Seismic Unix: Import from and export to SEG-Y format

1. Overview

This document explains how to import SEG-Y seismic data files to Seismic Unix (SU) format and how to export SU files to SEG-Y format. Section 1 is a quick look at import and export. Sections 2 (import) and 3 (export) are detailed discussions of the same topics.

Internally, many companies have their own proprietary (secret) format. The SEG-Y format (Society of Exploration Geophysicists "Y" format) is called an "exchange" format because SEG-Y is the standard format to send seismic data from one company to another. See the References section for SEG's official documents of the SEG-Y formats.

The Reference section has more than one SEG-Y format because while SEG-Y was created in 1975, it was revised for the first time in 2002, and then a second time in 2017. Because of this, the 2017 version is called "revision 2" (or "rev2"), the 2002 version is called "revision 1" (or "rev1"), and the original 1975 version is called "revision 0" (or "rev0").

A SEG-Y rev2 file can be much more complex than a rev1 or rev0 file. However, a rev2 file does not have to have all the optional parts. If someone tells you the file you received is rev1 or rev2, try to import it. If you find you are not able to import the file, ask the person who sent you the file to send you the same dataset in rev0 format.

1.1 Import a SEG-Y file to SU format with segyread & segyclean

Use program *segyread* to convert a SEG-Y file to SU format. When I import a SEG-Y file, I **almost always** also use *segyclean*. Program *segyclean* sets the following keys (trace headers) to zero:

```
d1, f1, d2, f2, ungpow, unscale, ntr, mark
```

Junk numbers in these keys can interfere with making plots. For example, my plots can have strange spacing, trace annotation can start with a wrong number, or SU will not make the plot.

> Two reference web sites for SU trace header keys are:
>
> Seismic Unix Wiki: https://wiki.seismic-unix.org/sudoc:su_data_format
>
> SeismicRocks.com: http://seismicrocks.com/segyformat.html

An example of a simple SEG-Y import to SU format is:

```
$ segyread tape=ts103828a.sgy | segyclean > ts103828a.su
```

If your only goal is to plot (view) your data, the command above is probably all you need.

An example of a complex SEG-Y import is:

```
$ segyread tape=ts103828a.sgy hfile=ts103828a.txt
bfile=ts103828a.bin byte=189l,105f,205f,181l
remap=cdpt,swdep,gwdep,sdel | segyclean > ts103828a.su
```

This complex import command is explained in Section 2. In the above example, I used parameters `byte` and `remap` to "rescue" (my term) trace header byte information from becoming lost during import. I often have to import a SEG-Y file more than once. The first time is to read the imported textual header file (see Figure 2 in Section 2.2) and to review trace headers with *surange*. If necessary, I will import the file a second time (and maybe a third time) to remap trace header information. (Keep trying until it works!)

1.2 Export an SU file to SEG-Y format

Use program *segywrite* to convert an SU file to SEG-Y format. In addition to the seismic data, *segywrite* requires a textual header file (parameter `hfile`) and a binary header file (parameter `bfile`). If the SU file was originally a SEG-Y file, I probably have the two header files from the import process. But, if the seismic file was created in SU, I do not have header files. (An SU dataset only has trace header and trace data pairs.) In that case, I use *segyhdrs* to create the two header files. Example commands are:

```
$ suplane > plane.su

$ segyhdrs < plane.su bfile=plane.bin hfile=plane.txt

$ segywrite < plane.su tape=plane.sgy hfile=plane.txt
bfile=plane.bin
```

The first command uses an SU program to create a seismic file. The second command uses *segyhdrs* to create the two header files to be used in *segywrite*. The header files created using *segyhdrs* have the bare minimum of useful information, just enough to satisfy *segywrite*.

> In both *segyread* and *segywrite*, for parameter `tape`, supply the name of the SEG-Y file.

If I need to set the "endianness" of the output file, I can use the *segywrite* parameter `endian`:

```
endian =(autodetected)
       =1 for big-endian byte order
       =0 for little-endian byte order
```

2. Import a SEG-Y file to SU format

2.1 Quick import

Program *segyread* converts a SEG-Y seismic data file to Seismic Unix (SU) format. When I import a SEG-Y file, I **almost always** also use *segyclean*. Program *segyclean* sets the following keys (trace headers) to zero:

`d1, f1, d2, f2, ungpow, unscale, ntr, mark`

Some of these keys are used by the main plotting programs. If there are junk numbers in these keys, my plots can have strange spacing (`d2`) or trace annotation can start with a wrong number (`f2`), or SU will not allocate space to run the program if `ntr` is too large.

A very simple import of a SEG-Y file is:

```
$ segyread tape=ei.sgy | segyclean > ei.su
```

Parameter `tape` is for the name of the input SEG-Y file.

> If I forgot to use *segyclean* when I used *segyread*, I either re-import the SEG-Y file or I use *segyclean* on the new .su file. For example:
>
> ```
> $ segyclean < ei.su > ei.clean.su
> ```

Successful use of *surange* is a good test of whether the imported file is usable in SU:

```
$ surange < ei.su
```

There is more to know about the SEG-Y file format and more to know about *segyread*. That necessary explanation continues below.

2.2 SEG-Y textual header and binary header

After importing the SEG-Y file, I find that my directory has the SU file ei.su. I also see file binary (400 bytes) and file header (3200 bytes).

```
-rw-r--r-- 1 df df 2413200 Aug 10 16:59 ei.sgy
-rw-rw-r-- 1 df df     400 Aug 10 16:59 binary
-rw-rw-r-- 1 df df    3200 Aug 10 16:59 header
-rw-rw-r-- 1 df df 2409600 Aug 10 16:59 ei.su
```

A simple view of a SEG-Y file (Figure 1, below) shows it starts with a 3200-byte textual (human-readable) block. It then has a 400-byte binary (not human-readable) block. Next, the file has trace header and trace data pairs that extend to the end of the dataset. Each trace header has 240 bytes. The trace data (the seismic signal) does not have a fixed length; the seismic data blocks have as many bytes as were recorded, but (almost always) all trace data blocks in the file have the same number of bytes; in other words, the same time length.

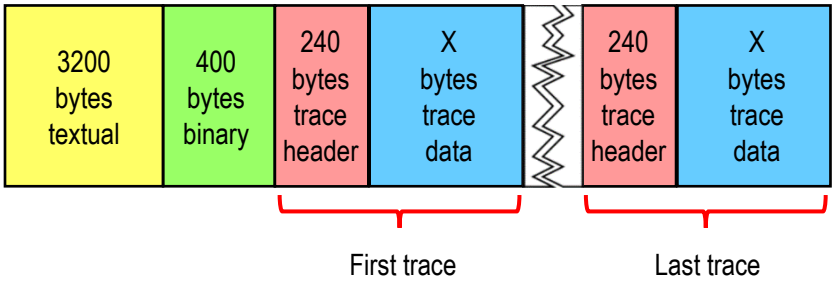

Figure 1. A SEG-Y file, 1975 (rev0) format.

Figure 1 in the SEG-Y revision 2 document (see the References) shows that a SEG-Y file can be more complex than this; however, Seismic Unix was only written to handle the original 1975 SEG-Y standard (revision 0). Since those days, there has been a revision 1 (created 2002) and a revision 2 (created 2017).

> I am unsure how to import SEG-Y rev1 data and I do not think SU recognizes SEG-Y rev2 format.

To summarize, SU assumes a SEG-Y file has

- A textual header block (3200 bytes),
- a binary header block (400 bytes), and
- pairs of trace header (240 bytes) and trace data (unknown length, but almost always the same length).

Program *segyread*

- copies the textual header block to create the header file,
- copies the binary header block to create the binary file, and
- copies the rest of the SEG-Y file into the output .su file.

Please see Figure 2, below.

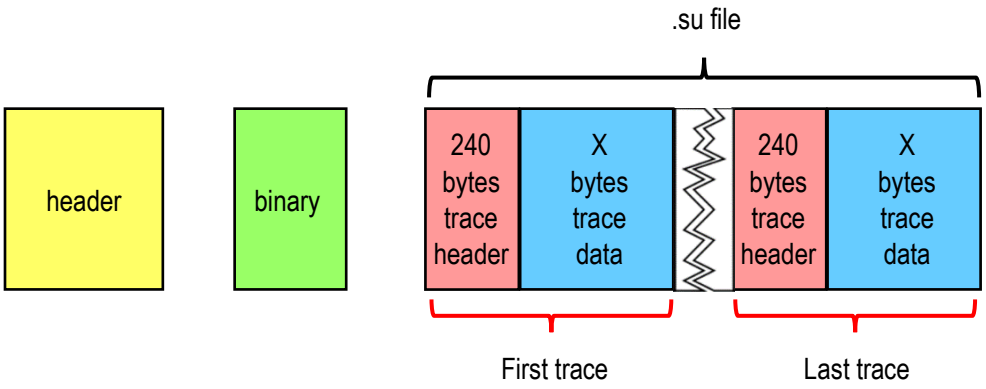

Figure 2. After *segyread*, the header file, the binary file, and the .su file.

The 3200-byte textual header (human-readable) has 40 rows, 80 characters per row (see the Appendix for a commercial example).

2.3 Save the SEG-Y textual header and binary header

Pretend at this point I have file header and file binary because I imported a SEG-Y file. If I do nothing with these two files, the next time I import a SEG-Y file *segyread* will overwrite these two files. This is bad because *segywrite* (convert SU format file to SEG-Y format file) requires a textual header file and a binary file. To prepare to export my SU data file, I have three options regarding these two header files:

1. Rename file header and file binary so these two files will not be overwritten with the next SEG-Y import.
2. During SEG-Y import, use parameters in *segyread* to assign useful names (instead of the default generic names) to header and binary.
3. Use *segyhdrs* to create a header file and a binary file from the SU file I plan to export.

Option 1 is quick, easy, and smart. For example:

```
$ mv header ei.txt
$ mv binary ei.bin
```

"txt" stands for textual header block. Remember that Linux does not assign meaning to file suffixes. ".txt" and ".bin" are convenient reminders for me, the human SU user.

Option 2 is the smartest way, if I can remember at the time I import a SEG-Y file. Program *segyread* has parameters `hfile` and `bfile` for assigning names to the two header blocks:

```
$ segyread tape=ei.sgy hfile=ei.txt bfile=ei.bin | segyclean > ei.su
```

Option 3 is the least desirable but it is a valuable tool, especially if I want to export a dataset that I created in SU. Program *segyhdrs* is discussed below in Section 3.

2.4 SU trace header and trace data programs

The trace header/trace data pairs are binary; that is, not human-readable. That is why SU has programs to let us see trace header information:

surange, *sucountkey*, *sukeycount*, *suascii*, *sudumptrace*, *sugethw*

And, SU has programs to let us see the trace data information:

suascii, *sudumptrace*

Of course, a plot is a nice way to see data:

suxwigb, *suximage*, *supswigp*, *supsimage*, *suxgraph*

> Documentation for these plotting programs tells only part of the story. To gain deep control of plotting options, check the documentation for the "inner" part of the plotting program. For example, *suxwigb* is a wrapper for program *xwigb*. While there are interesting options for *suxwigb*, there are also extensive options for program *xwigb*. This is true for many of the plotting programs.

These program lists are not meant to be all-inclusive. Wikipedia has a nice list of all SU programs:

https://en.wikipedia.org/wiki/Seismic_Unix

2.5 Trace header

There are many seismic data formats. Most companies have an internal format and frequently that format is proprietary (secret). In the early 1970s, the Society of Exploration Geophysicists (SEG) created the "Y" format (SEG-Y) after many meetings with various companies. The goal was to create an "exchange" format, a format that is not secret and is well documented. Since then, (usually but not always) when Company A wants to send a file to Company B, Company A will convert the file from their own format to SEG-Y. When Company B receives the file, it will convert the SEG-Y file to its own format for further processing or maybe just evaluation.

When referring to parts of trace headers:

- SEG-Y refers to byte sequences numerically; for example, bytes 1-4.
- Seismic Unix refers to byte sequences by names called "keys"; for example, the key for bytes 1-4 is `tracl`.

Each SEG-Y link in Section 1.1 has a valuable Trace Header table of the SEG-Y trace header, SU keys, the corresponding byte sequences, and definitions.

While the SEG-Y trace header has 240 bytes, SU only has keys for manipulating the first 180 bytes. To see my point, 2/3 of the way down the Trace Header table, after 2-byte key `ofrav` (bytes 179-180), the next line says:

-	60	181-240	SEG-Y Revison 0: Unassigned – for optional information.

Looking at the Trace Header table, we can see there are keys for bytes 181-240. In my experience, it is unwise to use these keys. Remember that I use *segyclean* to put zeros in some of these keys because (1) an imported SEG-Y file might have junk values in these keys and (2) I have found it dangerous to depend on SU to correctly use these bytes.

Before moving to the next section, look at the Trace Header table and see that, in bytes 1-180, there are only 2-byte and 4-byte keys. Naturally, 4-byte keys are used to hold larger numbers than can be in 2-byte keys.

2.6 Remap SEG-Y trace header bytes to SU keys

The SEG-Y import example below is copied from the example in the Introduction. (I added line breaks to make it easier to see various parts of the command.)

```
$ segyread tape=ts103828a.sgy
hfile=ts103828a.txt bfile=ts103828a.bin
byte=189l,105f,205f,181l remap=cdpt,swdep,gwdep,sdel
| segyclean > ts103828a.su
```

Here, I want to discuss the two parameters `byte` and `remap`. These are a pair; that is, for every `byte` value there must be a `remap` value. And, the first `byte` value must correspond to the first `remap` value, the second `byte` value must correspond to the second `remap` value, etc.

Parameter `byte` refers to locations in the SEG-Y trace headers and parameter `remap` refers to corresponding keys in the SU trace headers. The values of `byte` are numbers with a letter. The number is the first byte in the 2-byte or 4-byte sequence and the letter is the format. The table below, from the *segyread* documentation, explains the letter. From the example above, 189l means treat bytes 189-192 (4 bytes) as if they contain long integer values for key `cdpt`.

Letter	Meaning
f	float (4 bytes)
l	long int (4 bytes)
s	short int (2 bytes)
b	byte (1 bytes)

2.7 Import example SEG-Y file

Since the creation of the SEG-Y standard in 1975, the SEG-Y format has had revision 1 in 2002 and revision 2 in 2017. That "extended" information is also in the Trace Header table, but is meaningless to an SU trace because SU programs (and therefore SU traces) were never updated from revision 0. However, companies **do** put information in SEG-Y files in bytes 181-240.

Therefore, during SEG-Y import, I might want to "remap" information from trace header bytes 181-240 to bytes 1-180. This is a simple way of phrasing a trace header import problem. There are actually two possible trace header import problems:

Import Problem 1: The SEG-Y traces have information in bytes 181-240. After import, those bytes will become inaccessible to SU programs.

Import Problem 2: The SEG-Y trace header has 4-byte information in bytes 1-180 that will be placed in 2-byte keys in the SU trace headers, or the other way around.

These are *possible* problems. The challenge is to learn which trace header bytes have been used by the data supplier. That crucial information can be in documents that I get with the seismic datasets and it can be in the SEG-Y textual header. I review all the documentation I get with a dataset: internal reports in the form of paper or PDFs, technical articles, spreadsheets, etc.

> You might notice that I do not write about the contents of the binary header. It is valuable during file import, but not for seismic data processing. The binary header has information that *segyread* uses to correctly break the SEG-Y file into the header file, the binary file, and the .su trace header/trace data file; but the binary header does not contain any useful geophysical data.

To review the textual header file, I do a simple import of the SEG-Y file (see Section 2.1) to study the textual header file. For example:

```
$ segyread tape=ts103828a.sgy | segyclean > ts103828a.su
```

Or, slightly more complex, I do the import below to store the textual and binary header blocks into useful names:

```
$ segyread tape=ts103828a.sgy hfile=ts103828a.txt
bfile=ts103828a.bin | segyclean > ts103828a.su
```

If I decide bytes need to be remapped, I will do a second import adding the `byte` and `remap` parameters.

Below is the textual header file from dataset ts103828a.sgy.

```
C01  CLIENT: ERCH 4D-4C CONSORTIUM
C02  AREA: TEAL SOUTH
C03  PHASE 1   PZ SUMMED COMPONENT CDP GATHERS FROM INLINE 1104
C04                                  (CDP 1052-1188 BY 4)
C05  HEADER WORD TYPE                BYTE LOCATION
C06  ********************************************
C07  SHOT STATION NUMBER                9-12
C08  INLINE NUMBER                     189-192
C09  XLINE (CDP) NUMBER                 21-24
```

```
C10  X COORDINATE OF BIN CENTER         105-108
C11  Y COORDINATE OF BIN CENTER         205-208
C12  RECEIVER NUMBER                    181-185
C13
C14
C15
C16
C17
C18
C19
C20
C21
C22
C23
C24
C25
C26
C27
C28
C29
C30
C31
C32
C33
C34
C35
C36
C37
C38
C39
C40
```

For comparison, the Appendix is an example "typical" textual header. Compared to that "typical" textual header, I see that this textual header has very little information. I see:

- the name of the project (lines 1, 2),
- the type of data (line 3),
- all the gathers are from the same inline (line 3),
- this is a file of every fourth CDP from that inline (line 4), and
- a list of six headers with their start and end byte locations (lines 5-12):

Header Word	SEG-Y bytes	Above 180?
SHOT STATION NUMBER	9-12	No
INLINE NUMBER	189-192	Yes
XLINE (CDP) NUMBER	21-24	No
X COORDINATE OF BIN CENTER	105-108	No
Y COORDINATE OF BIN CENTER	205-208	Yes
RECEIVER NUMBER	181-185	Yes

In the Header Word table above:

- The first and second columns contain the information from the textual header.
- All the Header Words are four bytes long.
  - But what about Receiver Number? Although the textual header says it occupies five bytes, I am going to assume (!) that is a typing mistake. There are no 5-byte keys in SU. If I want to remap Receiver Number, I have to pick a 4-byte key or a 2-byte key.

- The third column indicates whether the Header Word occupies bytes less than or greater than 180. This indicates whether I might want to remap a Header Word during import from bytes that are above 180 to bytes below 181.

Next, I use *surange* to learn which SU keys have useful information and which keys are empty:

```
$ surange < ts103828a.su
```

```
644 traces:
tracl    1 644 (1 - 644)
tracr    1 644 (1 - 644)
fldr     1001071 1235039 (1025103 - 1235039)
tracf    1 24 (7 - 23)
ep       1039 1137 (1103 - 1039)
cdp      1052 1188 (1052 - 1188)
cdpt     1 24 (7 - 23)
trid     1
nvs      1
duse     1
offset   13 2501 (715 - 1974)
gelev    36 14133 (1491 - 14133)
selev    2822 3700 (3475 - 3131)
sdepth   82 85 (83 - 82)
gdel     1 26 (8 - 21)
swdep    80 86 (83 - 86)
gwdep    860
scalel   -1 0 (0 - -1)
scalco   -89 89 (1 - -54)
sx       569222 572148 (569519 - 572148)
sy       -57742 -56517 (-56942 - -57742)
gx       570224 571226 (570234 - 571015)
gy       -57330 -56122 (-56923 - -56125)
counit   1
sstat    734 854 (836 - 801)
gstat    839 861 (853 - 842)
tstat    3
laga     8
lagb     -19949 -18239 (-19947 - -18242)
muts     6 1750 (506 - 1504)
mute     106 1850 (606 - 1604)
ns       2000
dt       2000
sfs      0 2383 (1670 - 0)
slen     501 618 (513 - 618)
stas     20 69 (52 - 20)
afils    7777
lcf      168
hcf      31004 31140 (31004 - 31140)
hcs      1
day      1039 1137 (1103 - 1039)
minute   1001 1235 (1025 - 1235)

Shot coordinate limits:
       North(-570825,56942) South(5.07499e+07,-5.13904e+06) East(5.08214e+07,-
5.03224e+06) West(-571944,56942)

Receiver coordinate limits:
       North(-571025,56930) South(5.08034e+07,-5.10237e+06) East(5.08213e+07,-
5.10219e+06) West(-571225,56929)

Midpoint coordinate limits:
```

```
      North(-571228,56936) South(5.08036e+07,-5.0673e+06) East(5.08214e+07,-
5.06722e+06) West(-571584,56935.5)
```

I am surprised to see that so many SU keys have non-zero values. I had hoped there would be several 4-byte keys available (not listed, meaning the key values are zero) for me to remap the +180 bytes into. This makes my decisions more complicated.

- When a key does not appear in the *surange* output, it means none of the traces have a value for this key or all the traces have the value zero.
- When a key has a single value in the *surange* output (the minimum and maximum values are the same), it means either all the traces have a single value or at least one trace has only that value for the key. To know which of these two situations is true, I would use *sugethw* ("writes [print to screen or use redirect to print to disk] the values of the selected key words").

For my first trace header review, I look at Shot Station Number, SEG-Y bytes 9-12. Without remapping, these bytes will go into SU key `fldr`. I am happy with that.

Next, I look at Crossline (Xline) Number. Left alone, it will go from SEG-Y bytes 21-24 into the 4-byte key `cdp`, which is good. However, the related header, Inline Number, is in bytes 189-192, which will go into SU trace headers above 180 bytes. I would like to remap Inline Number close to Crossline Number, so I look for an unused or seldom used 4-byte key near key `cdp`. I see that `tracl` and `tracr` appear to be duplicates, and the same for `tracf` and `cdpt`. If I want to be certain about these duplicates, I would use *sugethw*; for example:

```
$ sugethw < ts103828a.su key=tracl,tracr,fldr,tracf,cdpt
```

or

```
$ sugethw < ts103828a.su key=tracl,tracr,fldr,tracf,cdpt output=geom
```

Notice I include key `fldr`. I am using `fldr` as a sort of control, not something I am testing. I like to include a control whenever I use *sugethw*.

I prefer the second example, with `output=geom`. The first example includes the key label on every output line (which is helpful), but every other output line is a blank line (which I find annoying).

I ran this *sugethw* test and found that, yes, `tracr` is a duplicate of `tracl`, and `cdpt` is a duplicate of `tracf`. In my experience, key `cdpt` is rarely used, so I will remap SEG-Y trace header bytes 189-192 to SU key `cdpt`. I also like the fact that `cdpt` is next to `cdp`.

For the sake of this import, I assume Header Words Inline Number and Crossline Number are "long integer" format, "l" format according to the table in Section 2.6.

> In my experience, "number" such as Inline Number is an integer number. Also in my experience, "coordinate" such as X-Coordinate is a floating point number.
>
> The only way I can be certain about "format" is to import the file, then compare the key values to documentation that came to me with the seismic file.
>
> **For this dataset, I will be wrong about the X-Coordinate and Y-Coordinate being floating point values. But for this learning experience, I will continue as if I am right. I will correct this in a third use of *segyread*.**

So far, my `byte` and `remap` values are:

```
byte=189l remap=cdpt
```

Let me review the line above. I want to remap SEG-Y trace header bytes 189-192 to 4-byte SU key `cdpt` as long integer ("l") format.

Next, I look at X-Coordinate Bin Center and Y-Coordinate Bin Center, bytes 105-108 and bytes 205-208. The X-Coord values occupy bytes less than 181, but if I do nothing, these values will overwrite two 2-byte keys, `laga` (bytes 105-106) and `lagb` (bytes 107-108). Refer to the wiki table. If I let this happen, the X-Coordinate data will become corrupted. (This is **Import Problem 2** discussed above.) Also, if I want to use the Y-Coordinate data, I must remap those bytes because they occupy bytes greater than 180. Finally, as I mentioned in the box above, because these are "coordinates," I assume these are floating point ("f") values.

Looking at 4-byte keys in the *surange* output, I see 4-byte keys `swdep` (bytes 61-64) and `gwdep` (bytes 65-68). I do not plan to use this dataset to study the water depth of the sources and receivers, so I feel fine overwriting these keys. Now my `byte` and `remap` values are:

```
byte=189l,105f,205f remap=cdpt,swdep,gwdep
```

There are no spaces around an equal sign and no spaces around a comma.

> Notice I decided to overwrite potentially useful data in keys `swdep` and `gwdep`. I do this reluctantly, but the SEG-Y data have to go somewhere, and coordinates have to go into 4-byte keys. If at some point in my research I need water depth of sources and receivers, I might do another import putting the X and Y coordinates into keys `ep` and `gdel`; that is a different overwrite. The problem is, this SEG-Y dataset has a lot of trace header information and SU trace headers have a limited number of 4-byte keys.

Last is Receiver Number. As I wrote earlier, because this is "Number," I assume the values are integers. The textual header says this Header Word occupies bytes 181-185. That is five bytes! I will assume this is a typing error, that it actually occupies four bytes, 181-184. A 4-byte key not being used is `sdel`. My final `remap` and `byte` values are:

```
byte=189l,105f,205f,181l remap=cdpt,swdep,gwdep,sdel
```

Header Word	Current bytes	Above 180?	start byte +format	remap [bytes]
SHOT STATION NUMBER	9-12	No	-	-
INLINE NUMBER	189-192	Yes	189l	cdpt [25-28]
XLINE (CDP) NUMBER	21-24	No	-	-
X COORDINATE OF BIN CENTER	105-108	No	105f	swdep [61-64]
Y COORDINATE OF BIN CENTER	205-208	Yes	205f	gwdep [65-68]
RECEIVER NUMBER	181-185	Yes	181l	sdel [57-60]

In the table above:

- The first and second columns contain the information from the textual header.
- The third column indicates whether the Header Word occupies bytes up to or greater than 180. This indicates whether a Header Word needs to be remapped during import. I remapped Header Word "X-COORDINATE OF BIN CENTER" even though this Header Word occupies bytes less than 180. A simple import will put this header into two 2-byte keys, destroying the information.
- The fourth and fifth columns are the `byte` and `remap` values for my second *segyread* (below).

```
$ segyread tape=ts103828a.sgy hfile=ts103828a.txt
bfile=ts103828a.bin byte=189l,105f,205f,181l
remap=cdpt,swdep,gwdep,sdel | segyclean > ts103828b.su
```

Below is the *surange* output from this second *segyread*. To save space, I only show SU keys for bytes 1 to 90.

```
$ surange < ts103828b.su
```

```
644 traces:
tracl    1 644 (1 - 644)
tracr    1 644 (1 - 644)
fldr     1001071 1235039 (1025103 - 1235039)
tracf    1 24 (7 - 23)
ep       1039 1137 (1103 - 1039)
cdp      1052 1188 (1052 - 1188)
cdpt     1104
trid     1
nvs      1
duse     1
offset   13 2501 (715 - 1974)
gelev    36 14133 (1491 - 14133)
selev    2822 3700 (3475 - 3131)
sdepth   82 85 (83 - 82)
gdel     1 26 (8 - 21)
sdel     721080 1681160 (1041080 - 1681144)
swdep    -2147483648 373358592 (0 - 0)
scalel   -1 0 (0 - -1)
scalco   -89 89 (1 - -54)
sx       569222 572148 (569519 - 572148)
sy       -57742 -56517 (-56942 - -57742)
gx       570224 571226 (570234 - 571015)
gy       -57330 -56122 (-56923 - -56125)
counit   1
```

Let us see how I did:

Key `cdpt` is supposed to be the Inline Number. It appears to be a constant value 1104. This makes sense because that is what the textual header says on line 3.

Key `swdep` is supposed to be the X-Coordinate. Program *surange* shows the values range from -2147483648 to +373358592. Maybe that is a valid range, but maybe not. Key `gwdep` is supposed to be the Y-Coordinate. This key is missing from the *surange* output, which means all the SU trace header values are zero. I do not like that!

Key `sdel` is supposed to be Receiver Number. Program *surange* shows the values range from 721080 to 1681160. Maybe that is a valid range, but maybe not.

Conclusion: All the gwdep values are zero, which makes me think my idea of importing SEG-Y X-Coordinate (trace bytes 105-108) and Y-Coordinate (trace bytes 205-208) as "float" was a bad idea. I will try the import again, but this time import the Coordinate bytes as "long integer" (the only other option for 4-byte trace headers).

```
$ segyread tape=ts103828a.sgy hfile=ts103828a.txt
bfile=ts103828a.bin byte=189l,105l,205l,181l
remap=cdpt,swdep,gwdep,sdel | segyclean > ts103828c.su
```

Below is the *surange* output from this third *segyread*. Again, to save space, I only show SU keys for bytes 1 to 90.

```
$ surange < ts103828c.su
```

```
644 traces:
tracl    1 644 (1 - 644)
tracr    1 644 (1 - 644)
fldr     1001071 1235039 (1025103 - 1235039)
tracf    1 24 (7 - 23)
ep       1039 1137 (1103 - 1039)
cdp      1052 1188 (1052 - 1188)
cdpt     1104
trid     1
nvs      1
duse     1
offset   13 2501 (715 - 1974)
gelev    36 14133 (1491 - 14133)
selev    2822 3700 (3475 - 3131)
sdepth   82 85 (83 - 82)
gdel     1 26 (8 - 21)
sdel     721080 1681160 (1041080 - 1681144)
swdep    569875 571585 (569877 - 571582)
gwdep    -56937 -56923 (-56933 - -56934)
scalel   -1 0 (0 - -1)
scalco   -89 89 (1 - -54)
sx       569222 572148 (569519 - 572148)
sy       -57742 -56517 (-56942 - -57742)
gx       570224 571226 (570234 - 571015)
gy       -57330 -56122 (-56923 - -56125)
counit   1
```

Let us see how I did:

Key `cdpt`, Inline Number, is, again, the constant value 1104. As before, this makes sense.

Key `swdep`, the X-Coordinate, the values range from 569875 to 571585. Key `gwdep`, the Y-Coordinate, has values (yay!) and they range from -56937 to -56923. I do not have any paper documents, PDFs, or spreadsheets about this dataset to confirm these results, but I see that these values are in the same range as the values in keys `sx`, `sy`, `gx`, and `gy`. These are much better results than I got in my previous import.

Key `sdel`, Receiver Number, again has values that range from 721080 to 1681160. Maybe that is a valid range, but maybe not.

2.8 Import summary:

Make sure SEG-Y 4-byte values go to SU 4-byte keys (and the same for 2-byte values/keys).

Should SEG-Y 4-byte values be imported as floats or long integers? That can be answered partly by importing the values as one format, then the other, looking for "reasonable" values after import. But this can only be answered definitively by reviewing the documentation that comes with the seismic data (if any).

Above, I imported `sdel` as long integer. The result is not satisfying, perhaps unrealistic. I tried the import one more time, this time setting SEG-Y bytes 181-184 as float. The SU *surange* result is:

```
sdel     -2147483648 0 (0 - -2147483648)
```

This, again, is unsatisfying. I hope I do not need Receiver Number.

3. Export an SU file to SEG-Y format

Use program *segywrite* to convert a Seismic Unix file to SEG-Y format. Program *segywrite* requires a textual header file and a binary header file. When I do not have them, I use *segyhdrs* to create them.

For an example export, I create an SU dataset from a simple SU program:

```
$ suplane > plane.su
```

To export this file to SEG-Y format, I need textual and binary header files:

```
$ segyhdrs < plane.su bfile=plane.bin hfile=plane.txt
```

As hoped (below), the size of the textual header block is 3200 bytes and the size of the binary header block is 400 bytes.

```
$ ls -l plane.*
```

```
-rw-rw-r-- 1 df df   400 Mar  9 12:03 plane.bin
-rw-rw-r-- 1 df df 15872 Mar  9 12:03 plane.su
-rw-rw-r-- 1 df df  3200 Mar  9 12:03 plane.txt
```

The textual header block has no carriage returns, no line feeds; just an EOF (end-of-file) at the end. This file is one long line. If I print this file to screen …

```
$ cat plane.txt
```

I suggest you set your terminal window to 80 characters wide. Do the same if you choose to edit the file. As you can see, the text generated by *segyhdrs* is not useful, only adequate to have made the (necessary) file. You can edit this file, but do NOT use the “Return” (or “Enter”) key anywhere in the file. And, when you finish editing, make sure the file still contains exactly 3200 bytes.

```
C      This tape was made at the
C
C      Center for Wave Phenomena
C      Colorado School of Mines
C      Golden, CO, 80401
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
C
```

Now that I have the necessary textual and binary header files, I can export the SU file to SEG-Y format:

```
$ segywrite < plane.su tape=plane.sgy hfile=plane.txt
bfile=plane.bin
```

The value of parameter `tape` is the name of the output SEG-Y file.

If I need to set the “endianness” of the output file, I can use the *segywrite* parameter `endian`:

```
endian =(autodetected)
       =1 for big-endian byte order
       =0 for little-endian byte order
```

For example, to export file plane.su in big-endian format:

```
$ segywrite < plane.su tape=plane.sgy hfile=plane.txt
bfile=plane.bin endian=1
```

References: SEG-Y Formats

SEG-Y revision 2.0 Data Exchange format, SEG Technical Standards Committee, January 2017

https://seg.org/Portals/0/SEG/News%20and%20Resources/Technical%20Standards/seg_y_rev2_0-mar2017.pdf

SEG-Y revision 1.0 Data Exchange format, SEG Technical Standards Committee, May 2002

https://www.seg.org/Portals/0/SEG/News%20and%20Resources/Technical%20Standards/seg_y_rev1.pdf

SEG-Y revision 0 Data Exchange format, SEG Technical Standards Committee, April 1975

Barry, K.M., Cavers, D.A., and Kneale, C.W., 1975, Special Report: Recommended Standards for Digital Tape Formats, Geophysics, Vol. 40, No. 2, 344-352.

Below is another reference for SEG-Y revision 0 (1975). I include this, with the permission of Prof. Doug Toomey at the University of Oregon, because the text is easier to read than the text in the Geophysics report. The figures are the same.

https://pages.uoregon.edu/drt/MGL0910_Science_Report/attachments/seg_y_rev0.pdf

Appendix: Typical SEG-Y textual header

The SEG-Y textual header, below, is from a marine acquisition project in Eugene Island, Gulf of Mexico. I include this example because, unlike the example above in Section 2.7, this file's format is typical of SEG-Y textual headers from commercial seismic data processors.

```
C 1 PGS TENSOR SEG-Y
C 2 LINE                 3440                                          C 2 L
C 3 ----------------------------------------
C 4 ACQ: DIAMOND GEO USING PGS EXPLORATION STRMR LEN:  6000 M
C 5 AREA:  EUGENE ISLAND PHASE I     STRMR DEPTH:  9 M
C 6 R/V:  MV EDISON & JONATHAN CHOUEST      GRP INT:  25 M
C 7 AZMTH:  269.202356 DEGREES            GRPS/STRMR:  240
C 8 SRC VOL & PRSR:   4770 CU IN, 2000 PSI    REC INSTR:  SYNTRAK 480
C 9 SRC DEPTH:  7.5 M               FILT:  3HZ 6DB/OCT-218HZ 484DB/OCT
C10 NO OF SRCS:  2                        REC LEN:  10752 MS
C11
C12 NR OFFSET:  185 M                     SAMP INT:  2 MS
C13 STRMR TYPE:  TELEDYNE DIGITAL
C14 NO OF STRMR:  3                       ACQ CLIENT CONTCT:  JOHN CRAMER,  DIAM
C15 ------------------------------------------------------------
C16 NAV:  STARFIX & WADGPS COMPARISON     CEN MRDN:  93 0 .000W
C17 DATUM:  SEA LEVEL                     ORIGIN COOR:  500000.00E
C18 UNIT:  METERS                         SCALE FCTR:   .9996000000
C19 PROCESSING:  PGS TENSOR HOUSTON, TX UNDER SUPERVISION OF  DIAMOND GEO
C20 PROJECT:  EUGENE ISLAND PHASE I
C21 PROCESSING SUPERVISOR:  DAVIS RATCLIFF
C22
C23 PROCESSING FLOW:
C24 SEG-D REFORMAT,       RESAMPLE TO 4 MS
C25 TRACE EDITING,        NAVIGATION MERGED WITH SEISMIC
C26 HIGH PASS FILTER:  LOW CUT 2 HZ, LOW PASS 4 HZ
C27 DATUM CORRECTION,     SPHERICAL DIVERGENCE & GAIN CORRECTION
C28 FIRST BREAK MUTE,     DECONVOLUTION:  SPIKE, 2 WINDOW
C29 TRACE BALANCING,      DMO STACK
C30 INTERPOLATION TO 20 M SUBLINE SPACING,   CLEAN-UP MUTE
C31 BANDPASS FILT:  LOW CUT          HIGH CUT
C32      0 SEC:       3/5              80/100
C33      5 SEC:       3/5              65/85
C34      6 SEC:       3/5              50/70
C35      8 SEC:       3/5              40/55
C36     10 SEC:       3/5              35/50
C37 GAIN DOWN:  -2 DB/SECOND
C38 MIGRATE: ONE PASS CASCADED FINITE DIFFERENCE
C39 GAIN RECOVERY: +2 DB/SECOND,  WATER BOTTOM REFERENCED CLEAN-UP MUTE
C40 GLOBAL RELATIVE AMPLITUDE COMPENSATION
```

The upper two-thirds usually has project information, acquisition parameters, and sometimes geometry information. The bottom one-third sometimes has processing information.

For a diagram of a typical SEG-Y textual header, see the SEG-Y rev2 reference document's Table 1, "Textual File Header" (page 5).