Tesina

Interpretazione e
implementazione in “ANSI C”
della raccomandazione
ITU-R BS.1387-1:
P e r c e p t u a l
E v a l u a t i o n
o f
A u d i o
Q u a l i t y

UNIVERSITÀ POLITECNICA DELLE MARCHE

Titolare del corso: Studente:
Prof. Ing. Francesco Piazza **Giuseppe Gottardi**
Anno Accademico 2002/2003

CHAPTER II
Acquisizione dati
2.1 Il formato dei file wave

I WAV files sono probabilmente il formato più semplice e diffuso per la memorizzazione di campioni audio. Così come MPEG ed altri formati di compressione, i WAVs memorizzano i campioni in modo “raw”, pertanto non è necessario un pre-processing ma è sufficiente la sola formattazione dei dati.

Un WAV file è strutturato secondo tre "chunks" di informazioni: Il RIFF chunk che identifica il file come WAV file, il FORMAT chunk che identifica i parametri come il sample rate e il numero di bit per campione, e il DATA chunk che contiene i dati effettivi dei campioni.

Una rappresentazione schematica di questi Chunk è la seguente:

RIFF Chunk (12 bytes)
Byte Number
0 - 3 "RIFF" (ASCII Characters)
4 - 7 Total Length Of Package To Follow (Binary, little endian)
8 - 11 "WAVE" (ASCII Characters)

FORMAT Chunk (24 bytes)
Byte Number
0 - 3 "fmt_" (ASCII Characters)
4 - 7 Length Of FORMAT Chunk (Binary, always 0x10)
8 - 9 Always 0x01
10 - 11 Channel Numbers (Always 0x01=Mono, 0x02=Stereo)
12 - 15 Sample Rate (Binary, in Hz)
16 - 19 Bytes Per Second

```
20 - 21 Bytes Per Sample: 1=8 bit Mono, 2=8 bit Stereo or 16 bit Mono,
4 =16 bit Stereo
22 - 23 Bits Per Sample
```

DATA Chunk

```
Byte Number
0 - 3 "data" (ASCII Characters)
4 - 7 Length Of Data To Follow
8 - end Data (Samples)
```

L'approccio più facile per l'analisi dei WAV files è quello di guardare come vengono memorizzati i dati al loro interno. In questo caso, esaminiamo il file "Utopia – Arresto critico.WAV" che è un file standard in ogni versione di Windows (e che si sente molto spesso).

Questo file ha 16-bit per campione, mono, campionato a 22.050 kHz di lunghezza 5824 bytes. Facendo un dump del file con un editor esadecimale come hexdump (un editor testuale fornito di default nella distribuzione slackware di linux) possiamo vedere i diversi chunk.

```
root@eax:~# hexdump -C -n 1000 /mnt/fat/winnt/Media/Utopia\ -\ Arresto\
critico.WAV | more
00000000 52 49 46 46 b8 16 00 00 57 41 56 45 66 6d 74 20 |RIFF....WAVEfmt |
00000010 10 00 00 00 01 00 01 00 22 56 00 00 44 ac 00 00 |........"V..D...|
00000020 02 00 10 00 64 61 74 61 8c 0e 00 00 e5 f5 ff f7 |....data........|
00000030 18 fa 7f fb 98 fd ff fe 17 01 31 03 4a 05 b1 06 |..........1.J...|
00000040 ca 08 ca 0f ca 16 97 12 17 0f fe 13 e4 18 7e 17 |..............~.|
00000050 17 16 64 15 64 15 e4 18 17 1d ca 1d 7e 1e ca 24 |..d.d.......~..$|
00000060 17 2b e4 26 64 23 31 2d fe 36 7e 33 b1 30 17 32 |.+.&d#1-.6~3.0.2|
00000070 7e 33 64 2a 4a 21 64 15 7e 09 97 04 b2 ff b2 f1 |~3d*J!d.~.......|
00000080 b2 e3 ff e2 4b e2 32 d9 cb d0 7f d1 e5 d2 18 d0 |....K.2.........|
```

Come atteso, il file inizia con la stringa di 4 caratteri ASCII "RIFF" che identifica il WAV file. I successivi 4 bytes 0x000016b8 (essendo little endiand) ci dicono che la lunghezza dei dati nel file è di 5816 bytes (5824 – 5816 = 8 bytes).

Segue la stringa "WAVE" e "fmt ". Alla linea 2 troviamo il valore 0x00000010 nei primi 4 bytes che rappresenta la lunghezza del FORMAT chunk (sempre costante a 0x00000010, 16 in decimale). I successivi 4 bytes sono 0x0001 (rappresenta il tipo di codifica, ed è sempre ad uno per la PCM) e 0x0001 che rappresenta il numero di canali (in questo caso è mono).

I successivi 4 bytes sono 0x00005622 che corrispondono al sample rate di 22050 in decimale, e poi troviamo 0x0000ac44, cioè il numero di bytes per secondo (16 poiché il file è campionato a 22.050 kHz ed è a 16 bit). I prossimi 2 bytes che sono 0x0002 mostrano il numero di bytes per campione e 0x0010 è il numero di bit per campione (16- bit).

Infine, la stringa "data" seguita da 0x00000e8c (3724 in decimale) che rappresenta il numero di bytes di dati che seguono. Il dato è un numero signed che va da 0x0000 a 0xffff, e quindi in

complemento a 2, questo significa che 0x8000 è uno 0 mentre 0x0000 è il massimo negativo e 0xffff è il massimo positivo.

Un possibile tipo di dato strutturato in C che rappresenta il formato di un WAV file può essere:

```
struct WavHeader {
char ChunkType1[4]; /* 'R', 'I', 'F', 'F' */
int length; /* length of everything after this in file */
char ContainerType[4]; /* 'W', 'A', 'V', 'E' */
char ChunkType2[4]; /* 'f', 'm', 't', ' ' */
int FormatChunkDataLength;
struct FormatChunkData fcd;
char ChunkType3[4]; /* 'd', 'a', 't', 'a' */
int SoundLength; /* in bytes */
/* After this, n sound samples of data in file */
};
struct FormatChunkData {
short CompressionCode; /* 0x0001 for PCM */
short NumberOfChannels;
int SamplesPerSecond;
int AvgNumberBytesPerSecond;
short BlockAlignment;
short SignificantBitsPerSample;
/* 16 bytes total */
};
```

2.2 Little Endian o Big Endian

I nomi big-endian e little-endian si riferiscono all'ordine con cui sono ordinati i bytes nei nei tipi multi-byte.

- Nell'architettura big-endian i bytes più a sinistra (più significativi) sono messi per primi nella memoria (*big-endfirst*);
- Nell'architettura little-endian i bytes più a destra (meno significativi) sono messi per primi nella memoria (*little-endfirst*);

Per illustrare i due ordinamenti consideriamo il numero intero 123456 memorizzato in tipo intero di 32 bit, ossia in un intero di 4-byte:

```
123456 = 00000000 00000001 11100010 01000000
```

Nelle due architetture i 4 bytes sono organizzazti in memoria come segue:

Indirizzo	Little-endian	Big-endian
0xbffffb14	01000000	00000000
0xbffffb15	11100010	00000001
0xbffffb16	00000001	11100010
0xbffffb17	00000000	01000000

Il termine big-endian e little-endian sono derivanti dai Lillipuziani dei *Viaggi di Gulliver*, il cui problema principale era se le uova bollite dovessero essere aperte dal lato grande (bigendian) o da quello piccolo (little-endian).

Il problema della conversione tra un ordinamento e l'altro è noto come il *NUXI problem*. Infatti se la parola *UNIX* è memorizzata in due 2-byte variabili, allora in un sistema big-endian in memoria questa appare UNIX, mentre in un sistema little-endian apparirebbe NUXI.

Osserviamo che l'ordinamento big-endian e little-endian si riscontra anche nel modo di scrivere le date. Gli Europei scrivono la data come gg/mm/aa quindi un ordinamento little-endian. Per contro i Giapponesi la scrivono come aa/mm/gg e quindi un ordinamento bigendian.

Gli Americani la scrivono come mm/gg/aa e quindi non bigendian e non little-endian, ordinamento middle-endian.

Ordinamenti middle-endian, con ordinamenti perversi dei bytes, sipossono trovare anche su alcuni computers.

Anche i bit all'interno di un byte possono essere ordinati in maniera diversa, ad esempio big-endian o little-endian. L'ordinamento dei bit non deve necessariamente riflettere quello dei bytes.