Date: Tue, 13 Aug 91 09:18:07 WST
From: Peter N Lewis <peter@cujo.curtin.edu.au>
Subject: info-mac/tech/binhex-definition.txt

Hi All,

This is a definition of the BinHex 4.0 standard as I see it. When I first tried to write DeHQX, I had to post several questions to the net to get a full definition of this standard. Hopefully this file will make it easier for anyone who wants to add BinHex compatability to there application.

Have fun,
Peter <Lewis_P@cc.curtin.edu.au>

---

BinHex 4.0 Definition by Peter N Lewis, Aug 1991.

For a long time BinHex 4.0 has been the standard for ASCII encoding of Macintosh files. To my knowledge, there has never been a full definition of this format. Info-Mac had an informal definition of the format, but this lacked a description of the CRC calculation, as well as being vague in some areas. Hopefully this document will fully define the BinHex 4.0 standard, and allow more programmers to fully implement it. Note, however, that this definition is how I see the BinHex standard, and since I had no part whatsoever in defining it initially, this document can have no real claim to being the one true definition. If anyone feels that I have not got the facts straight, or that it is ambiguous in any details, please contact me at the address at the bottom of this document.

Format:

It is necessary to distinguish between the encoding format and decoding format since we wish to allow all decoders to read all versions of the BinHex format, while trying to reduce the variation in encoding.

All numbers are decimal unless they have the format 0xFF, in which case they are hex.

<tab> is a tab, character value 9.
<lf>  is a linefeed, character value 10.
<cr> is a carriage return, character value 13.
<spc> is a space, character value 32.
<return> means to the encoder:
The sequence <cr> <lf>. Either (but not both) may be omitted. Use whatever is appropriate for your system (<cr> for Mac, <lf> for Unix, <cr><lf> for MS-DOS).
<return> means to the decoder:
Any sequence of zero or more of <cr>, <lf>, <tab>, <spc>. (The <tab> and <spc> are required because some old programs produced these characters).
For example, <cr> <tab> <lf> <spc> <lf> <cr> would be perfectly acceptable.

A hqx file begins with a description which should be ignored by the decoder (and generally left blank by encoding software). The hqx file proper consists of the sequence:

<start-of-line>(This<spc>file<spc>must<spc>be<spc>converted<spc>with<spc>
BinHex<spc>4.0)<return>:<hqx-file>:<return>

When encoding, DO NOT mess with the "(This file..." string. There are a large number of automated programs that use it, and they may stumble over anything other than this exact string.

When decoding, you should only check the string up to "with BinHex", then skip until either a <cr> or <lf>, then skip <return> characters, and check for the colon. Also, be careful with the <start-of-line>, this can be either a <return> character, or the start of the file. Some old programs produced an extra exclamation mark (!) immediately before the final colon (:). When decoding, after all data is read, skip any <return> characters, and then allow a single optional exclamation mark (and then skip <returns> again) before checking for the terminating colon. Don't check for a <return> after the colon.

<hqx-file> is a sequence of 6-bit encoded characters.

When encoding, a <return> should be inserted after every 64 characters. The first character should follow immediately after the first colon (without a <return>), and the first line should be 64 characters long, (unless its also the last line obviously) including the colon. The final colon should go on the same line as the last character and there should be a return after it. Thus, the final line must be between 2 and 65 (inclusive) characters long.

When decoding, lines of any length should be accepted, and <return> characters should be ignored everywhere (before and after the first colon, between any two hqx characters, and before the trailing colon.

This string defines the valid characters, in order:
!"#$%&'()*+,-012345689@ABCDEFGHIJKLMNPQRSTUVXYZ[`abcdefhijklmpqr
(ie, ! is 0, " is 1, ..., r is 63).

When decoding, any character other that those 64 and the <return> characters indicates an error and the user should be notified of such.

The format also supports run length encoding, where the character to be repeated is followed by a 0x90 byte then the repeat count. For example, FF9004 means repeat 0xFF 4 times. The special case of a repeat count of zero means it's not a run, but a literal 0x90. 2B9000 => 2B90.

*** Note: the 9000 can be followed by a run, which means to repeat the 0x90 (not the character previous to that). That is, 2B90009005 means a 2B9090909090.

Before encoding into RLE, 6-bits (or after if decoding), the file is formatted into a header, data fork, and resource fork.  All three are terminated with a two byte CRC.

The header consists of a one byte name length, then the file name, then a one byte 0x00.  The rest of the header is 20 bytes long, and contains the file type, creator, file flags, data and resource lengths, and the two byte CRC value for the header.

When encoding, the flags should be copied directly from the Finder Info. When decoding, the OnDesk, Invisible, and Initted flags should be cleared.

Also, when decoding, the file name should be validated for the given OS. For example, a Mac program should replace a full stop (.) at the front of a file name with a bullet (option-8), and should replace colons (:) with dashes (-) and, if running under A/UX, slashes should be replaced by dashes (-).

The data fork and resource fork contents follow in that order.  If a fork is empty, there will be no bytes of contents and the checksum will be two bytes of zero.

So the decoded data between the first and last colon (:) looks like:

```
         1       n       4    4    2    4    4   2      (length)
        +-+---------+-+----+----+----+----+----+--+
        |n| name... |0|TYPE|AUTH|FLAG|DLEN|RLEN|HC|     (contents)
        +-+---------+-+----+----+----+----+----+--+

                        DLEN                     2      (length)
        +--------------------------------------+--+
        |       DATA FORK                      |DC|     (contents)
        +--------------------------------------+--+

                        RLEN                     2      (length)
        +--------------------------------------+--+
        |       RESOURCE FORK                  |RC|     (contents)
        +--------------------------------------+--+
```

CRCs:

BinHex 4.0 uses a 16-bit CRC with a 0x1021 seed.  The general algorithm is to take data 1 bit at a time and process it through the following:

1) Take the old CRC (use 0x0000 if there is no previous CRC) and shift it to the left by 1.

2) Put the new data bit in the least significant position (right bit).

3) If the bit shifted out in (1) was a 1 then xor the CRC with 0x1021.

4) Loop back to (1) until all the data has been processed.

Or in pseudo pascal:

```
var crc:integer; { init to zero at the beginning of each of the three
forks }

procedure CalcCRC (v: integer);  { 0<=v<=255 }
        var
                temp: boolean;
                i: integer;
begin
        for i := 1 to 8 do begin
                temp := (crc AND 0x8000) <> 0;
                crc := (crc << 1) OR (v >> 7);
                if temp then
                        crc := crc XOR 0x1021;
                v := (v << 1) AND 0xFF;
        end;
end;
```

When encoding, include two bytes of 0x00 where the CRC should be, and use them in the calculation of the CRC before writing it to the file. Similarly, when decoding, calculate the CRC over all the bytes in the fork (header, data, or resource) except the last two CRC bytes, then continue the CRC calculation with two 0x00 bytes, then compare it to the CRC stored in the file.

Parts:

If you wish to support segmented files in the same way that comp.binaries.mac files are segmented,  use the following line to note the end of a hqx part:
<return>--- end of part NN ---<return>

Note: When decoding, it is only necessary to check for "<return>--- end of part".

Recommence decoding (either later in this file, or in the next file) when you encounter the line:
<return>---<return>

Note:  In this case, when decoding, demand either a <cr> or <lf> both immediately before, and immediately after the ---.  It is unfortunate that this sequence is in fact valid hqx data, but there should not be any problems with this.

Sources to refer to:

FTP from sumex-aim.stanford.edu
    A file called hqx-format.txt which I can no longer find.
    info-mac/unix/xbin
    info-mac/unix/mcvert
    info-mac/source/pascal/dehqx

Author:

Peter N Lewis <Lewis_P@cc.curtin.edu.au>
10 Earlston way,
Booragoon 6154 WA,
AUSTRALIA

Contributors:

Roland Mansson <Roland.Mansson@LDC.lu.se>
Steve Dorner <dorner@pequod.cso.uiuc.edu>
Sak Wathanasin <sw@network-analysis-ltd.co.uk>
Dave Green <daveg@Apple.com>
Tom Coradeschi <tcora@PICA.ARMY.MIL>
Howard Haruo Fukuda <physi-hf@garnet.berkeley.edu>
Michael Fort <mfort@ub.d.umn.edu>
Dave Johnson <ddj%brown@csnet-relay.ARPA>

Note: I attempted to contact all these people, but only a few replied. Some of them may not wish to be associated with this document, and certainly they should not be seen as endorsing it in any way.  Obviously, any errors in this document are my own!