SEGD On Disk – A basis for discussion

- *Stewart A. Levin* 19 Nov, 2007

Beginning with Revision 2, SEGD has supported “Byte stream devices” for storage of SEGD data. This format is designed for disk storage and network transmission of SEGD data and consists of all the bytes of one or more shot records on the SEGD tape concatenated together after a 128 byte label. In addition, that standard promised that the standards subcommittee was working towards an, as yet to be released, mapping to the RODE encapulation structure.

Experience has shown that a byte stream disk file is a fragile format. It requires that the information in the general headers, channel sets and trace headers be correct in the specfication(s) of trace length, sample rate and bytes per sample. If, as usual, the disk file is a bit copy of a full tape, the disk file becomes unmanageably huge for modern terabyte capacity tapes. The byte stream format is also potentially awkward for data distribution onto CD and DVD media, forcing the transcription program to have to know the details and quirks of the SEGD format in order to separate the input stream at shot record boundaries and prefix them with 128 byte labels.

Encapsulation formats abound in our industry. Here is my incomplete list:

I. *Fortran sequential unformatted*. Perhaps the oldest encapsulation in widespread use, this format places a byte length in front of and in back of each tape record. (In some variants, the number of 32 bit words in each record is used.) This facilitates moving forward and backwards through the file sequentially. A file mark (EOF) is represented by a zero length record. In the early days, the byte ordering of the length fields was that of the machine on which the file was created. More recently, the big endian convention has taken hold, though there are still exceptions and it is best to check the length of the first record dynamically at run time to see which ordering has been used. Tape I/O errors in the transcription are sometimes noted by setting the length fields to a negative value.

II. *VBS*. (Variable Blocked Spanned) Another encapsulation format dating from the IBM mainframe heyday, this was originally designed for storing variable length records using fixed length blocks. A length field and flag bits are placed before each record or piece of a record telling whether that segment is the beginning, end, both, or neither of the full record.

III. *Lacey*. An 8 byte prefix appears before each record. The first 4 bytes are a record number, starting from 1, and the second 4 bytes are the number of bytes in the record. Again a file mark is captured as a zero length record and the byte ordering of the prefix is not defined and is often determined dynamically by inspecting the first 4 bytes of the dataset to see if it reads in as a “1” or as a large integer taking the value +/– 2,147,483,648.

IV. *Byte stream with index*. This class strengthens the otherwise fragile byte stream disk file with a separate index file telling where each record is in the file. This index file tells, at minimum, the byte location within the disk file at which each record begins and its length. Generally, the index will also provide a record counter and some seismic index, e.g. Field Record number or CDP, to assist software sorting and random access to the byte stream disk file.

V. *Header and data file pairs*. In this approach, each field file is split into one disk file containing its header, i.e. the set of SEGD general headers, channel sets, extended and external headers, and a second disk file consisting of the traces with their individual trace headers. Some preagreed naming convention is used to help identify which header file goes with which trace file. One common one is to name the disk files F###.hb and F###.sb where ### is the field record number of that shot. This convention automatically leads to an alphabetical directory listing in which the field records appear in increasing order with the header file preceding the data file name.

VI. *Tape Image Format (TIF)*. This encapsulation format is more common for well logs rather than seismic traces and, indeed, was developed by Dresser Atlas for disk storage of well logs. Each record is preceded by three 4 byte little endian integers, the first being a 0 (normal record) or a 1 (file mark), the second the byte offset of the start of the previous record and the third the byte offset of the start of the next record. Because it uses absolute offsets into the file, it is limited to datasets with a maximum size of 4GB. To extend TIF to larger files, a TIF8 has been proposed which uses a 24 byte prefix with the ASCII characters “TIF8” followed by the 4 byte record type and the two file offsets now each 8 bytes long instead of 4. Again, little endian byte ordering is used for the record type and file offsets.

VII. *RODE*. Like TIF, this too evolved from the well log world as an outgrowth, denoted RP66, of the Schlumberger DLIS format. RP66, documented at http://www.posc.org/technical/data_exchange/RP66/V2/rp66v2.html, is a very general, *aka* complex, mechanism for capturing data and metadata into a byte stream as opposed to a specific encapsulation recipe for field data. RODE is the SEG’s recommendation on how to use RP66 with seismic data, with SEG-Y as the single example given. (And that example actually violates RP66!) As noted in the introduction, the SEG Technical Standards Committee has committed to provide a corresponding mapping for SEG-D data.

What are desirable features of a disk encapsulation format? Again, here’s my list:

A. Support for any input record length and output file size. For me this means record lengths up to 2^{32} bytes and file sizes up to 2^{64} bytes, though others might well be happy with something smaller, say a small multiple of the capacity of a DVD.

B. Support for tape-like operations such as rewind, skip forward one or more records, skip backwards one or more records, move to a file mark, and move to end of data.

C. Clear specification of byte ordering within the encapsulation coding. (The data bytes, of course, remain in their original order.)

D. A format easily recognized by inspection of the first few bytes of an encapsulated file.

E. The ability to flag records that did not read cleanly in transcription from tape to the encapsulated output.

F. Sample open source software for reading and writing the format and/or a simple to understand format that can be easily coded into software.

G. The ability to concatenate two or more encapsulated files and have the resulting file be itself a properly encapsulated file.

So how do the formats measure up?

	A	B	C	D	E	F	G[1]
Byte stream	+	−	*n/a*	−	−	+	+
Fortran seq	+	+	−[2]	+	−[3]	+	+
VBS	+	−[4]	+	+	−	+	+
Lacey	+	−[4]	−[2]	+	−[3]	+	−
Byte str w/index	+	+	−	−	−[3]	+	−
Header/data file pairs	+	−	*n/a*	−[5]	−	+	+[6]
TIF	−	+	+	+	−[3]	+	−[7]
TIF8	+	+	+	+	−[3]	+	−[7]
RODE	+	+	+	+[8]	+	−[9]	−[8]

Notes:

1. The SEG-D standards requirement that the data in the file begin with a SEG-D tape label makes all concatenation output nonstandard due to the embedded 128 byte labels.
2. As noted earlier, the initial bytes of the file in this format will allow runtime determination of the actual endianness in which it was written.
3. A convention such as using a negative length to flag transcription errors could be used if parties agree to it.
4. Partial support, specifically forward space operations and rewinding, is allowed.
5. File name patterns from a directory listing can be helpful, though tricky to automate in software.
6. Concatenation here means adding new files into the directory. This is safe so long as there aren't duplicate file names.
7. Because absolute offsets are used in the TIF formats, concatentation will wrongly reset offsets to zero within the file. This can be detected and programmed around with a bit of care.
8. Every RODE file must begin with a 128 byte label. The SEG-D tape label also uses this format, but doesn't replace it. This interferes with concatenation comparable to note 1.

9. There is some public domain RP66 software available, but it isn't fully complete.

It is clear there is no one format that hits the mark on all counts. Which one(s) should we recommend with SEG-D? Can we devise a better one? And are there other important criteria not in the above list?