

만든이 : 전호철 (<http://hybridego.net/>)

분석은 어느 정도 다 했으나, 정리를 제대로 못했다.
추후 정리되면 업데이트 할 수도 있다.

MATROSKA 뽀개기 -_-;;

1. Mplayer에서 matroska로 진입과정 및 전체 흐름.

demuxder_list에

```
demuxer_desc_lavf
demuxer_desc_ogg
demuxer_desc_audio
demuxer_desc_h264_es
demuxer_desc_aac
```

이런게 배열로 들어있다.

맨처음 demuxer.c 파일의

demux_open() 함수에서 get_demuxer_type_from_name() 으로 demuxer를 찾아서 연다.

demux_open() -> get_demuxer_type_from_name()

-> get_demuxer_desc_from_type()

get_demuxer_desc_from_type()에서 demuxer type을 찾는다.

이 과정을 루프를 돌면서 계속 찾다가 mkv를 찾으면 mkv를 연다.

demux_mkv_open() 안에서는

```
demux_mkv_read_seekhead
demux_mkv_read_cues
demux_mkv_read_info
demux_mkv_read_tracks
demux_mkv_read_trackentry
demux_mkv_read_trackvideo
demux_mkv_read_trackaudio
```

등의 과정을 거치면서 헤더부분의 EBML을 해석하게 된다.

그리고 해석이 다 끝나면 TRACKEntry를 작성하고 Video와 Audio를 오픈하면서 비디오를 재생한다.

```
display_create_tracks()
demux_mkv_open_video()
demux_mkv_open_audio()
```

재생중에는 계속 버퍼를 관리하는 작업을 한다.

```
demux_mkv_control()
demux_mkv_fill_buffer
add_cluster_position
handle_clokc
demux_mkv_read_block_lacing
demux_mkv_decode
```

가 반복된다.

마지막에는

```
demux_close_mkv()
free_cashed_dps()
demux_mkv_free_trackentry()
demux_mkv_free_encodings()
등이 실행되면서 종료된다.
```

2. MKV 헤더 추출

demux_mkv_open()을 호출하는 부분은

demux_mkv.c 파일의

const demuxer_desc_t demuxer_desc_matroska 배열안에 들어있다.

이 demuxer_desc_matroska는 demuxer.c 의

const demuxer_desc_t * const demuxer_list[] 배열에 주소로 등록되어 있다.

demuxer.h안에 demuxers_desc_st 구조체가 정의되어 있다.

demuxer.c의 demux_open_stream()에서 demuxer_desc로 demuxer_list[i]가
대입된다.

demuxer_desc->check_file(demuxer) 에 의해 실행된다.

3. MKV File의 전체 구조



이 그림과 같다고 써있는데 내가 샘플로 갖고 있는 mkv 파일에는 meta Seek Information, Attachment, Tagging 같은부분은 없었다.

<p>첫 번째 영역은 MKV 의 헤더.</p> <pre> EBML_ID_HEADER EBML_ID_DOCTYPE EBML_ID_DOCTYPEVERSION EBML_ID_DOCTYPEREADVERSION MATROSKA_ID_SEGMENT MATROSKA_ID_SEEKHEAD MATROSKA_ID_SEEKENTRY MATROSKA_ID_SEEKID MATROSKA_ID_SEEKINFO MATROSKA_ID_SEEKPOINT MATROSKA_ID_SEEKENTRY MATROSKA_ID_SEEKID MATROSKA_ID_TRACKS MATROSKA_ID_SEEKPOSITION </pre> <p>이런식으로 줄줄 읽어남.</p>
<p>두 번째 영역은 트랙의 정보이다. 오디오와 비디오트랙의 압축코덱이나 자막, 트랙넘버,트랙UID, 트랙언어 등이 들어있다.</p>
<p>본문</p> <p>클러스터</p> <ul style="list-style-type: none"> 블록그룹 <ul style="list-style-type: none"> 블록 <ul style="list-style-type: none"> 프레임. <p>클러스터는 블록을 분리하고 찾고 에러보호 하는 기능이 있다. 모든 클러스터의 시작은 timecode 이다.</p>

파일의 맨 마지막에서 약간 위에 부분이 CUE 이다.
인덱스라고 많이 알려진 그것이다. 타임코드에 해당하는 정확한 위치가 저장되어 있다.
파일의 맨 마지막 부분은 클러스터의 주소목록이 있다.

4. EBML 읽어내는 구조

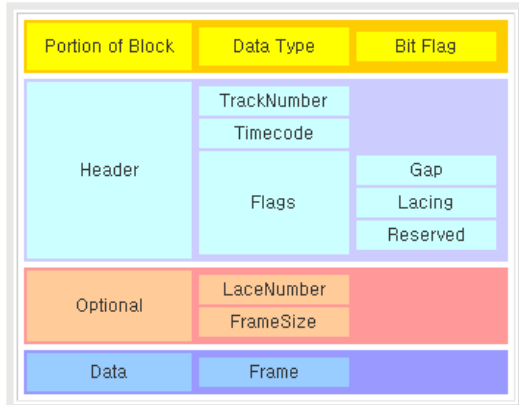
EBML은

ID 읽어낼 덩어리 길이 ID 읽어낼 길이 ID 읽어낼 길이 ID 읽어낼 길이 ID 읽어낼 길이
ID 읽어낼 덩어리 길이 ID 읽어낼 길이 ID 읽어낼 길이 ID 읽어낼 길이 ID 읽어낼 길이

이런식의 반복이다.

mplayer에서 ebml 함수들은
 ebml_read_ascii(스트림, 읽어낸 길이) 리턴값은 읽어낸 내용
 ebml_read_id(스트림, 읽어낸 길이) 리턴값은 읽어낸 내용
 ebml_read_unit(스트림, 읽어낸 길이) 리턴값은 읽어낸 내용
 이런방식으로 쓴다.

Level 0	Grouping	Level 1	Level 2	Level 3
EBML	Header	EBMLVersion		
		DocType		
Segment	Meta Seek Information	SeekHead	Seek	SeekID SeekPosition
			Seek	SeekID SeekPosition
	Segment Information	Info	Title SegmentUID	
	Track	Tracks	TrackEntry	Name TrackNumber TrackType
			TrackEntry	Name TrackNumber TrackType
	Chapters	Chapters	Edition Entry	
	Clusters	Cluster	Timecode	
			BlockGroup	Block
			BlockGroup	Block ReferenceBlock
			BlockGroup	Block
		Cluster	Timecode	
			BlockGroup	Block
			BlockGroup	Block
			BlockGroup	Block BlockDuration
Cueing Data	Cues	CuePoint	CueTime CuePosition	
		CuePoint	CueTime CuePosition	
Attachment	Attachments	AttachedFile	FileName FileData	
		AttachedFile	FileName FileData	
Tagging	Tags	Tag	MultiTitle Language	
		Tag	MultiTitle Language	



2 EBML - Basic

EBML 파일들은 변화가능한 크기의 정수를 사용한다. UTF-8 encoding format 으로부터 영감을 받은 것 같다.

Unsigned Integer Values of Variable Length ("vint")

하나의 정수 길이는 $length = 1 + [number_of_leading_zero_bits]$

2.1 Unsigned Integer Values of variable Length ("vint")

정수 하나의 길이는 $length = 1 + [number_of_leading_zero_bits]$ 이다. 모든 정수들은 big endian을 사용한다. 여러분은 7개의 leading zeros 보다 더 많이 사용 할 수 있다. 그러면 첫 번째 바이트는 0x00 이 될 것이다. 그러나. 이것은 56비트를 요구하는 것보다 더 긴 정수가 필요할 때만 쓰일 것이다.

ex) 3A 41 FE

첫 번째 바이트 3A (0011 1010) 은 2개의 leading zeros를 갖고 있다. (전체길이 3바이트 중에서.) 0011 중에 처음 1은 단지 leading zero들의 연속을 끝내는데만 필요하고 값을 저장하는 데는 사용할 수 없다. 그래서 이 연속으로 표현한 바이트에서 값을 획득하기 위해서 리셋한다. 그 결과가 0x1A41FE. 보는바와 같이 여러분은 한 숫자의 길이를 알기위해서 바이트당 한비트를 잃는다. 그리고 정수의 값을 저장하기 위해서 한 바이트당 7비트를 사용 할 수 있다.

물론 0x1A41FE 값은 10 1A 41 FE 또는 08 00 1A 41 FE(만약 이것이 명확하지 않다면 종이에 디코딩해라)처럼 쓰여질 수 있다. 그러나 EBML 파일들을 쓸때는 가능한 최대로 짧게 encoding 해서 공간의 낭비를 피하도록 한다. 이것은 이 coding scheme의 포인트이다.

Unknown Length

leading zeros 다음의 모든 비트들은 1로 세팅한다. FF나 7F FF처럼, 하나의 알려지지 않은 길이를 가리킨다. Muxers는 알려지지 않은 길이 값을 쓰길 꺼려할 것이다. 한 가지 예외는 파일의 마지막 Level 0 요소이다. 하나의 숫자를 위에서 설명한 결과처럼 인코딩했다면, 그것은 반드시 더 큰 용도의 길이로 다시 인코딩 되어야 한다. 예를 들면 : 16383을 위에서처럼 인코딩 할 때 결과는 7F FF이다. 이 7F FF에서 leading zero 다음의 모든 비트들은 1이다. 이것은 알려지지 않은 길이를 가리킬 것이다. 이 의미는 길이가 3으로

증가고, 그 숫자는 다시 20 3F FF로 인코드 된다는 것이다.

Note

첫 번째 바이트로부터 전체 길이를 결정하는데 lookup table을 사용하는 것이 가능하다.

Matroska 파일 포맷은 정수 길이가 8을 넘어가는 것을 허용하지 않는다. 이 의미는 leading zeors의 개수가 7개를 넘어가면 안된다는 것이다 그리고 전체 길이는 항상 첫 번째 바이트로부터 복구될 수 있다.

2.2 EBML elements

Information의 한 조각 은 다음의 방법으로 저장된다.

```
typedef struct {  
    vint    ID        //EBML-ID  
    vint    size      //size of element  
    char[size] data   //data  
} EBML_ELEMENT;
```

ID의 길이는 s_ID라고 부른다. size의 길이는 s_size라고 부른다. Element가 다른 EBML Element들을 포함하고 있으면 그것을 EBML Master elements 라고 부른다.

일반적으로 부모 element의 안에 있는 EBML element들의 순서는 고정되어 있지 않다. 몇몇 경우에는 정확한 순서가 추천된다. 그러나 의무는 아니다. 특별히 작은 부모 element 안에 가정된 element 순서가 없다.

2.3 Signed Interger Values of Variable Length (svint)

(중략)

3 Matroska Top level elements

Matroska 파일들은 단지 두 개의 top level elements를 갖고 있다.

- EBML

EBML 헤더는 EBML 파일의 콘텐츠를 기술한다. 파일 하나에 하나의 EBML헤더가 있다.

여러개가 있을때에는 어플리케이션에서 무시될 수 있다.

copy /b 명령으로 두 개이상의 파일이 결합되었을 때에는 EBML헤더가 두 개이상 발결될 수 있다.

파일 타입, EBML 버전, 파일 타입 이름, 파일 타입 버전 등이 들어 있다.

- SEGMENT

SEGMENT는 멀티미디어 데이터를 담고 있다. 뿐만 아니라 replay를 위한 헤더 데이터도 갖고 있다. 하나의 MATROSKA파일안에 하나이상의 SEGMENT들이 있을 수 있다. 하지만 이것은 권장되지 않는다. 많은 툴에서 MultiSegment를 지원하지 않는다. 여러분이

multisegment MATROSKA 파일들을 윈도우에서 replay 하고 싶다면 Haali Media Slitter를
사용하길 바란다.

4 EBML - The EBML File Header

EBML의 Top level element는 파일 타입, EBML 버전, File type 이름, File type 버전들을
갖고 있다.

EBMLVERSION	42 86	EBML의 버전을 표시한다. 파일을 생성하는데 사용된다.
EBMLREADVERSION	42 F7	EBML의 최하버전을 가리킨다. Parser가 파일을 읽어내는데 꼭 필요하다.
EBMLMAXIDLENGTH	42 F2	파일안에 있는 가장 긴 EBML-ID의 길이를 지정한다. MATROSKA는 이 값이 4이다. 어떤 EBML-ID도 이 길이를 넘어가면 잘못된 것으로 간주된다.
EBMLMAXSIZELENGTH	42 F3	최대 s_size 값을 가리킨다.