Innovative Performance Improvements Through Automated Flowcharts In SAS®

Steven First, Systems Seminar Consultants, Inc.

ABSTRACT

One of the tedious but necessary things that SAS programmers must do is to trace and monitor SAS runs: counting observations and columns, checking performance, drawing flow charts, and diagramming data flows. This is required to be able to audit results, find errors, find long running steps, identify large data files, and to simply understand what a particular job does. Enterprise Guide includes functionality to help with some of this, on some SAS jobs. This paper presents an innovative custom tool that automatically produces flow charts — showing all SAS steps in a job, file counts in and out, variable counts, recommendations and generated code for performance improvements, and more. The system was written completely in SAS using SAS source programs, SAS logs, PROC SCAPROC output, and more as input. This tool can eliminate much of the manual work needed in job optimization, space issues, debugging, change management, documentation, and job checkout. As has been said many times before: "We have computers, let’s use them".
INTRODUCTION

How did this all begin?

A very common task is to look at a SAS job and try to make sense of what it is doing. Many questions come up:

1. Did it have errors?
2. Are the row counts reasonable?
3. The job ran out of space, where was the problem, and what do I do to fix it?
4. Does my job carry along extra baggage like, files no longer needed, files and columns that are never used?
5. Are columns too wide?
6. Would compression help or hurt?
7. Where is every place that a file is used?
8. What is the data lineage of a column that has passed through multiple steps?
9. A graphical representation may be easier to understand and help with documentation.
10. Or the worst sentence you want to hear yourself or others ask: Just what does the job do?

As I have looked at my own SAS jobs, or other user’s jobs, one of the first things that I would do, is to start sketching a rough flow chart with rectangles to represent SAS steps, other rectangles to represent SAS tables and database tables, tape symbols for tape files, disk symbols, and reports. I just used the standard flow chart symbols for each of them, but then I started adding row counts in and out, column counts out, and connected them with lines showing sources and destinations.

The SAS source may be a start, but the SAS log contains much more than just source code. Hopefully enough of the notes and source options are turned on and the log is available for review. If the log is available, I would manually write in row counts for most of the files in and out. If space was an issue, I would estimate used space and perhaps draw a rectangle showing used and available space at each step.

Below is an abbreviated SAS log. Note that lines have been deleted and the line numbers don’t necessarily run consecutively. I have bolded some of the great information that the log contains.
Physical Name: /ssctest/dir

libname filetemp '/ssctest/temp';

NOTE: Libref FILETEMP was successfully assigned as follows:
Engine: V9
Physical Name: /ssctest/temp

172       +    data winfile.cust_info ;
173       +    %let _EFIERR_ = 0; /* set the ERROR detection macro variable */
174       +    infile '/ssctest/dir/cust_info.csv' delimiter = ',' MISSOVER
DSD lrecl=32767 firstobs=2 ;
175       +    informat cust_id_ $31. ;
176       +    informat Account_Number 9. ;
177       +    informat Account_Name $30. ;
178       +    format cust_id_ $31. ;
179       +    format Account_Number $30. ;
180       +    input
181       +    cust_id_ $;
182       +    Account_Number $;
183       +    Account_Name $;
184       +    if _ERROR_ then call symputx('_EFIERR_',1);
185       +    run;

NOTE: The infile '/ssctest/dir/cust_info.csv' is:
Filename=/ssctest/dir/cust_info.csv,
Owner Name=bsweinb,Group Name=sasdev,
Access Permission=rw-r--r--,
Last Modified=Mon Oct 19 09:13:37 2015,
File Size (bytes)=118098274

NOTE: 796781 records were read from the infile '/ssctest/dir/cust_info.csv'.
The minimum record length was 0.
The maximum record length was 222.

NOTE: The data set winfile.cust_info has 796781 observations and 3 variables.
NOTE: DATA statement used (Total process time):
real time 1.99 seconds
cpu time 1.89 seconds

210       +data winfile.cust_info_adj;
211       +length cust_id    $128.;
212       +set winfile.cust_info;
213       +cust_id = lowcase(cust_id_);
214       +run;

NOTE: There were 796781 observations read from the data set winfile.cust_info.
NOTE: The data set winfile.cust_info_ADJ has 796781 observations and 4 variables.
NOTE: DATA statement used (Total process time):
real time 0.76 seconds
cpu time 0.57 seconds

217       +proc sort data=winfile.cust_info_adj;
218       +by cust_id;
219       +run;

NOTE: There were 796781 observations read from the data set winfile.cust_info_adj.
NOTE: The data set winfile.cust_info_ADJ has 796781 observations and 4 variables.
NOTE: PROCEDURE SORT used (Total process time):
real time 1.02 seconds
cpu time 1.64 seconds
data winfile.CONTACTcust_ids;
   %let_EFIERR_ = 0; /* set the ERROR detection macro variable */
   infile '/ssctest/dir/Contactcust_ids.csv' delimiter = ',';
   MISSOVER DSD lrecl=32767 firstobs=2;
   informat First_Name $13.;
   informat Last_Name $12.;
   informat cust_id_ $37.;
   format First_Name $13.;
   format Last_Name $12.;
   format cust_id_ $37.;
   format Contact_Number 10.;
   format Account_Number 9.;
   format Account_Name $31.;
   input First_Name $
                        Last_Name $
                        cust_id_ $
                        Contact_Number 
                        Account_Number 
                        Account_Name $
                        Parent_Account_Number 
                        Parent_Account $
                        Type $
                        Account_Messaging $
                        Service_Team $
                        Mail_Code $
                        createdDate $
                        lastModifiedDate $;
   if _ERROR_ then call symputx('_EFIERR_',1);
run;
NOTE: The infile '/ssctest/dir/Contactcust_ids.csv' is:
Filename=/ssctest/dir/Contactcust_ids.csv,
Owner Name=bsweinb,Group Name=sasdev,
Access Permission=rw-r--r--, Last Modified=Mon Oct 19 09:31:32 2015,
File Size (bytes)=648408736
NOTE: 3724028 records were read from the infile '/ssctest/dir/Contactcust_ids.csv'.
The minimum record length was 104.
The maximum record length was 271.
NOTE: The data set winfile.CONTACTcust_ids has 3724028 observations and 14 variables.
NOTE: DATA statement used (Total process time):
   real time           8.34 seconds
   cpu time            7.83 seconds

   +data winfile.Contactcust_ids_adj;
   +length cust_id    $128.;
   +set winfile.Contactcust_ids ;
   +cust_id = lowcase(cust_id);
   +run;
NOTE: There were 3724028 observations read from the data set winfile.CONTACTcust_ids.
NOTE: The data set winfile.CONTACTcust_ids_ADJ has 3724028 observations and 15 variables.
NOTE: DATA statement used (Total process time):
   real time           4.47 seconds
   cpu time            3.63 seconds

   +proc sort data=winfile.Contactcust_ids_adj;
   +by cust_id;
   +run;
WHY A FLOWCHART?

One of the best training exercises for intermediate students is to display an actual SAS log and then as a group describe what it does and perhaps document it. As students work it out, I would draw on a board flows and libraries with input and output files. I always thought that it would be useful to automatically generate those charts, but the task of connecting all the boxes and overlapping lines seemed like too difficult a problem.

One day it occurred to me, that if I limited the lines connecting steps and files to a single page and if the data ran off the page to use off page connectors, the problem became much simpler. More of this will be discussed later.

SAS Enterprise Guide does a very nice job of showing process flows and if a SAS job is imported, and run, it can show very useful and interactive diagrams showing connections and data flows. That process is much more advanced than what I did with the flow charts and I would love to have a process like that in our system. It does require something more than SAS such as JAVA, and we are basically SAS folks. We do have the underlying data though and could conceivably add a process flow sometime.

One disadvantage of using the EG process flow, is that the job would have to be rerun, and also items like row counts are not included. Our process relies at minimum on the SAS log and doesn’t require much user change if any. We have had surprising good results with the flow chart program in some cases creating charts for hundreds of pages in length.

It is almost automatic documentation to run the logs through the flow charter and include the charts in documentation.

PARSING THE LOGS

An advantage to using the SAS log, is in most cases it is already there and doesn’t require any user program changes. In addition row counts are available and are available nowhere else. PROC SCAPROC does add some additional capabilities that will be addressed later.

There is probably no better tool to parse the SAS log than SAS itself. As part of the SSC SAS Job Performance Tool, we have written SAS jobs that do parse the files and extract job, step, and file information. At first glance it would seem to be a relatively easy program to write, but there are lots of interesting nuances and questions. Even if reading a single job’s log, there are considerable challenges.

1. The log was not intended to be program input, but rather to be read by a human.
2. Things like upper/lower case usage can be inconsistent, different releases alter what is written to the log.
3. Each operating system has differences that have to be dealt with.
4. If SAS/CONNECT remotely submits to another machine, there is effectively a new log in the middle of the old one.
5. If a SAS/GRID is in place, again there are multiple logs to deal with. All of this can complicate even a single job log.

There is also a very extensive logging capture feature, where the SAS installer keeps a copy of all SAS logs without the users doing anything or even knowing about the capture. This is a wonderful source for SAS administrators to review job history, trends, and extract performance information. These logs are much like the normal SAS logs, but also have date stamping and other information preceding the normal log lines. There are also provisions to break long running job’s logs into separate files at midnight or other times, or when a certain file size is reached. The effect of this, is that now there are multiple logs for the same job that have to be related to each other. It actually is a quite complex bit of logic to do this correctly, but we were able to do it. After a few years of using our parsing programs we are getting quite dependable results.

As an aside, a SAS administrator that I know recently had a user who lost three months worth of programming when her SAS enterprise guide project became unreadable. The administrator was able to
go the system captured log, and with a small amount of editing recover her programs. This is not a recommended method of backup, but to quote the user: "I cried when I lost it, and I cried again when you recovered it".

**ADVANTAGES OF PARSING LOGS**

One of the most obvious reasons to parse a log, is that there is a tremendous amount of information that may be needed to read, but the row counts, timing and other good timing information is buried within lots of clutter. A recent log that we were examining for performance was over 50 megabytes and 1.4 million lines of text. Parsing this down to the thousands of steps and files was relatively fast and like all summarizing processes makes things easier to handle. It was very easy to spot the trends, locate longest running steps, and look for trouble spots from the input jobs.

Even though the topic of this paper is graphical, it is the gathering of good underlying data that was the most work and has the most value. The primary tables coming out of the log processing are a job level table, a step level table, and a file level table. The STEP and DATAFILE tables are then merged to create a table that we called JOB_FLOW.

**THE STEPS DATA TABLE**

STEPS contains a row for each SAS step detected in the job. Each step is numbered, line numbers where the step was detected are recorded, a PROC name, Real (elapsed) time), along with CPU time for the steps. Some operating systems and system options allow user times, system time, and excps (number of blocks of data read/written). Steps can easily be sorted and selected to find the most expensive steps in a job or installation.

<table>
<thead>
<tr>
<th>Obs</th>
<th>sas_step_no</th>
<th>Actual_log_line</th>
<th>Sas_log_line</th>
<th>mprocname</th>
<th>rsecs</th>
<th>csecs</th>
<th>u_csecs</th>
<th>s_csecs</th>
<th>excp</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>8</td>
<td></td>
<td>Initialization</td>
<td>0:00:00.03</td>
<td>0:00:00.02</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>16</td>
<td>27</td>
<td>SCAPROC</td>
<td>0:00:00.00</td>
<td>0:00:00.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>80</td>
<td>27</td>
<td>DATASTEP</td>
<td>0:00:01.99</td>
<td>0:00:01.99</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>92</td>
<td>210</td>
<td>DATASTEP</td>
<td>0:00:00.76</td>
<td>0:00:00.57</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>101</td>
<td>210</td>
<td>SORT</td>
<td>0:00:01.02</td>
<td>0:00:01.64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>166</td>
<td>210</td>
<td>DATASTEP</td>
<td>0:00:08.34</td>
<td>0:00:07.83</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>178</td>
<td>268</td>
<td>DATASTEP</td>
<td>0:00:04.47</td>
<td>0:00:03.63</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>188</td>
<td>274</td>
<td>SORT</td>
<td>0:00:04.77</td>
<td>0:00:08.01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>200</td>
<td>278</td>
<td>DATASTEP</td>
<td>0:00:04.14</td>
<td>0:00:03.35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>210</td>
<td>283</td>
<td>SORT</td>
<td>0:00:04.80</td>
<td>0:00:07.01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>236</td>
<td>353</td>
<td>DATASTEP</td>
<td>0:00:00.00</td>
<td>0:00:00.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>246</td>
<td>364</td>
<td>SORT</td>
<td>0:00:00.00</td>
<td>0:00:00.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td>256</td>
<td>368</td>
<td>PRINT</td>
<td>0:00:00.03</td>
<td>0:00:00.01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>13</td>
<td>312</td>
<td>403</td>
<td>DATASTEP</td>
<td>0:00:00.55</td>
<td>0:00:00.49</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>320</td>
<td>403</td>
<td>IMPORT</td>
<td>0:00:03.41</td>
<td>0:00:02.23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>15</td>
<td>332</td>
<td>403</td>
<td>DATASTEP</td>
<td>0:00:00.41</td>
<td>0:00:00.36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>16</td>
<td>345</td>
<td>65</td>
<td>SORT</td>
<td>0:00:00.00</td>
<td>0:00:00.01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>17</td>
<td>359</td>
<td>469</td>
<td>DATASTEP</td>
<td>0:00:00.04</td>
<td>0:00:00.00</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>18</td>
<td>370</td>
<td>476</td>
<td>SORT</td>
<td>0:00:00.03</td>
<td>0:00:00.00</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
THE DATAFILE DATA TABLE

DATAFILE contains a row for each file read or written to in the job. Again, each step is numbered, along with path, type information, file information, but row and column counts if available. Most procs and data steps report rows in and out and variables are reported for output files. For some reason PROC SQL does not display row counts in which is a big drawback. PROC SCAPROC can address some of those issues, but does require source changes.

<table>
<thead>
<tr>
<th>Obs</th>
<th>path</th>
<th>pathtype</th>
<th>engine</th>
<th>sas_step_no</th>
<th>ddn</th>
<th>member</th>
<th>doba</th>
<th>memtype</th>
<th>L_o</th>
<th>dvars</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>/sctest/dir/cust_info.csv</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td>cust_info</td>
<td>796781</td>
<td>RAW</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SAS LIB</td>
<td>2</td>
<td>winfile</td>
<td>cust_info</td>
<td></td>
<td>796781</td>
<td>SAS DS</td>
<td>Out</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>SAS LIB</td>
<td>3</td>
<td>winfile</td>
<td>cust_info</td>
<td></td>
<td>796781</td>
<td>SAS DS</td>
<td>In</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SAS LIB</td>
<td>3</td>
<td>winfile</td>
<td>cust_info_ADJ</td>
<td></td>
<td>796781</td>
<td>SAS DS</td>
<td>Out</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>SAS LIB</td>
<td>4</td>
<td>winfile</td>
<td>cust_info_ADJ</td>
<td></td>
<td>796781</td>
<td>SAS DS</td>
<td>In</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SAS LIB</td>
<td>4</td>
<td>winfile</td>
<td>cust_info_ADJ</td>
<td></td>
<td>796781</td>
<td>SAS DS</td>
<td>Out</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>7</td>
<td>/sctest/dir/Contactcust_idS.csv</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>cust_info</td>
<td>796781</td>
<td>RAW</td>
<td>In</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>SAS LIB</td>
<td>5</td>
<td>winfile</td>
<td>CONTACTcust_idS</td>
<td></td>
<td>3724028</td>
<td>SAS DS</td>
<td>Out</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>9</td>
<td>SAS LIB</td>
<td>6</td>
<td>winfile</td>
<td>CONTACTcust_idS</td>
<td></td>
<td>3724028</td>
<td>SAS DS</td>
<td>In</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SAS LIB</td>
<td>6</td>
<td>winfile</td>
<td>CONTACTcust_idS_ADJ</td>
<td></td>
<td>3724028</td>
<td>SAS DS</td>
<td>Out</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>11</td>
<td>SAS LIB</td>
<td>7</td>
<td>winfile</td>
<td>CONTACTcust_idS_ADJ</td>
<td></td>
<td>3724028</td>
<td>SAS DS</td>
<td>In</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>SAS LIB</td>
<td>7</td>
<td>winfile</td>
<td>CONTACTcust_idS_ADJ</td>
<td></td>
<td>3724028</td>
<td>SAS DS</td>
<td>Out</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>13</td>
<td>SAS LIB</td>
<td>8</td>
<td>winfile</td>
<td>CONTACTcust_idS_ADJ</td>
<td></td>
<td>3724028</td>
<td>SAS DS</td>
<td>In</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>SAS LIB</td>
<td>8</td>
<td>winfile</td>
<td>cust_info_ADJ</td>
<td></td>
<td>796781</td>
<td>SAS DS</td>
<td>In</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>SAS LIB</td>
<td>8</td>
<td>winfile</td>
<td>cust_idS_ALL_SF</td>
<td></td>
<td>3724028</td>
<td>SAS DS</td>
<td>Out</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>16</td>
<td>SAS LIB</td>
<td>9</td>
<td>winfile</td>
<td>cust_idS_ALL_SF</td>
<td></td>
<td>3724028</td>
<td>SAS DS</td>
<td>In</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>SAS LIB</td>
<td>9</td>
<td>winfile</td>
<td>cust_idS_ALL_SF</td>
<td></td>
<td>3724028</td>
<td>SAS DS</td>
<td>Out</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>18</td>
<td>/sctest/dir/exclude.txt</td>
<td></td>
<td>10</td>
<td>WORK SPECIAL_EXCLUDES</td>
<td></td>
<td>2</td>
<td>RAW</td>
<td>In</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>WORK SPECIAL_EXCLUDES</td>
<td>10</td>
<td>SAS DS</td>
<td>Out</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>WORK SPECIAL_EXCLUDES</td>
<td>11</td>
<td>SAS DS</td>
<td>In</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>WORK SPECIAL_EXCLUDES</td>
<td>11</td>
<td>SAS DS</td>
<td>Out</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>WORK SPECIAL_EXCLUDES</td>
<td>12</td>
<td>SAS DS</td>
<td>In</td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

THE JOBFLOW DATA TABLE

JOBFLOW is a simple join of the two earlier files. At this point, we have rough information for each step and file along with whether a file is input or output to the step. This file is adequate for a lot of the performance reports and can be used as a text report to tie out for audit purposes. The beginnings of files going out and into another step are beginning to be apparent, but in order to draw a flowchart, more values need to be computed.
To simplify the first iteration of the flow chart, I did the following:

1. The layout of the flowchart page would have three columns and eight rows.
2. All charts are created as Landscape PDF files.
3. All steps are shown whether they have any input or output files.
4. Input files are normally on the left, output files on the right.
5. If the last output file from one step is used in the very next step, a diagonal arrow is drawn to the next step if possible. This logic was the most time consuming.
6. If a file comes from or goes to a different step on the same page, a “On page” connector is used.
7. If a file comes from or goes to a step on a different page, an “Off Page” Connector is used.
8. Pages are constructed and printed separately, any number of pages is allowed.
9. Row and column counts, timings, line numbers, and the last reference to a file are annotated.

---

**DESIGNING THE FLOW CHART**

To simplify the first iteration of the flow chart, I did the following:

1. The layout of the flowchart page would have three columns and eight rows.
2. All charts are created as Landscape PDF files.
3. All steps are shown whether they have any input or output files.
4. Input files are normally on the left, output files on the right.
5. If the last output file from one step is used in the very next step, a diagonal arrow is drawn to the next step if possible. This logic was the most time consuming.
6. If a file comes from or goes to a different step on the same page, a “On page” connector is used.
7. If a file comes from or goes to a step on a different page, an “Off Page” Connector is used.
8. Pages are constructed and printed separately, any number of pages is allowed.
9. Row and column counts, timings, line numbers, and the last reference to a file are annotated.

---

Page 9 of the sample flow chart is shown below in Output 5.
Output 5

PROCESSING THE JOBFLOW DATA TABLE AND DESIGNING THE FLOW CHART.

Further refining of the JOBFLOW data to create effectively three columns and 8 rows to be drawn on the page is necessary. The file contains the data needed for drawing and annotating the chart. A series of steps basically created various three by eight element arrays to fill each page and then store those results in a file. The logic at times got intense, especially with diagonal line processing.

Along with the names, steps etc. the file contains the row where the symbols will be drawn (TROW), along with INSYMBOL and OUTSYMBOL values set as follows for files:

**INSYMBOL:**
- 'None' if never output earlier,
- 'Conn' if referenced earlier on same page
- 'Diag' if referenced one step back by Outfile set to Diag
- 'Coff' if referenced by earlier page.

**OUTSYMBOL:**
- 'None' if never output later.
- 'Conn' if referenced later on same page
- 'Diag' if referenced one step later in next input file.
- 'Coff' if referenced by later page.

Below is the underlying data to draw page eight.
A CUSTOM FONT

So that I could easily draw flowchart symbols in a variety of sizes, I decided to build a SAS/GRAPH font. I generated all the points need to draw the following: Disk file, off page connector, step box, printed report, SAS table, tape reel, and on page connector. Other symbols can be added as needed.

THE LAST DATASET

The final dataset is a standard SAS/GRAPH annotation dataset that contains the instructions to draw the lines, use the font for symbols, and place all the text on each page. The code is tedious but not especially difficult. The final chart was drawn with PROC GSLIDE and the annotation data set.
Everything produced so far was captured from the SAS log and drawn with SAS/GRAPH. It is not especially elegant or advanced in this day of interactivity and web output, but still the underlying data can in future be sent to a more robust graphical output.

PROC SCAPROC

PROC SCAPROC can be inserted at the beginning of a SAS job and create a text file containing step information and timing, library and file information, macro information, but also column information.

A SAS statement such as `SET;` reads the last dataset built and all of the columns within that dataset. Because PROC SCAPROC has access to the internals of SAS, there is no other easy way to gain column information from the open file which of course might vary widely.

An example of PROC SCAPROC output for the first couple of steps follows.

```sas
/* JOBSPLIT: JOBSTARTTIME 21OCT2015:13:10:08.58 */
/* JOBSPLIT: TASKSTARTTIME 21OCT2015:13:10:08.59 */
/* JOBSPLIT: CATALOG OUTPUT WORK.SASMACR.M000_FILE_INFORMATION.MACRO */
/* JOBSPLIT: Libname WORK V9 'saswrk/SAS_work73AF000063F8_leussa01' */
/* JOBSPLIT: CATALOG INPUT WORK.SASMACR.M000_FILE_INFORMATION.MACRO */
/* JOBSPLIT: Libname WORK V9 'saswrk/SAS_work73AF000063F8_leussa01' */
/* JOBSPLIT: CATALOG OUTPUT WORK.SASMACR.SYSTEMP.MACRO */
```
A fairly easy program to parse the SCAPROC txt file can create tables at the step, file, and now column levels. There is some overlapping information from what was captured from the log and some unique information.
ADVANTAGES TO USING SCAPROC

1. More accurate information in some cases.
2. Files read via PROC SQL are reported.
3. File and column attributes are reported. These values can be used to trace back column lineage.

DISADVANTAGES TO USING SCAPROC

1. Source code must be altered and rerun.
2. Row counts are not reported

COMBINING SCAPROC WITH LOG DATA

It would seem like joining log data with SCAPROC data would give the most information available. We are attempting to do just that, but it is not a trivial join.

The most difficult issue is the SCAPROC sometimes reports steps in a different order than the log does. The whole question of which step elements belong to becomes very tedious. The most common occurrence of this is when a PROC IMPORT is executed, sometimes SAS creates a “Scrubber” data step.
to clear input files. The PROC step is reported as well as the generated DATA step, but in different orders in the log and SCAPROC.

CODE IMPROVEMENTS

Since the underlying data contains indicators that a step is the “Last Reference” of a file, there is the possibility that the file is not needed in the original job. Of course, the file may be stored for archival purposes, or could be referenced by other jobs. Therefore, the program can’t prudently delete code or delete files. It can however generate Proc datasets statements to delete possible unneeded files that the user can manually verify as unneeded, and then manually insert the cleanup code. Other possibilities would be to determine minimum lengths needed for columns, perhaps identify unneeded columns, or even unneeded steps.

It is relatively easy to identify long running steps and point them out to the user, but we have found that quite often a macro may generate a series of individually small steps, that together are repeated thousands of times and overall represent a significant performance degradation that perhaps some pattern detection might point out.

There are virtually unlimited opportunities for programs to examine run time statistics and make or at least suggest improvements automatically.

CONCLUSION

- There is a lot of great run time information in the SAS log and PROC SCAPROC output files.
- A graphical display with annotation can build a pdf of a flow chart easily.
- Other data flow diagrams can be built from the underlying data.
- There is much to be gained for documentation, performance, and data lineage.
- Improvement code can be generated by programs.

RECOMMENDED READING

Previous SAS Global Forum papers by Steven First


CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Steven First
President
Systems Seminar Consultants
608 278-9964
sfirst@sys-seminar.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

Other brand and product names are trademarks of their respective companies.