Unusual Uses of SAS

Input Data

Old Process → Old result file

New Process → Equal? → New result file

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.”


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.

```sas
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).

The Reversing Program
Be careful with @@.

## Systems Seminar Consultants, Inc
13 14 15 16
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

```sas
// EXEC SAS
// TAPE1 DD UNIT=TAPE, VOL=SER=A,B,C,D,E, DISP=OLD
PROC TAPELABEL DDNAME=TAPE1;
RUN;
```

Copying Tapes

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

**TAPECOPY under MVS**

```sas
// EXEC SAS
// VOLIN DD DSN=firstdsn.on.tape, UNIT=TAPE, VOL=SER=XXXXX,
// DISP=OLD
// VOLOUT DD UNIT=TAPE, VOL=SER=YYYYY, DISP=(,KEEP)
PROC TAPECOPY;
RUN;
```

A Hex Dumping Program

Display any file in a hexadecimal format.

```sas
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; /* STOP AFTER 50 */
RUN; /* END OF STEP */
```

A Copying Program

Simply copy any sequential file.

```sas
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.

```sas
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.

```sas
DATA _NULL_; /* DON'T NEED DATASET */
INFILE IN; /* RAW FILE IN */
FILE OUT; /* RAW FILE OUT */
INPUT 05 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.

```sas
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.

```cobol
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.

```cobol
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.

```cobol
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.

```cobol
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

<table>
<thead>
<tr>
<th>Operator</th>
<th>TERMID</th>
<th>0001</th>
<th>0002</th>
<th>0003</th>
<th>0004</th>
<th>...</th>
<th>0200</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0002</td>
<td>24</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0003</td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0150</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A PROC FREQ Alternative

```cobol
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

```plaintext
DATA _NULL_; /* DON'T NEED DATASET */
INFILE JFCBIN=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='.' 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

![Diagram showing the process of using PAN#3 and SAS]

---

The Panvalet/SAS Source

```plaintext
// EXEC PGM=PAN#3
// Do not interrupt
// /* USE PAN#3 TO CREATE DIRECTORY */
// #--------------------------#
// /SYSPRINT DD SYSOUT=*
// /SYSPPUNCH DD DSN=&PANDIREC,DISP=(,PASS),
// UNIT=DISK,SPACE=(TRK,25)
// /PANDD1 DD DSN=PANVALET.LIBRARY,DISP=SHR
++CONTROL ....
++PRINT 0-UP
// CREATE REPORTS WITH SAS */
// #--------------------------#
// /SAS EXEC SAS
// /PANDD1 DD DSN=&PANDIREC,DISP=(OLD,DELETE)
```

---

The Panvalet/SAS Source

```plaintext
DATA PANDIREC;
INFILE PANDD1 ;
INPUT @01 PNPGMNM $10.
   @11 PNLEVEL  3.
   @19 PNPJOBYP $5.
   @27 PNMDATE MMDDYY8.
   @35 PNACDATE MMDDYY8.
   @24 PNMSTATUS $3.
   @43 PNNOOBLK 4.
   @48 PNOBLK 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

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

**Notes:**

- Similar results can be produced under all platforms.
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
./ ADD NAME = A
./ . . . .
./ ADD NAME = B

DATA STEP SCAN REPL
./ ADD NAME = A
./ . . . .
./ ADD NAME = B

IEBUPDTE

SOME PDS
A  B

ANOTHER PDS
A  B

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=XXXX
/-----------------*/
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 TCCHHR)
FMT3(KEEP=CCHHR ALLOC3); LENGTH DEFAULT=4;
```
Free Programs (continued)

RETAI1 TRKCY1 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=`4' THEN GOTO FORMAT4;
IF DSCBTYPE=`5' THEN GOTO FORMAT5;
IF DSCBTYPE=`6' THEN RETURN;
_FAILED_=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=* 
//&SOUT DD DSN=&RSVPOUT,DISP=(,PASS),UNIT=DWKO,
//   SPACE=(TRK,50)
//&OUTPUT DD SYSOUT=* 
//SYSTSIN DD CHANGE VOLSER OF DISK IN ***** 
$RSVP VOL(VOLO01) - 
TRK PRINT (NEW (DSNAME CDATE LDATE DSORG RECFM BLKSIZE - 
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 */
   /* FILE FROM $RSVP */
   /* DELETE JUNK */
   /* HEADERS */
   /* MORE */
   /* SUMMARY LINES */
   /* DETAIL LINES */
INFILE RSOUT;
   IF COL2_7 NE '$RSOA0';
   IF COL2_7 NE 'DSNAME';
   IF COL2_7 NE 'TOTAL '; 
INPUT &2 DSN $CHAR44. 
   &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?

A Possible Use for SAS?

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

Auditing Other Software Packages

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

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 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 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 deleted rushfl;
  Title 'Compare old to new results';
run;
```

Verify Audit Report

<table>
<thead>
<tr>
<th>Compare old to new results</th>
</tr>
</thead>
<tbody>
<tr>
<td>The COMPARE Procedure</td>
</tr>
<tr>
<td>Comparison of WORK.OLDFILE with WORK.NEWFILE (Method=EXACT)</td>
</tr>
</tbody>
</table>

Data Set Summary

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Created</th>
<th>Modified</th>
<th>NVar</th>
<th>NObs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK.OLDFILE</td>
<td>21FEB03:08:39:02</td>
<td>21FEB03:08:39:02</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>WORK.NEWFILE</td>
<td>21FEB03:08:39:03</td>
<td>21FEB03:08:39:03</td>
<td>7</td>
<td>2</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Observation</th>
<th>Base</th>
<th>Compare</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Obs</td>
<td>1</td>
<td>1</td>
<td>Custnum=001</td>
</tr>
<tr>
<td>Last Obs</td>
<td>2</td>
<td>2</td>
<td>Custnum=002</td>
</tr>
</tbody>
</table>

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.

Parallel Process Code - Read Input File Into SAS

<table>
<thead>
<tr>
<th>Data file original</th>
</tr>
</thead>
<tbody>
<tr>
<td>001 acme 15.35 120 5683.87 02032003 Y</td>
</tr>
<tr>
<td>002 widgit 12.43 10 634.52 02042003 N</td>
</tr>
</tbody>
</table>

```
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 mmdy10.
    @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

<table>
<thead>
<tr>
<th>Data file ACTOUT</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>001 acme</td>
<td>15 120</td>
<td>5683.87</td>
<td>02/03/2003 Y 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>002 widgit</td>
<td>12 10</td>
<td>634.52</td>
<td>02/04/2003 N 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Created</th>
<th>Modified</th>
<th>NVar</th>
<th>NObs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK.EXPECTED</td>
<td>24FEB03:07:45:48</td>
<td>24FEB03:07:45:48</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>WORK.ACTUAL</td>
<td>24FEB03:07:45:49</td>
<td>24FEB03:07:45:49</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Dataset</th>
<th>Type</th>
<th>Length</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>deldt</td>
<td>WORK.EXPECTED</td>
<td>Num</td>
<td>8</td>
<td>DATE9.</td>
</tr>
<tr>
<td></td>
<td>WORK.ACTUAL</td>
<td>Num</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Observation Summary

<table>
<thead>
<tr>
<th>Observation</th>
<th>Base</th>
<th>Compare</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Obs</td>
<td>1</td>
<td>1</td>
<td>Custnum=001</td>
</tr>
<tr>
<td>First Unequal</td>
<td>2</td>
<td>2</td>
<td>Custnum=002</td>
</tr>
<tr>
<td>Last Unequal</td>
<td>2</td>
<td>2</td>
<td>Custnum=002</td>
</tr>
<tr>
<td>Last Obs</td>
<td>2</td>
<td>2</td>
<td>Custnum=002</td>
</tr>
</tbody>
</table>

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 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

<table>
<thead>
<tr>
<th>Data file actout</th>
<th>001 acme</th>
<th>15</th>
<th>120</th>
<th>5683.87</th>
<th>02/03/2003</th>
<th>Y</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>002 widgit</td>
<td>12</td>
<td>10</td>
<td>634.52</td>
<td>02/04/2003</td>
<td>N</td>
<td>3</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Created</th>
<th>Modified</th>
<th>NVar</th>
<th>Obs</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK.EXPECTED</td>
<td>24FEB03:07:45:48</td>
<td>24FEB03:07:45:48</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>WORK.ACTUAL</td>
<td>24FEB03:07:45:49</td>
<td>24FEB03:07:45:49</td>
<td>8</td>
<td>2</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Variable</th>
<th>Dataset</th>
<th>Type</th>
<th>Length</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>deldt</td>
<td>WORK.EXPECTED</td>
<td>Num</td>
<td>8</td>
<td>DATE9.</td>
</tr>
<tr>
<td></td>
<td>WORK.ACTUAL</td>
<td>Num</td>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

Observation Summary

<table>
<thead>
<tr>
<th>Observation</th>
<th>Base</th>
<th>Compare</th>
<th>ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Obs</td>
<td>1</td>
<td>1</td>
<td>Custnum=001</td>
</tr>
<tr>
<td>Last Obs</td>
<td>2</td>
<td>2</td>
<td>Custnum=002</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Old</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs</td>
<td>Id</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>b</td>
</tr>
<tr>
<td>3</td>
<td>c</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>d</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>d</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>New</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Obs</td>
<td>Id</td>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>3</td>
<td>c</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>d</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>d</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th></th>
<th>Id</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>A</th>
<th>X_new</th>
<th>Y_new</th>
<th>Z_new</th>
<th>A_new</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>a</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>b</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>3</td>
<td>c</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>c</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>d</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>d</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>d</td>
</tr>
</tbody>
</table>

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
  %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;

The Rest of The Macro
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

<table>
<thead>
<tr>
<th>Obs</th>
<th>Id</th>
<th>Variable</th>
<th>Diff</th>
<th>Old_value</th>
<th>New_value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>Y</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>Z</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>a</td>
<td>A</td>
<td>0</td>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>X</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>b</td>
<td>Y</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>b</td>
<td>Z</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>b</td>
<td>A</td>
<td>1</td>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>9</td>
<td>c</td>
<td>X</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>c</td>
<td>Y</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>11</td>
<td>c</td>
<td>Z</td>
<td>-2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>c</td>
<td>A</td>
<td>0</td>
<td>c</td>
<td>c</td>
</tr>
</tbody>
</table>

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 / nopercent
  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

<table>
<thead>
<tr>
<th>Variable: A</th>
</tr>
</thead>
<tbody>
<tr>
<td>difference</td>
</tr>
<tr>
<td>amount</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Variable: Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>difference</td>
</tr>
<tr>
<td>amount</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Variable: X</th>
</tr>
</thead>
<tbody>
<tr>
<td>difference</td>
</tr>
<tr>
<td>amount</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Variable: Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>difference</td>
</tr>
<tr>
<td>amount</td>
</tr>
<tr>
<td>Lt -1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

---

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:

<table>
<thead>
<tr>
<th>Invoice No. 6884</th>
<th>Invoice Date 01/31/03</th>
</tr>
</thead>
<tbody>
<tr>
<td>Your Order PC 3C0000000405</td>
<td></td>
</tr>
<tr>
<td>Terms</td>
<td>Due Upon Receipt</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Invoice Date 01/31/03 Workbooks and Manuals</th>
<th>0.00 0.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01/31/03 MATERIAL RATE</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>NEWSLTR</th>
<th>Frequency</th>
<th>Percent</th>
<th>Cumulative Frequency</th>
<th>Cumulative Percent</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>497</td>
<td>8.11</td>
<td>1845</td>
<td>30.11</td>
</tr>
<tr>
<td>Y</td>
<td>4283</td>
<td>69.89</td>
<td>6128</td>
<td>100.00</td>
</tr>
</tbody>
</table>

Data Cleaning continued

- Logic to recode values when needed

data central.mailinglist;
    set central.mailinglist;
    if newsltr="" then newsltr='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