OBJETIVO: Exercitar a manipulação de arquivos/dados binários através da decodificação/divisão 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).

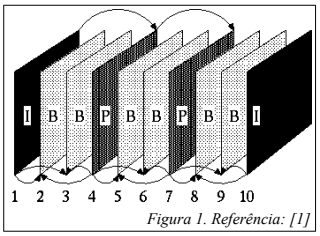

Figura 1. Referência: [1]

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.

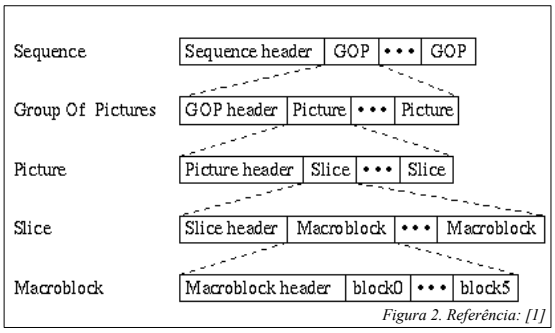

Figura 2. Referência: [1]

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]

Neste trabalho, seu objetivo será pegar um arquivo de vídeo MPEG (`argv[1]`) e dividi-lo em várias partes de tamanho máximo definido na linha de comando (`argv[2]`), em MB. Os tamanhos das partes não precisam ser exatamente iguais entre si, mas precisam ser o máximo possível próximo do tamanho máximo definido na linha de comando (entretanto, menor ou igual a ele).

Como as partes do vídeo precisam funcionar independentemente, a grande questão é aonde exatamente você pode dividir um arquivo de vídeo. No caso de um arquivo MPEG, a melhor parte para dividir um vídeo é ao iniciar um novo fluxo do tipo "`Sequence`", pois ele contém informações de tamanho das imagens. Um arquivo MPEG contém vários fluxos `Sequence`. Arquivos muito pequenos podem conter apenas um (e.g., `dodo.mpeg`).

Salve as partes do vídeo com o nome `video_parte_%d.mpg`, onde `%d` é o número da parte (`1`, `2`, `3`, `4`, ...).

Exemplo de Execução:

```
$ ls -lh
total 14M
-rw-r--r-- 1 horacio horacio  14M Nov 11 10:58 leon.mpg
-rwxr-xr-x 1 horacio horacio  14K Nov 12 10:58 mpg_split_size
-rwxr-xr-x 1 horacio horacio 3,3K Nov 12 10:58 mpg_split_size.c

$ ./mpg_split_size leon.mpg 4
Criando arquivo video_parte_1.mpg ..
Criando arquivo video_parte_2.mpg ..
Criando arquivo video_parte_3.mpg ..
Criando arquivo video_parte_4.mpg ..

$ ls -lh
total 35M
-rw-r--r-- 1 horacio horacio  14M Nov 11 10:58 leon.mpg
-rwxr-xr-x 1 horacio horacio  14K Nov 12 10:58 mpg_split_size
-rwxr-xr-x 1 horacio horacio 3,3K Nov 12 10:58 mpg_split_size.c
-rw-r--r-- 1 horacio horacio 4,0M Nov 12 10:59 video_parte_1.mpg
-rw-r--r-- 1 horacio horacio 4,0M Nov 12 10:59 video_parte_2.mpg
-rw-r--r-- 1 horacio horacio 4,0M Nov 12 10:59 video_parte_3.mpg
-rw-r--r-- 1 horacio horacio 1,9M Nov 12 10:59 video_parte_4.mpg
```

Um possível algoritmo:

- Abre o arquivo de entrada
- Abre o primeiro *arquivo de saída* (`video_parte_1.mpg`)
- Aloca memória para um buffer (tamanho passado no `argv[2]` convertido para **bytes**). Este buffer armazenará o conteúdo completo de um único fluxo "**`Sequence`**" retirado do arquivo de entrada.

- Enquanto verdadeiro
  - Lê quatro bytes (`fread`)
  - Se não for código de `Sequence` (`memcmp` com a string `"\x00\x00\x01\xB3"`) nem o final do arquivo de entrada (`feof`)
    - Salve o primeiro byte lido no buffer
    - Retorna três bytes
    - Próxima iteração do while (continue)

  - Se chegar aqui, então temos o início de um novo fluxo `Sequence` ou o final do arquivo de entrada

  - Se o tamanho do *arquivo de saída* atual mais o tamanho do buffer (que contém um único **sequence**) for maior que o tamanho máximo, teremos que salvar o buffer atual já em um novo arquivo
    - Fecha o arquivo de saída atual
    - Crie uma string para ter o nome do novo arquivo de saída (`sprintf`)
    - Abra o novo arquivo de saída
    - Escreva todo o buffer nele (`fwrite`)
    - Sete o tamanho do arquivo atual para o tamanho do buffer recém-salvo

  - Se não (o tamanho ainda é menor)
    - Salve o buffer lido no arquivo já aberto (`fwrite`)
    - Incremente o tamanho do arquivo atual com o tamanho do buffer

  - Se for o final do arquivo de entrada
    - Feche o arquivo de saída
    - Saia do loop

  - Se chegou aqui, então nós lemos os quatro bytes do código `Sequence` e não retornamos os três bytes.
    - Copie o código `Sequence` lido para o buffer (`memcpy`)
    - Sete o tamanho do buffer para 4, de forma que o buffer comece já com o código `Sequence` lido.

- Libere o buffer
- Feche o arquivo de entrada

Dica:

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

ENTREGA DO LABORATÓRIO

O laboratório é presencial e deve ser entregue durante o horário da aula. Para entregar, envie o código-fonte para horacio@icomp.ufam.edu.br com o assunto “Entrega do 9o Laboratório de LPA”.

Depois de entregue, você pode sair ou, se desejar, pode ajudar algum(a) colega a terminar o trabalho dele(a), desde que não haja cópia do seu código (plágio).