BSDIFF 4.0 FORMAT

This describes the format used by version 4.0 of the bsdiff/bspatch utilities,
published in 2003, and which was the first (and as of early 2012, only) widely
used version.

For historical reference: bsdiff/bspatch operate on files of 8-bit bytes.

In the standard usage "bspatch source target patch", the file "target" is
created following instructions in the file "patch", with reference to the
file "source".

Patch file

offset	size	data type	value
0	8	ASCII characters	"BSDIFF40"
8	8	INTEGER	size of compressed control block
16	8	INTEGER	size of compressed diff block
24	8	INTEGER	size of target file
32	?	bzip2	compressed control block
?	?	bzip2	compressed diff block
?	to EOF	bzip2	compressed extra block

The patch file is processed by:
1. Reading the header,
2. Parsing the INTEGER values (see below),
3. Locating the three bzip2 blocks using the parsed sizes,
4. Decompressing the three bzip2 blocks,
5. Processing the control block (see below).

INTEGER type

offset	size	data type	value
0	1	byte	x0
1	1	byte	x1
2	1	byte	x2
3	1	byte	x3
4	1	byte	x4
5	1	byte	x5
6	1	byte	x6
7	1	byte	x7 + 128 * s

The values x0, x1, x2, x3, x4, x5, x6 are between 0 and 255 (inclusive). The
value x7 is between 0 and 127 (inclusive). The value s is 0 or 1.

The INTEGER is parsed as:

```
(x0 + x1 * 256 + x2 * 256^2 + x3 * 256^3 + x4 * 256^4 +
    x5 * 256^5 + x6 * 256^6 + x7 * 256^7) * (-1)^s
```

(In other words, an INTEGER is a 64-bit signed integer in sign-magnitude
format, stored in little-endian byte order.)

Control block

The control block consists of triples (mixlen, copylen, seeklen) stored as:

offset	size	data type	value
0	8	INTEGER	mixlen
8	8	INTEGER	copylen
8	8	INTEGER	seeklen

The control block is processed according to the following algorithm:
1. Open the source file for reading, and set the read pointer to position 0.
2. Open the target file for writing.
3. Start at the beginning of the control, diff, and extra blocks.
4. While there is data left in the control block:
4.1. Read and parse a (mixlen, copylen, seeklen) triple from the control block.
4.2. Read mixlen bytes from the diff block and mixlen bytes from the current
position in the source file; add the bytes pairwise (modulo 256), and write

the resulting mixlen bytes to the target file.
4.3. Read copylen bytes from the extra block and write them to the target file.
4.4. Adjust the read pointer in the source file by seeklen bytes.
5. Verify that the number of bytes written to the target file matches the file
size stored in the patch file header.

For clarity, the word "read X bytes" above means "read X bytes and advance the
read pointer by X bytes"; the only time the same bytes of input can be used
more than once is via a negative seeklen value (which will result in part of
the source file being read at more than one iteration of step 4.2).