streams* do tipo "`Picture`", e assim por diante, como mostra a Figura 2, a seguir.

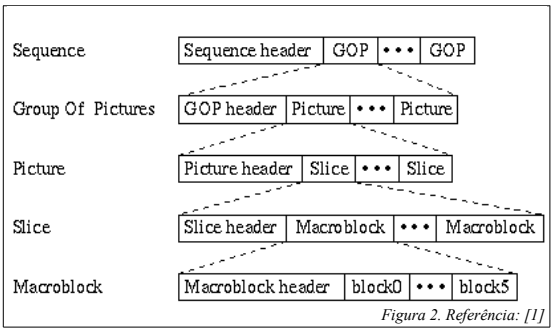

Figura 2. Referência: [1]

O MPEG possui diversos tipos de *streams* (os da figura ao lado e mais outros), mas todos começam com o mesmo código, conhecido como *start code prefix*:

"`\x00\x00\x01`" – em caracteres hexadecimais
"`0000 0000 0000 0000 0000 0001`" – em Binário

Desta forma, sempre que este código (três caracteres/bytes) aparecer em um lugar no arquivo MPEG, já se sabe que o que segue é um *stream*. Após o *start code prefix*, vem um byte identificando o tipo de *stream*, conforme mostra a tabela (incompleta) abaixo:

Código (hexa)	*Tipo do Stream*	*Descrição*
`0xBA`	*`Pack`*	*Contém algumas referências de tempo e multiplexação.*
`0xBB`	*`System`*	*Algumas informações de áudios disponíveis e taxas de multiplexação.*
`0xB3`	*`Sequence`*	*Contém informações de largura e altura das figuras seguintes (normalmente é o mesmo para todas as figuras no vídeo) bem como a taxa de frames por segundo (frame rate).*
`0xB8`	*`Group of Pictures`*	*Marca o início de uma série de figuras. Contém a marcação de tempo da primeira figura.*
`0x00`	*`Picture`*	*Contém, dentre outras coisas, o tipo de figura que segue (I, B ou P).*
`0x01` até `0xAF`	*`Slice`*	*Contém informações para decodificar uma parte da figura atual.*
`0xC0` até `0xDF`	*`Packet Video`*	*Contém informações para decodificar o vídeo.*
`0xE0` até `0xEF`	*`Packet Audio`*	*Contém informações para decodificar o áudio.*

Referências e mais detalhes: [2], [3]

Se o *stream* for do tipo `Sequence`, os dados que seguem (após o *start code prefix* + `0xB3`) possuem o seguinte formato (incompleto, mas o que importa para este trabalho):

Byte 1								**Byte 2**								**Byte 3**								**Byte 4**							
1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Horizontal Size (largura das figuras)												*Vertical Size* (altura das figuras)												*Aspect Ratio*				*Frame Rate*			

Note como os dados estão "desalinhados" com os bytes (caracteres) do arquivo. O valor `Horizontal Size`, por exemplo, ocupa "um byte e meio", enquanto que o `Frame Rate` ocupa "meio byte". Para converter estas informações para inteiro, algumas manipulações binárias são necessárias. Por exemplo, para pegar o `Horizontal Size` é preciso pegar o valor (em inteiro) do primeiro byte e multiplicá-lo por 16, uma vez que este está 4 bits mais "significante" e, em seguida, somar este valor com o valor dos quatro primeiros (mais significativos) bits do segundo byte. Pra fazer isso, basta fazer quatro "shifts à direita" neste segundo byte (considerando que o tipo do byte é `unsigned char`). Exemplo:

```
unsigned char byte1 = fgetc(mpg_file);
unsigned char byte2 = fgetc(mpg_file);
unsigned char byte3 = fgetc(mpg_file);
unsigned char byte4 = fgetc(mpg_file);
unsigned int largura = byte1 * 16 + (byte2 >> 4);
unsigned int altura = ( byte2 & 0x0F) * 256 + byte3;
unsigned int frame_rate = byte4 & 0x0f;
```

Sobre o *frame rate*, se ele for `1`, o *frame rate* do vídeo é `23.976fps`. Se for `2`, ele é `24.000fps`. `3=25.000fps`, `4=29.970fps`, `5=30.000fps`, `6=50.000fps`, `7=59.940fps`, `8=60.000fps`.

Já se o *stream* for do tipo `Picture`, ele possuirá a informação de tipo de figura (I, B ou P) no segundo byte, conforme a figura ao lado. Novamente, note como o dado que queremos está “desalinhado” com o byte. Neste caso, para pegar o valor basta fazer dois “shifts à esquerda” no segundo byte, converter o resultado para `unsigned char` e, em seguida, fazer 5 “shifts à direita”. É importante converter o resultado intermediário para `unsigned char` porque, pela especificação de C, se o número não for `unsigned`, o “shift à direita” preenche os bits à esquerda com `1`'s, e não com `0`'s, que é o que queremos. Após pegar este número, se ele for 1, a figura é do tipo “`I`”, se for 2 ela é do tipo “`P`” e, se for 3, ela é do tipo “`B`”. Obs: uma outra solução para pegar o `tipo` (talvez melhor) é fazer três “shifts à direita” e depois “`& 0x07`”.

Byte 1								Byte 2							
1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
...										TIPO			...		

Neste trabalho, você irá ler um arquivo MPEG (`argv[1]`), byte por byte, e procurar por um *start code prefix*, que indicará o início de um novo *stream*. Ao encontrar, leia o próximo byte e imprima qual é este tipo de *stream* (com base na tabela apresentada). Se o *stream* for do tipo `Sequence`, imprima também a largura, altura e *frame rate* contidos no cabeçalho deste *stream*. Se for um *stream* do tipo `Picture`, imprima também o tipo de figura (I, P ou B). Para qualquer outro tipo de *stream*, imprima “`--> Código: %.2x -- Tipo de stream não implementado`”.

Exemplo de Execução:

```
./mpeg_info dodo.mpg
--> Código: b3 -- Sequence Header -- Width = 360, Height = 288 -- Frame rate = 25.000fps
--> Código: b8 -- Group of Pictures
--> Código: 00 -- Picture -- Tipo = I
--> Código: 01 -- Slice
--> Código: 00 -- Picture -- Tipo = P
--> Código: 01 -- Slice
--> Código: 00 -- Picture -- Tipo = B
--> Código: 01 -- Slice
--> Código: 00 -- Picture -- Tipo = B
--> Código: 01 -- Slice
--> Código: 00 -- Picture -- Tipo = P
--> Código: 01 -- Slice
--> Código: 00 -- Picture -- Tipo = B
--> Código: 01 -- Slice
--> Código: 00 -- Picture -- Tipo = B
--> Código: 01 -- Slice
--> Código: 00 -- Picture -- Tipo = P
--> Código: 01 -- Slice
--> Código: 00 -- Picture -- Tipo = B
--> Código: 01 -- Slice
--> Código: 00 -- Picture -- Tipo = B
--> Código: 01 -- Slice
--> Código: b8 -- Group of Pictures
--> Código: 00 -- Picture -- Tipo = I
--> Código: 01 -- Slice
--> Código: 00 -- Picture -- Tipo = B
--> Código: 01 -- Slice
--> Código: 00 -- Picture -- Tipo = B
...
```

Dicas:

Uma possível solução é ler o arquivo MPEG de três em três bytes (`fread`) e, se esses três bytes lidos não forem igual ao *start code prefix* (`memcmp`), retorne dois bytes (`fseek`) e siga para a próxima iteração (lendo novamente os três próximos bytes). Outra possibilidade é manter em variáveis os dois últimos bytes lidos das iterações anteriores.

O arquivo usado no exemplo (`dodo.mpeg`) pode ser baixado em `https://beans.icomp.ufam.edu.br/dodo.mpg`

Referências

[1] Soderquist, P., Leeser, M. (1997). *Optimizing the data cache performance of a software MPEG-2 video decoder*. MULTIMEDIA '97: Proceedings of the fifth ACM international conference on Multimedia. Pg 291-301. New York, NY, USA. ACM

[2] DVD-Video Information. *MPEG Headers Quick Reference*. http://dvd.sourceforge.net/dvdinfo/mpeghdrs.html. Acessado em Novembro de 2020.

[3] Duncan, Andrew. *MPEG-1 Pictorial Guide - A graphical guide to the ISO 11172 (MPEG-1) digital audio/video standard*. http://andrewduncan.net/mpeg/mpeg-1.html. Acessado em Novembro de 2020.

A seguir, cenas do próximo capítulo …

→ Dividir um arquivo MPEG em vários menores (que funcionem!).

ENTREGA DO LABORATÓRIO

Envie, até 03/06/2021 às 23:59, o código-fonte para horacio@icomp.ufam.edu.br com o assunto “Entrega do 8o Laboratório de LPA”.