Due 4 Feb 2016

Compose your own MIDI file as text-based hex bytes and then compile this file into a binary Standard MIDI File (SMF) with the *binasc* program (program details below). Your MIDI file should have at least 20 MIDI events excluding overtly copy/pasted content (note-ons and note-offs count as separate events, of course). Incorporate at least 4 of the following features into your MIDI file:

(1) Notes in more than one channel with a different instrument on each channel.
(2) A drum track (General MIDI channel 10 (or channel 09 for programmers).
(3) Multiple tracks (Type-1 MIDI file).
(4) One or more pitch-bend messages to adjust a pitch after it starts.
(5) Use Pitch-bend messages to compose a melody containing some quarter-tones (pitch-bend range is +/- 200 cents (2 half-steps), so set pitch bend to +/- 25% to get a quarter tone.
(6) Any meta message (tempo, key signature, time signature, track name).
(7) A VLV greater than 128 ticks.
(8) If your MIDI file is notation-based (quantized to a set of printable note values as opposed to more randomly distributed performance-based timings), load the file into MuseScore/Finale/Sibelius and print the result.
(9) Include a chord (set of simultaneous notes).

E-mail your ASCII data and compiled Standard MIDI Files before class on Feb 4th so we can play them in class.

The *binasc* program should be installed on all of the lab computers for use in a unix terminal. You can also download executable versions for linux, OS X or Windows from the webpage:
https://code.google.com/p/binasc/downloads/list
(or download the source code from https://github.com/craigsapp/binasc and compile yourself). For OS X, for example, save the file named "binasc.osx" to your desktop. The geeky way to download in OS X is to open the App called /Applications/Utilities/Terminal.app, then go to the desktop and download the program by typing in these commands:

```
cd ~/Desktop
curl https://binasc.googlecode.com/files/binasc.osx –o binasc
chmod 0755 binasc
```

Optionally to allow running binasc from any directory on your computer, copy into the directory /usr/bin:

```
sudo cp binasc /usr/bin
```

You can look at sample MIDI text files found in the examples wikipage for binasc as a template for ideas:
https://github.com/craigsapp/binasc/blob/wiki/examples.md

Documentation:
https://github.com/craigsapp/binasc/blob/wiki/mainpage.md

As an example usage for the binasc program, here is the MIDI file we parsed in class as ASCII characters:

```
4d 54 68 64 00 00 00 06 00 00 00 01 00 80 4d 54 72 6b 00 00 00 8c 00 ff 58
04 04 02 30 08 00 ff 59 02 00 00 00 90 3c 28 81 00 90 3c 00 00 90 3c 1e 81
00 90 3c 00 00 90 43 2d 81 00 90 43 00 00 90 43 32 81 00 90 43 00 00 90 45
2d 81 00 90 45 00 00 90 45 32 81 00 90 45 00 00 90 43 23 82 00 90 43 00 00
90 41 32 81 00 90 41 00 00 90 41 2d 81 00 90 41 00 00 90 40 32 40 90 40 00
40 90 40 28 40 90 40 00 40 90 3e 2d 40 90 3e 00 40 90 3e 32 40 90 3e 00 40
90 3c 1e 82 00 90 3c 00 00 ff 2f 00
```

If this content is stored in a file called "twinkle.txt" then the binasc program can compile it into a Standard MIDI File with the command-line call:

```
binasc twinkle.txt –c twinkle.mid
```

If you downloaded the binasc program to your desktop, and are located in the terminal within the Desktop directory, you will most likely have to add ./ in front of the command name so the terminal shell knows where to find the command:

```
./binasc twinkle.txt –c twinkle.mid
```

The binasc program can also convert decimal values into hex bytes in binary files. If a number is prefixed by a single quote then the number is a decimal value. E.g., FF = '255. The size in bytes of a decimal integer can be placed before the single-quote (such as 2'128 for a two-byte integer which will be converted to the hex bytes "00 80"). Also, an ASCII character can be indicated by placing a plus sign in front of it (such as +A converts to the hex byte "41"). Comments start with a semi-colon and continue to the end of the line. Below is a slightly higher-level version of the Twinkle MIDI file that can also be compiled to an identical binary file as the hex bytes above by binasc:

```
+M +T +h +d          ; The MIDI track header chunk ID
4'6                  ; Header size (6) stored in 4-bytes (big endian).
2'0                  ; format (type 0) stored in two bytes
2'1                  ; track count (1) stored in two bytes
2'128                ; ticks per quarter note (128)

+M +T +r +k          ; The MIDI track chunk ID
4'140                ; # of bytes to follow in track (you have to count yourself)

; MIDI events (delta time (first column),
; MIDI/meta message following on one or more indented lines):

0                    ; a delta time (starts immediately)
   ff 58 04          ; Time signature meta event 4 more bytes expected
   04 02             ; the key signature 4/4  (4 / 2^2)
   '48               ; 48 clock ticks per quarter note
   8                 ; 8 32nd notes per quarter note
0
   ff 59 02          ; key signature meta event
   0                 ; no sharps or flats
   0                 ; major key (therefore C major)

; play some notes:

0                    ;
   90 '60 '40        ; C4 note-on (quarter note)
81 0                 ; delta time (128 ticks) given as a VLV
   90 '60 0          ; C4 note-off
0                    ;
```

```
   90 '60 '30       ; C4 note-on
81 0                ;
   90 '60 0         ; C4 note-off
0
   90 '67 '45       ; G4 note-on
81 0                ;
   90 '67 0         ; G4 note-off

0
   90 '67 '50       ; G4 note-on
81 0                ;
   90 '67 0         ; G4 note-off
0
   90 '69 '45       ; A4 note-on
81 0                ;
   90 '69 0         ; A4 note-off
0
   90 '69 '50       ; A4 note-on
81 0                ;
   90 '69 0         ; A4 note-off
0
   90 '67 '35       ; G4 note-on
82 0                ;
   90 '67 0         ; G4 note-off
0
   90 '65 '50       ; F4 note-on
81 0                ;
   90 '65 0         ; F4 note-off
0
   90 '65 '45       ; F4 note-on
81 0                ;
   90 '65 0         ; F4 note-off
0
   90 '64 '50       ; E4 note-on
40
   90 '64 0         ; E4 note-off
40
   90 '64 '40       ; E4 note-on
40
   90 '64 0         ; E4 note-off
40
   90 '62 '45       ; D4 note-on
40
   90 '62 0         ; D4 note-off
40
   90 '62 '50       ; D4 note-on
40
   90 '62 0         ; D4 note-off
40
   90 '60 '30       ; C4 note-on
82 0                ;
   90 '60 0         ; C4 note-off
0
   ff 2f 00         ; end-of-track meta event (NOT optional)
```

More binasc compiling tips given below:

Automatic VLV creation:
https://github.com/craigsapp/binasc/blob/wiki/mainpage.md#36-variable-length-values

A handy feature of the binasc program is that if you prefix a number with the letter ‘v’, it will interpret that number as a decimal value which needs to be converted in a set of VLV bytes. For example, the string ‘v128’ will be converted into the hex bytes ‘81 00’ when compiling an ASCII file into a binary file.

Automatic pitch-bend data bytes creation:
https://github.com/craigsapp/binasc/blob/wiki/mainpage.md#37-midi-pitch-bend-data-bytes

The MIDI command byte for pitch bend is hex E0. This command requires two data bytes to follow it. Encoded in these two bytes is a 14-bit value from 0 to 2^14-1. This 14-bit number is split into two 7-bit pieces (top bit is set to 0 since it is a MIDI data byte). The smallest portion of the number is the first byte after the E0 command byte, and the largest portion is second (big-endian ordering instead of the little-endian method for most everything else in MIDI files). The binasc program maps a floating point value in the range from -1.0 to +1.0 if it is prefixed by the letter 'p' to this 14-bit number, and then packs it into the two MIDI data bytes. The pitch-bend range is programmable on MIDI synthesizers (and how to program it is device independent). The default is typically +/- 200 cents (a whole tone or two semitones). For example if the note 60 (middle C) is played with maxium pitch bend, the MIDI message would be "E0 p1.0" which will be translated into the bytes "E0 7f 7F". This will typically sound as a D pitch. The lowest bend would be "E0 p-1.0" which becomes "E0 00 00". This typically sounds as a B-flat. To turn off the pitch bend (for the current note and any following notes, the command would be "E0 p0" which translates into "E0 00 40".

Automatic Meta-message tempo creation:
https://github.com/craigsapp/binasc/blob/wiki/mainpage.md#38-midi-tempo-meta-message-data-bytes

Tempo control in MIDI files is done with the Meta message hex 51. The tempo is given as the duration of a beat in microseconds. Here are three equivalent representations of a tempo meta message when binasc compiles a file:

FF 51 03 07 A1 20
FF 51 03 3'500000
FF 51 03 t120

The tempo 120 beats-per-minute translates into 0.5 seconds a beat, or 500000 microseconds. This in turn is converted into a three-byte hex integer as data in the tempo meta message. Prefixing a decimal number with "t" will cause the three-hex byte value microsecond value to be compiled into the resulting MIDI file.

Chunk byte counts

The binasc program does not handle counting chunk byte sizes. The header byte size is always 6, but individual track chunk sizes have to be counted by yourself… The easiest counting method is to compile a file into an SMF with a dummy chunk bytesize field (4 bytes), then convert back to ASCII bytes with binasc. Then edit the text file and save each track chunk hex values into a separate file and use the wc command to count the number of "words" in the file:

```
wc –w chunkbytes.txt
```

The resulting number should replace the dummy value in the original text file. If you use MS Word or similar program, there should be a word-count feature that can be used in a similar manner.