scl2txt program:

is used as follows:

```
./scl2txt run_low run_high file.out
```

One must specify 3 parameters: two run numbers and a file name of the output file. The two run numbers specify a range of runs to be processed. And the output file is a text file where the resulting summary will be saved.

---

source file is: **`scl2txt.cpp`**
can be compiled by running: **`>make`**
or by **`>g++ scl2txt.cpp -o scl2txt`**

Source file can be modified easily for any input scaler file format:
1. modification of the file name for each run:
   default is: "run**.scalers" where "**" is run number
can be modified easily in the following section of the source code:

```
for(i=low_i;i<=high_i;i++)   // <-- looping through all scaler files (runs)
        {
          sprintf(filename,"run");      //   <<-- Putting the scaler file name together
          sprintf(numstring,"%d",i);    //  this part needs to be modified in case
          strcat(filename,numstring);   //  scaler file-names change
          strcat(filename,".scalers");      //
          printf("%s\n",filename);      //  Scaler file name is here
          ifstream fp(filename,ios::in|ios::nocreate);   // opening the scaler file
          if(!fp.is_open())
            cerr << "Error: Scaler file: " << filename << " was not open! (run is
skipped!)\n";
          else
```

The two lines in bold show where changes should be made if the scaler files are called differently. First one: "prefix" of the filename is "run". The latter one "suffix" of the filename is ".scalers". In between we insert the run number, which is true in almost all cases.

2. Modification of the format of the scaler file. Changes will be made in the following section of the source code:

```
// ******************* Reading the scaler file: ***********************
              cout << "Converting file: " << filename << endl;
              fp.getline(line,256);
              fp.getline(line,256);
              fp.getline(line,256);
              fp.getline(line,256);
              fp.getline(line,256);
              fp.getline(line,256);
              fp.getline(line,256);   // ignoring first 7 lines from the scaler file
              // these lines contain various descriptions
              names.clear();
              values.clear();
              names.push_back("RUN");
              values.push_back(i);
              while (!fp.eof())   // reading the rest of the file
                {
```

```
    fp.getline(line, 256);
   // formatting is contained here:
   sscanf(line, "%s\t%d\t%f\t%f", scaler_name, &scaler_value, &junk, &junk);
//^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
    string name(scaler_name);
    names.push_back(name);
    values.push_back(scaler_value);
  }
```

The lines in bold face are the most important ones. The first part (fp.getline(line, 256)) is where I ignore the first 7 lines of the scaler file, by simply reading 7 lines in and never doing anything with it. But the most important is the sscanf(..) line where I actually defined the formatting for reading in the scaler file. This line assumes the scalers to be in the following format:

```
                     HiRA+s800 coincidence.  Beam on plastic target.
                     Run: 120
                     Started: Wed Jul 27 08:47:02 EDT 2005
                     Ended:  Wed Jul 27 10:01:08 EDT 2005
                     Duration: 0 01:13:54
 Scaler Name        Scaler Total  Avg Rate     Std. Dev
------------------------------------------------------------------
    Bad_CRDC          149         0.03         0.13
    COM_STOP            0         0.00         0.00
 CRDC1.Anode      1293772       291.78        30.75
 CRDC2.Anode      1281483       289.01        30.70
 Coinc.Trigger       211558        47.71         6.86
   CsI1_Gate       211542        47.71         6.85
   CsI2_Gate       211541        47.71         6.85
  CsI_Center      1152929       260.02        32.28
     E1.Down      1270694       286.58        30.45
       E1.Up      1248339       281.54        29.94
     E2.Down            0         0.00         0.00
       E2.Up            0         0.00         0.00
     E3.Down            0         0.00         0.00
       E3.Up            0         0.00         0.00
 Ext1.Source          972         0.22         0.33
 Ext1.Trigger            0         0.00         0.00
 Ext2.Source            0         0.00         0.00
 Ext2.Trigger            0         0.00         0.00
 HiRA_Trigger      4128569       931.12       130.30
  Live.Clock     40363163      9103.10       365.47
 Live.Trigger       211535        47.71         6.85
        MCP0    898875998    202723.50     27231.09
        MCP1    854538155    192723.99     28706.49
   OBJ.Scint   1093969519    246722.94     31363.24
      OBJ.Si            0         0.00         0.00
   OBJ_SCINT            0         0.00         0.00
   Raw.Clock     44342908     10000.66       364.78
 Raw.Trigger       211558        47.71         6.86
 S800.Source      1251032       282.15        29.93
 S800.Trigger            0         0.00         0.00
 Second.Source      4059035       915.43        45.28
 Second.Trigger            0         0.00         0.00
    TOW0_CsI       361774        81.59        11.38
     TOW0_Si       912441       205.78        16.67
   TOW0_Trig       914751       206.30        19.38
    TOW1_CsI       747075       168.49        22.20
     TOW1_Si      1388976       313.26        31.06
   TOW1_Trig      1390214       313.53        30.32
    TOW2_CsI       767037       172.99        22.18
     TOW2_Si      1527097       344.41        34.89
   TOW2_Trig      1527034       344.39        31.79
    TOW3_CsI       282360        63.68         9.21
     TOW3_Si      1103697       248.92        23.11
   TOW3_Trig      1401002       315.97       200.30
      TPPAC1     46469824     10480.34      1077.72
      TPPAC2     10561700      2381.98       261.89
```

```
XFP.Scint   1479688500    333714.14     37660.25
XFP_SCINT            0         0.00         0.00
```