ASCII-Hex, ASCII-85 and reverse ASCII-85 encoding

Dipl.-Ing. D. Krause

February 17, 2009

Contents

1 Overview

Binary data is encoded as ASCII text for multiple reasons, i. e.:

- A human readable form is needed to compare binary data against other binary data.
  A typical example is file integrity verification where a message digest of a downloaded archive is compared against the message digest published by the original author of the archive.
- Some networking protocols can handle 7-bit ASCII text only.
  An example are old printers connected to terminals.

For binary-to-ASCII encoding we require:

- Encoding and decoding must be unique transformations (this means after encoding and decoding we want to have exactly the original data).
- The encoded text must contain characters in the range $0x21 \leq c \leq 0x7F$ only (7-bit characters without Ctrl-A... Ctrl-Z).
- The encoded text must not contain whitespaces, it must be exactly one text string.

2 ASCII-Hex encoding

2.1 Encoding

Each byte is splitted into 2 half-bytes (sometimes referred to as “nibbles”) consisting of 4 bits each. Each half-byte is represented by the hexadecimal character corresponding to the numeric value.
The byte 0x1F for example is encoded as the text “1F” or “1f”. The most significant half-byte is printed first.

2.2 Decoding

The text is splitted into pairs of two hexadecimal characters, each pair represents one byte of the binary data. The numeric values corresponding to the hexademial characters are merged to build the byte. In the example the text “1F” is splitted into the hexadecimal characters “1” and “F”, the corresponding numeric values are 0x01 and 0x0F. Merging the half-bytes results in 0x1F.

3 ASCII-85 encoding

3.1 Encoding

The input is grouped into DWORDs (group of 4 bytes a, b, c and d). The first byte is used as the *most* significant byte, the fourth byte is the least significant byte in the DWORD.

$$z = a \cdot 256^3 + b \cdot 256^2 + c \cdot 256^1 + d \cdot 256^0$$

The base-85 representation of z is calculated

$$z = e \cdot 85^4 + f \cdot 85^3 + g \cdot 85^2 + h \cdot 85^1 + i \cdot 85^0$$

The encoded characters for the DWORD are build as (char)$(33+e)$, (char)$(33+f)$, (char)$(33+g)$, (char)$(33+h)$ and (char)$(33+i)$.
If the last 4-byte group is incomplete the encoded data contains

- (char)$(33+e)$, (char)$(33+f)$, (char)$(33+g)$, (char)$(33+h)$ for three input bytes,
- (char)$(33+e)$, (char)$(33+f)$, (char)$(33+g)$ for two input bytes or
- (char)$(33+e)$, (char)$(33+f)$ for one input byte.

3.2 Decoding

3.2.1 Complete groups of 5 encoded characters

For a complete group of 4 binary bytes (5 encoded characters) we can retrieve e, f, g, h and i and calculate z as

$$\begin{aligned} z = z' &= e \cdot 85^4 + f \cdot 85^3 + g \cdot 85^2 + h \cdot 85^1 + i \cdot 85^0 \\ &= a \cdot 256^3 + b \cdot 256^2 + c \cdot 256^1 + d \cdot 256^0 \end{aligned}$$

The decoded bytes can be retrieved from the DWORD z, starting with the *most* significant byte.

3.2.2 Incomplete groups

If the last group is incomplete and contains

- e and f (1 binary byte / 2 encoded characters),
- e, f and g (2 binary bytes / 3 encoded characters) or
- e, f, g and h only (3 binary bytes / 4 encoded characters)

the coefficients

- g, h and i,
- h and i or
- i

are unknown.
Instead of

$$\begin{aligned} z &= a \cdot 256^3 + b \cdot 256^2 + c \cdot 256^1 + d \cdot 256^0 \\ &= e \cdot 85^4 + f \cdot 85^3 + g \cdot 85^2 + h \cdot 85^1 + i \cdot 85^0 \end{aligned}$$

we can only calculate

$$\begin{aligned} z' &= a' \cdot 256^3 + b' \cdot 256^2 + c' \cdot 256^1 + d' \cdot 256^0 \\ &= \begin{cases} e \cdot 85^4 + f \cdot 85^3 + g \cdot 85^2 + h \cdot 85^1 & \text{for 3 binary bytes} \\ e \cdot 85^4 + f \cdot 85^3 + g \cdot 85^2 & \text{for 2 binary bytes} \\ e \cdot 85^4 + f \cdot 85^3 & \text{for 1 binary byte} \end{cases} \end{aligned}$$

If the skipped coefficients are not 0 we have

$$z' < z$$

For a group of 3 binary bytes only i is omitted, we have

$$z - z' \leq 84 < 255$$

After calculating a', b', c' and d' we can find z:

$$\begin{aligned} z &= a \cdot 256^3 + b \cdot 256^2 + c \cdot 256^1 \\ &= \begin{cases} a' \cdot 256^3 + b' \cdot 256^2 + c' \cdot 256^1 & \text{if } d' = 0 \\ a' \cdot 256^3 + b' \cdot 256^2 + c' \cdot 256^1 + 256 & \text{otherwise} \end{cases} \end{aligned}$$

For a group of 2 binary bytes h and i are skipped, we have

$$z - z' \leq 84 \cdot 85 + 84 < 255 \cdot 256 + 255$$

After calculating a', b', c' and d' we can find z:

$$\begin{aligned} z &= a \cdot 256^3 + b \cdot 256^2 \\ &= \begin{cases} a' \cdot 256^3 + b' \cdot 256^2 & \text{if } c' = 0 \wedge d' = 0 \\ a' \cdot 256^3 + b' \cdot 256^2 + 256^2 & \text{otherwise} \end{cases} \end{aligned}$$

For one single binary byte g, h and i are omitted, we have

$$z - z' \leq 84 \cdot 85^2 + 84 \cdot 85 + 84 < 255 \cdot 256^2 + 255 \cdot 256 + 255$$

After calculating a', b', c' and d' we can find z:

$$\begin{aligned} z &= a \cdot 256^3 \\ &= \begin{cases} a' \cdot 256^3 & \text{if } b' = 0 \wedge c' = 0 \wedge d' = 0 \\ a' \cdot 256^3 + 256^3 & \text{otherwise} \end{cases} \end{aligned}$$

4 Reverse ASCII-85 encoding

4.1 Encoding

The input is grouped into DWORDs (group of 4 bytes d, c, b and a). The first input byte is used as the *least* significant byte, the fourth byte is the most significant byte in the DWORD.

$$z = a \cdot 256^3 + b \cdot 256^2 + c \cdot 256^1 + d \cdot 256^0$$

The base-85 representation of z is calculated

$$z = e \cdot 85^4 + f \cdot 85^3 + g \cdot 85^2 + h \cdot 85^1 + i \cdot 85^0$$

The encoded characters for the DWORD are build as (char)$(33+i)$, (char)$(33+h)$, (char)$(33+g)$, (char)$(33+f)$ and (char)$(33+e)$.
If the last 4-byte group is incomplete the encoded data contains

- (char)$(33+i)$, (char)$(33+h)$, (char)$(33+g)$, (char)$(33+f)$ for three input bytes,
- (char)$(33+i)$, (char)$(33+h)$, (char)$(33+g)$ for two input bytes or
- (char)$(33+i)$, (char)$(33+h)$ for one input byte.

4.2 Decoding

4.2.1 Complete groups of 5 encoded characters

For a complete group of 4 binary bytes (5 encoded characters) we can retrieve e, f, g, h and i and calculate z as

$$\begin{aligned} z = z' &= e \cdot 85^4 + f \cdot 85^3 + g \cdot 85^2 + h \cdot 85^1 + i \cdot 85^0 \\ &= a \cdot 256^3 + b \cdot 256^2 + c \cdot 256^1 + d \cdot 256^0 \end{aligned}$$

The decoded bytes can be retrieved from the DWORD z, starting with the *least* significant byte.

4.2.2 Incomplete groups

If the last group is incomplete and contains

- h and i (1 binary byte / 2 encoded characters)
- g, h and i (2 binary bytes / 3 encoded characters) or
- f, g, h and i (3 binary bytes / 4 encoded characters)

we know the skipped coefficients are zero

- $e = f = g = 0$ (1 binary byte)
- $e = f = 0$ (2 binary bytes)
- $e = 0$ (3 binary bytes)

If we calculate

$$z' = \begin{cases} h \cdot 85^1 + i \cdot 85^0 & \text{(1 binary byte)} \\ g \cdot 85^2 + h \cdot 85^1 + i \cdot 85^0 & \text{(2 binary bytes)} \\ f \cdot 85^3 + g \cdot 85^2 + h \cdot 85^1 + i \cdot 85^0 & \text{(3 binary bytes)} \end{cases}$$

we know

$$z = z'$$

and

$d = d'$	1, 2 or 3 binary bytes
$c = c'$	2 or 3 binary bytes
$b = b'$	3 binary bytes

We do not have to take care of different cases.

5 Summary

The ASCII-Hex encoding is an easy-to-understand encoding. The relation between the number of original bytes l_o and the number of encoded bytes l_e is

$$l_e = 2 \cdot l_o$$

The ASCII-85 encoding is described in the PostScript and PDF file format reference. Encoding and decoding is more complicated than ASCII-Hex encoding but output is smaller.

$$l_e = 1.25 \cdot l_o$$

The reverse ASCII-85 encoding produces output of the same length as the original ASCII-85 encoding. As the skipped coefficients for an incomplete final group are known to be zero the decoding routine is a little bit simpler than the decoding routine of the ASCII-85 encoding.

$$l_e = 1.25 \cdot l_o$$