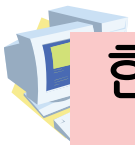
Sound Programming

제4장 Loading Wave Files in OpenAL



헌신에는 신의 섭리가 빚어내는 결실이 따른다

- 모든 시작과 창조 활동에는 한가지 진실이 있다.
  진정으로 자신의 모든 것을 바쳐
  완전히 헌신했을 때 하늘도 움직인다.
  예전에는 발생하지 않았던 그 모든 것들이
  그 사람을 돕기 위해 발생한다.
  모든 일은 결심에서 시작되며,
  이전에 그가 믿지 않았던 사건들이나
  만남 그리고 모든 물질적 수단들이
  그에게 이익이 되고 일이 잘되도록 도와준다.

  - 히말라야 탐험가, W.H. 머레이

꿈, 목표, 그리고 인생

- 꿈은 돋보기로 종이를 태우는 것과 같다

'꿈'은 돋보기로 종이를 태우는 것에 비유할 수 있다.
초점을 맞추고 햇볕을 한 곳에 모은 다음
종이가 탈 때까지 돋보기를 손에 꼭 쥐는 것처럼 우리의 꿈도 마찬가지다.
꿈을 이루려면 굳은 신념을 끝까지 손에 쥐고 포기하면 안 된다.

- 미국 지질학자 *Preston Cloud*

꿈, 목표, 그리고 인생

- 나에게 꿈을 팔아주세요

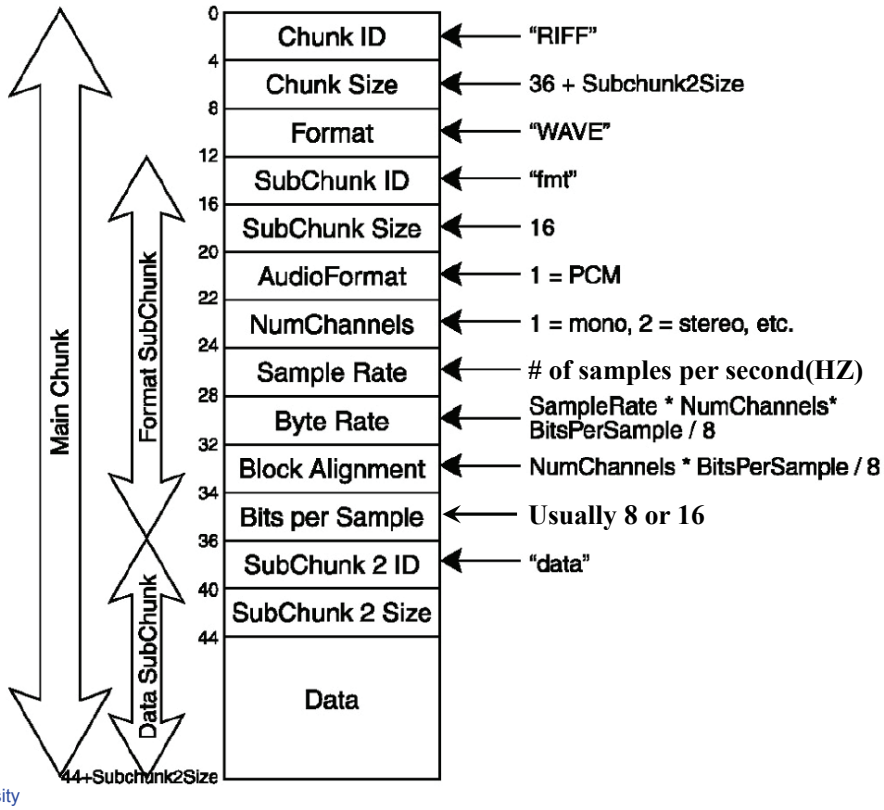
"내게 옷을 팔려 하지 말고,
매혹적인 외모에 대한 기대를 팔아주세요.
내게 장난감을 팔려 하지 말고,
내 아이들이 즐거워하는 모습을 팔아주세요.
내게 물건을 팔려고 하지 마세요.
대신 꿈과 느낌과 자부심과 일상의 행복을 팔아주세요.
제발 내게 물건을 팔려고 하지 마세요."

- 마이클 르뵈프 교수, '평생의 고객으로 만드는 방법'에서

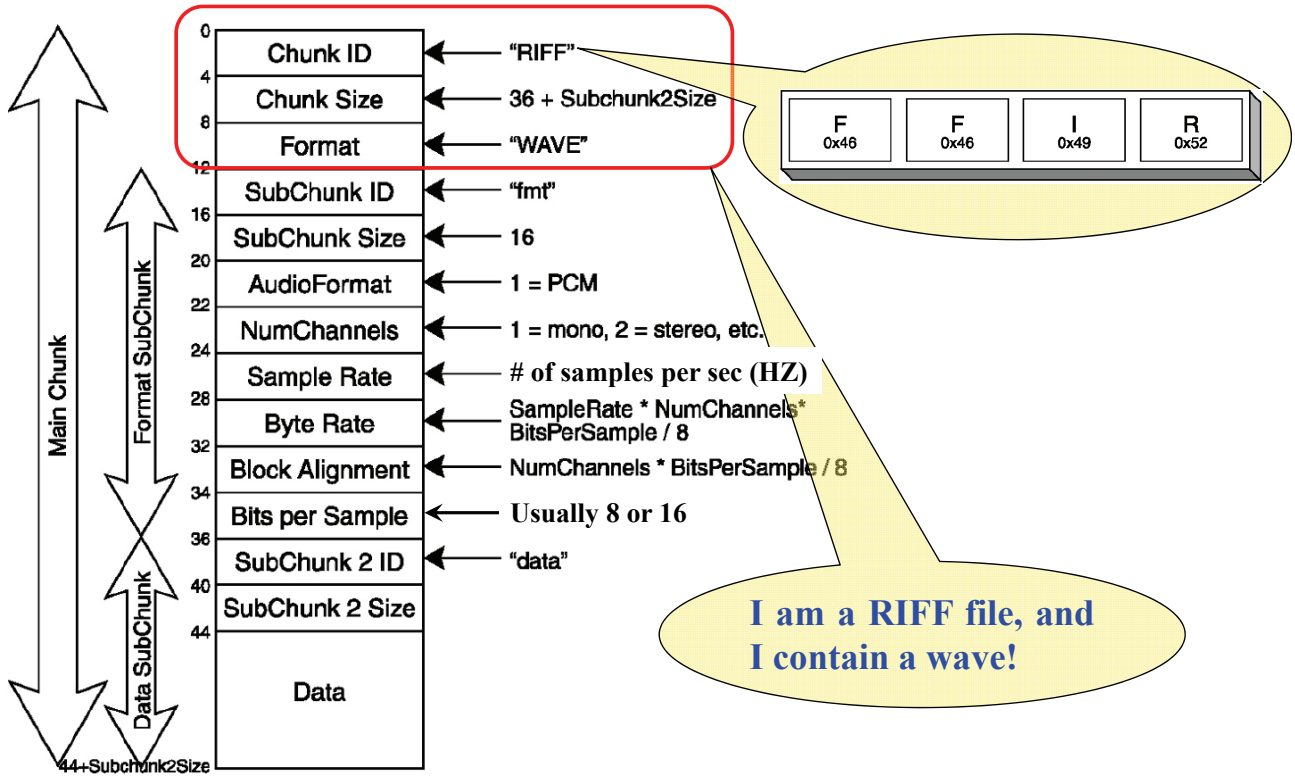
WAV 파일 포맷 분석

- **OpenAL**에서 **Wav** 파일 적재하는 **CWaves** 클래스를 분석하여 **Wav** 파일 포맷을 이해한다.
  - 이를 바탕으로 Wav 파일 편집 프로그램을 작성한다.
- **RIFF**
  - 마이크로소프트가 제정한 멀티미디어 파일 포맷
  - Resource Interchange File Format
  - Wav는 RIFF의 subset에 해당
- **Wav** 파일 포맷 이해
  - chunk 단위로 이해하는 것이 바람직함!

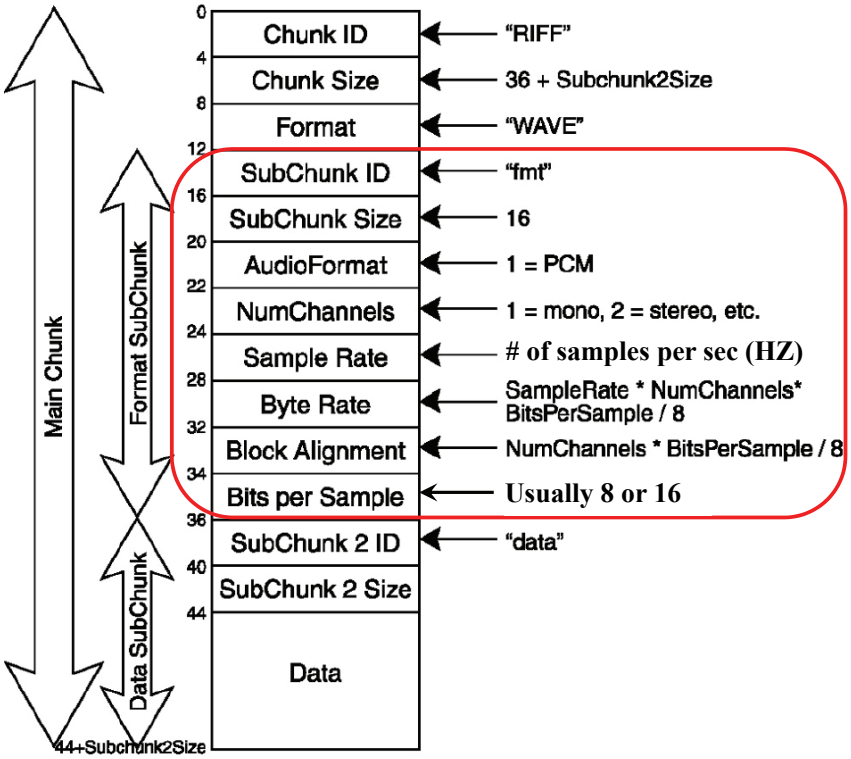
압축 안 된 WAV 파일의 포맷



Main Chunk



Format Chunk



Format Chunk (계속)

- **Audio Format**

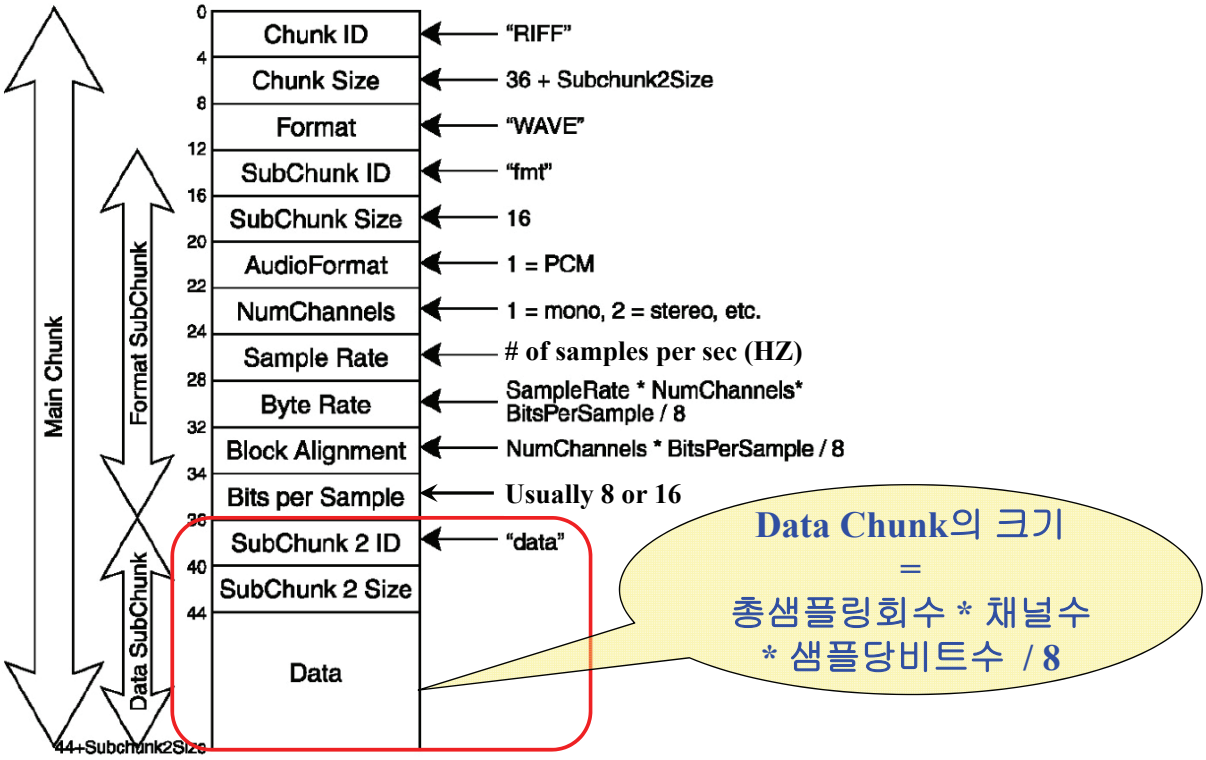
Wave Format	#define in MMREG.H	Number
Unknown (this is bad)	WAVE_FORMAT_UNKNOWN	0
Pulse Code Modulation (PCM)	WAVE_FORMAT_PCM	1
Adaptive Differential PCM	WAVE_FORMAT_ADPCM	2
32-bit floating point	WAVE_FORMAT_IEEE_FLOAT	3
CCITT G.711 A-law	WAVE_FORMAT_ALAW	6
CCITT G.711 u-law	WAVE_FORMAT_MULAW	7
MPEG Layer 3 (MP3)	WAVE_FORMAT_MPEGLAYER3	55

- **Number of Channels**
  - How many channels the WAV file has
  - 1 : mono, 2 : stereo, …
- **Sample Rate**
  - 8000 : 8000 Hz, 44100 : 44.1KHz, …

Format Chunk (계속)

- **Byte Rate**
  - 이 사운드 재생을 위해 1초당 소요되는 바이트 수
    = SampleRate * NumChannels * BitsPerSample / 8
- **Block Align**
  - 모든 채널에 대한 샘플링 1번에 필요한 바이트 수
    = NumChannels * BitsPerSample / 8
- **Bits per Sample**
  - 샘플링 1번에 소요되는 비트 수 (8 또는 16)

Data Chunk



Wav 파일 읽고 해석하기

- **CWaves 클래스**
  - OpenAL framework 라이브러리의 일부로 Wav 제어 클래스
  - Wav 파일을 읽어 해석하고 분해하여 재생을 돕는 클래스
  - framework의 CWaves.h와 CWaves.cpp에 정의되어 있음

```
class CWaves {
public:
    CWaves();
    virtual ~CWaves();

    WAVERESULT LoadWaveFile(const char *szFilename, WAVEID *WaveID);
    WAVERESULT OpenWaveFile(const char *szFilename, WAVEID *WaveID);
    WAVERESULT ReadWaveData(WAVEID WaveID, void *pData, unsigned long ulDataSize,
                            unsigned long *pulBytesWritten);
    WAVERESULT SetWaveDataOffset(WAVEID WaveID, unsigned long ulOffset);
    WAVERESULT GetWaveDataOffset(WAVEID WaveID, unsigned long *pulOffset);
    WAVERESULT GetWaveType(WAVEID WaveID, WAVEFILETYPE *pwfType);
    WAVERESULT GetWaveFormatExHeader(WAVEID WaveID, WAVEFORMATEX *pWFEX);
    WAVERESULT GetWaveFormatExtensibleHeader(WAVEID WaveID,
                                             WAVEFORMATEXTENSIBLE *pWFEXT);
.....
```

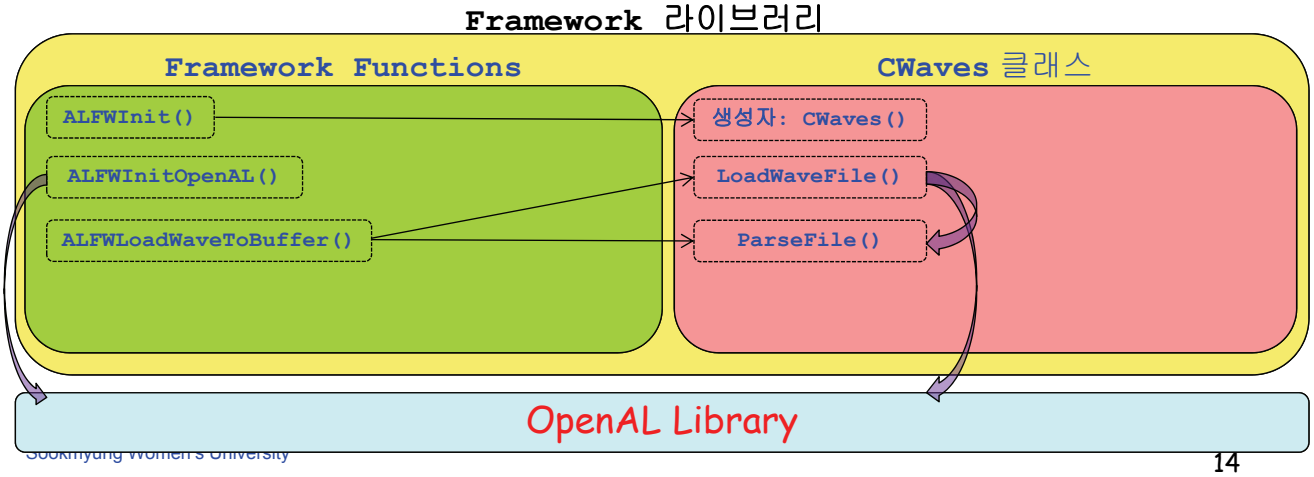
CWaves 클래스 (계속)

```
……
    WAVERESULT GetWaveData(WAVEID WaveID, void **ppAudioData);
    WAVERESULT GetWaveSize(WAVEID WaveID, unsigned long *pulDataSize);
    WAVERESULT GetWaveFrequency(WAVEID WaveID, unsigned long *pulFrequency);
    WAVERESULT GetWaveALBufferFormat(WAVEID WaveID,
                    PFNALGETENUMVALUE pfnGetEnumValue, unsigned long *pulFormat);
    WAVERESULT DeleteWaveFile(WAVEID WaveID);
    WAVERESULT SaveWaveFile(const char *szFilename, WAVEID WaveID);
    WAVERESULT SaveWaveFile(const char *szFilename, WAVEID WaveID, unsigned long StartSec,
                                    unsigned long EndSec);

    char *GetErrorString(WAVERESULT wr, char *szErrorString,
                                unsigned long nSizeOfErrorString);
    bool IsWaveID(WAVEID WaveID);

private:
    WAVERESULT ParseFile(const char *szFilename, LPWAVEFILEINFO pWaveInfo);
    WAVEID InsertWaveID(LPWAVEFILEINFO pWaveFileInfo);

    LPWAVEFILEINFO m_WaveIDs[MAX_NUM_WAVEID];
};
```


Review – 3장 PlayStatic 실행흐름

3장의 PlayStatic 예제의 main() 흐름 review

- ALFWInit(): CWaves 인스턴스 생성. 전역변수 g_pWaveLoader에 포인터를 저장
- ALFWInitOpenAL(): OpenAL 객체들을 생성하고 초기화
- alGenBuffers(): OpenAL buffer 생성
- ALFWLoadWaveToBuffer(): Wav 파일을 파싱하고 Wav 데이터를 buffer로 적재한다.
- alGenSources() ➔ alSourcei() ➔ alSourcePlay() ➔ alGetSourcei() ➔ alSourceStop() ➔ alDeleteSource() ➔ alDeleteBuffer()



Wav 파일 적재 관련 Framework 함수

- **ALFWLoadWaveToBuffer()**

```
ALboolean ALFWLoadWaveToBuffer(const char *szWaveFile, ALuint uiBufferID,
                                        WAVEID *pWaveID, ALenum eXRAMBufferMode)
{
   WAVEID WaveID;
   ALint iDataSize, iFrequency;
   ALenum eBufferFormat;
   ALchar* pData;
   ALboolean bReturn;

   bReturn = AL_FALSE;
   if (g_pWaveLoader) {
      if (SUCCEEDED(g_pWaveLoader->LoadWaveFile(szWaveFile, &WaveID))) {
         if ((SUCCEEDED(g_pWaveLoader->GetWaveSize(WaveID, (unsigned long*)&iDataSize))) &&
             (SUCCEEDED(g_pWaveLoader->GetWaveData(WaveID, (void**)&pData))) &&
             (SUCCEEDED(g_pWaveLoader->GetWaveFrequency(WaveID, (unsigned long*)&iFrequency))) &&
             (SUCCEEDED(g_pWaveLoader->GetWaveALBufferFormat(WaveID, &alGetEnumValue,
                                                    (unsigned long*)&eBufferFormat)))) {
            // Set XRAM Mode (if application)
            if (eaxSetBufferMode && eXRAMBufferMode)
               eaxSetBufferMode(1, &uiBufferID, eXRAMBufferMode);

            ......
```


Wav 파일 적재 관련 Framework 함수

- **ALFWLoadWaveToBuffer() (계속)**

```
            ......
            alGetError();
            alBufferData(uiBufferID, eBufferFormat, pData, iDataSize, iFrequency);
            if (alGetError() == AL_NO_ERROR) {
               bReturn = AL_TRUE;
               if (pWaveID)                    // 지금 로드한 Wave 파일을 프로그램에서 제어할 수 있게
                  *pWaveID = WaveID;           // Wave ID를 인자를 통해 리턴하게 함.
            }
         } // end of if (SUCCEED(...
      }
   } // end of if (g_pWaveLoader)

   return bReturn;
}
```

ALenum alGetError(ALvoid)
- 현재의 오류 상태를 리턴한 후 에러 상태를 리셋한다.

```
void alBufferData(      // fills a buffer with audio data
   ALuint buffer,       // 데이터가 채워질 버퍼 ID
   ALenum format,       // 채울 데이터의 Wav 포맷 ID
   const ALvoid *data,  // 채울 오디오 데이터 포인터
   ALsizei size,        // 오디오 데이터 길이(바이트)
   ALsizei freq         // 오디오 데이터 Sampling Rate
);
```

Wav 파일 적재 관련 CWaves 함수

- **CWaves::LoadWaveFile()**

```
WAVERESULT CWaves::LoadWaveFile(const char *szFilename, WAVEID *pWaveID)
{
   WAVERESULT wr = WR_OUTOFMEMORY;
   LPWAVEFILEINFO pWaveInfo;

   pWaveInfo = new WAVEFILEINFO;
   if (pWaveInfo) {
      if (SUCCEEDED(wr = ParseFile(szFilename, pWaveInfo))) {
         // Allocate memory for sample data
         pWaveInfo->pData = new char[pWaveInfo->ulDataSize];
         if (pWaveInfo->pData) {
            // Seek to start of audio data
            fseek(pWaveInfo->pFile, pWaveInfo->ulDataOffset, SEEK_SET);

            // Read Sample Data
            if (fread(pWaveInfo->pData, 1, pWaveInfo->ulDataSize, pWaveInfo->pFile) ==
                                                                  pWaveInfo->ulDataSize) {
               long lLoop = 0;
               for (lLoop = 0; lLoop < MAX_NUM_WAVEID; lLoop++) {
                  if (!m_WaveIDs[lLoop]) {
                     m_WaveIDs[lLoop] = pWaveInfo;
                     *pWaveID = lLoop;
                     wr = WR_OK;
                     break;
                  }
               }
               ……
```


Wav 파일 적재 관련 CWaves 함수

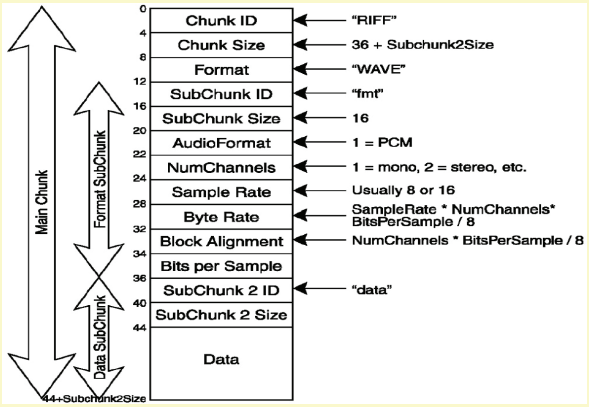
- **CWaves::LoadWaveFile() (계속)**

```
               ……
               if (lLoop == MAX_NUM_WAVEID) {
                  delete pWaveInfo->pData;
                  wr = WR_OUTOFMEMORY;
               }
            }
            else {
               delete pWaveInfo->pData;
               wr = WR_BADWAVEFILE;
            }
         }
         else
            wr = WR_OUTOFMEMORY;

         fclose(pWaveInfo->pFile);
         pWaveInfo->pFile = 0;
      }

      if (wr != WR_OK)
         delete pWaveInfo;
   }

   return wr;
}
```

Wav 파일 적재 관련 CWaves 함수

- **CWaves::ParseFile()**

```
WAVERESULT CWaves::ParseFile(const char *szFilename, LPWAVEFILEINFO pWaveInfo)
{
   WAVEFILEHEADER waveFileHeader;
   RIFFCHUNK      riffChunk;
   WAVEFMT        waveFmt;
   WAVERESULT     wr = WR_BADWAVEFILE;

   if (!szFilename || !pWaveInfo)
      return WR_INVALIDPARAM;

   memset(pWaveInfo, 0, sizeof(WAVEFILEINFO));

   // Open the wave file for reading
   fopen_s(&pWaveInfo->pFile, szFilename, "rb");
   if (pWaveInfo->pFile) {
      // Read Wave file header
      fread(&waveFileHeader, 1, sizeof(WAVEFILEHEADER), pWaveInfo->pFile);
      if (!_strnicmp(waveFileHeader.szRIFF, "RIFF", 4) &&
          !_strnicmp(waveFileHeader.szWAVE, "WAVE", 4)) {
         while (fread(&riffChunk, 1, sizeof(RIFFCHUNK), pWaveInfo->pFile) == sizeof(RIFFCHUNK)) {
            if (!_strnicmp(riffChunk.szChunkName, "fmt ", 4)) {
               if (riffChunk.ulChunkSize <= sizeof(WAVEFMT)) {
                  ……
```



Wav 파일 적재 관련 CWaves 함수

- **CWaves::ParseFile() (계속)**

```
                  ……
                  fread(&waveFmt, 1, riffChunk.ulChunkSize, pWaveInfo->pFile);

                  // Determine if this is a WAVEFORMATEX or WAVEFORMATEXTENSIBLE wave file
                  if (waveFmt.usFormatTag == WAVE_FORMAT_PCM)  {
                     pWaveInfo->wfType = WF_EX;
                     memcpy(&pWaveInfo->wfEXT.Format, &waveFmt, sizeof(PCMWAVEFORMAT));
                  }
                  else if (waveFmt.usFormatTag == WAVE_FORMAT_EXTENSIBLE) {
                     pWaveInfo->wfType = WF_EXT;
                     memcpy(&pWaveInfo->wfEXT, &waveFmt, sizeof(WAVEFORMATEXTENSIBLE));
                  }
               } // end of if (riffChunk.ulChunkSize <= sizeof(WAVEFMT))
               else // 이상한 포맷임. format chunk 건너뜀.
                  fseek(pWaveInfo->pFile, riffChunk.ulChunkSize, SEEK_CUR);
            } // if (!_strnicmp(riffChunk.szChunkName, "fmt ", 4))
            else if (!_strnicmp(riffChunk.szChunkName, "data", 4)) {       // 데이터 청크 발견
               pWaveInfo->ulDataSize = riffChunk.ulChunkSize;
               pWaveInfo->ulDataOffset = ftell(pWaveInfo->pFile);
               fseek(pWaveInfo->pFile, riffChunk.ulChunkSize, SEEK_CUR); // 혹시 다음 data chunk가 있으면
            }
            else // format chunk도 아니고 data chunk도 아님
               fseek(pWaveInfo->pFile, riffChunk.ulChunkSize, SEEK_CUR);
            ……
```

Wav 파일 적재 관련 CWaves 함수

- **CWaves::ParseFile() (계속)**

```
            ......
            // Ensure that we are correctly aligned for next chunk
            if (riffChunk.ulChunkSize & 1)
            fseek(pWaveInfo->pFile, 1, SEEK_CUR);
         } // end of while

         if (pWaveInfo->ulDataSize && pWaveInfo->ulDataOffset &&
             ((pWaveInfo->wfType == WF_EX) || (pWaveInfo->wfType == WF_EXT)))
            wr = WR_OK;
         else
            fclose(pWaveInfo->pFile);
      }
   }
   else
      wr = WR_INVALIDFILENAME;

   return wr;
}
```


Wav 파일 적재 관련 CWaves 구조체

- **CWaves::ParseFile()**에서 사용하는 **Wav** 헤더 관련 자료구조

```
typedef struct {
    char            szRIFF[4];
    unsigned long   ulRIFFSize;
    char            szWAVE[4];
} WAVEFILEHEADER;

typedef struct {
    char            szChunkName[4];
    unsigned long   ulChunkSize;
} RIFFCHUNK;
```

```
typedef struct {
    unsigned short  usFormatTag;
    unsigned short  usChannels;
    unsigned long   ulSamplesPerSec;
    unsigned long   ulAvgBytesPerSec;
    unsigned short  usBlockAlign;
    unsigned short  usBitsPerSample;
    unsigned short  usSize;
    unsigned short  usReserved;
    unsigned long   ulChannelMask;
   GUID         guidSubFormat;
} WAVEFMT;
```

CWaves 클래스에 Save 메소드 추가하기

- Wav 파일을 읽고 재생하는 것까지는 되었으니 *로딩된 Wav 파일을 파일로 저장하는 메소드*를 만들어 본다.
- 로딩된 Wav 파일의 헤더 정보들은 CWaves.h에 정의된 WAVEFILEINFO 구조체에 저장됨

```
typedef struct tWAVEFORMATEX {
    WORD    wFormatTag;
    WORD    nChannels;
    DWORD   nSamplesPerSec;
    DWORD   nAvgBytesPerSec;
    WORD    nBlockAlign;
    WORD    wBitsPerSample;
    WORD    cbSize;
} WAVEFORMATEX;

typedef struct {
    WAVEFORMATEX    Format;
    union {
        WORD wValidBitsPerSample;// bits of precision
        WORD wSamplesPerBlock;   // valid if wBitsPerSample
                                 //   ==0
        WORD wReserved;     // If neither applies, set to zero.
    } Samples;

    DWORD   dwChannelMask;       // which channels are
                                 // present in stream
    GUID    SubFormat;
} WAVEFORMATEXTENSIBLE,
  *PWAVEFORMATEXTENSIBLE;
```

```
typedef struct {
    WAVEFILETYPE            wfType;
    WAVEFORMATEXTENSIBLE wfEXT;
    char                *pData;
    unsigned long       ulDataSize;
    FILE                *pFile;
    unsigned long       ulDataOffset;
} WAVEFILEINFO, *LPWAVEFILEINFO;
```


Step 1 – CWaves.h 변경

```
class CWaves {
public:
        CWaves();
        virtual ~CWaves();

        WAVERESULT LoadWaveFile(const char *szFilename, WAVEID *WaveID);
        WAVERESULT OpenWaveFile(const char *szFilename, WAVEID *WaveID);
        WAVERESULT ReadWaveData(WAVEID WaveID, void *pData, unsigned long ulDataSize,
                                unsigned long *pulBytesWritten);
        WAVERESULT SetWaveDataOffset(WAVEID WaveID, unsigned long ulOffset);
        WAVERESULT GetWaveDataOffset(WAVEID WaveID, unsigned long *pulOffset);
        WAVERESULT GetWaveType(WAVEID WaveID, WAVEFILETYPE *pwfType);
        WAVERESULT GetWaveFormatExHeader(WAVEID WaveID, WAVEFORMATEX *pWFEX);
        WAVERESULT GetWaveFormatExtensibleHeader(WAVEID WaveID, WAVEFORMATEXTENSIBLE *pWFEXT);
        WAVERESULT GetWaveData(WAVEID WaveID, void **ppAudioData);
        WAVERESULT GetWaveSize(WAVEID WaveID, unsigned long *pulDataSize);
        WAVERESULT GetWaveFrequency(WAVEID WaveID, unsigned long *pulFrequency);
        WAVERESULT GetWaveALBufferFormat(WAVEID WaveID, PFNALGETENUMVALUE pfnGetEnumValue,
                                         unsigned long *pulFormat);
        WAVERESULT DeleteWaveFile(WAVEID WaveID);
        WAVERESULT SaveWaveFile(const char *szFilename, WAVEID WaveID);
        WAVERESULT SaveWaveFile(const char *szFilename, WAVEID WaveID, unsigned long StartSec,
                                unsigned long EndSec);

        char *GetErrorString(WAVERESULT wr, char *szErrorString, unsigned long nSizeOfErrorString);
        bool IsWaveID(WAVEID WaveID);

private:
        WAVERESULT ParseFile(const char *szFilename, LPWAVEFILEINFO pWaveInfo);
        WAVEID InsertWaveID(LPWAVEFILEINFO pWaveFileInfo);

        LPWAVEFILEINFO  m_WaveIDs[MAX_NUM_WAVEID];
};
```

Step 2 - Save() 메소드 설계

목표: 이미 CWaves 클래스로 로드한 Wav 파일을 다른 이름으로 저장한다.

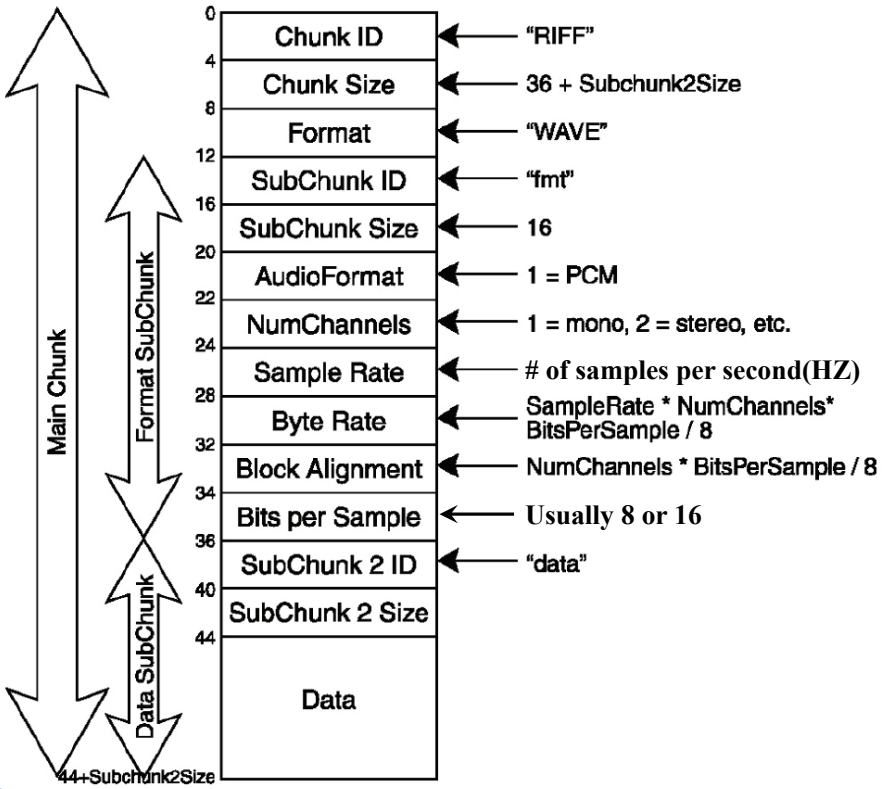
- 인자로 받은 저장할 파일 이름과 Wave ID가 유효한지 검사
- 저장할 파일 생성 후 Main Chunk 저장
- Format Chunk 중 RIFF Chunk 저장
- Format Chunk 중 WAVEFORMAT 16바이트를 저장한다.
- Data Chunk 저장
- Wav 데이터 저장

Step 3 - Save() 메소드 구현

```
WAVERESULT CWaves::SaveWaveFile(const char *szFilename, WAVEID WaveID)
{
   LPWAVEFILEINFO pWaveInfo;
   WAVERESULT wr = WR_BADWAVEFILE;
   FILE *fpSave;
   WAVEFILEHEADER waveFileHeader;
   RIFFCHUNK riffChunk;
   WAVEFMT waveFmt;

   if (!szFilename)
      return WR_INVALIDPARAM;

   if (IsWaveID(WaveID)) {                    // 주어진 WaveID가 유효한 것인지 검사
      pWaveInfo = m_WaveIDs[WaveID];          // 유효하면 CWave 클래스 객체 포인터를 얻어 온다.
      fopen_s(&fpSave, szFilename, "wb");     // 저장용 wav 파일을 오픈
      if (fpSave) {
         // main chunk 저장
         ::strncpy(waveFileHeader.szRIFF, "RIFF", 4);
         waveFileHeader.ulRIFFSize = 36 + pWaveInfo->ulDataSize;
         ::strncpy(waveFileHeader.szWAVE, "WAVE", 4);
         fwrite(&waveFileHeader, sizeof(WAVEFILEHEADER), 1, fpSave);

         // format chunk 중 riff chunk 저장
         ::strncpy(riffChunk.szChunkName, "fmt ", 4);
         riffChunk.ulChunkSize = sizeof(PCMWAVEFORMAT);          // It should be 16.
         ::fwrite(&riffChunk, sizeof(RIFFCHUNK), 1, fpSave);
         .....
```

Step 3 – Save() 메소드 구현(계속)

```
        ……
        // format chunk 중 WAVEFORMAT 16바이트를 저장한다.
        waveFmt.usFormatTag = pWaveInfo->wfEXT.Format.wFormatTag;
        waveFmt.usChannels = pWaveInfo->wfEXT.Format.nChannels;
        waveFmt.ulSamplesPerSec = pWaveInfo->wfEXT.Format.nSamplesPerSec;
        waveFmt.ulAvgBytesPerSec = pWaveInfo->wfEXT.Format.nAvgBytesPerSec;
        waveFmt.usBlockAlign = pWaveInfo->wfEXT.Format.nBlockAlign;
        waveFmt.usBitsPerSample = pWaveInfo->wfEXT.Format.wBitsPerSample;
        ::fwrite(&waveFmt, riffChunk.ulChunkSize, 1, fpSave);

        // data chunk 저장
        ::strncpy(riffChunk.szChunkName, "data", 4);
        riffChunk.ulChunkSize = pWaveInfo->ulDataSize;
        ::fwrite(&riffChunk, sizeof(RIFFCHUNK), 1, fpSave);

        // Wave 데이터 저장
        if (pWaveInfo->pData) {
           ::fwrite(pWaveInfo->pData, pWaveInfo->ulDataSize, 1, fpSave);
           wr = WR_OK;
        }
        else wr = WR_BADWAVEFILE;

        ::fclose(fpSave);
     }
     else wr = WR_INVALIDFILENAME;
  }
  else wr = WR_INVALIDWAVEID;

  return wr;
}
```

Home Works

- 4장 보고서에서 사용할 음악 Wav 파일을 준비
- 이 Wav 파일을 로드한 후 이 Wav 파일을 4등분 하는 프로그램을 작성한다.
  - 4등분된 Wav 파일 4개는 서로 다른 이름의 Wav 파일로 저장되어야 한다.
- 4등분된 Wav 파일 각각을 순차적으로 재생하는 프로그램을 작성한다.

Home Works Hints

- 4장 보고서에서 사용할 음악 Wav 파일을 준비
  - 사운드 파일 포맷 변환 프로그램을 이용하여,
  - 각자 가지고 있는 Mp3 파일을 Wav 파일로 변환한다.
- Wav 파일을 로드한 후 이 Wav 파일을 4등분 하는 프로그램을 작성한다.
  - CWaves 클래스의 멤버 함수에

    `WAVERESULT SaveWaveFile(const char *szFilename, WAVEID WaveID, unsigned long StartSec, unsigned long EndSec);` 메소드를 추가한다.
    - 사운드 데이터 길이가 바뀌었으므로, SubChunk2Size 필드, 즉, Wav 데이터 길이가 저장되는 4바이트 필드 값을 수정해야 함.
    - nStartSec이 가리키는 Data 섹션 내의 위치를 알아내야 함.
    - nEndSec이 가리키는 Data 섹션 내의 위치를 알아내야 함.
    - 이 두 위치 사이에 있는 데이터만 파일로 저장하면 됨.

Home Works Hints (계속)

- **4**등분된 **Wav** 파일 각각을 순차적으로 재생하도록 **main()** 함수를 수정한다.

Q & A