PROC PUTSPEK, A SAS TO SPEAKEASY INTERFACE

John W. Davison, Jr., Federal Reserve Board

Speakeasy is an extendable computer language based, like APL, upon the concept of arrays and matrices as natural entities, but without the terseness of APL notation. Like SAS, Speakeasy may be extended by any FORTRAN program through the use of standard linkages, and over several years the combined efforts of the Speakeasy user community have created a very extensive vocabulary of mathematical and statistical operators and an effective modeling system. Its ease of use, particularly in the interactive environment, and the extent of its vocabulary make Speakeasy a useful complement to SAS in many situations. Such complementary use requires a convenient interface for the movement of data between the two systems. It is of course easy to read and not too difficult to create Speakeasy objects in a SAS DATA step, given the SAS formatting capability. The latter, however, requires knowledge and programming ability beyond the naivete which both systems are meant to satisfy. This paper describes the SAS procedure PUTSPEK to create a Speakeasy CHECKPOINT data set which may be retrieved by the Speakeasy RESTORE statement.

SPEAKEASY

Speakeasy is a computer language designed to provide users with a simple and effective means to enter data into a computer and to manipulate and transform it and display the result. While easy to use in a batch environment, the language and its underlying structure are oriented toward interactive use, and it is in the interactive mode that it is most powerful as an extension of the user's intellect.

The structure of the language itself is similar to that of APL, without the terseness of notation and strict right to left parsing which characterize the latter. Like APL, Speakeasy operates on arrays but also distinguishes between singly or doubly subscripted arrays on the one hand and scalars, vectors, and matrices on the other. Such objects are stored in the allocator, working storage under control of the Speakeasy processor. The allocator is two sections, one containing control information reflecting the current state of the processor, and the other containing all currently defined objects in a linked list structure. Each object is preceded by a three doubleword header containing its name, type, and structure as well as the list pointers. The list structure permits dynamic creation and freeing of data objects.

Many single character APL operators are represented in Speakeasy by named functions. Consequently, a Speakeasy input line is longer than a comparable APL input line and mixes prefix and infix notation, but is more readily suggestive of the intended result to the reader who is not especially attuned to mathematical notation.

Like SAS and unlike APL, Speakeasy is an extendable language. While it is possible and often desirable to write Speakeasy programs, new Speakeasy functions called linkules may be written as function subprograms, usually in FORTRAN, and their load modules stored in a library from which they are dynamically linked to the Speakeasy processor when invoked by name, like native Speakeasy functions. Linkules use a standardized calling sequence to communicate with the Speakeasy processor for access to arguments and the allocator. The linkule facility permits Speakeasy to be extended by the addition of functions designed from scratch, and by the inclusion of existing programs by making them subroutines to a driver function incorporating the standardized linkule calling sequence. In this way, Speakeasy has acquired many of the features of packages such as EISPACK, and an extensive modeling and simulation facility developed at the Federal Reserve Board.

INTERFACE REQUIREMENTS

It is unnecessary to expound the capabilities of SAS to this audience; however, SAS does not offer the convenience of a "super desk calculator", as does Speakeasy. On the other hand, Speakeasy can be much less efficient than SAS in many situations in which user interaction is not really required. Frequently the most effective approach is the use of either Speakeasy or SAS for data entry and preliminary transformations, followed by the use of SAS procedures, and then further interactive processing with Speakeasy. Each system includes a supervisor that manages its own data storage and interacts with the operating system, so it is not practical either to run Speakeasy as a SAS PROC or SAS as a Speakeasy linkule. Such a style of operation therefore requires alternate use, with a convenient interface to move the data from one system to the other.

It is trivial to read Speakeasy data with SAS. A Speakeasy object, whether scalar, vector, matrix, or array, can be output to a partitioned data set by the KEEP command. In this storage it is represented in 80 character records by a header of two doublewords containing its structure, followed by the data elements as floating point doublewords. To read this into SAS it is necessary only to override the DCB in the INFILE statement as shown. In the JCL:

```
//DD1 DD DSN=SPEAKEZ.DATA(X),DISP=SHR
```

where LRECL=80. The SAS DATA statement should read:

```
DATA; INFILE DD1 LRECL=8 FIRSTOBS=3;
      INPUT X1 RB8.;
```

Speakeasy matrices and doubly subscripted arrays are stored in row major order, so several

related objects of the same type could be stored as columns of the same array or matrix and read with a single INPUT statement. The structure information could be read as IB4. fields of the second doubleword to determine the number of columns to be read. Clearly, no special interface is required to read a single Speakeasy object into SAS.

On the other hand, it is not practical to read SAS datasets with Speakeasy directly, and to design a linkule to do so seemed excessively cumbersome, considering the output capabilities of SAS. One of the useful features of Speakeasy is its CHECKPOINT command, to save the current status of the system and an image of the allocator for subsequent recovery by a RESTORE command. Inspection of the structure of a CHECKPOINT dataset suggested that the simplest approach would be to develop a SAS PROC to create a dataset in the image of a CHECKPOINT, which Speakeasy could load by a RESTORE action. The result is PROC PUTSPEK, described below.

PROC PUTSPEK

The procedure PUTSPEK was developed to create a data set which is modeled upon an actual Speakeasy CHECKPOINT data set. The control information of the data set is an exact copy of the corresponding lines of a CHECKPOINT, differing only in the fields containing the date and time of creation and the count of doublewords in the data section. One other field contains a coded form of the Speakeasy level number. This must be modified in the source code and recompiled when a new release of Speakeasy is installed. The CHECKPOINT data sets used as a model for those created by the procedure were taken immediately after initial entry to Speakeasy during a TSO session, so this is the state to which the control information will be set by the RESTORE statement.

After the fixed control information is output to the CHECKPOINT, the entire SAS data set is brought into main memory. Because the word count field is only a halfword integer, the total number of doublewords in this part of the CHECKPOINT, including headers, is limited to 32767. The lengths of the variables in each observation are determined to calculate the offset of each in the work area. For each, a header is created specifying the name and describing the object as a real column vector, and the individual items are concatenated with their headers and each other. The actual transfer is done by using offset pointers to pick the elements from their respective observations and move them to their positions in an 80 character buffer; each time this is filled, a record is written to the output data set.

PROC PUTSPEK options and parameters;

Output a SAS data set in the image of a Speakeasy CHECKPOINT. The following options are accepted, with defaults as indicated.

DATA= data set name

Use DATA= to specify the SAS data set to be processed. If omitted, the most recently created SAS data set is used.

DDNAME= ddname
DD= ddname
DSN= ddname

The required parameter DDNAME is the file name allocated to the sequential data set 'userid.dsname.SPEKSAVE', or a member of the partitioned data set 'userid.SPEAKEZ.DATA', to receive the CHECKPOINT image. If the data set is sequential, recover data in Speakeasy by 'RESTORE name', and by 'KEPTALL name' if it is in SPEAKEZ.DATA. The data set should be allocated with LRECL=80; one way to ensure that it exists in correct form is to enter Speakeasy and issue a 'CHECKPOINT name' or 'KEEPALL name' statement and then allocate the resulting data set or member to the desired ddname.

NOMISS

The NOMISS option is not yet implemented, but is intended for use if you want to include only those observations with no missing values.

Procedure Information Statements

VAR variables;

List the variables to be included in the CHECKPOINT data set. Both numeric and alphabetic variables may be processed. If omitted, all variables are output.

```
 /*-------------------------------------------------------------------+
 |  PROC PUTSPEK: CREATES A SPEAKEASY CHECKPOINT FROM A SAS DATASET.  |
*+-------------------------------------------------------------------+
*|               PARSING MODULE FOR PROC PUTSPEK                     |
*+-------------------------------------------------------------------+
         PRINT NOGEN
PROC     SASPROC NAME=PUTSPEK,LOADMOD=PUTSPEK2,DEFLIST=1,                *
              DEFMODE=ALPHANUM
LISTS    SASLIST VARIABLES,1,VARIABLE,1,VAR,1,MODE=ALPHANUM
PARMS    SASLIST DDNAME,1,DD,1,DSN,1,MODE=ALPHANUM
OPTS     SASLIST NOMISS,1
         SASEND
         END
 +-------------------------------------------------------------------+
 |                      THE PROCEDURE PROPER                         |
 +-------------------------------------------------------------------+
 |      DECLARE BUILTIN AND INTERNAL PROCEDURES                      |
 +-------------------------------------------------------------------*/
 PUTSPEK: PROC OPTIONS(MAIN);
 DCL (CEIL,DATE,SUBSTR,SUM,TIME,UNSPEC) BUILTIN;
 PUTCARD:  PROC; WRITE FILE(SPKSAVE) FROM(CARD0); IX=0; RETURN; END;
 ERRMSG: PROC (NR);
 DCL MSG(4) CHAR(48)
           INITIAL(' ERROR: OUTPUT DDNAME REQUIRED.',
                   ' ERROR: CHECKPOINT LIMITED TO 32767 DOUBLEWORDS.',
                   ' ERROR: SPEKSAVE FILE IS UNDEFINED.',
                   ' NOTE: PROC PUTSPEK CREATED A CHECKPOINT.'),
     MSGLEN(4) FIXED BIN(31) INITIAL(31,48,35,41),
     MESG(6) FLOAT(12), NR FIXED;
 DO I=1 TO 6; UNSPEC(MESG(I))=UNSPEC(SUBSTR(MSG(NR),8*(I-1)+1,8)); END;

 CALL SASLOG(MESG(1),MSGLEN(NR)); STOP; END;
 /*-------------------------------------------------------------------+
 |      DECLARE LINKAGE TO SAS ROUTINES                              |
 +-------------------------------------------------------------------*/
 DCL GETMEM  ENTRY(FIXED BIN(31),FIXED BIN(31),PTR,FIXED BIN(31));
 DCL INPUT   RETURNS(FIXED BIN(31));
 DCL IOPT    ENTRY(FIXED BIN(31)) RETURNS(FIXED BIN(31));
 DCL MEMERR  ENTRY(FIXED BIN(31));
 DCL NAMES   ENTRY(FIXED BIN(31),FIXED BIN(31),FLOAT BIN(53),
                   FIXED BIN(31),FIXED BIN(31));
 DCL NOVAR   ENTRY(FIXED BIN(31)) RETURNS(FIXED BIN(31));
 DCL PARM    ENTRY(FIXED BIN(31)) RETURNS(FLOAT BIN(53));
 DCL SASLOG  ENTRY(FLOAT(12),FIXED BIN(31));
 DCL VAR     ENTRY(FIXED BIN(31),FIXED BIN(31),FLOAT BIN(53));
 DCL VARY    ENTRY(FIXED BIN(31),FIXED BIN(31),FLOAT BIN(53));
 DCL (SASPLO,OBSERR,VARERR) ENTRY;
 /*-------------------------------------------------------------------+
 |      DECLARE DYNAMIC STRUCTURES                                   |
 +-------------------------------------------------------------------*/
 DCL VNAME(1) FLOAT BIN(53) BASED(NPTR);
 DCL LENTH(1) FIXED BIN(31) BASED(LPTR);
 DCL VTYPE(1) FIXED BIN(31) BASED(TPTR);
 DCL VALUE(1) FLOAT BIN(53) BASED(VPTR);
 /*-------------------------------------------------------------------+
 |      DECLARE INTERNAL STRUCTURES                                  |
 +-------------------------------------------------------------------*/
 DCL SPKSAVE FILE RECORD OUTPUT;
 DCL (NPTR,LPTR,TPTR,VPTR) PTR;
 DCL TYME CHAR(16) INIT((16)'0'), DAYT CHAR(8) INIT((8)' ');
 DCL CHECK FLOAT BIN(53) INIT(0);
 DCL (I,ISW,NBYTES,MBYTES) FIXED BIN(31) INIT(0);
 DCL (IX,J,JX,K,KX,LENRW,NVAR,NWDI) FIXED BIN(15) INIT(0);
```

```
/*-------------------------------------------------------------------+
|  DESCRIPTOR CARD, WITH TIME STAMP AS 'HH:MM XM' IN COL 9-16,      |
|  DATE AS 'MM/DD/YY' IN COL 17-24, AND NWORDS IN COLS 79-80.       |
+-------------------------------------------------------------------*/
DCL 1 CARD1,
        2 ID FIXED BIN(31) INIT(-12369856),
        2 RELEASE FIXED BIN(31) INIT(3131000),     /* LEVELS NU, NU+ */
        2 NOW CHAR(16) INIT((16)' '),                /* SET BY PROC */
        2 BLNK16 CHAR(16) INIT((16)' '),
        2 P(10) FIXED BIN(15)    INIT(7,168,6,48,46,44,56,376,0,0),
        2 BLNK18 CHAR(18) INIT((18)' '),
        2 NWDS FIXED BIN(15) INIT(0);                /* SET BY PROC */
/*-------------------------------------------------------------------+
|  CARDS REPRESENTING CHECKPOINT CONTROL INFORMATION                 |
+-------------------------------------------------------------------*/
DCL   CARD2(20) FIXED BIN(31)
        INIT(1,2,0,1,1,1,1,348,1,30,0,1,0,1,0,67108864,0,258,50,1);
DCL   CARD3(20) FIXED BIN(31)
        INIT(0,0,-681158351,-677363216,83776,
         -681158638,315911,-673181696,-1861169080,1192284848,
           1091623096,1090574304,319883794,318718448,214896,
            -681143807,-674215920,-1068347376,-668413439,-1069623258);

DCL   CARD4(20) FIXED BIN(31) INIT(1168168322,1206966116,(18)0);
DCL   CARD5(20) FIXED BIN(31)
        INIT(0,5,1197432336,975893277,1646093097,-238855003,
          529228081,1002093794,975893277,0,529228081,1002093794,(8)0);

DCL   CARD6(20) FIXED BIN(31)INIT((13)0,2,574536,(3)0,1342177280,0);
DCL 1 CARD7,
        2 P(6) FIXED BIN(31) INIT((6)0),
        2 CHECKPOI CHAR(8) INIT('CHECKPOI'),
        2 DDNAME CHAR(8) INIT((8)' '),               /* SET BY PROC */
        2 BLNK16 CHAR(16) INIT((16)' '),
        2 TRACEBAC CHAR(16) INIT('TRACEBACON      '),
        2 V(2) FIXED BIN(31) INIT(2,1);
DCL   CARD8(20) FIXED BIN(31) INIT(17,18,(18)0);
DCL   CARD9(20) FIXED BIN(31) INIT((20)0); /* 10 AND 11 ARE THE SAME */
DCL   CARD0(10) FLOAT BIN(53) INIT((10)0);
/*-------------------------------------------------------------------+
|  ACTUAL DATA CHECKPOINT STRUCTURE. FIRST TWO DOUBLEWORDS CONTAIN   |
|  DESCRIPTOR INFORMATION, WITH COUNT IN BYTES 3-4 AND 13-14.        |
|  HEADER FOR FIRST OBJECT STARTS IN BYTE 17, DATA IN BYTE 41.       |
+-------------------------------------------------------------------*/
DCL 1 CHECKHD,
        2 WORD1(4) FIXED BIN(15) INIT(-192,0,0,21),
        2 WORD2(4) FIXED BIN(15) INIT(23,1,0,1);
DCL 1 OBJCTHD,
        2 POINTRS,
            3 BPTR FIXED BIN(15) INIT(0),
            3 FPTR FIXED BIN(15) INIT(0),
            3 LEVL FIXED BIN(15) INIT(10),
            3 PRTY FIXED BIN(15) INIT(0),
        2 DSCRPTR,
            3 KIND FIXED BIN(15) INIT(0),
            3 KLAS FIXED BIN(15) INIT(0),
            3 NRWS FIXED BIN(15) INIT(0),
            3 NCLS FIXED BIN(15) INIT(0);
/*-------------------------------------------------------------------+
|   1. SET SAS ENVIRONMENT                                           |
|   2. DETERMINE NUMBER OF VARIABLES: FATAL ERROR IF DATASET EMPTY   |
|   3. DETERMINE IF NOMISS OPTION SET (ACTION NOT YET IMPLEMENTED)   |
|   4. DETERMINE OUTPUT DATASET: FATAL ERROR IF NOT NAMED            |
|   5. ALLOCATE STORAGE FOR VARIABLE DESCRIPTIONS                    |
+-------------------------------------------------------------------*/
/*1*/ CALL SASPLO;
/*2*/ NVAR=NOVAR(1); IF NVAR=0 THEN DO; CALL VARERR; STOP; END;
/*3*/ ISW=IOPT(1);
/*4*/ CHECK=PARM(1);
      IF CHECK=0 & UNSPEC(CHECK)^='0'B THEN CALL ERRMSG(1);    /*EXIT*/
      UNSPEC(DDNAME)=UNSPEC(CHECK);
```

```
/*5*/ NBYTES=8*NVAR;
/* NAMES */  CALL GETMEM(NBYTES,MBYTES,NPTR,0);
      IF NBYTES~=MBYTES THEN DO; CALL MEMERR(NBYTES-MBYTES); STOP; END;
      NBYTES=4*NVAR;
/* TYPE */   CALL GETMEM(NBYTES,MBYTES,TPTR,0);
      IF NBYTES~=MBYTES THEN DO; CALL MEMERR(NBYTES-MBYTES); STOP; END;
/* LENGTH */ CALL GETMEM(NBYTES,MBYTES,LPTR,0);
      IF NBYTES~=MBYTES THEN DO; CALL MEMERR(NBYTES-MBYTES); STOP; END;
/*-------------------------------------------------------------------+
|      FOR EACH VARIABLE, DETERMINE NAME, NCLS, AND KIND             |
/*-------------------------------------------------------------------*/
LENRW=0;
DO I=1 TO NVAR;
   CALL NAMES(1,I,VNAME(I),VTYPE(I),LENTH(I));
   IF VTYPE(I)=1 THEN VTYPE(I)=2; ELSE VTYPE(I)=6;
   LENTH(I)=CEIL(LENTH(I)/8.);
   LENRW=LENRW+LENTH(I);
END;
/*-------------------------------------------------------------------+
|    ALLOCATE STORAGE DATASET AND COPY VALUES INTO WORKING STORAGE    |
|    NOTE: 32767 DOUBLEWORD LIMIT IMPOSED BY SPEAKEASY               |
|    NOMISS OPTION NOT YET IMPLEMENTED; SHOULD COME IN THIS SECTION  |
+-------------------------------------------------------------------*/
NBYTES=262043;
CALL GETMEM(NBYTES,MBYTES,VPTR,0);
IF NBYTES~=MBYTES THEN DO; CALL MEMEPR(NBYTES-MBYTES); STOP; END;
NRWS=0; NWDS=3*NVAR; J=1;
DO WHILE(INPUT=0);
     NRWS=NRWS+1;
     NWDS=NWDS+LENRW; IF NWDS<0 THEN CALL ERRMSG(2);          /*EXIT*/
     DO I=1 TO NVAR;
        IF LENTH(I)=1 THEN DO;
            CALL VAR(1,I,VALUE(J));
            IF VALUE(J)=0 & UNSPEC(VALUE(J))~='0'B THEN VALUE(J)=-1E41;
            J=J+1;          END;
                    ELSE DO;
            DO K=0 TO LENTH(I)-1;
              UNSPEC(VALUE(J+K))=UNSPEC('        ');
            END;
            CALL VARY(I,I,VALUE(J));
            J=J+LENTH(I);  END;
END; END;
IF NRWS=0 THEN DO; CALL OBSERR; STOP; END;
WORD1(2),WORD2(3)=NWDS;
/*-------------------------------------------------------------------+
|      COMPLETE AND OUTPUT THE CHECKPOINT CONTROL CARDS (1-11)       |
+-------------------------------------------------------------------*/
TYME=TIME();
DAYT=DATE();
NOW=SUBSTR(TYME,1,2) || ':' || SUBSTR(TYME,3,2) || ' ON' ||
    SUBSTR(DAYT,3,2) || '/' || SUBSTR(DAYT,5,2) || '/'   ||
    SUBSTR(DAYT,1,2);
ON UNDEFINEDFILE (SPKSAVE) CALL ERRMSG(3);                  /*EXIT*/
OPEN  FILE (SPKSAVE) TITLE(DDNAME);
WRITE FILE (SPKSAVE) FROM (CARD1);
WRITE FILE (SPKSAVE) FROM (CARD2);
WRITE FILE (SPKSAVE) FROM (CARD3);
WRITE FILE (SPKSAVE) FROM (CARD4);
WRITE FILE (SPKSAVE) FROM (CARD5);
WRITE FILE (SPKSAVE) FROM (CARD6);
WRITE FILE (SPKSAVE) FROM (CARD7);
WRITE FILE (SPKSAVE) FROM (CARD8);
WRITE FILE (SPKSAVE) FROM (CARD9);
WRITE FILE (SPKSAVE) FROM (CARD9);
WRITE FILE (SPKSAVE) FROM (CARD9);
```

```
/*-------------------------------------------------------------------+
|   1. BUILD THE CHECKPOINT DATA HEADER FIELDS                       |
|   2. OUTPUT EACH OBJECT IN TURN TO THE CHECKPOINT RECORDS          |
|      A. BUILD THE OBJECT HEADER FIELDS                             |
|      B. OUTPUT RECORDS TO CHECKPOINT DATASET AFTER EVERY TENTH ITEM |
|      C. MOVE DATA ITEMS                                            |
+-------------------------------------------------------------------*/
/*1*/ UNSPEC(CARD0(1))=UNSPEC(WORD1(1)) || UNSPEC(WORD1(2)) ||
                       UNSPEC(WORD1(3)) || UNSPEC(WORD1(4));
      UNSPEC(CARD0(2))=UNSPEC(WORD2(1)) || UNSPEC(WORD2(2)) ||
                       UNSPEC(WORD2(3)) || UNSPEC(WORD2(4));
      FPTR=NRWS*LENTH(NVAR)+3;
      IX=2; KX=1;
/*2*/ DO I=1 TO NVAR;
/*A*/    NCLS=LENTH(1);
         BPTR=FPTR;
         FPTR=NRWS*NCLS+3;
         KIND=VTYPE(I);
         IF NCLS>1 THEN KLAS=6; ELSE KLAS=5;
         IX=IX+1; UNSPEC(CARD0(IX))=UNSPEC(BPTR) || UNSPEC(FPTR) ||
                                    UNSPEC(LEVL) || UNSPEC(PRTY);
/*B*/    IF IX=10 THEN CALL PUTCARD;
         IX=IX+1; CARD0(IX)=VNAME(I);
         IF IX=10 THEN CALL PUTCARD;
         IX=IX+1; UNSPEC(CARD0(IX))=UNSPEC(KIND) || UNSPEC(KLAS) ||
                                    UNSPEC(NRWS) || UNSPEC(NCLS);
         IF IX=10 THEN CALL PUTCARD;
/*C*/      JX=0;
           DO J=1 TO NRWS;
                DO K=0 TO NCLS-1;
                   IX=IX+1; CARD0(IX)=VALUE(JX+KX+K);
                   IF IX=10 THEN CALL PUTCARD;
                END; JX=JX+LENRW;
           END; KX=KX+NCLS;
      END; IF IX<10 THEN CALL PUTCARD;
      CLOSE FILE (SPKSAVE);
      CALL ERRMSG(4);                                    /*EXIT*/
      END;
```