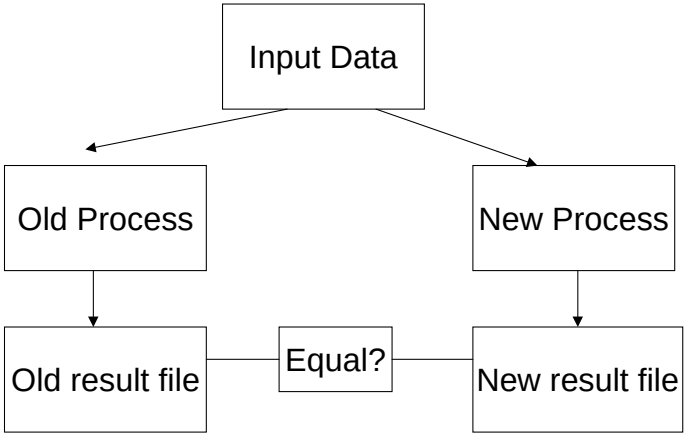
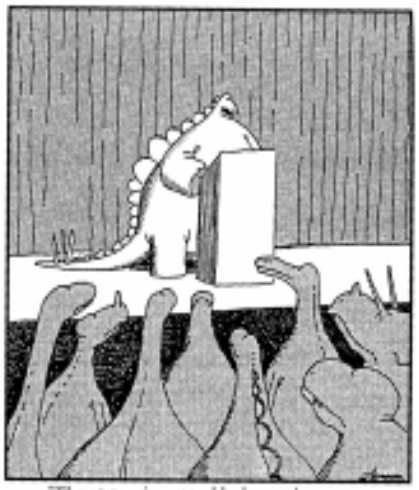
Unusual Uses of SAS



SYSTEMS SEMINAR CONSULTANTS, INC.
Presented by: Steve First
2997 Yarmouth Greenway Drive, Madison, WI 53711
Phone: (608) 278-9964 • Web: www.sys-seminar.com

SUGI ?



"The picture's pretty bleak, gentlemen. ... The world's climates are changing, the mammals are taking over, and we all have a brain about the size of a walnut."

Introduction

- Influences
- Contrarian examples
- Wrap up

What is Wisconsin Famous For?

- Cheese
- Beer
- Bratwurst
- Cold Weather (How cold?)
- Snow plow hockey
- Packer Football
- Illinois tourists
- University of Wisconsin
- Madison
- A hotbed of SAS users
- Not nearly as cold as Minnesota
- Not nearly as boring as Iowa
- ?

Who Said This And When?

"The process of preparing programs for a digital computer is especially attractive because it not only can be economically and scientifically rewarding, it can also be an aesthetic experience much like composing poetry or music."

Donald Knuth - The Art of Computer Programming Volume 1: Fundamental Algorithms (1969)

Influences

- Started as a Math major
- Moved to Computer Science (open book tests!)

- First Comp Sci Professor
- Donald Knuth
- Lemuel Jones
- SAS Authors
- Allen Hershey
- Joe Guepfer
- Gary Larson
- My brothers-in-law
- Many others

Contrarian Uses For SAS

- Industry is constantly changing
- New software all over the place
- Still a shortage of good software tools.
- New programmers know SQL, Web programming
- Many recent projects have different flavor.

Different Uses For SAS

SAS is still the best tool for many applications

Examples:

- Conversion programs with SAS
- Utility applications for programmers
- Auditing other software packages with SAS
- Production reporting systems with SAS.
- Source control with SAS
- Cross referencing data sources with SAS

Conversion Programs

SAS can:

- SAS can read virtually any file format.
- SAS can also write virtually any file format.
- SAS can easily read and write from third party databases
- SAS allows access to system information.
- SAS provides ease of use as a programming language
- There is a general lack of other good tools in the industry.

A Conversion Project

A SAS system of 30 files were being converted to DB2.

- Input files were SAS, so SAS was necessary.
- Contractor was not a SAS user
- Estimated to be less than 40 hours work.

SAS Code Required

The following job reads SAS and adds rows to existing DB2 tables.

```
libname oldsas ‘old.sas.files’;
libname sasdb2 db2 ssid=z1q1 location=z1q2
                 authid='z1q2001$';
data temp;
 set oldsas.xys;
  . . .   change as necessary
run;
proc append base=sasdb2.xysnewtable
              data=temp;
run;
```


A Conversion Project

The results of the project.

- The SAS work to read and write took less than 8 hours .
- The clean-up of data took more than 500 hours.

Notes:

- These techniques can be applied to many conversions.

FILE and PUT Statements

FILE names a NON-SAS file to be written.
PUT moves the data to the file.

Syntax:

FILE '*filespec*' options;
PUT *variables and constants* ;

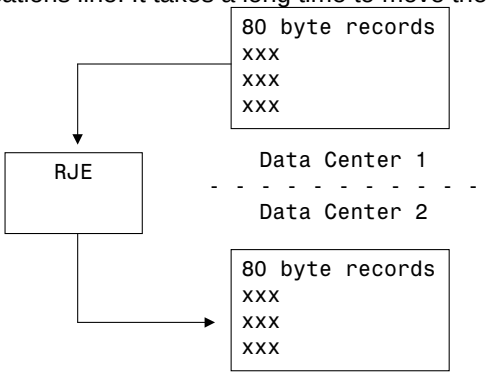
Notes:

- FILE and put have numerous options.
- The output file is NOT a SAS file.
- Care must be taken not to overwrite important data.

My First SAS Job

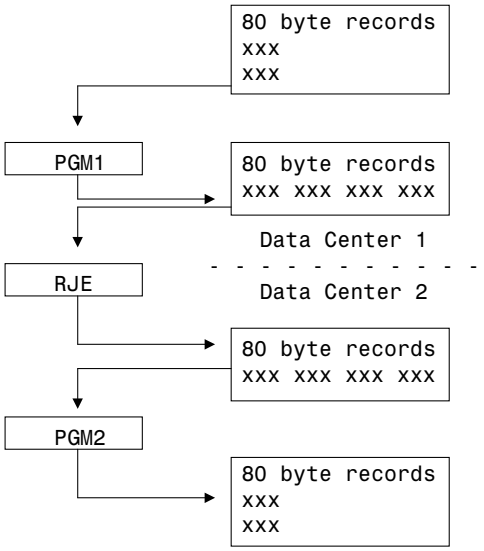
I spent 10 years in Iowa (in a 3 year period)

A data center transfers thousands of mostly blank records down a communications line. It takes a long time to move the data.



A Cobol solution

Write two programs to "pack" and "unpack" data. (8 hours)



A SAS Solution – Two Small Programs

(JCL defined both files with LRECL=80).

```
DATA _NULL_;                              /* DON'T NEED DS     */
  INFILE IN;                              /* RAWFILE IN        */
  FILE OUT;                               /* RAWFILE OUT       */
  INPUT @1 TWENTY $CHAR20.;               /* READ 20 CHAR      */
  PUT      TWENTY $CHAR20. @;             /* 20 OUT, HOLD PTR  */
RUN;                                      /* END OF STEP       */
```

The Reversing Program

Be careful with @@.

```
DATA _NULL_;                              /* DON'T NEED DATASET*/
 INFILE IN;                               /* RAW FILE IN       */
 FILE OUT LRECL=80;                       /* FILEOUT           */
 INPUT                                    /* READ TWENTY/LOOP  */
     TWENTY  $CHAR20.                     /* FIXED PTRS CAUSE  */
    @@ ;                                  /* LOOPS, BE CAREFUL */
 IF TWENTY NE ' ' THEN                    /* IF NONBLANK ?     */
       PUT                                /* OUTPUT 20         */
          @1 TWENTY  $CHAR20.;            /* $CHAR SAVES BLANKS*/
 IF _N_ > 20 THEN                         /* A GOOD IDEA WHILE */
    STOP;                                 /* TESTING           */
RUN;                                      /* END OF STEP       */
```

The Dreaded "Foreign Tape"

PROC TAPELABEL will list:

- DSNAME
- DCB attributes
- block count
- estimated length in feet
- creation date
- expiration date
- creating job
- more.

PROC TAPELABEL under MVS

```
//  EXEC SAS
//TAPE1  DD UNIT=TAPE,VOL=SER=ABCDE,DISP=OLD

PROC TAPELABEL DDNAME=TAPE1;
RUN;
```


Copying Tapes

PROC TAPECOPY copies one or more tapes to a single volume.

TAPECOPY under MVS

```
//  EXEC SAS
//VOLIN DD DSN=first.dsn.on.tape,UNIT=TAPE,VOL=SER=xxxxxx,
//          DISP=OLD
//VOLOUT  DD UNIT=TAPE,VOL=SER=yyyyyy,DISP=(,KEEP)
 PROC TAPECOPY;
 RUN;
```

A Hex Dumping Program

Display any file in a hexadecimal format.

```
DATA _NULL_;                    /* DON'T NEED DATASET */
  INFILE IN;                    /* RAW FILE IN       */
  INPUT;                        /* READ A RECORD     */
  LIST;                         /* LIST BUFFER IN LOG*/
  IF _N_ > 50 THEN              /* STOP AFTER 50     */
     STOP;                      /* ADJUST AS NEEDED  */
RUN;                            /* END OF STEP       */
```


A Copying Program

Simply copy any sequential file.

```
DATA _NULL_;                    /* DON'T NEED DATASET */
  INFILE IN;                    /* RAW FILE IN        */
  FILE OUT;                     /* RAW FILE OUT       */
  INPUT;                        /* READ A RECORD      */
  PUT _INFILE_;                 /* WRITE IT OUT       */
RUN;                            /* END OF STEP        */
```

Changing DCB While Copying

Additional columns will be padded.

```
DATA _NULL_;                    /* DON'T NEED DATASET*/
  INFILE IN;                    /* RAW FILE IN       */
  FILE OUT LRECL=90             /* INCREASE DCB AS   */
           BLKSIZE=9000         /* NEEDED            */
           RECFM=FB;
  INPUT;                        /* READ A RECORD     */
     PUT _INFILE_;              /* WRITE IT OUT      */
   RUN;                         /* END OF STEP       */
```


A Subsetting Program

Select part of a file.

```
DATA _NULL_;                    /* DON'T NEED DATASET  */
  INFILE IN;                    /* RAW FILE IN         */
  FILE OUT;                     /* RAW FILE OUT        */
  INPUT @5 ID $CHAR1.;          /* INPUT FIELDS NEEDED */
  IF ID='2';                    /* WANT THIS RECORD?   */
  PUT _INFILE_;                 /* YEP, WRITE IT OUT   */
RUN;
```

Selecting a Random Subset

Randomly select about 10% of a file.

```
DATA _NULL_;                    /* NO DATASET NEEDED   */
  INFILE IN;                    /* RAW FILE IN         */
  FILE OUT;                     /* RAW FILE OUT        */
  INPUT;                        /* READ A RECORD       */
  IF RANUNI(0) LE .10;          /* TRUE FOR APP. 10%   */
  PUT _INFILE_;                 /* WRITE OUT OBS       */
RUN;                            /* END OF STEP         */
```

Adding Sequence Numbers

Write out a buffer, then overlay.

```
DATA _NULL_;                          /* NO DATASET NEEDED  */
  INFILE IN;                          /* RAW FILE IN        */
  FILE OUT;                           /* RAW FILE OUT       */
  INPUT;                              /* READ A RECORD      */
  SEQ=_N_*100;                        /* COMPUTE SEQ NO     */
  PUT _INFILE_                        /* OUTPUT INPUT REC   */
      @73 SEQ Z8.;                    /* OVERLAY WITH SEQ   */
RUN;                                  /* END OF STEP        */
```

Writing Literals in Every Record

Put 'SSC' in columns 10-12 of every line.

```
DATA _NULL_;                          /* NO DATASET NEEDED  */
  INFILE IN;                          /* RAW FILE IN        */
  FILE OUT;                           /* RAW FILE OUT       */
  INPUT;                              /* READ A RECORD      */
  PUT _INFILE_                        /* OUTPUT INPUT REC   */
      @10 'SSC';                      /* OVERLAY WITH CONST.*/
RUN;                                  /* END OF STEP        */
```


Printing a File with Carriage Control

Report files, microfiche files etc. can be handled.

```
DATA _NULL_;                          /* DON'T NEED DATASET  */
  INFILE IN;                          /* INPUT FILE IN       */
  FILE PRINT NOPRINT;                 /* DON'T ADD CC        */
  INPUT;                              /* READ A RECORD       */
  PUT _INFILE_;                       /* WRITE IT OUT        */
RUN;                                  /* END OF STEP         */
```

Correcting a Field on a File

Logic can be altered to match any situation.

```
DATA _NULL_;                          /* DON'T NEED DATASET  */
  INFILE IN;                          /* INPUT FILE IN       */
  FILE OUT;                           /* OUTPUT FILE         */
  INPUT @5 ID $CHAR1.;                /* INPUT FIELDS NEEDED */
  IF ID='2'                           /* CHANGE AS NEEDED    */
     THEN ID='3';
  PUT   _INFILE_                      /* OUTPUT FILE         */
     @5 ID CHAR1.;                    /* OVERLAY CHANGED ID  */
RUN;                                  /* END OF STEP         */
```


A Cobol Generated Report

A sparse matrix crossing two variables

```
Number of Transactions by Termid and Operator

       Operator
TERMID   0001 0002 0003 0004 . . . 0200

  0001   55
  0002   24         12
  0003                   2
     .
     .
  0150                              1
```

A PROC FREQ Alternative

```
PROC FREQ;
  TABLE TERMID * OPERATOR / LIST;
RUN;
```


Accessing System Control Blocks

The MVS Job File Control Block provides:

- 176 bytes for each DD card
- datasetname
- device type
- catalog status
- SYSIN or SYSOUT status
- label processing options

Possible uses of the JFCB

- Access dataset name from JCL for titles
- Program can determine whether it is reading a live VSAM file, a sequential backup disk file, or a tape file
- May require bit testing
- Determine the date a dataset was created

Possible uses of the JFCB

A JFCB example

Determine the DSNAME and DSORG

```
DATA _NULL_;                          /* DON'T NEED DATASET */
 INFILE IN JFCB=JFCBIN;               /* ASK FOR JFCB       */
 LENGTH TITLDSN $ 44;                 /* SET LENGTHS AS     */
 LENGTH DSORG1 $1.;                   /* REQUIRED           */
 IF _N_ = 1 THEN                      /* FIRST TIME IN ?    */
  DO;                                 /* YES, DO BLOCK      */
   TITLDSN=SUBSTR(JFCBIN,1,44);       /* EXTRACT DSNAME     */
   DSORG1=SUBSTR(JFCBIN,99,1);        /* AND DSORG BYTE 1   */
    IF DSORG1='.1......'B THEN        /* BIT TEST AS NEEDED */
        DSORGOUT='PS';                /* MUST BE SEQUENTIAL */
 END;                                 /* END OF BLOCK       */
 INPUT etc. ;                         /* REST OF PROGRAM    */
  . . .
 RETAIN TITLDSN DSORGOUT;             /* RETAIN             */
RUN;                                  /* END OF STEP        */
```


Programs for Data Managers & Systems Programmers

SAS Sample Programs

- 1000’s of programs available
- Old SAS sample library
- shipped with SAS
- member INDEX prints index
- PDSLIST and SPFMLIST read PDSs
- MAPDISK will read MVS VTOC
- LOADMAP maps load modules
- reading VSAM catalogs

Reading Other Program’s Output

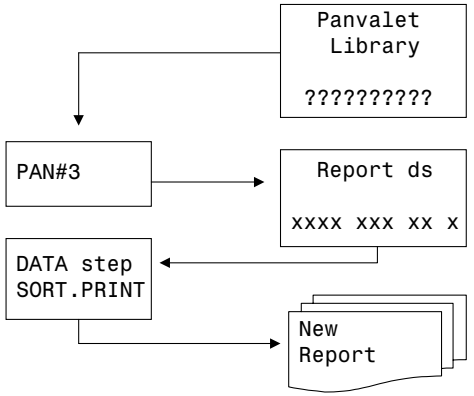
Examples:

- Read VTOC listing programs output
- source management systems
- security systems
- financial packages
- IBM utilities
- virtually any program product

A Panvalet® Example

The user wanted a different sort sequence than PANVALET provided.

Solution: Use PAN#3, then SAS



The Panvalet/SAS Source

```
// EXEC PGM=PAN#3
//*************************************/
//*   USE PAN#3 TO CREATE DIRECTORY    */
//*************************************/
//SYSPRINT DD SYSOUT=*
//SYSPUNCH DD DSN=&&PANDIREC,DISP=(,PASS),
//            UNIT=DISK,SPACE=(TRK,25)
//PANDD1   DD DSN=PANVALET.LIBRARY,DISP=SHR
++CONTROL ....
++PRINT O-UP
//*************************************/
//*   CREATE REPORTS WITH SAS          */
//*************************************/
//SAS   EXEC SAS
//PANDD1   DD DSN=&&PANDIREC,DISP=(OLD,DELETE)
```


The Panvalet/SAS Source

```
DATA PANDIREC;
INFILE PANDD1 ;
INPUT @01 PNPGMNM  $10.
      @11 PNLEVEL  3.
      @19 PNPGMTYP $5.
      @27 PNMTDATE MMDDYY8.
      @35 PNACDATE MMDDYY8.
      @24 PNSTATUS $3.
      @43 PNNOBLK  4.
      @48 PNNOBLK  8.;
RUN;
PROC PRINT DATA=PANDIREC;
  TITLE 'PAN DIRECTORY';
RUN;
```


Programs that Write Other Programs

1. Use SAS to read data, figure logic.
2. Build another program.
3. Submit 2nd program to Internal reader or file.

Applications:

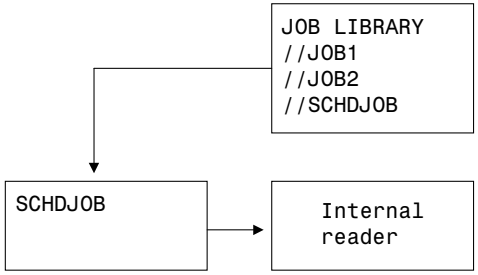
- job schedulers
- disk management programs
- many others

Be careful if deleting data!

Programs that Write Other Programs

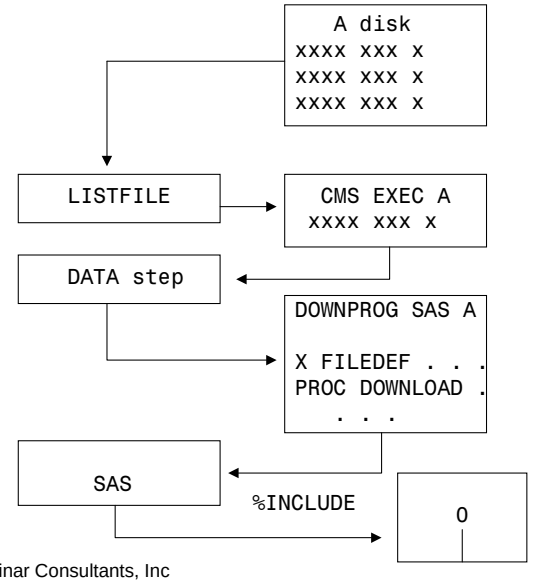
There goes the JES Queue.

A system to automate job submission



Downloading All SAS Programs Under CMS

PC SAS and RESUBMIT automate transfer



The Download Program

```
X LISTFILE  SAS A (EXEC;                        /* CMS COMMAND    */
X FILEDEF FLIST DISK CMS EXEC A;                /* LISTFILE OUTPT */
X FILEDEF GENPRG DISK DOWNPRG SAS A;            /* TEMP PROGRAM   */
DATA _NULL_;                                    /* BUILD FILEDEFS */
 INFILE FLIST;                                  /* PROC DOWNLOADS */
 INPUT @8 FN $8. @17 FT $8.;
 FILE GENPRG;
 PUT 'X FILEDEF HOSTFILE DISK ' FN FT ' L;'
   /
   'PROC DOWNLOAD INFILE=HOSTFILE OUTFILE="C:\DOWNDIR\'
   FN '.SAS";'
    /  'RUN;';
RUN;
%INCLUDE GENPRG;                                /* INC GENED CODE  */
RUN;
```

Notes:

- Similar results can be produced under all platforms.

A Mass Change Program

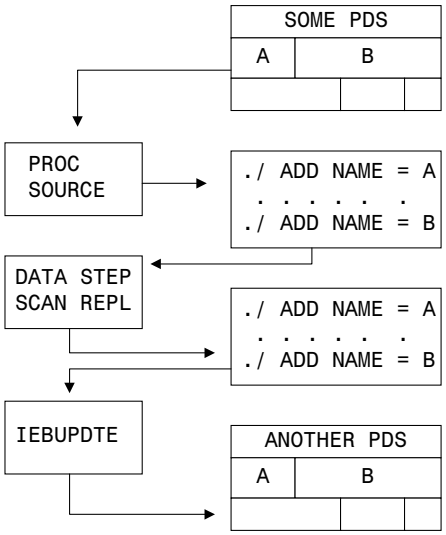
Change all SYSOUT=A to SYSOUT=* in a PROC Library

Issues:

- many PDS members
- production is affected
- change logic may be complex
- back up before, be careful
- LRECL=80, use PROC SOURCE
- sample lib PDSFIND, PDSCHANG

A Mass Change Program (continued)

Mass Change Flow



The Mass Change SAS Code

```
// EXEC SAS
//PDSIN  DD  DSN=SOMEPDS,DISP=SHR
//SEQIN  DD  UNIT=DISK,SPACE=(TRK,500)
//SEQOUT DD  DSN=&&SEQOUT,DISP=(,PASS),
//         UNIT=DISK,SPACE=(TRK,500)

PROC SOURCE INDD=PDSIN OUTDD=SEQIN
 SELECT . . .; / MODIFY AS NEEDED /
RUN;

DATA _NULL_;
  INFILE SEQIN;
  INPUT @1 WHOLE $CHAR80.;
  COL=INDEX(WHOLE,'SYSOUT=A');
  IF COL NE 0 THEN
     SUBSTR(WHOLE,COL,8)='SYSOUT=*';
  FILE SEQOUT;
  PUT @1 WHOLE $CHAR80.;
```


The Mass Change SAS Code (continued)

```
RUN;

// EXEC PGM=IEBUPDTE
//SYSPRINT DD SYSOUT=*
//SYSUT2 DD DSN=SOMEOTHR.PDS,DISP=OLD
//SYSIN  DD DSN=&&SEQOUT,DISP=(OLD,DELETE)
```

Notes:

- Backup before starting, be careful!!

Free Programs

The MAPDISK program reads VTOCs.

```
/*----------- MAPDISK ----------------------------*/
 THIS PROGRAM READS THE DSCBS IN A VTOC AND PRODUCES A LISTING
OF ALL DATA SETS WITH THEIR ATTRIBUTES AND ALLOCATION DATA. THE
VOLUME TO BE MAPPED MUST BE DESCRIBED BY A DISK DD STMT.:
//DISK DD DISP=SHR,UNIT=SYSDA,VOL=SER=XXXXXX
/*------------------------------------------------*/
DATA FREE(KEEP=LOC CYL TRACK TOTAL F5DSCB)
     DSN (KEEP=DSNAME CREATED EXPIRES LASTREF LASTMOD
          COUNT EXTENTS DSORG RECFM1-RECFM4 ALOC BLKSIZE
          LRECL SECALOC TT R TRACKS VOLUME)
     FMT1(KEEP=DSNAME CREATED EXPIRES LASTREF LASTMOD
          COUNT EXTENTS DSORG RECFM1-RECFM4 ALOC BLKSIZE
          LRECL SECALOC TT R TRACKS VOLUME CCHHR)
     FMT2(KEEP=CCHHR TOCCHHR)
     FMT3(KEEP=CCHHR ALLOC3);  LENGTH DEFAULT=4;
```

Free Programs (continued)

```
RETAIN TRKCYL 0;         /* ERROR IF NO FORMAT 4 ENCOUNTERED */
  LENGTH VOLUME VOLSER1 $ 6  CCHHR CCHHR1 $ 5 ;
  FORMAT CCHHR CCHHR1 $HEX10. DSCBTYPE $HEX2. ;

  *-------READ DSCB AND DETERMINE WHICH FORMAT------*;
  INFILE DISK VTOC CVAF CCHHR=CCHHR1 VOLUME=VOLSER1
         COLUMN=COL ;
  INPUT @45 DSCBTYPE $CHAR1. @;  VOLUME=VOLSER1;
  CCHHR=CCHHR1;
  IF DSCBTYPE='00'X THEN DO; NULL+1;
        IF NULL>200 THEN STOP;
        RETURN;  END;  NULL=0;
      IF DSCBTYPE= '1' THEN GOTO FORMAT1;
      IF DSCBTYPE= '2' THEN GOTO FORMAT2;
      IF DSCBTYPE= '3' THEN GOTO FORMAT3;
      IF DSCBTYPE= '5' THEN GOTO FORMAT5;
      IF DSCBTYPE= '4' THEN GOTO FORMAT4;
      IF DSCBTYPE= '6' THEN RETURN;
      _ERROR_=1;RETURN;
FORMAT1:                   *---REGULAR DSCB TYPE--*;
  INPUT @1 DSNAME $CHAR44.
```


An Easier Way

Pre-process with a VTOC reading utility

- most shops have one
- usually simpler, faster
- may have weak reporting

$RSVP followed by SAS

Other VTOC reading programs can be handled similarly.

```
//TMP EXEC PGM=IKJEFT01,DYNAMNBR=30,REGION=1024K
//SYSTSPRT DD SYSOUT=*
//$RSOUT DD  DSN=&&RSVPOUT,DISP=(,PASS),UNIT=DWKO,
//           SPACE=(TRK,50)
//$OUTPUT  DD SYSOUT=*
//SYSTSIN  DD       CHANGE VOLSER OF DISK IN  *****
 $RSVP VOL(VOL001) -
 TRK PRINT (NEW (DSNAME CDATE LDATE DSORG RECFM BLKSZ -
LRECL ALLOC USED USECNT VOLUME MDATE CAT LMTIM))
//STEP02   EXEC SAS
//*******************************************
//* RUN SAS TO GET IT TOGETHER FOR PRINTING
//*******************************************
//SAS.RSOUT   DD DSN=&&RSVPOUT,DISP=(OLD,DELETE)
/*******************************************/
/* BUILD SAS DS WITH INDIVIDUAL DS RECS IN IT
/*******************************************/
```


$RSVP followed by SAS (continued)

```
DATA SASDISK;                          /* DATASET RECORDS     */
 INFILE RSOUT;                         /* FILE FROM $RSVP     */
 INPUT @2 COL2_7 $CHAR6. @ ;           /* DELETE JUNK         */
 IF COL2_7 NE '$RSOA0';                /* HEADERS             */
 IF COL2_7 NE 'DSNAME';                /* MORE                */
 IF COL2_7 NE 'TOTAL ';                /* SUMMARY LINES       */
 INPUT @2  DSN         $CHAR44.        /* DETAIL LINES        */
       @48 CDATE       5.
       @54 LDATE   ?? 5.
       @61 DSORG       $CHAR2.
       @66 RECFM       $CHAR1.
       @72 BLKSIZE     5.
       @78 LRECL       5.
       @87 ALLOC       5.
       @96 USED        5.
      @104 USECNT      6.
      @111 VOLUME     $CHAR6.
      @118 MDATE   ?? 5.
      @125 CAT         $CHAR1.
       . . .
```

Other Utility Programs

Other utilities we have written:

- print mainframe reports
- download and rebuild entire pds to a windows environment including html, source, and graphics..
- Print all programs in a directory
- Shrink a dataset by reducing to smaller variables.
- SAS log and dictionary table queries to automate job flow, performance bottlenecks.

A Possible Use for SAS?

Testing whether fish have feelings

Auditing Other Software Packages

While SAS is great in a production environment, SAS can also be used for one-time, quick comparison and conversion jobs.

Features:

- SAS can read and write most files
- Prototyping and comparison jobs are quick and easy
- PROC COMPARE can compare every part of a SAS dataset
- PROC MEANS/ SUMMARY, FREQ and others can give summary reports and datasets
- Use SAS Data Step to perform more flexible comparison analysis

Overview - Audit Problem # 1

We have purchased a “Black Box” software package upgrade. Our task is to verify that the new version doesn’t change the results we get from our existing version.

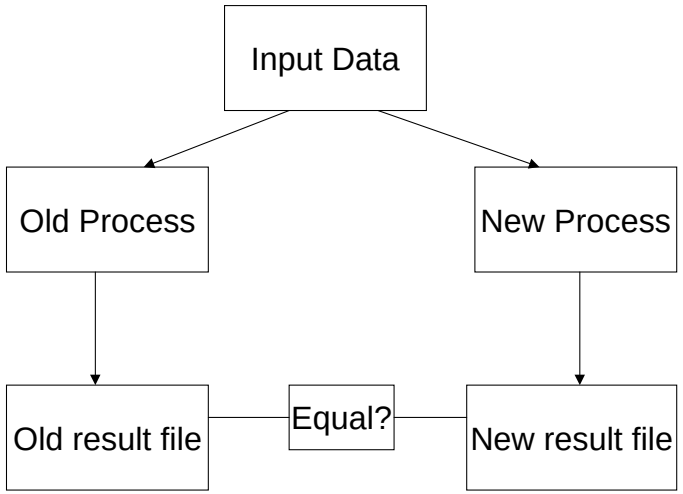
Problem:

We are not allowed to view the source code.

Resources available:

We have a flat file for input and the output files generated from the new and old versions of the system.

Overview



Overview - Audit Solution # 1

Solution:

We use SAS to read the old results file and the new results file and then use PROC COMPARE to spot inconsistent results.

Why use SAS to audit package software?

- SAS can read many different file types
- SAS is easy to use and allows a flexible reporting structure
- SAS procedures simplify tasks (PROC COMPARE)

Verification Audit Code

First, read the old system's results files into SAS using common variable names.

```
Data file old
001 acme            120 5683.87   02032003    Y
002 widgit          10  634.52    02042003    N
```

```
data oldfile;
  format deldt date9.;
  infile old;
  input @1  Custnum   $3.
        @5  compnam   $15.
        @21 orders     3.
        @25 amount     8.
        @35 deldt      mmddyy10.
        @47 rushfl    $1.
        ;
run;
```


Verification Audit Code Continued

Second, read the new system's results files into SAS using common variable names.

```
Data file new
001 acme            15.35 120 5683.87   02032003    Y
002 widgit          12.43 10  634.52    02042003    N
```

```
data newfile;
 format deldt date9.;
 infile new;
 input @1  Custnum   $3.
       @5  compnam   $15.
       @21 unitpr     5.
       @27 orders     3.
       @31 amount     8.
       @40 deldt      mmddyy10.
       @51 rushfl    $1.       ;
run;
```

Verification Audit Code Compare

Next, sort the files by identifying fields. Then use PROC COMPARE, specify how rows are matched (ID), and which variables(VAR) to compare.

```
proc sort data=oldfile;
  by compnam;
run;
proc sort data=newfile;
  by compnam;
run;

Proc compare base=oldfile
             compare=newfile;
  id custnum;
  var Compnam orders amount deldt rushfl;
  Title 'Compare old to new results';
run;
```


Verify Audit Report

```
                Compare old to new results

                    The COMPARE Procedure
Comparison of WORK.OLDFILE with WORK.NEWFILE (Method=EXACT)

                        Data Set Summary
Dataset             Created          Modified   NVar   NObs

WORK.OLDFILE  21FEB03:08:39:02  21FEB03:08:39:02   6      2
WORK.NEWFILE  21FEB03:08:39:03  21FEB03:08:39:03   7      2

                              Variables Summary
            Number of Variables in Common: 6.
            Number of Variables in WORK.NEWFILE but not in
                    WORK.OLDFILE: 1.
            Number of ID Variables: 1.
            Number of VAR Statement Variables: 5.
```


Verify Audit Report Continued

```
                    Observation Summary

            Observation      Base  Compare  ID

            First Obs           1        1  Custnum=001
            Last  Obs           2        2  Custnum=002

    Number of Observations in Common: 2.
    Total Number of Observations Read from WORK.OLDFILE: 2.
    Total Number of Observations Read from WORK.NEWFILE: 2.

Number of Observations with Some Compared Variables
  Unequal: 0.
Number of Observations with All Compared Variables Equal:
  2.
NOTE: No unequal values were found. All values compared are
  exactly equal.
```


Overview Audit Problem #2

We want to verify that the 'Black Box' software is working correctly.

We have a Data Dictionary and system documentation. Our task is to verify that all logic is correctly applied in the black box software.

Problem:

The black box software isn't open for us to review and test code.

Resources available:

Data Dictionary and system documentation of each process rule (General Definition) data files.

Overview Audit Solution #2

Solution:

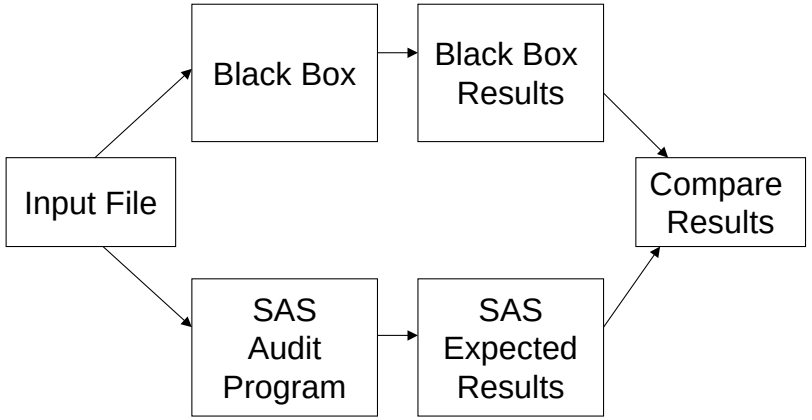
Create parallel process in SAS that duplicates the business rules.

- flexible
- ability to read a wide range of file formats
- easily allows logic to be built
- has a flexible reporting structure.

Now provides an opportunity to compare expected results with actual results.

Overview Flow

Read in the same input file and use the audit program to get the expected results. Then compare the actual black box results with the SAS process expected results.



Parallel Process Code - Read Input File Into SAS

```
Data file original
001 acme            15.35 120 5683.87  02032003   Y
002 widgit          12.43 10  634.52   02042003   N
```

```
data start;
 format deldt date9.;
 infile original;
 input @1  Custnum   $3.
       @5  compnam   $15.
       @21 unitpr     5.
       @27 orders     3.
       @31 amount     8.2
       @40 deldt      mmddyy10.
       @51 rushfl    $1.
       ;
```


Parallel Process Code - Apply Bus. Rules in SAS

```
  if (amount GT 5000
  or  unitpr GT 15)
  AND rushfl='Y' then
     priority='1';
  else
     priority='3';
run;

proc sort data=expected;
  by custnum;
run;
```

Parallel Process Code - Read Actual Results

Data file ACTOUT

```
001 acme                 15 120  5683.87 02/03/2003 Y 1
002 widgit               12  10   634.52 02/04/2003 N 2
```

```
data actual;
 infile actout;
 input  @1  Custnum   $3.
        @5  compnam   $15.
        @21 unitpr     5.
        @27 orders     3.
        @31 amount     8.2
        @40 deldt      mmddyy10.
        @51 rushfl    $1.
        @53 priority  $1. ;
run;
proc sort data=actual;
 by custnum;run;
```


Parallel Process Compare Actual and Expected

```
title 'compare expected to actual';
proc compare base=expected
             compare=actual;
 id custnum;
 var compnam orders amount deldt rushfl priority;
run;
```


Parallel Process Report

```
                compare expected to actual

                  The COMPARE Procedure
         Comparison of WORK.EXPECTED with WORK.ACTUAL
                     (Method=EXACT)

                        Data Set Summary
Dataset           Created          Modified      NVar  NObs
WORK.EXPECTED  24FEB03:07:45:48  24FEB03:07:45:48   8     2
WORK.ACTUAL    24FEB03:07:45:49  24FEB03:07:45:49   8     2

                       Variables Summary

   Number of Variables in Common: 8
   Number of Variables with Differing Attributes: 1.
   Number of ID Variables: 1.
   Number of VAR Statement Variables: 6.
```


Parallel Process Report - Continued

```
   Listing of Common Variables with Differing Attributes

      Variable  Dataset        Type  Length  Format
      deldt     WORK.EXPECTED  Num        8  DATE9.
                WORK.ACTUAL    Num        8

                   Observation Summary
                Observation      Base  Compare  ID
          First Obs           1        1  Custnum=001
          First Unequal       2        2  Custnum=002
          Last  Unequal       2        2  Custnum=002
          Last  Obs           2        2  Custnum=002

       Number of Observations in Common: 2.
  Total Number of Observations Read from WORK.EXPECTED: 2.
  Total Number of Observations Read from WORK.ACTUAL: 2.
```

Parallel Process Report - Continued

```
Number of Observations with Some Compared Variables
  Unequal: 1.
Number of Observations with All Compared Variables Equal:
  1.

                        Values Comparison Summary

Number of Variables Compared with All Observations Equal:5.
Number of Variables Compared with Some Observations
  Unequal: 1.
Total Number of Values which Compare Unequal: 1.
Maximum Difference: 0.

                     Variables with Unequal Values
                Variable  Type  Len  Ndif   MaxDif
                priority  CHAR    1     1
```


Parallel Process Report - Continued

```
                compare expected to actual

                  The COMPARE Procedure
          Comparison of WORK.EXPECTED with WORK.ACTUAL
                     (Method=EXACT)

               Value Comparison Results for Variables


_____________________________________________________________

                    ||  Base Value          Compare Value
          Custnum   ||  priority             priority
          _______   ||  _                    _
                    ||
          002       ||  3                    2

_____________________________________________________________
```


Parallel Process code - Logic Problem

Actual results do not match expected results. An investigation into the black box software discovered a business rule was not applied correctly.

After fixing the rules testing is resumed.

Parallel Process Code - Read Corrected Results

Data file actout

```
001 acme                 15 120  5683.87 02/03/2003 Y 1
002 widgit               12  10   634.52 02/04/2003 N 3
```

```
data actual;
 infile actout;
 input  @1  Custnum   $3.
        @5  compnam   $15.
        @21 unitpr     5.
        @27 orders     3.
        @31 amount     8.2
        @40 deldt      mmddyy10.
        @51 rushfl    $1.
        @53 priority  $1. ;
run;
proc sort data=actual;
 by custnum;run;
```

Parallel Process Compare Actual and Expected

```
title 'compare expected to actual';
proc compare base=expected
             compare=actual;
 id custnum;
 var compnam orders amount deldt rushfl priority;
run;
```


Parallel Process Report

```
                compare expected to actual

                  The COMPARE Procedure
          Comparison of WORK.EXPECTED with WORK.ACTUAL
                       (Method=EXACT)

                       Data Set Summary
Dataset                  Created          Modified  NVar  Obs
WORK.EXPECTED  24FEB03:07:45:48  24FEB03:07:45:48   8     2
WORK.ACTUAL    24FEB03:07:45:49  24FEB03:07:45:49   8     2
                     Variables Summary
           Number of Variables in Common: 8.
           Number of Variables with Differing Attributes: 1.
           Number of ID Variables: 1.
           Number of VAR Statement Variables: 6.
```


Parallel Process Report - continued

```
    Listing of Common Variables with Differing Attributes

            Variable  Dataset        Type  Length  Format
             deldt     WORK.EXPECTED  Num        8  DATE9.
                       WORK.ACTUAL    Num        8

                     Observation Summary
          Observation      Base  Compare  ID
          First Obs           1        1  Custnum=001
          Last  Obs           2        2  Custnum=002

  Number of Observations in Common: 2.
  Total Number of Observations Read from WORK.EXPECTED: 2.
  Total Number of Observations Read from WORK.ACTUAL: 2.
```


Parallel Process Report Continued

```
    Number of Observations with Some Compared Variables
                        Unequal: 0.
Number of Observations with All Compared Variables Equal:
  2.

NOTE: No unequal values were found. All values compared
  are exactly   equal.
```

Types of Audits Performed

Two types of Audit were performed in these cases:

1) verification of results - Determine if old result file is equal to new result file.

- Read both into matching SAS data sets
- use PROC compare

(Allows exclusion of new variables, or non matching values like date/time stamps)

2) Parallel Process.

- Use System Documentation to create a SAS External Audit system.
- Processing the same input files by the same business rules yielding the same results.

What If?

- PROC COMPARE does not meet the required reporting needs
- Data requires manipulation before PROC COMPARE may be used
  - Comparison variables do not have common names across the data sets
  - Conversion of values is needed before the comparison

Solution:

- Use Data Step processes to complete the data transformation
- Use a MACRO to perform the comparison
- Use other SAS procedures (FREQ, SUMMARY, REPORT) to create the required reports

Alternative to PROC COMPARE

A new system is implemented that should give exact duplicate values.
We can create SAS data sets of the two files.

```
data old;
  infile olddd;
   input Id $ X Y Z A $;
run;
proc print data=old;
 Title "Old";
run;
data new;
  infile newdd;
   input Id $ X Y Z A $;
run;
proc print data=new;
 title 'New';
run;
```


The Two Files With Some Differences

Old

Obs	Id	X	Y	Z	A
1	a	1	1	1	a
2	b	2	**2**	2	**b**
3	c	3	3	**3**	c
4	d	4	4	4	d

New

Obs	Id	X	Y	Z	A
1	a	1	1	1	a
2	b	2	**1**	2	**a**
3	c	3	3	**5**	c
4	d	4	4	4	d

Merge The Two Files

```
data both;
 merge old
       new(rename=(x=X_new y=Y_new z=Z_new a=A_new));
  by id;
run;
proc print data=both;
title 'Both';
run;
```


The Merged File

Both

Obs	Id	X	Y	Z	A	X_new	Y_new	Z_new	A_new
1	a	1	1	1	a	1	1	1	a
2	b	2	2	2	b	2	1	2	a
3	c	3	3	3	c	3	3	5	c
4	d	4	4	4	d	4	4	4	d

Notes: PROC MEANS, FREQ, etc. could be run against this data set.

A Macro to Calculate Differences

```
%macro calc_diff(mvar=,mtype=);
 /*************************************************/
 /* macro to calculate difference between two vars if*/
 /* numeric.  if character difference is set to 0 if */
 /* different and 0 if the same.                    */
 /*************************************************/
 %if &mtype=n %then
  %do;
   Variable = "&mvar";
   Diff = &mvar - &mvar._new;
   Old_value = put(&mvar,3.);
   New_value = put(&mvar._new,best3.);
  %end;
 %else
```


The Rest of The Macro

```
%do;
    Variable= "&mvar";
    if &mvar = &mvar._new then
       Diff=0;
    else
       Diff = 1;
    Old_value = &mvar;
    New_value = &mvar._new;
  %end;
output compds;
%mend calc_diff;
```

Calculate Differences and Transpose The Data Set

```
data compds(keep=id variable old_value new_value diff);
 set both;
%calc_diff(mvar=X,mtype=n)
%calc_diff(mvar=Y,mtype=n)
%calc_diff(mvar=Z,mtype=n)
%calc_diff(mvar=A,mtype=c)
run;
proc print data=compds;
Title 'Compds';
run;
```


The Transposed Data Set

Compds

Obs	Id	Variable	Diff	Old_ value	New_ value
1	a	X	0	1	1
2	a	Y	0	1	1
3	a	Z	0	1	1
4	a	A	0	a	a
5	b	X	0	2	2
6	b	Y	1	**2**	**1**
7	b	Z	0	2	2
8	b	A	1	**b**	**a**
9	c	X	0	3	3
10	c	Y	0	3	3
11	c	Z	-2	**3**	**5**
12	c	A	0	c	c

. . . . .

Sort The Data Set And Build A Format

```
proc sort data=compds;
  by variable;
run;
/*********************************************************/
/* create user defined format to apply to diffs in the  */
/* frequency reports.                                    */
/*********************************************************/
proc format;
  value fmt_diff
     low -< -1 = 'lt -1'
     -1        = '-1'
     0         = '0'
     1         = '1'
     1 <- high = 'gt 1'
  ;
run;
```


Count With FREQ, Print a Report

```
proc freq data=compds;
  by variable;
  table diff / noprint missing
               out=freqdiff;
  format diff fmt_diff.;
run;
options nobyline;
proc report data=freqdiff split='*' nowd;
  by variable;
  column diff count percent;
  define diff  / display  'difference*amount' width=10
                 format=fmt_diff.;
  define count  / analysis 'frequency'  width=9  format=8.;
  define percent/ analysis 'percent'    width=9 format=9.2;
  rbreak after / ol dul summarize;
  title "Variable: #byval1";
run;
```

The Resulting Reports

Variable: A

difference amount	frequency	percent
0	3	75.00
1	1	25.00
	4	100.00

Variable: X

difference amount	frequency	percent
0	4	100.00
	4	100.00

The Resulting Reports Continued

Variable: Y

difference amount	frequency	percent
0	3	75.00
1	1	25.00
	4	100.00

Variable: Z

difference amount	frequency	percent
lt -1	1	25.00
0	3	75.00
	4	100.00

Audit Summary

Using SAS to audit package software has several advantages

- allows ‘what - if’ testing outside the product environment.
- ability to use sample files
- repeatable test results
- test actual versus expected results
- report differences

SAS For Reporting And Production

- A wonderful reporting tool
- ODS makes output extremely flexible
- Proc simplify life
- Data step gives flexibility
- Excellent for batch reporting
- Get the IT folks involved with SAS.

But Seriously Folks

"Oh, lovely — just the hundredth time you've managed to cut everyone's head off."

Saving Resources

Focus to SAS conversion

- Use of SAS Views to emulate hierarchies
- Macros to use dictionaries and code
- Efficiencies

Conclusions

- The SAS system is still a very vibrant and flexible system
- Applications are not always where you would expect them
- I would encourage you to look for those “contrarian” applications
- Keep a good sense of humor.

Invoice Printing Program

- Company was outsourcing invoices
- Invoices created from complicated IMS dataset
- Complicated logic to bring data together
- Used data step and ODS to create invoices in SAS
- Company saved over $1 million a year

Invoice Printing Program continued

EXAMPLE:

```
Invoice No.  6884
                                        Invoice Date 01/31/03
                                        Your Order PC 3C000000405
                                        Terms    Due Upon Receipt

       Your Company Here
       Procurement Purchasing Unit
       123 Main Street
       AnyTown, CA 98765

       01/31/03 Workbooks and Manuals                  0.00      0.00
6      01/31/03 MATERIAL RATE                         50.00    300.00
                                                                -------
                                        TOTAL                  $300.00
```


Data Cleaning

Robust logic for cleaning data

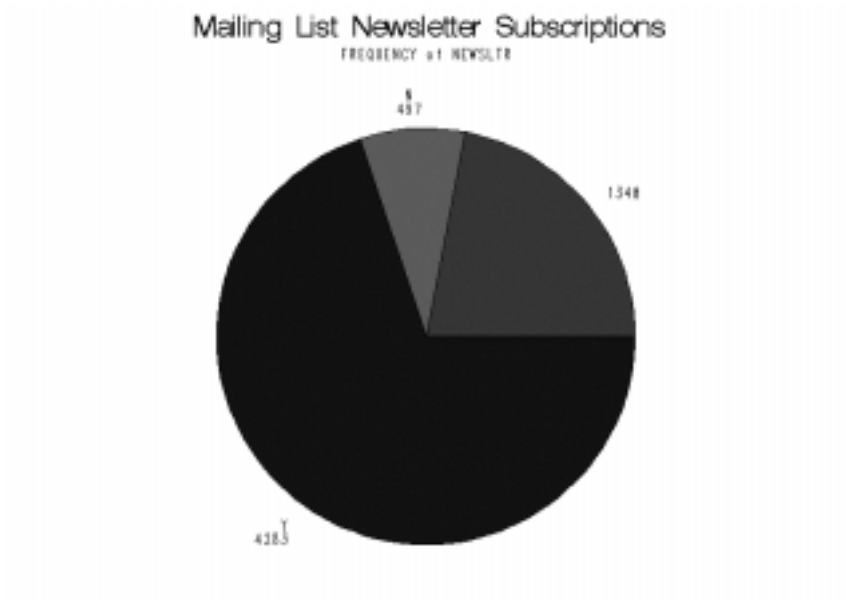
- Great language for finding and reporting on outliers
- Frequencies, plots, STD
- Cleaning data - replacing missing or outlying values

Mailing List Newsletter Subscriptions

The FREQ Procedure

NEWSLTR	Frequency	Percent	Cumulative Frequency	Cumulative Percent
	1348	22.00	1348	22.00
N	497	8.11	1845	30.11
Y	4283	69.89	6128	100.00

Data Cleaning continued



Data Cleaning continued

- Logic to recode values when needed

```
data central.mailinglist;
  set central.mailinglist;
  if newsltr='' then newsletr='N';
run;
```

Emailing From SAS

EMAIL capabilities allow direct communication users from SAS program

- Was job successful?
- What were the errors, if any?
- If job successful, attach report or send link to reports

```
filename emailf email to="sfirst@sys-seminar.com"
              subject="The Report you requested";
ods listing close;                /*close listing dest*/
ods html body=emailf;             /*opens html dest   */
proc means sum data=softsale;     /*means procedure   */
 title 'Sales Summary';           /*title             */
   var sales;                     /*analysis variable */
   class division;run;            /*class by division */
ods html close;                   /*close html dest   */
ods listing;                      /*back to listing   */
```


FTPing Files Across Platforms

The FTP engine allows movement of files across platforms.

- Write output DIRECTLY to web server
- Write flat files directly to other platforms

MVS -> UNIX
MVS -> LAN Server
UNIX -> LAN Server
1 LAN Server -> 2nd LAN Server

and more….!

FTPing Files Across Platforms continued

```
filename myfile ftp 'filename.html'
        cd='/y/mydir/'        /* unix directory      */
        host='host1'          /* host name           */
        user='unxusr'         /* user name           */
        prompt                /* ask for password    */
        rcmd='site umask 022' /* set unix permissions*/
        recfm=s               /* binary data         */
        debug;                /* show messages       */
```


Calling Other Software

Exit SAS to run other software and come back!

Application needed addresses standardized for mailing.

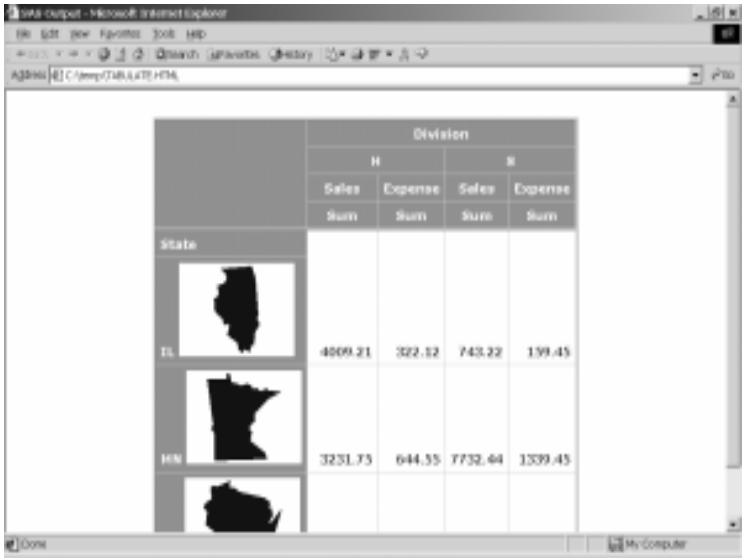
1) Pull desired population
2) Send output to file
3) Use X command to execute outside software to clean addresses
4) Read file created from other software back in
5) Continue with rest of program

Traffic Lighting Problematic Data

SAS can highlight significant values. In this example, total sales of less than $1,000 has a red background.

Embedding Pictures in Reports

Could embed pictures of sales regions, products, and more in reports.



Automating Processes

Many organizations spent too much time each month running reports, manually entering parameters, data, and distributing the results.

- Manually creating reports leaves room for errors / miskeying results
- Wastes time! Let analysts be analysts!
- Use the software that you have paid for!

Automating Processes continued

Let SAS do what SAS does well

- Read data
- Pull different ranges of data depending on system date
- Cleanse the data
- Analyze the data
- Create reports
- Create graphs
- Distribute the reports
- Highlight important information
- Choice of HTML, Excel, PDF, RTF, flat files and more!
- Send emails when complete!
- Automate and schedule