University of Wollongong

School of Computer Science and Software Engineering (SCSSE)

CSCI124/MCS9124 Applied Programming Spring 2015

Lab 6 (1 mark)

Due 11:59pm Thurs 23rd April

Objective:

To work with binary files

Task:

MP3 is a commonly used file format for Audio files (tracks). Typically a single MP3 file represents one track. When MP3 was first popularised, people were converting much of their music collection on preexisting media to this format for use in software applications such as WinAMP and iTunes.

The problem however was there was no mechanism to store information about individual tracks e.g. artist, genre etc. The solution to this was simple – attach some metadata describing the track.

ID3 tags are used today to record information about tracks. There are a number of versions of the ID3 standard. Most MP3 players today use the ID3v2 standard which encodes track information at the beginning and end of the MP3 file.

The version of ID3 which we will use for this task however is Version 1.1 which is fairly simple. ID3v1.1 places track information at the end of the file.

An ID3v1.1 tag occupies the last 128 bytes of the MP3 file. An ID3v1.1 tag has the following structure:

```
Header          3 bytes
Song Title      30 bytes
Artist          30 bytes
Album           30 bytes
Year            4 bytes
Comment         29 bytes
Track Number    1 byte
Genre           1 byte
```

In order to validate the ID3 information, the value for the Header (3 bytes) is `TAG`. It should also be noted that attributes in a ID3 tag are always padded with `\0` bytes. The sample file provided has the following ID3information.

```
TAGPerfect Timing (This Morning)\0Orba
Squara\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0Sunshyness\0\0\0\0\0\0\
0\0\0\0\0\0\0\0\0\0\0\0\0\02006spinner.com\0\0\0\0\0\0\0\0\0\0\0\
0\0\0\0\0Z'
```

It should be noted the `\0` represents a NULL byte. When viewed through a text editor you may not see this byte as it has a zero ASCII value.

The question is how do you represent track number and genre in 1 byte? Well that's easy. The track number is cast to a character in the file so `Z` represents track number 90 (that's Z's ASCII value). For 126 genres, the genre byte when cast gives you the value in the following array.

```
char genre[126][128] = {“Blues”, “Classic Rock”, “Country”,
“Dance”, “Disco”, “Funk”, “Grunge”, “Hip-Hop”, “Jazz”, “Metal”,
“New Age”, “Oldies”, “Other”, “Pop”, "R&B", "Rap", "Reggae",
"Rock","Techno", "Industrial", "Alternative","Ska", "Death
Metal","Pranks", "Soundtrack", "Euro-Techno","Ambient", "Trip-Hop",
"Vocal", "Jazz+Funk", "Fusion", "Trance","Classical",
"Instrumental", "Acid", "House",
"Game", "Sound Clip","Gospel", "Noise", "AlternRock","Bass",
"Soul", "Punk", "Space", "Meditative", "Instrumental Pop",
"Instrumental Rock", "Ethnic", "Gothic", "Darkwave", "Techno-
Industrial", "Electronic","Pop-Folk", "Eurodance", "Dream",
"Southern Rock", "Comedy", "Cult", "Gangsta", "Top 40", "Christian
Rap", "Pop/Funk", "Jungle", "Native American", "Cabaret", "New
Wave", "Psychadelic", "Rave", "Showtunes", "Trailer", "Lo-Fi",
"Tribal", "Acid Punk", "Acid Jazz", "Polka", "Retro", "Musical",
"Rock & Roll", "Hard Rock", "Folk", "Folk-Rock", "National Folk",
"Swing", "Fast Fusion", "Bebop", "Latin", "Revival", "Celtic",
"Bluegrass", "Avantgarde", "Gothic Rock", "Progressive Rock",
"Psychedelic Rock", "Symphonic Rock", "Slow Rock", "Big Band",
"Chorus", "Easy Listening", "Acoustic", "Humour", "Speech",
"Chanson", "Opera", "Chamber Music", "Sonata", "Symphony", "Booty
Bass", "Primus", "Porn Groove", "Satire", "Slow Jam", "Club",
"Tango", "Samba", "Folklore", "Ballad", "Power Ballad", "Rhythmic
Soul", "Freestyle", "Duet", "Punk Rock", "Drum Solo", "Acapella",
"Euro-House", "Dance Hall"};
```

When you typecast ' to an integer you get 39 which translates to genre Noise.

Given this, your challenge is to write a program which does the following:
1. Prompt the user for a filename which ends with the .mp3 extension. If the file does not end with the .mp3 file extension the program should terminate.

2. Pull out the ID3v1.1 information using the specification above – make sure the tag is legitimate by looking for the TAG header. If it does not exist, print an error and terminate. The information read should be stored in a buffer then printed out.

An example run of the program would yield:

```
Enter filename: sample.mp3
File is OK and has valid ID3v1.1 tag
Title: Perfect Timing (This Morning)
Artist: Orba Square
Album: Sunshynesss
Year: 2006
Genre: Noise
Comment: www.spinner.com
Track Number: 90
```

3. Once you have displayed the information as above – ask the user if they wish to change the comment field. If they do, get the new comment and store it and then write it (**only the comment**) to the mp3 file.

```
Do you wish to update the comment field Y/N? Y
Comment: I've heard better.
```

4. Once complete re-print the updated ID3 tag to the display

```
Title: Perfect Timing (This Morning)
Artist: Orba Square
Album: Sunshynesss
Year: 2006
Genre: Noise
Comment: I've heard better.
Track Number: 90
```

In order to do this, you will need to use binary I/O along with various seeking routines. Place your code into the file `mp3.cpp`.

Submission Procedure:

You must be in attendance of the lab to submit the task. Non attendance at the lab will result in no mark being awarded. Tutors will be in the lab to help you with the task. Submission procedure is as follows.

`submit –u <username> –c CSCI124 –a lab6 mp3.cpp`