Audio I/O

Speaker : Wei-Shin Pan

DATE : 2008.12.23

Outline

- Multimedia API (Input)
- Media file saving
  - Wave file format
  - Demo (Record and save)
- Multimedia API (Output)
- Demo (Play media file)

MultiMedia API (Input)

WAVEFORMATEX

- **It defines the format of waveform-audio data.**
- typedef struct {

```
  WORD    wFormatTag;
  WORD    nChannels;
  DWORD nSamplesPerSec;
  DWORD nAvgBytesPerSec;
  WORD   nBlockAlign;
  WORD    wBitsPerSample;
  WORD   cbSize;
} WAVEFORMATEX;
```

WAVEHDR

- **It defines the header used to identify a waveform-audio buffer.**
- **typedef struct wavehdr** {

```
  LPSTR lpData;
  DWORD dwBufferLength;
  DWORD dwBytesRecorded;
  DWORD_PTR dwUser;
  DWORD dwFlags;
  DWORD dwLoops;
  struct wavehdr_tag * lpNext;
  DWORD_PTR reserved;
 } WAVEHDR, *LPWAVEHDR;
```

waveInOpen()

- **Open the given waveform-audio input device for recording.**
- **MMRESULT waveInOpen(**
  LPHWAVEIN *phwi*,
  UINT_PTR *uDeviceID*,
  LPWAVEFORMATEX *pwfx*,
  DWORD_PTR *dwCallback*,
  DWORD_PTR *dwCallbackInstance*,
  DWORD *fdwOpen* **);**

waveInprepareHeader()

- **It prepares a buffer for waveform input .**
- **MMRESULT waveInPrepareHeader**(HWAVEIN *hwi*, LPWAVEHDR *pwh*, UINT *cbwh*);

waveInAddBuffer()

- **Send an input buffer to the specified waveform-audio input device.**
- **MMRESULT waveInAddBuffer(HWAVEIN** *hwi*, **LPWAVEHDR** *pwh*, **UINT** *cbwh* **);**

waveInStart()

- **Start input on the specified waveform input device.**
- **MMRESULT waveInStart(HWAVEIN** *hwi* **);**

waveInReset()

- **Stop input on the given waveform-audio input device and resets the current position to zero.**
- **MMRESULT waveInReset(HWAVEIN** *hwi* **);**

waveUnprepareheader()

- **Clean up the preparation performed by waveInPrepareHeader.**
- **MMRESULT waveInUnprepareHeader**(HWAVEIN *hwi*, LPWAVEHDR *pwh*, UINT *cbwh*);

waveInclose()

- **Close the specified waveform-audio input device .**
- **MMRESULT waveInClose**(HWAVEIN hwi);

Process of recording

- waveInOpen() -> waveInPrepareHeader() -> waveInAddBuffer() -> waveInStart() -> waveInStop() -> waveInClose()
- Send voice stream by oRTP

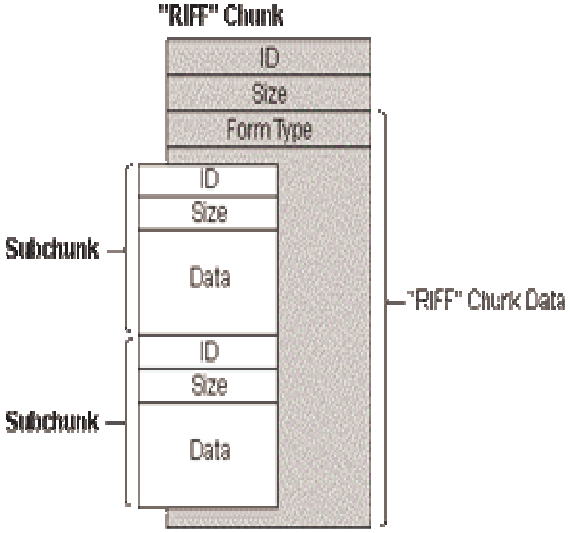
Wave file format

- **RIFF (Resource Interchange File Format)**
- RIFF file =

  One Header (id,size,form type) + Subchunks (id,size,data)
- WAV、AVI files…etc

Wave file format

http://ccrma.stanford.edu/CCRMA/Courses/422/projects/WaveFormat/



The Canonical WAVE file format

endian	File offset (bytes)	field name	Field Size (bytes)	
big	0	ChunkID	4	The "RIFF" chunk descriptor
little	4	ChunkSize	4	The Format of concern here is "WAVE", which requires two sub-chunks: "fmt " and "data"
big	8	Format	4	
big	12	Subchunk1ID	4	The "fmt " sub-chunk
little	16	Subchunk1Size	4	describes the format of the sound information in the data sub-chunk
little	20	AudioFormat	2	
little	22	NumChannels	2	
little	24	SampleRate	4	
little	28	ByteRate	4	
little	32	BlockAlign	2	
little	34	BitsPerSample	2	
big	36	Subchunk2ID	4	The "data" sub-chunk
little	40	Subchunk2Size	4	Indicates the size of the sound information and contains the raw sound data
little	44	data	Subchunk2Size	

RIFF chunk spec.

- ChunkID (4 bytes) // "RIFF"
- ChunkSize (4 bytes)
- Format (4 bytes) // "WAVE"

Fmt chunk spec.

- Subchunk1ID (4 bytes) ex: 'fmt '(+ space)
- Subchunk1Size (4 bytes) ex: 16 or 18
- AudioFormat (2 bytes) ex: PCM = 1 (0x0001)
- NumChannels (2 bytes) ex: 1-mono 2-Stereo
- SampleRate (4 bytes) ex: 8000 ~ 44100
- ByteRate (4 bytes) ex: SampleRate * NumChannels * BitsPerSample/8
- BlockAlign (2 bytes) ex: == NumChannels * BitsPerSample/8
- BitPerSample (2 bytes) ex: 8 bits = 8 , 16bits...etc

- (optional) ExtraparamSize (2 bytes) (if dump file in hex mode : 00 00)若是PCM，則沒有額外資料
- ! (x) ExtraParams ! X bytes

Data chunk spec.

- subchunk2 ID (4 bytes) // 'data'
- subchunk2 Size (4 bytes)
  // == NumSamples * NumChannels * BitsPerSample/8
- Data (* bytes)

MMCKINFO

- The structure contains information about a chunk in a RIFF file.
- typedef struct {
  FOURCC ckid;
  DWORD cksize;
  FOURCC fccType;
  DWORD dwDataOffset;
  DWORD dwFlags;
  } MMCKINFO;

mmioFOURCC()

- The **mmioFOURCC** macro converts four characters into a four-character code.
- **FOURCC mmioFOURCC(** CHAR *ch0*, CHAR *ch1*, CHAR *ch2*, CHAR *ch3*);
- It returns the four-character code created from the given characters.

mmioOpen()

- Open a file.
- **HMMIO mmioOpen(LPSTR** *Filename*, **MMIOINFO** *mmioinfo*, **DWORD** *dwOpenFlags* **);**
- Returns a handle of the opened file.
- If the file cannot be opened, the return value is NULL.
- *dwOpenFlags* :MMIO_CREATE、MMIO_WRITE、MMIO_READ …etc.

mmioCreateChunk()

- Creates a chunk in a RIFFfile that was opened ,and the current file position is the beginning of the data portion of the new chunk.
- **MMRESULT mmioCreateChunk(HMMIO** *hmmio*, **MMCKINFO** *&ck*, **UINT** *wFlags* **);**
- *wFlags* :MMIO_CREATERIFF 、MMIO_CREATELIST
- It returns MMSYSERR_NOERROR if successful

mmioWrite()

- Writes a specified number of bytes to a file opened.
- **LONG mmioWrite(HMMIO** *hmmio*, **char *** *pch*, **LONG** *cch* **);**
- Returns the number of bytes actually written.
- If it's an error, return -1.

mmioAscend()

- Ascends out of a chunk in a RIFF file , and the current file position is the location following the end of the chunk
- **MMRESULT mmioAscend(HMMIO** *hmmio*, **MMCKINFO** *&ck*, **UINT** *wFlags* **);**
- *wFlags* : must be zero.
- It returns MMSYSERR_NOERROR if successful or an error otherwise.

mmioClose()

- closes a file that was opened by using the mmioOpen() function.
- **MMRESULT mmioClose(HMMIO** *hmmio*, **UINT** *wFlags* **);**
- It returns zero if successful or an error otherwise.

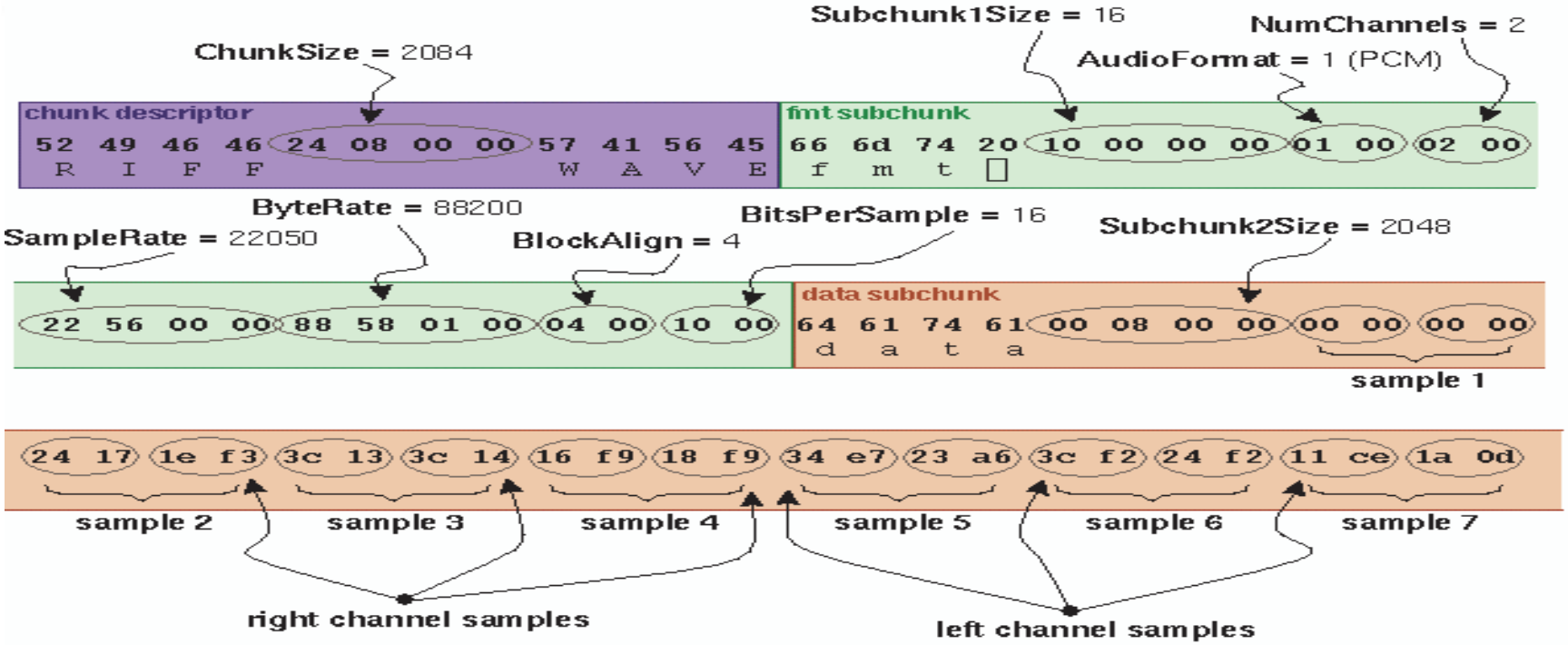
Process of saving file

- mmioOpen() -> mmioFOURCC() -> mmioCreateChunk() -> mmiowrite ->mmioAscend() -> mmioClose()

Demo & dump file

52 49 46 46 24 08 00 00 57 41 56 45 66 6d 74 20 10 00 00 00 01 00 02 00
22 56 00 00 88 58 01 00 04 00 10 00 64 61 74 61 00 08 00 00 00 00 00 00
24 17 1e f3 3c 13 3c 14 16 f9 18 f9 34 e7 23 a6 3c f2 24 f2 11 ce 1a 0d

(Sample shown in hex number)
http://ccrma.stanford.edu/CCRMA/Courses/422/projects/WaveFormat/



MultiMeadia API(Output)

waveOutOpen()

- Open a specified waveform output device for playback.
- **MMRESULT waveOutOpen(LPHWAVEOUT** *phwo*, **UINT** *uDeviceID*, **LPWAVEFORMATEX** *pwfx*, **DWORD** *dwCallback*, **DWORD** *dwInstance*, **DWORD** *fdwOpen* **);**
- It Return MMSYSERR_NOERROR if successful.

waveOutPrepareHeader()

- Prepare a waveform data block for playback.
- **MMRESULT waveOutPrepareHeader(HWAVEOUT** *hwo*, **LPWAVEHDR** *pwh*, **UINT** *cbwh* **);**
- It return MMSYSERR_NOERROR if successful.

waveOutWrite()

- Send a data block to the specified waveform output device.
- **MMRESULT waveOutWrite(HWAVEOUT** *hwo*, **LPWAVEHDR** *pwh*, **UINT** *cbwh* **);**
- It return MMSYSERR_NOERROR if successful.

waveOutReset()

- Stop playback on a specified waveform output device and resets the current position to 0.
- **MMRESULT waveOutReset(HWAVEOUT** *hwo* **);**
- It return MMSYSERR_NOERROR if successful.

waveOutUnprepareHeader()

- Clean up the preparation performed by **waveOutPrepareHeader**.
- **MMRESULT waveOutUnprepareHeader(HWAVEOUT** *hwo*, **LPWAVEHDR** *pwh*, **UINT** *cbwh* **);**
- It return MMSYSERR_NOERROR if successful.

waveOutClose( )

- Close the specified waveform output device.
- **MMRESULT waveOutClose(HWAVEOUT** *hwo* **);**
- It returns MMSYSERR_NOERROR if successful.

mmioDescend()

- Descends into a chunk of a RIFF file that was opened. It can also search for a given chunk.
- **MMRESULT mmioDescend(HMMIO** *hmmio*, **MMCKINFO** *lpck*, **MMCKINFO** *lpckParent*, **UINT** *wFlags* **);**
- *wFlags:MMIO_FINDCHUNK* 、MMIO_FINDRIFF
- Return MMSYSERR_NOERROR if successful
- If the chunk is not founded, return MMIOERR_CHUNKNOTFOUND.

mmioRead()

- reads a specified number of bytes from a file opened by using the mmioOpen function.

- **LONG mmioRead(HMMIO** *hmmio*, **HPSTR** *pch*, **LONG** *cch* **);**
- Return the number of bytes actually read.
- If there is an error, then return value -1.

Process of play file

- mmioOpen() -> mmioDescend() -> mmioRead() -> mmioAscend -> mmioClose()

waveOutOpen() ->waveOutPrepareHeader()-> waveOutWrite()->waveOutPause()-> waveOutUnprepareHeader()-> waveOutClose()

Demo (play file)

Reference

- http://ms11.voip.edu.tw/~beautidays/program/waveout.txt (WAVEOUT)
- http://ms11.voip.edu.tw/~beautidays/program/recordingmakingwave.txt (WAVEIN + SAVE)
- http://ccrma.stanford.edu/CCRMA/Courses/422/projects/WaveFormat/
- http://ms11.voip.edu.tw/~beautidays/mydocument/20061113_ChingChen_MultimediaFileIO.ppt (競真學姐的簡報)
- http://msdn.microsoft.com/ (msdn)