Seismic Unix Programming, Part 1

This document explains how to convert an existing Seismic Unix (SU) program into your own.

Always (**always!**) make your own copy of a program before you begin your modifications.

1. General statements about SU programming

In my mind, there are two reasons to create an SU program:

A. Create a process that does not currently exist within SU.
B. Explore how a current program works. By this I mean, I want to understand how a particular SU program (algorithm) works, so I modify the program to print information to the screen or to a file as the algorithm progresses. For example, I might want to understand finite-difference migration, so I create a personal version of *sumigfd*.

The basic steps of programming are:

1. Modify an existing program or create one from scratch (from nothing). This is where the programmer modifies or writes the C-code. (Most SU programs are written in C.)
2. Compile the program. This step checks the syntax of the program.
3. Link the program. This step connects (bundles) the program with the libraries that it calls.

When the compile and link steps are successful, I have an executable form of the program.

2. SU source program (C file) directories

Most of the SU source programs (the C files) are in directories under:

```
$CWPROOT/src/su/main
```

`$CWPROOT` is the SU "export" direction I put in my `.bashrc` file. In my case:

```
export CWPROOT=/usr/su44r19
```

Which means most of the SU source files are in application directories under:

```
/usr/su44r19/src/su/main
```

Some of the application directories are:

```
$CWPROOT/src/su/main/amplitudes
$CWPROOT/src/su/main/attributes_parameter_estimation
$CWPROOT/src/su/main/convolution_correlation
$CWPROOT/src/su/main/data_compression
```

Some graphics source files are:

```
$CWPROOT/src/su/graphics/psplot
$CWPROOT/src/su/graphics/xplot      example: suxwigb.c
```

The above two directories hold the wrappers for the programs in the directories below.

```
$CWPROOT/src/psplot/main
$CWPROOT/src/xplot/main              example: xwigb.c
```

I mention the plot wrapper program and the wrapped program for two reasons. (a) The plot programs are not in the application program directories. (b) If I want to modify a plot program, I need to decide whether I want to modify the wrapper program or the wrapped program.

Another interesting repository of source files are the directories under:

```
$CWPROOT/src/par/main
```

3. Compile and link a program

Let us pretend I "cd" into a source code directory to make my own version of program `oso.c`. I copy that program to make my own version, `oso2.c`. I can edit `oso2.c` anywhere, but after I finish my edits, I put `oso2.c` in the same directory as `oso.c`.

My version, `oso2.c`, is different from `oso.c` but it uses the same libraries as `oso.c`.

> This document is called "Part 1" because the examples only use libraries that are in the original program. A "Part 2" document would explain how to access libraries that are not in the copied program. Honestly, today I do not know how to do that.

All the directories that contain source code have a file called `Makefile`. The `Makefile` in a directory was built to link the programs in its directory with a specific set of libraries. The `Makefile` in the directory with my imaginary program `oso.c` has a row with `oso.c`. To compile and link my `oso2.c`, I need to create an `oso2.c` row in this `Makefile`.

A `Makefile` is delicate. Spaces and tabs in a `Makefile` are critically important. To make the `oso2.c` row in `Makefile`, I copy the `oso.c` row, paste it back in the `Makefile`, then I edit the new row to be my new program name. After I add the row for `oso2.c`, I use the arrow keys to navigate around to make sure my `oso2.c` row has tabs and spaces just like the `oso.c` row (and all the other rows with program names).

After I am sure my `Makefile` has been edited correctly, I exit the `Makefile` and in this same directory, I give this command:

```
$ make
```

This command compiles and links any modified versions of any programs in the `Makefile` list. For example, `oso.c` was not modified, so it will not be recompiled and relinked.

All executable programs are in:

```
$CWPROOT/bin
```

4. Application program example

To modify `sugain.c`, I go into the "amplitudes" directory and create my own version, `sugain2.c`:

```
cd $CWPROOT/src/su/main/amplitudes
cp sugain.c sugain2.c
```

For the simple edits I plan, I will edit my program with gedit:

```
$ gedit sugain2.c &
```

Below are the normal header lines of a released SU program. These lines can be replaced by credits from me. When the program is released by SU, whatever I put here will be replaced by the SU release machinery.

```
1  /* Copyright (c) Colorado School of Mines, 2011.*/
2  /* All rights reserved.                       */
3
4  /* SUGAIN: $Revision: 1.63 $ ; $Date: 2019/07/22 16:31:00 $     */
5
6  #include "su.h"
```

Below is the single header line I put in my version of the gain program. Give yourself credit for the program so the SU managers can show your contribution in the release notes.

```
1  /* David Forel, Seismic Rocks LLC, 2020. */
2
3  #include "su.h"
```

Below, I renamed the program in the "self documentation" lines. Search and replace through the source code to make sure you replace the old name with the new one.

```
9  /************************ self documentation ************
10 char *sdoc[] = {
11 "                                    ",
12 " SUGAIN2 - apply various types of gain
13 "                                    ",
14 " sugain2 <stdin >stdout [optional parameters]
15 "                                    ",
```

Below, I added a comment to the "Credits" section. Describe what you did (very briefly) so you get full credit for your work when this is released. This also makes clear to future users why this new program or edited program is useful.

```
75 /* Credits:
76  *  SEP: Jon Claerbout
77  *  CWP: Jack K. Cohen, Brian Sumner, Dave Hale
78  *  David Forel: Minor comment. 2020.
79  *
```

Below, line 237, is my "Minor comment."

```
233     if (jon) {
234         tpow  = 2.0;
235         gpow  = 0.5;
236         qclip = 0.95;
237         warn("\n You chose the 'jon' option.\n");
238     }
```

Below is a look at my edited `Makefile` file. This is the Makefile in the `amplitude` directory, the directory that contains the source file (the C program) `sugain.c`. You can see where I copied the line for `sugain`, pasted it below `sugain`, and renamed it `sugain2`.

```
11 PROGS =          \
12     $B/sucentsamp    \
13     $B/sudipdivcor   \
14     $B/sudivcor \
15     $B/sugain    \
16     $B/sugain2   \
17     $B/sunan     \
```

As I wrote in the previous section, **carefully** review the line to be sure it uses tabs and spaces the same way they are used in the other lines in this section of `Makefile`. After I am sure my `Makefile` has been edited correctly, I exit `Makefile` and enter this command:

```
$ make
```

The screen output is below:

```
df@sp3:/usr/su44r18/src/su/main/amplitudes$ make
gcc -I/usr/su44r18/include -O  -std=c99 -Wall -pedantic -Wno-long-long  -D_FILE_
OFFSET_BITS=64 -D_LARGEFILE64_SOURCE -DCWP_LITTLE_ENDIAN   -D_BSD_SOURCE -D_POSI
X_SOURCE sugain2.c  -L/usr/su44r18/lib -lsu -lpar -lcwp -lm  -o /usr/su44r18/bin
/sugain2
In file included from /usr/include/x86_64-linux-gnu/bits/libc-header-start.h:33,
                 from /usr/include/stdio.h:27,
                 from /usr/su44r18/include/cwp.h:12,
                 from /usr/su44r18/include/par.h:12,
                 from /usr/su44r18/include/su.h:14,
                 from sugain2.c:3:
/usr/include/features.h:187:3: warning: #warning "_BSD_SOURCE and _SVID_SOURCE a
re deprecated, use _DEFAULT_SOURCE" [-Wcpp]
  187 | # warning "_BSD_SOURCE and _SVID_SOURCE are deprecated, use _DEFAULT_SOU
RCE"
      |   ^~~~~~~
sugain2 installed in /usr/su44r18/bin
df@sp3:/usr/su44r18/src/su/main/amplitudes$
```

That output confuses me, but I only see warnings (no errors). Also, the last line (see the arrow) means I succeeded.

To test whether my gain program is accessible, I check for the selfdoc by typing the name of the program:

```
$ sugain2
```

In the source file, as a bit of a joke, I added a smiley face to the end of the line that documents the “jon” parameter:

```
39 "   bias=0.0    bias data by adding an overall bias value   ",
40 "   jon=0       flag; 1 means tpow=2, gpow=.5, qclip=.95  ;-) ",
41 "   verbose=0   verbose = 1 echoes info                     ",
```

In the selfdoc, it appears as:

```
        bias=0.0        bias data by adding an overall bias value
        jon=0           flag; 1 means tpow=2, gpow=.5, qclip=.95   ;-)
        verbose=0       verbose = 1 echoes info
```

I am satisfied that my version of `sugain2` is in the SU bin directory.

Because all executable programs are in `$CWPROOT/bin`, and because that bin directory is in my PATH, I can run my gain program from any directory. For example, below I run it from the directory that contains my datasets:

```
~/work$ sugain2 < ozle01.su jon=1 | suxwigb perc=99 title="jon=1" &
```

Below is the screen output:

```
sugain2:
   You chose the 'jon' option
```

5. Plot program example

For a plot example, I will create my own version of `xwigb`. But because `xwigb` is wrapped by `suxwigb`, I will also create my own version of `suxwigb` that will run my version of `xwigb`. Making this a little more complicated, these two programs are in separate directories.

1. I make sure I know which versions of these two programs are in the bin directory:

```
$ cd $CWPROOT/bin
$ ls *xwig*
```

This returned `suxwigb` and `xwigb`, so I know there will not be a naming conflict if I name my new files with "2" (for example).

2. I make my own copies of these programs and the `Makefile` in these two directories:

```
$ cd /usr/su44r18/src/su/graphics/xplot
$ cp suxwigb.c suxwigb2.c
$ cp Makefile Makefile_backup

$ cd /usr/su44r18/src/xplot/main
$ cp xwigb.c xwigb2.c
$ cp Makefile Makefile_backup
```

3. I carefully add my new program names to the two `Makefile` files.

4. I modify `suxwigb2` to call `xwigb2` (replace the name), then I run `$ make` in the suxwigb2.c directory to ensure this (easy part) works. Screen output from $ make gives only warnings, but does not tell me it installed the executable version in the bin directory. I check (`ls`) the $CWPROOT/bin directory and see a new `suxwigb2` there.

5. I modify xwigb2.c to reflect the name change and add a simple print message.

```
313         /* if necessary, determine clip from percentile */
314         if (!getparfloat("clip",&clip)) {
315                 perc = 100.0;  getparfloat("perc",&perc);
316                 temp = ealloc1float(nz);
317                 warn("    -->  1  perc = %g\n",perc);
318                 for (iz=0; iz<nz; iz++)
319                         temp[iz] = fabs(z[iz]);
320                 iz = (nz*perc/100.0);
321                 if (iz<0) iz = 0;
```

6. I run `$ make`. Screen output gives warnings and the happy message:

```
xwigb2 installed in /usr/su44r18/bin
```

7. I run `suxwigb2`:

```
$ suxwigb2 < ozle01.su perc=99 &
```

My comment is printed to the screen by the wrapped program:

```
xwigb2:     -->  1  perc = 99
```

8. I change the comment, run `$ make` for `xwigb2` but ***not*** for `suxwigb2`.

9. I run `suxwigb2`:

```
$ suxwigb2 < ozle01.su perc=99 &
```

I find that my changed comment in `xwigb2` is written to the screen:

```
xwigb2: -->  -->  2  perc = 99
```

This means an SU program finds the most recent version of any function that it calls.

Appendix. Minimum SU keys

For a trace to exist in SU, it must have appropriate values in keys `ns` and `dt`. Values in other keys are often useful, but values in these keys are mandatory.

In the sample interval key `dt`, the unit is microseconds. This means when a trace has sample interval 2 milliseconds, the value for `dt` is 2000.

SU key	Length	Bytes	Description
ns	2	115-116	Number of samples in this trace
dt	2	117-118	Sample interval in µsec for this trace