Mainframe's Basic SAS Programming

Author:

Venkata Rama Rajesh

DB2 DBA for Z/OS

vmallina@in.ibm.com

Objectives

1. Introduction
2. File processing
3. SAS – File allocation
4. SAS Procedures
5. Variables & Arrays
6. Macros & Macro statements
7. SAS – DB2 interface
8. SAS – TSO interface
9. SAS – REXX interface
10. SAS – ISPF interface
11. SAS – Z/OS functions
12. E-mail through SAS

3 days

3 days

Introduction - What is SAS (Statistical Analysis System) ?

The SAS is a programming language used for statistical analysis, It can read in data from common spreadsheets, datasets and databases and output can put in datasets, tables, graphs, and as HTML and PDF documents. The SAS has compilers are available in Windows, Linux, UNIX and Z/OS.

How to execute SAS

```
//SAS01   EXEC SAS
//SYSIN   DD *
OPTIONS NOCENTER;
DATA RAJU;
    YEAR =  YEAR(TODAY());
    MONTH = MONTH(TODAY());
    DAY   = TODAY() ;
PROC PRINT DATA=RAJU;
```

Here the PROC specified at EXEC step will have SAS compiler logic. Which will used to compile and execute the SAS program coded in SYSIN

Introduction - SAS Fundamentals

- Every instruction in SAS program should by ended with semi-colon (;)
- SAS is the combination of set of global statements, DATA and PROC steps.
- We can use data step for file reading , file processing, file writing, computing values for new variables, using SAS buit-in functions and so on.
- The names of data steps are user defined. (except _NULL_ step)
- Some of the uses of PROC step or procedure step is mentioned below
  - Print variables used in data step (PROC PRINT step),
  - Execute SQL statements (PROC SQL step)
  - Execute TSO commands like LISTCAT (PROC IDCAMS step)
  - To sort the data (PROC SORT step)
  - Used to draw graphs/plots (PROC PLOT step)
  - To Histograms and bar charts (PROC CHART step)
- The names of PROC steps are pre-defined.

Introduction - Sample SAS program

```
//SAS01     EXEC SAS
//SYSIN     DD *
DATA RAJU;
      YEAR =  YEAR(TODAY());
      MONTH = MONTH(TODAY());
      DAY   = TODAY() ;
PROC PRINT DATA=RAJU;
```

→ This SAS program has one data step and one PROC step

→ Raju is user-defined name for the data step

→ PROC PRINT step will print all the variables in data step RAJU. PROC PRINT is a special type of PROC step, which will used for print.

PROC PRINT output in FT12F001

```
---+----6----+----7----+----8----+----9----+---10----+---11----+---12----+---1
          THE SAS SYSTEM                          09:45 TUESDAY, OCTOBER 25, 2016

 OBS    YEAR    MONTH     DAY

  1     2016      10     20752
************************************ Bottom of Data ************************************
```

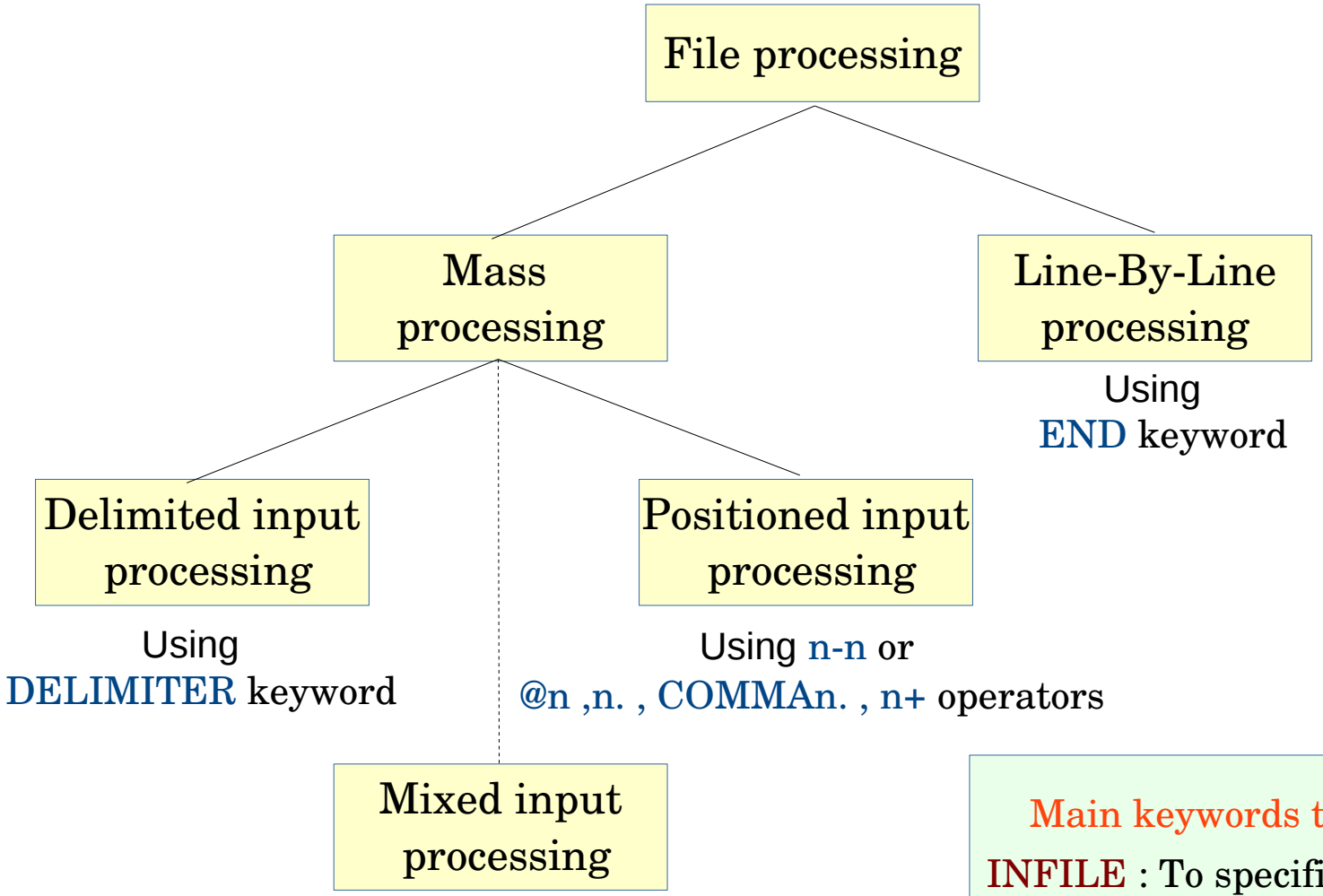
Introduction - SAS JOB output in SPOOL

- This is screen-shot from spool regarding for the submitted SAS job.

```
Cmd DDName   StepName ProcName Que* Step Program  C Destination
___ ________/↑↑↑↑↑↑↑↑ ________ ____ ____ _______ _ ___________
    JESMSGLG                   HLD                W ANYLOCAL
    JESJCL                     HLD                W ANYLOCAL
    JESYSMSG                   HLD                W ANYLOCAL
    FT11F001 SAS01     SAS     HLD     1 SAS      W ANYLOCAL
    FT12F001 SAS01     SAS     HLD     1 SAS      W ANYLOCAL
********************************* Bottom of Data ****************
```

- Here we can see some implicit DD statements (stating with FT%) included by SAS PROC
- Here FT11F001 will have SAS compilation output. If we have any errors in SAS program we can see the error information here.
- Here FT12F001 will have data prints . If we specify any SAS print statements in program.

File processing



Main keywords to know for file processing

INFILE : To specifiy DD name of input dataset

FILE : To specifiy DD name of output dataset

INPUT : To read data from input dataset

PUT : To write data in output dataset

Delimited input processing – Example:1

```
//SAS01     EXEC SAS
//INPUT1    DD     DSN=VX$B5B1.TEST,DISP=SHR
//SYSIN     DD *
 DATA RAJU;
    INFILE INPUT1;
    INPUT NAME $   AGE TEMP;
 PROC PRINT DATA=RAJU;
```

```
INPUT:
----+----1----+--
******************
MALLINA   21 5.92
VENKATA   17 6.12
RAMA   17 6.23
******************
```

```
OUTPUT:
   THE SAS SYSTEM

 OOBS     NAME        AGE     TEMP

 1        MALLINA     21      5.92
 2        VENKATA     17      6.12
 3        RAMA        17      6.23
```

Notes:

→ Here NAME, AGE and TEMP are variables

→ Data read into variables with spaces as delimiter

→ $ specifies that NAME is character variable, without $ symbol SAS can not load char data into variable in readable format.

→ The space b/w character variable name (NAME) and $ is optional

Delimited input processing – Example:2

```
//SAS01     EXEC SAS
//INPUT1    DD     DSN=VX$B5B1.TEST,DISP=SHR
//SYSIN     DD *
 DATA RAJU;
    INFILE INPUT1 DELIMITER='-';
    INPUT NAME $ AGE TEMP COMMA4.;
 PROC PRINT DATA=RAJU;
```

```
INPUT:
----+----1----+--
******************
MALLINA -21-5.92
VENKATA -17-6.12
RAMA -17-6.23
******************
```

```
OUTPUT:
   THE SAS SYSTEM

 OOBS     NAME        AGE     TEMP

 1        MALLINA     21      5.92
 2        VENKATA     17      6.12
 3        RAMA        17      6.23
```

Notes:

SAS will expect integer data into TEMP variable.

If it contains special characters like dot (decimal data) or comma, then it may not read into variable (in certain kind of reads like List input read with delimiter).

COMMA4. refers 4 digit data including special chars like dot and comma. After COMMA4 we should specify dot(.) otherwise COMMA4 considered as another variable name

Positioned input processing – Example:1

```
//SAS01     EXEC SAS
//INPUT1    DD     DSN=VX$B5B1.TEST,DISP=SHR
//SYSIN     DD *
 DATA RAJU;
    INFILE INPUT1;
    INPUT NAME $ 1-7 AGE 10-11  TEMP 13-16;
 PROC PRINT DATA=RAJU;
```

```
INPUT
----+----1----+--
******************
MALLINA   21 5.92
VENKATA   17 6.12
RAMA      17 6.23
******************

OUTPUT:
   THE SAS SYSTEM

 OBS     NAME         AGE     TEMP

 1      MALLINA       21     5.92
 2      VENKATA       17     6.12
 3      RAMA          17     6.23
```

Notes:

→ NAME reads data from position 1 to 7 and AGE reads data from position 10 to 11 from input dataset.. so on

Positioned input processing – Example:2

```
//SAS01     EXEC SAS
//INPUT1    DD     DSN=VX$B5B1.TEST,DISP=SHR
//SYSIN     DD *
 DATA RAJU;
    INFILE INPUT1;
    INPUT @1 NAME $CHAR7. @10 AGE 2.  +2 TEMP COMMA4.;
 PROC PRINT DATA=RAJU;
```

```
INPUT
----+----1----+--
******************
MALLINA   21 5.92
VENKATA   17 6.12
RAMA      17 6.23
******************

OUTPUT:
   THE SAS SYSTEM

 OBS     NAME         AGE     TEMP

 1      MALLINA       21     5.92
 2      VENKATA       17     6.12
 3      RAMA          17     6.23
```

Notes:

In above example

→ @ specifies position number in input file

→ $CHAR7. Or $7. specifies read upto 7 chars

→ 2. specifies read upto 2 numbers

→ +2 refers move 2 positions forward

Mixed input processing – Example

```
//SAS01     EXEC SAS
//INPUT1    DD     DSN=VX$B5B1.TEST,DISP=SHR
//SYSIN     DD *
 DATA RAJU;
    INFILE INPUT1;
    INPUT NAME  @10 AGE 2.  +2 TEMP COMMA4.;
 PROC PRINT DATA=RAJU;
```

```
INPUT
----+----1----+--
******************
MALLINA   21 5.92
VENKATA   17 6.12
RAMA      17 6.23
******************

OUTPUT:
   THE SAS SYSTEM

 OBS     NAME         AGE     TEMP

 1      MALLINA       21     5.92
 2      VENKATA       17     6.12
 3      RAMA          17     6.23
```

Notes:

In above example

→ @ specifies position number in input file

→ 2. specifies read upto 2 numbers

→ +2 refers move 2 positions forward

Line-by-Line processing – Example:1

```
//SAS01     EXEC SAS
//INPUT1   DD     DSN=VX$B5B1.TEST,DISP=SHR
//OUTPUT  DD SYSOUT=*
//SYSIN     DD *
 DATA RAJU;
        INFILE INPUT1 END=DONE;
        FILE OUTPUT;
        DO WHILE (NOT DONE);
            INPUT;
            LINE = RESOLVE(_INFILE_);
            NAME = SUBSTR(LINE,1,7);
            AGE  = SUBSTR(LINE,10,2);
            IF (AGE > 20) THEN STATUS = 'ELIGIBLE';
                        ELSE STATUS = 'NOT ELIGIBLE';
            PUT NAME AGE STATUS;
    END;
 PROC PRINT DATA=RAJU;
```

```
INPUT
----+----1----+--
******************
MALLINA   21 5.92
VENKATA   17 6.12
RAMA      17 6.23
******************

OUTPUT:
  1THE SAS SYSTEM
 MALLINA 21 ELIGIBLE
 VENKATA 17 NOT ELIG
 RAMA 17 NOT ELIG
```

Note:

→ NOT PRINT output this was the data present in OUTPUT dataset

→ FILE keyword is to point output dataset

→ RESOLVE function used for LINE to string convertion

Line-by-Line processing – Example:2

```
//SAS01     EXEC SAS
//INPUT1   DD     DSN=VX$B5B1.TEST,DISP=SHR
//OUTPUT  DD SYSOUT=*
//SYSIN     DD *
 DATA RAJU;
        INFILE INPUT1 END=DONE;
        FILE OUTPUT;
        DO WHILE (NOT DONE);
            INPUT;
            INPUT NAME $1-7 AGE 10-12;
            IF (AGE > 20) THEN STATUS = 'ELIGIBLE';
                    ELSE STATUS = 'NOT ELIGIBLE';
            PUT NAME AGE STATUS;
    END;
 PROC PRINT DATA=RAJU;
```

```
INPUT
----+----1----+--
******************
MALLINA   21 5.92
VENKATA   17 6.12
RAMA      17 6.23
******************

OUTPUT:
  1THE SAS SYSTEM
 MALLINA 21 ELIGIBLE
 VENKATA 17 NOT ELIG
 RAMA 17 NOT ELIG
```

Note:

→ NOT PRINT output this was the data present in OUTPUT dataset

SAS – File allocation

Example:1

```
OPTIONS NOCENTER;
FILENAME RAW 'VX$B5B1.X' DISP=(NEW,CATLG) SPACE=(TRK,(5,1)) RECFM=FB
         LRECL=80 BLKSIZE=800;
DATA RAJU;
     FILE RAW;
     PUT 'HI';
RUN;
```

Example:2

```
OPTIONS NOCENTER;
FILENAME RAW 'VX$B5B1.JCLC' DISP=SHR;
DATA RAJU;
    FILE RAW(TEST);
    PUT 'HI';
RUN;
```

Example:3

```
FILENAME tempfile '&mytemp' ;
data out;
     ...;
run;
```

SAS – Procedures

SAS has many procedures(PROC STEPS) which are portable across platforms and certain procedures which are available only in Z/OS. Example of procedures are shown below.

1. PROC PRINT
2. PROC SORT
3. PROC PLOT
4. PROC GPLOT
5. PROC CHART
6. PROC GCHART
7. PROC IDCAMS
8. PROC PDS
9. PROC PDSCOPY
10. PROC RELEASE
11. PROC SQL

PROC PRINT - Without parameters

```
//SAS01     EXEC SAS
//SYSIN     DD *
DATA RAJU;
      YER =   YEAR(TODAY()-1);
      MONH = MONTH(TODAY()-1);
      DY    = TODAY()-1 ;
DATA AHAN;
      YEAR =   YEAR(TODAY());
      MONTH = MONTH(TODAY());
      DAY    = TODAY() ;
PROC PRINT
```

In this example, If we not specify what DATA step name needs to print in PROC PRINT step then it will print data step just above the PROC PRINT STEP. Specifying Data pram in PROC PRINT Step is recommended

PROC PRINT output in FT12F001

```
---+----6----+----7----+----8----+----9----+---10----+---11----+---12---
          THE SAS SYSTEM                          10:08 TUESDAY, OCTOBER 25

 OBS     YEAR     MONTH      DAY

  1      2016       10      20752
************************************ Bottom of Data ************************************
```

PROC PRINT – VAR statement ; Example:1

```
//SAS01     EXEC SAS
//SYSIN     DD *
DATA RAJU;
      YER =   YEAR(TODAY()-1);
      MONH = MONTH(TODAY()-1);
      DY    = TODAY()-1 ;
DATA AHAN;
      YEAR =   YEAR(TODAY());
      MONTH = MONTH(TODAY());
      DAY    = TODAY() ;
PROC PRINT DATA = RAJU ;
      VAR YER MONH;
```

If we do not want to print all variables in data step, we can choose the variable names needs to print using VAR statement (or VAR parameter) under PROC PRINT step.

PROC PRINT output in FT12F001

```
+----6----+----7----+----8----+----9----+---10----+---11----+---12---
      THE SAS SYSTEM                              10:21 TUESDAY, OCTOBER 25

   OBS      YER     MONH

     1      2016      10
********************************** Bottom of Data **********************************
```

PROC PRINT – VAR statement; Example:2

```
//SAS01     EXEC SAS
//INPUT1  DD   DSN=VX$B5B1.TEST,DISP=SHR
//SYSIN     DD *
DATA RAJU;
     INFILE INPUT1;
     INPUT @1 CITY $2. @6 YEAR $4. @14 STATE $2. ;
PROC PRINT DATA=RAJU;
       TITLE 'REVENUE AND EXPENSE REPORT';
```

INPUT1

```
**************************
CA0002013032 0WB100000.55
CA0002013032 0WB200000.55
MI0002013012 0KA300000.55
MI0002013032 0AP400000.55
CA0002013012 0TN500000.55
CA0002013032 0KA600000.55
RR0002013012 0AP700000.55
ST0002013032 0TN800000.55
**************************
```

PROC PRINT output in FT12F001

REVENUE AND EXPENSE REPORT

OBS	CITY	YEAR	STATE
1	CA	2013	WB
2	CA	2013	WB
3	MI	2013	KA
4	MI	2013	AP
5	CA	2013	TN
6	CA	2013	KA
7	RR	2013	AP
8	ST	2013	TN

PROC PRINT Output Customization

```
PROC PRINT DATA=RAJU;
     WHERE AGE > 20 ;
```

WHERE clause can have relational, AND, OR, IN, BETWEEN operators

NOTE: You can try following parameters also in PROC PRINT

- BY Statement
- ID Statement
- PAGEBY Statement
- SUM Statement
- SUMBY Statement

PROC SORT

```
//SAS01    EXEC SAS
//INPUT1   DD DSN=VX$B5B1.GDG.TEST.G0001V00,DISP=SH
//SYSIN    DD *
OPTIONS NOCENTER NODATE NONOTES NONUMBER;
DATA RAJU;
        INFILE INPUT1;
        INPUT @1 CITY $2. @6 YEAR $4. @14 STATE $2. ;
 PROC SORT DATA=RAJU;
        BY STATE;
 PROC PRINT DATA=RAJU;
```

INPUT1

```
**********************
CA00020130320WB100000.55
CA00020130320WB200000.55
MI00020130120KA300000.55
MI00020130320AP400000.55
CA00020130120TN500000.55
CA00020130320KA600000.55
RR00020130120AP700000.55
ST00020130320TN800000.55
**********************
```

PROC PRINT output in FT12F001

OBS	CITY	YEAR	STATE
1	MI	2013	AP
2	RR	2013	AP
3	MI	2013	KA
4	CA	2013	KA
5	CA	2013	TN
6	ST	2013	TN
7	CA	2013	WB
8	CA	2013	WB

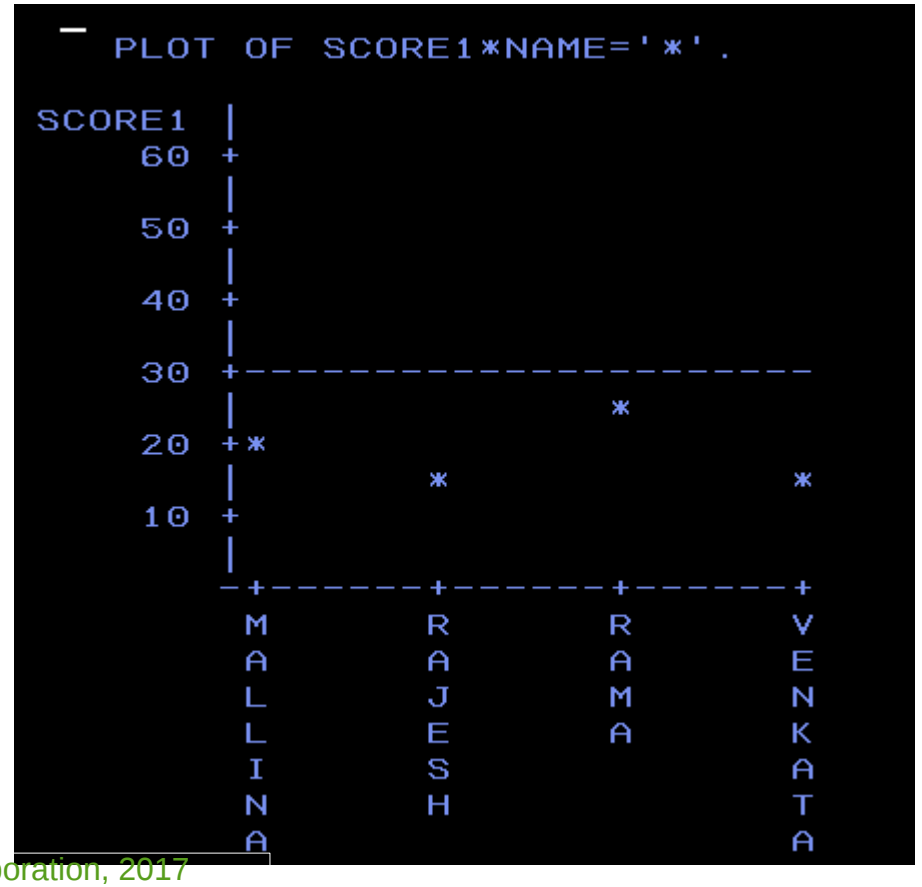
PROC PLOT

```
DATA RAJU;
    INFILE INPUT1;
    INPUT NAME $ SCORE1 SCORE2 ;
PROC PRINT DATA=RAJU;
PROC PLOT DATA=RAJU HPERCENT=25 VPERCENT=25;
    PLOT SCORE1*NAME='*'  vaxis=10 to 60 by 10 vref=30;
```

```
1THE SAS SYSTEM

 OBS     NAME        SCORE1     SCORE2

  1      MALLINA       21         23
  2      VENKATA       17         34
  3      RAMA          27         45
  4      RAJESH        14         56
```



You can also try

- PROC CHART
- PROC GPLOT
- PROC GCHART

PROC IDCAMS

This PROC executes TSO commands written in SYSIN and put output in SYSPRINT.

```
//SAS01    EXEC SAS,OPTIONS='SYSIN=SASIN'
//INPUT1   DD *
VX$B5B1.JCLC
VX$B5B1.JCLC.JCLLIB
//SYSPRINT DD SYSOUT=*
//SASIN    DD *
OPTIONS NOCENTER;
FILENAME SYSIN '&SYSIN1';
FILENAME SYSPRINT '&SYSPRINT';
DATA RAJU;
   FILE SYSIN;
   INFILE INPUT1 END=EOF;
   DO WHILE (NOT EOF);
      INPUT DS $ 1-64;
      OUT = "LISTCAT ALL ENT('"||TRIM(DS)||"')";
      PUT @3 OUT;
   END;
PROC IDCAMS;
RUN;
```

PROC PDS

PROC Pds is to Lists, deletes, or renames members of a partitioned data set

Example:

```
PROC PDS DDNAME='VX$B5B1.A.SAS.JOBS.PROCS';
run;
```

SAS LOG:

```
SAS PROC PDS VERSION 9.20   20100324

  DSNAME=VX$B5B1.A.SAS.JOBS.PROCS  VOL=SER=CTSOCC

  MEMBERS (ALIASES)

  JOB      SASCODE  SASJOB   SASJOBF  SASREXX  TEST     Z1HRPRD  Z1HRPRDA

  TRACKS USED        0.9
         UNUSED     14.1
         TOTAL      15.0
  EXTENTS              1

  DIRECTORY BLKS      10
  BLOCKS USED          2
```

PROC PDS

Example:

```
OPTIONS NOCENTER;
PROC PDS DDNAME='VX$B5B1.A.SAS.JOBS.PROCS';
DELETE TESTD1 TESTD2;
CHANGE TESTR1=TESTRX TESTR2=TESTRY ;
EXCHANGE TESTEX1=TESTEX2;
run;
```

What does it do ?

- DELETE TESTD1 TESTD2 members
- Rename members TESTR1 to TESTRX, TESTR2 to TESTRY
- Exchange data between members TESTEX1 and TESTEX2

PROC PDSCOPY

This was analogous to IEBCOPY z/os utility

Syntax

```
PROC PDSCOPY INDD=file-specification
     OUTDD=file-specification <options >;
EXCLUDE member-name-1 < . . .
       member-name-n>;
SELECT member-name-1 <. . .
        member-name-n >;
```

Example:

```
OPTIONS NOCENTER;
PROC PDSCOPY
      INDD='VX$B5B1.A.REXX.PROCS'
      OUTDD ='VX$B5B1.A.SAS.JOBS.PROCS';
      EXCLUDE JOB JOB1;
run;
```

SAS LOG:

```
INPUT  DSNAME=VX$B5B1.A.REXX.PROCS
       VOL=SER=CTSOCC
 OUTPUT DSNAME=VX$B5B1.A.SAS.JOBS.PROCS
       VOL=SER=CTSOCC
    MEMBER     TRACKS        SIZE
    ALIAS
    ORPHAN       0.5         13360
    ORPHANE      0.7         20080
    SASCODE      0.1          2160 REPLACED
    SASJOB       0.0           480 REPLACED
    SASJOBF      0.2          4320 REPLACED
    SASJOB1      0.0           480
    SASREXX      0.0           560 REPLACED
    SASRX        0.0           240
    SPROC        0.1          1600
WARNING:  NO MEMBERS FOUND TO MATCH RUN
12 THE SAS SYSTEM
    TRACKS USED      2.5
           UNUSED   12.5
           TOTAL    15.0
    EXTENTS          1
NOTE: THE PROCEDURE PDSCOPY USED 0.00 CPU
      SECONDS.
```

PROC RELEASE

Releases unused space at the end of a disk data set

Example:

OPTIONS NOCENTER;

```
PROC RELEASE DDNAME='VX$B5B1.A.REXX.PROCS';
run;
```

SAS LOG:

```
PROC RELEASE   DSNAME=VX$B5B1.A.REXX.PROCS  VOL=SER=CTSOCC

NOTE:  15 TRACKS ALLOCATED, 3 USED, 1 EXTENTS, BEFORE RELEASE.
NOTE:  15 TRACKS ALLOCATED, 3 USED, 1 EXTENTS, AFTER RELEASE.
NOTE: THE PROCEDURE RELEASE USED 0.00 CPU SECONDS.
```

Variables

Different types of variables in SAS

Data step variables / Local variables

Scope of variables are in data step only.

Macro varibiles / Global varibiles

Variables by default will have local scope. If we need global scope variables macro variables are the best solution.

1. System macro variables
2. User-defined macro variables

System macro varibiles

Automatically populated variables while program execution. These variables can put in SYSLOG using %PUT macro statement

- Portable Macro variables
- Z/OS Macro variables

Local variables

INT_VAR and CHAR_VAR are initiated in data step DATAX. So the scope of this Variables are with in this data strp only as shown in below example.

```
OPTIONS NOCENTER NONUMBER;
DATA DATAX;
     FILE OUTPUT1;
     INT_VAR=12345678 ;
     CHR_VAR='12345678';
     PUT INT_VAR ;
     PUT CHR_VAR ;
DATA DATAY;
     FILE OUTPUT2;
     PUT INT_VAR ;
     PUT CHR_VAR ;
```

OUTPUT1

```
----+----1----+--
1THE SAS SYSTEM
 12345678
 12345678
**********************
```

OUTPUT2

```
----+----1----+-
1THE SAS SYSTEM
 .
 .
**********************
```

Portable System Macro variables

Following are some of the MACRO variables

which are portable across the enviorments

```
OPTIONS NOCENTER;
%PUT &SYSJOBID;
%PUT &SYSENV;
%PUT &SYSSCPL;
%PUT &SYSRC;
%PUT &SYSCC;
```

SYSJOBID	VX$B5B1L	JOBID, You cannot change the value of this variable.
SYSENV	BACK	FORE if you are running SAS under TSO. BACK Otherwise. You cannot change the value of this variable.
SYSSCPL	z/OS	Operating system name, You cannot change the value of this variable, You cannot change the value of this variable.
SYSRC	0	contains the RC from the most recent command
SYSCC	0	contains the current SAS condition code that SAS translates into a meaningful RC

Z/OS System Macro variables

Following are some of the MACRO variables which are available only in Z/OS

```
DSNEXST 'VX$B5B1.PROCS.OUT';
%PUT &SYSDEXST;
%PUT &SYSUID;
RUN;
```

SYSDEXST	1	Contains the value that is returned by the DSNEXST statement. SYSDEXST has a value of 1 if the data set specified in the DSNEXST statement is currently available, otherwise 0
SYSUID	VX$B5B1	Contains the value of the TSO user ID that is associated with the SAS session.

User-defined Macro variables

%LET :

This is the global statement used to declare user-defined macro variable.

Syntax : %LET macro-variable =<value>

Ex : %LET G_COUNT =1;

SYMGET

Routine used to read macro variable in data step

Syntax : SYMGET(argument)

Ex : X = SYMGET('G_COUNT');

SYMPUT

Routine used to write macro variable in data step

Syntax : CALL SYMPUT(macro-variable, value);

Ex : CALL SYMPUT('G_COUNT',X);

Example

```
OPTIONS NOCENTER NONUMBER;
%LET G_COUNT =1;
DATA DATAX;
     FILE OUTPUT1;
     X = SYMGET('G_COUNT');
     X= X+1;
     CALL SYMPUT('G_COUNT',X);
     PUT X ;
DATA DATAY;
     FILE OUTPUT2;
     X = SYMGET('G_COUNT');
     PUT X ;
```

LENGTH statement

- Refer slide number 13, instead of wrting NOT ELIGIBLE status variable writing as NOT ELIG (first 8 charcters only). To avoid this we can specify stoarge required for varibale using LENGTH keyword.

```
LENGTH STATUS $25;
```

- For Numeric literal valid length value is 2-8

ARRAY statement

```
Defining Arrays
 array rain {3} janr febr marr;
 array days{7} d1-d7;
 array month{*} jan feb jul;
  array a{10};
Assigning Values
array test{4} t1 t2 t3 t4 (90 80 70 70);
array test{4} t1-t4 (90 80 2*70);
array test2{*} $ a1 a2 a3 ('a','b','c');
```

```
Example:
DATA RAJU;
data test (drop=i);
    array a{10} A01-A10;
    do i=1 to 10;
        a{i}=i;
    end;
run;
proc print noobs data=test;
run;
```

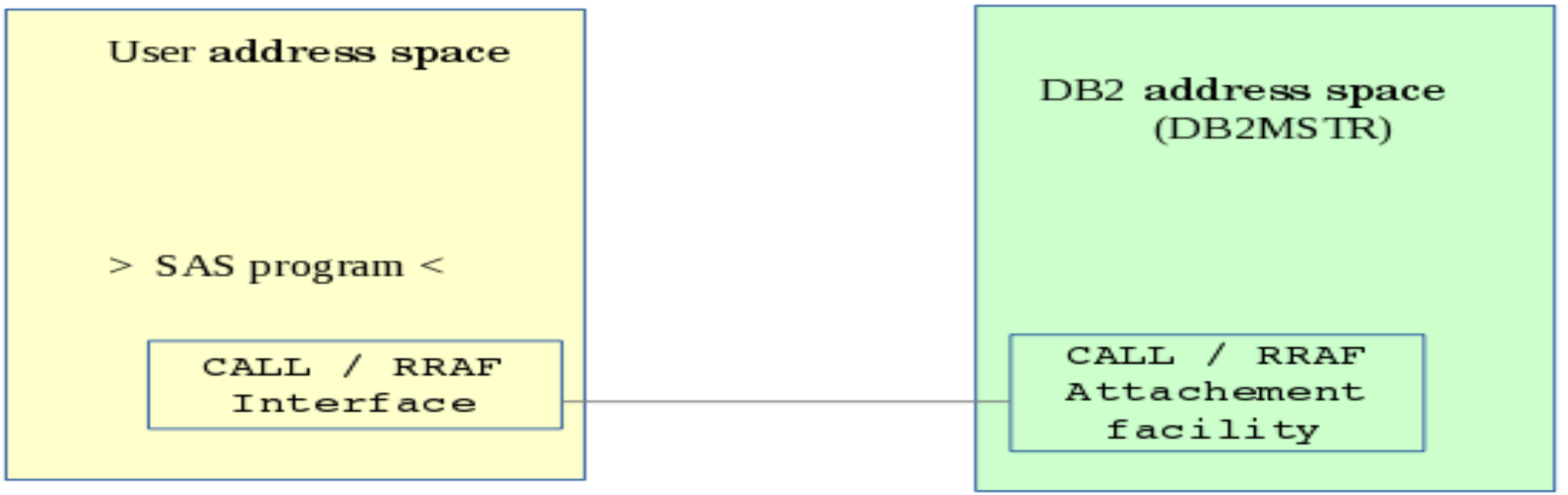
MACRO's

- Macro's should be in beginning of the program
- %MACRO & %MEND are macro demilitors
- & operator used to substitute macro arguments in macro instuctions.
- CALL EXECUTE used to call macro as string.
- While calling macro with variables, use concat operators as shown in below.

```
CALL EXECUTE('%RSEL('||X||','||Y||','||Z||')');
```

```
OPTIONS NOCENTER NONUMBER;
%MACRO RSEL(L,B,H);
 DATA DATAY;
     FILE OUTPUT1 ;
     PUT @2 'WELCOME TO MACRO';
     Z=&L*&B*&H;
     PUT Z;
%MEND RSEL;
DATA DATAX;
    CALL EXECUTE('%RSEL(1,2,3)');
```

SAS – DB2 Interface



Below are the some of ways, in-which we can access DB2 data using above interface.

- The LIBNAME statement
- The MODIFY and UPDATE statements in a DATA step.
- LIBNAME - PROC SQL
- CONNECT TO - PROC SQL
- CONNECT TO - EXECUTE
- Some legacy procedures like ACCESS, DBLOAD, and DBUTIL.

SAS – TSO Interface

Following are different ways to execute TSO in **TSO-SAS session**

S.NO	Method	Syntax
1	SYSTEM function	RC=SYSTEM(cmd)
2	SYSTEM Routine	CALL SYSTEM(cmd);
3	X Statement	X <'cmd'>;
4	TSO Statement	TSO <command>;
5	TSO Routine	CALL TSO(cmd);

```
data _null_;
   rc=system('alloc f(study)da(''userid.my.library'')');
Run;
```

Restriction: Above statements / routines / functions executes successfully only in a TSO SAS session. In a non-TSO session, the command is disabled and the return code is set to 0. '!' refers that instruction has disables features

```
3          TSO ALLOC F(INDD2) DA('VX$B5B1.DDL2') NEW DSORG(PS) SPACE(10 1) CYL
3        !                                                                   ;
```

SAS – REXX Interface

```
//SAS01    EXEC SAS,OPTIONS='SYSIN=SASIN'
//SASREX DD DSN=VX$B5B1.A.SAS.JOBS.PROCS,DISP=SHR
//SYSIN DD *
//INPUT1   DD *
 VX$B5B1.JCLC
 VX$B5B1.JCLC.JCLLIB
 VX$B5B1.LOB.JOBS
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SASIN    DD *
OPTIONS NOCENTER;
OPTIONS REXXMAC REXXLOC=SASREX;
SASREXX;
DATA RAJU;
     PUT 'DONE' ;
RUN;
```

SASREXX member

```
/*REXX*/
SAY 'WELCOME'
```

SAS – REXX Interface (passing variables)

```
//SAS01    EXEC SAS,OPTIONS='SYSIN=SASIN'
//SASREX DD DSN=VX$B5B1.A.SAS.JOBS.PROCS,DISP=SHR
//SYSIN DD *
//INPUT1   DD *
 VX$B5B1.JCLC
 VX$B5B1.JCLC.JCLLIB
 VX$B5B1.LOB.JOBS
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SASIN    DD *
OPTIONS NOCENTER;
OPTIONS REXXMAC REXXLOC=SASREX;
SASREXX 'HI';
DATA RAJU;
     PUT 'DONE' ;
RUN;
```

SASREXX member

```
/*REXX*/
Arg IN
SAY 'WELCOME'
SSY IN
```

SAS – REXX Interface (SAS from REXX)

- `ADDRESS MVS "SUBCOM SAS"` is used to check weather SAS can invoke from SAS or not.
- Subsquent instuctions after `ADDRESS SAS '++SASLOG'` instuction will execute in SAS
- `ADDRESS SAS '++NOLOG'` will terminate SAS session in rexx.

```
/*  REXX  */
ADDRESS MVS "SUBCOM SAS"
SAY "SUBCOM SAS RC:" RC
IF RC = 1
   THEN SAS="NOT "
   ELSE SAS=""
SAY "SAS ENVT. IS "SAS" AVAILABLE"
```

```
/*  REXX  */
ADDRESS SAS '++SASLOG'
"DATA RESULTS; "
"     TOT=100; "
"     RUN;     "
" PROC PRINT;  "
" RUN;         "
```

SAS – ISPF Interface

To invoke ISPEXEC/ ISPLINK from a SAS DATA step, use a CALL statement

Syntax : call ispexec(value1,value2);

```
OPTIONS NOCENTER;
DATA RAJU;
     DSNA='VX$B5B1.A.SAS.JOBS.PROCS';
     CALL ISPEXEC(
     'LMDINIT LISTID(DSLIST) LEVEL(&DSNA)',
     'ISPEXEC LMDFREE LISTID(&DSLIST)'
     );
     RC=ISP_RC;
PROC PRINT DATA=RAJU;
```

```
1          OPTIONS NOCENTER;
2          DATA RAJU;
3               DSNA='VX$B5B1.A.SAS.JOBS.PROCS';
4               CALL ISPEXEC(
5               'LMDINIT LISTID(DSLIST) LEVEL(&DSNA)',
6               'ISPEXEC LMDFREE LISTID(&DSLIST)'
7               );
8               RC=ISP_RC;

NOTE: VARIABLE ISP_RC IS UNINITIALIZED.
ERROR: ISPF IS NOT ACTIVE. THE ISPLINK AND ISPEXEC FUNCTIONS CANNOT BE USED.
ERROR: ISPF IS NOT ACTIVE. THE ISPLINK AND ISPEXEC FUNCTIONS CANNOT BE USED.
```

x=ANYPUNCT(string<,start>)

Searches a string for a punctuation character and returns the first position at which that character is found.

Ex:

```
DATA _NULL_;
     STRING='NEXT = _N_ + 12E3;';
     J=0;
     DO UNTIL(J=0);
        J=ANYPUNCT(STRING,J+1);
        IF J=0 THEN PUT +3 "THAT'S ALL";
        ELSE DO;
           C=SUBSTR(STRING,J,1);
           PUT +3 J= C=;
        END;
     END;
RUN;
```

Output

```
J=6 C==
J=8 C=_
J=10 C=_
J=12 C=+
J=18 C=;
THAT'S ALL
```

NOTE : **NOTPUNCT** function searches a character expression for a character that is not a punctuation character.

Some other string functions

CAT - concatenates character strings without removing leading or trailing blanks

CATS - concatenates character strings and removes leading & trailing blanks

CATT - concatenates character strings and removes trailing blanks

CATX - concatenates strings, removes leading & trailing blanks, and inserts separators

TRIM – Used to trim spaces on both sides

FILEEXIST / DSNCATLGD function

DSNCATLGD or FILEEXIST are functions used to check weather dataset is present or not.

Example

```
DATA _NULL_;
    D1='VX$B5B1.A.REXX.JOBS';
    F1 = DSNCATLGD(D1);
    IF F1 = 1 THEN  PUT 'F1: D1 IS THERE' ;
              ELSE  PUT 'F1: D1 IS NOT THERE';
    F2 = FILEEXIST(D1);
    IF F2 = 1 THEN  PUT 'F2: D1 IS THERE' ;
              ELSE  PUT 'F2: D1 IS NOT THERE';
RUN;
```

Note: DSNCATLGD does not cause dynamic allocation to occur.

Filename function

- Filaname function is used to allocate or deallocate the dataset
- Filanme function is alternative for FILENAME global statement
- fileref in rexx is analogous to DD name in JCL. (In below example MYDIR is the fileref)

```
OPTIONS NOCENTER;
DATA RAJU;
     /* ALLOCATE DIRECTORY */
     RC=FILENAME('MYDIR', 'VX$B5B1.A.SAS.JOBS.PROCS');
     /* DEALLOCATE THE DIRECTORY */
     RC=FILENAME('MYDIR');
RUN;
PROC PRINT DATA=RAJU;
```

FOPEN / FCLOSE / FOPTNUM / FOPTNAME / FINFO – file functions

Directory-id = FOPEN(fileref)

Opens a sequential file and returns a file identifier value

Infocount = FOPTNUM(directory-id)

Returns the number of information items that are available for a file

Option = FOPTNAME(directory-id,i)

Returns the name of a file information item

Option_value =FINFO(directory-id,info-item)

Returns information about a file for given item.

FCLOSE

Close sequential file

FOPEN / FCLOSE / FOPTNUM / FOPTNAME / FINFO – file functions

```
OPTIONS NOCENTER;
DATA RAJU;
     DS='VX$B5B1.SQLOUT';
     OPEN_RC=FILENAME('MYFILE',DS);
     FID=FOPEN('MYFILE');
     /* GET NUMBER OF INFORMATION ITEMS */
     INFOCNT=FOPTNUM(FID);
     /* RETRIEVE INFORMATION ITEMS AND PRINT TO LOG */
     PUT @1 'SEQUENTIAL FILE INFO.';
     DO J=1 TO INFOCNT;
        OPT=FOPTNAME(FID,J);
        OPTVAL=FINFO(FID,UPCASE(OPT));
        PUT @1 OPT @20 OPTVAL;
     END;
     RC=FCLOSE(FID);
     RC=FILENAME('MYFILE');
RUN;
```

Log:

```
SEQUENTIAL FILE INFO
DSNAME          VX$B5B1.SQLOUT
UNIT            3390
VOLUME          CTSOC1
DISP            SHR
BLKSIZE         4096
LRECL           4092
RECFM           VB
CREATION        2015/12/11
```

DOPEN / DCLOSE / DOPTNUM /DOPTNAME / DINFO – file functions

Directory-id = DOPEN(fileref)

Opens a directory and returns a directory identifier value

Infocount = DOPTNUM(directory-id)

Returns the number of information items that are available for a directory

Option = DOPTNAME(directory-id,i)

Returns the name of a directory information item

Option_value =DINFO(directory-id,info-item)

Returns information about a directory for given item.

DCLOSE

Close diectory

DOPEN / DCLOSE / DOPTNUM /DOPTNAME / DINFO – file functions

```
OPTIONS NOCENTER;
DATA RAJU;
     DS='VX$B5B1.A.SAS.JOBS.PROCS';
     OPEN_RC=FILENAME('MYDIR',DS);          /* ALLOCATE DIRECTORY */
     DIRID=DOPEN('MYDIR');                  /* OPEN DIRECTORY */
     INFOCNT=DOPTNUM(DIRID);                /* GET No. OF INFO. ITEMS */
     PUT @1 'INFORMATION FOR A PDS:';
     DO J=1 TO INFOCNT;
        OPT=DOPTNAME(DIRID,J);
        OPTVAL=DINFO(DIRID,UPCASE(OPT));
        PUT @1 OPT @20 OPTVAL;
     END;
     DIR_RC=DCLOSE(DIRID);
     CLOSE_RC=FILENAME('MYDIR');
RUN;
PROC PRINT DATA=RAJU;
```

Log:

```
INFORMATION FOR A PDS:
DSNAME       VX$B5B1.SAS.JOBS
UNIT         3390
VOLUME       CTSOCC
DISP         SHR
BLKSIZE      800
LRECL        80
RECFM        FB
CREATION     2016/12/10
```

Routines

SLEEP(...) routine:

Suspends the execution of a program that invokes this call routine for a specified period of time.

SYSTEM(...) routine / function:

Submits an operating system command for execution.

TSO(...) routine / function:

Issues a TSO command

CALL WTO(...) routine / function:

Sends a message to the system console.

Examples to Send email from SAS

```
FILENAME MAILBOX EMAIL 'venkataramarajesh@gmail.com'
         SUBJECT='TEST MAIL MESSAGE';
DATA _NULL_;
FILE MAILBOX;
PUT "HELLO";
PUT "THIS IS A MESSAGE FROM THE DATA STEP";
RUN;
```

Log:

```
NOTE: THE FILE MAILBOX IS:
      E-MAIL ACCESS DEVICE
MESSAGE SENT
      TO:          "venkataramarajesh@gmail.com"
      CC:
      BCC:
      SUBJECT:     TEST MAIL MESSAGE
      ATTACHMENTS:
NOTE: 2 RECORDS WERE WRITTEN TO THE FILE MAILBOX
```

Examples to Send email from SAS

Example:1

```
FILENAME MAILBOX EMAIL 'venkataramarajesh@gmail.com'
          SUBJECT='TEST MAIL MESSAGE';
DATA _NULL_;
     FILE MAILBOX;
     PUT "HELLO";
     PUT "MGE FROM THE DATA STEP";
 RUN;
```

Example:2

```
FILENAME Mailbox EMAIL;
DATA _NULL_;
FILE Mailbox TO='vmallina@in.ibm.com'
            SUBJECT='Test Mail message';
PUT "Hello";
PUT "This is a message from the DATA step";
RUN;
```

Examples to Send email from SAS

Example:1

```
FILENAME MAILBOX EMAIL 'venkataramarajesh@gmail.com'
          SUBJECT='TEST MAIL MESSAGE';
DATA _NULL_;
     FILE MAILBOX;
     PUT "HELLO";
     PUT "MGE FROM THE DATA STEP";
 RUN;
```

Example:2

```
FILENAME Mailbox EMAIL;
DATA _NULL_;
FILE Mailbox TO='vmallina@in.ibm.com'
            SUBJECT='Test Mail message';
PUT "Hello";
PUT "This is a message from the DATA step";
RUN;
```

Examples to Send email from SAS

Example:3

```
FILENAME Mailbox1 EMAIL;
FILENAME Mailbox2 EMAIL;
DATA _NULL_;
FILE MailBox1 TO=('arjunramesh@in.ibm' 'vmallina@in.ibm.com')
     CC='rajesh.mvr62@gmail.com'
     BCC='vmallina@in.ibm.com'
     REPLYTO='My anonymous address <vmallina@in.ibm.com>'
     SUBJECT='SSC ID';
     ATTACH='VX$B5B1.GDG.TEST.G0001V00';
PUT "Hi ARJUN";
PUT "Thanks, I will check with you on monday";
PUT " ";
PUT "Thanks & Regards";
PUT "Rajesh";
FILE MailBox2 TO='vmallina@in.ibm.com'
     SUBJECT='Third addressee';
PUT "Hello Mailbox2 ";
RUN;
```

Examples to Send email from SAS

Example:4

```
FILENAME Mailbox1 EMAIL;
DATA _NULL_;
FILE MailBox1 TO='vmallina@in.ibm.com'
              CC='rajesh.mvr62@gmail.com'
              BCC='vmallina@in.ibm.com'
              REPLYTO='<vmallina@in.ibm.com>'
              SUBJECT='TEST MAIL'
              ENCODING='UNICODE';
              ATTACH=('VX$B5B1.GDG.TEST.G0001V00' Ext='xls'
                      'VX$B5B1.GDG.TEST.G0001V00' Ext='ODS'
                      'VX$B5B1.GDG.TEST.G0001V00' Ext='PDF') ;
PUT "Thanks, I will check with you on monday";
PUT " ";
PUT "Thanks & Regards";
PUT "Rajesh";
RUN;
```

Bibliography

https://support.sas.com/documentation/cdl/en/hosto390/61886/PDF/default/hosto390.pdf

http://analytics.ncsu.edu/sesug/2005/PS04_05.PDF

http://www2.sas.com/proceedings/forum2008/038-2008.pdf