We tried challenging SAS's retrieval capabilities with some pretty tough system data sets.

We wanted some reports on our program libraries: what were the members? how many tracks did each take? list the members sorted by name, address, and loading size.... We wanted something a little more flexible than FENLIST. So we coded up a brief SAS job to read the directory of a partitioned data set and produce the desired reports. The directory has fixed length records, but you have to scan over a variable number of variable length subfields: not a job for just any retrieval package....

Next assignment: generate reports on what data sets are on a disk volume (i.e. read the VTOC--volume table of contents). The VTOC is a mixed file: the records are various types of DSCB's. The big problem is that when a data set has more than three extents, you have to look for a "format 3" extension and match it with the original "format 1" DSCB in order to get all the space allocation data. SAS's sort and merge facilities came in handy here. It took a little work, but SAS mastered the VTOC report--it even performed more efficiently than one of the custom-programmed utilities!

Now for something more challenging. The load modules on a program library are themselves interesting data sets. The ESD and RLD records contain all the information needed to reconstruct a linkage edit map and cross-reference for the module. And SAS is adept at retrieving this data. Once it is in SAS it can be manipulated, and reports can be generated. We have an application written in SAS that:

1) constructs a module map almost exactly replicating the one produced by the linkage editor;
2) produces a cross-reference (both ways) that we think is more concise and informative than that produced by the linkage editor;
3) it even reconstructs the linkage edit control cards needed to specify the existing overlay structure of the program.

I enclose some listings of the code for doing these tasks. These are already distributed in SAS.SAMPLE on the distribution tape.

Note: A minor correction to the LOADMAP application must be made if you work from SAS.SAMPLE. Change the MAXCOL logic to the statements enclosed here.
This sample is called "IEHLIST2"; it reads the directory of a partitioned data set and produces various reports on it. Some of the information is relevant only to compressed load module libraries.

*--------- SAMPLE DEMONSTRATING READING OF PDS DIRECTORY ---------;

DATA DIRECTRY; * PDS MUST BE A DDNAME FOR A PARTITIONED DATA SET;
INFILE PDS RECFM=U BLKSIZE=256 LRECL=256 COLUMN=C; COL=3;
INPUT COUNT PIB2. @; IF COUNT<14 THEN STOP;

LOOP: INPUT &COL MEMBER $8. TT PIB2. R PIB1. IND PIB1. @;
      IF MEMBER='99999999' THEN STOP;
      ALIAS='*'; IF IND>128 THEN ALIAS='*';
      LENG=MOD(IND,32); COL=C+LENG*2; SSICOL=COL-4;
      IF LENG>0 THEN INPUT +8 ATTRIB $2. SIZE PIB3. @;
      SS1='*';
      IF C<SSICOL THEN INPUT &SSICOL SSI SCHAR4. @;
      OUTPUT; IF COL <COUNT THEN GOTO LOOP;

DROP COUNT IND COL LENG SSICOL;
FORMAT ATTRIB $HEX4. S51 $8EX8.,
PROC PRINT; TITLE MEMBERS IN LIBRARY IN ALPHABETICAL ORDER;
PROC SORT; BY DESCENDING TT DESCENDING R;
DATA; SET; TRACKS=LASTTT-TT; RECS=0;
      IF TRACKS<1 THEN RECS=LASTR-R;
      OUTPUT; RETAIN;
      IF ALIAS='*' THEN RETURN; LASTTT=TT; LASTR=R;
      * THE TRACKS USED WILL BE CORRECT ONLY IF PDS IS COMPRESSED;
      DROP LASTTT LASTR;
PROC PRINT; TITLE MEMBERS IN LIBRARY BY REVERSE PHYSICAL ORDER;
PROC SORT ; BY DESCENDING TRACKS DESCENDING RECS;
PROC PRINT; TITLE MEMBERS IN LIBRARY SORTED BY TRACKS USED;
PROC SORT; BY DESCENDING SIZE;
PROC PRINT; TITLE MEMBERS IN LIBRARY SORTED BY CORE SIZE NEEDED;
This example called "MAPDISK" reads the VTOC of a disk pack, and produces reports on the data sets residing on that pack.

**----------------------------- MAPDISK -----------------------------**

This code reads in the DSCBs of the VTOC and produces a listing of all data sets with their attributes and allocation data.

A VTOC DD statement should be in the JCL like:

```
//VTOC DD DSN=VTOC VOL=SER=THISVOL UNIT=DISK DISP=SHR
```

The CCHHR option on the INFILE statement field as well as the key and data in the offsets are 5 more than indicated in the system control blocks.

The data set serial number at TUEC is used to store the usage count and date of last use.

**FORMAT 3 DSCBS and FORMAT 1 DSCBS for data sets with more than three extents are output to separate files for later merging to find out all the allocation data.**

```
DATA DSN(KEEP=DSNAME CREATED EXPIRES LASTUSE COUNT
         EXTENTS DSORG RECFM1-RECFM4 ALOC BLKSIZE
         LRECL SECALOC TT R TRACKS VOLSER)
FREE(KEEP=LOC CYL TRACK TOTAL RECN)
FMT1(KEEP=DSNAME CREATED EXPIRES LASTUSE COUNT
         EXTENTS DSORG RECFM1-RECFM4 ALOC BLKSIZE
         LRECL SECALOC TT R TRACKS CCHHR VOLSER)
FMT3(KEEP=CCHHR ALLOC3 VOLSER); LENGTH DEFAULT=4;

*--------READ DSCB AND DETERMINE WHICH FORMAT--------*
INFILE VTOC VTOC CCHHR COLUMN=COL;
INPUT @50 DSCBTYPE 11. @,
  IF DSCBTYPE=1 THEN GOTO FORMAT1;
  IF DSCBTYPE=3 THEN GOTO FORMAT3;
  IF DSCBTYPE=5 THEN GOTO FORMAT5;
  IF DSCBTYPE=4 THEN INPUT @70 TRKCYL PIB2.; RETAIN TRKCYL;
  IF DSCBTYPE=. THEN NULL=1; _ERROR_0;
  IF NULL>500 THEN STOP; RETURN;

FORMAT1: *----REGULAR DSCB TYPE----*
INPUT DSNAME $6-49
  @51 YU PIB1. DAYU PIB2. COUNT PIB2. IND PIB1. 
  @59 YC PIB1. DAYC PIB2. 
  YE PIB1. DAYE PIB2. 
  EXTENTS PIB1. @88 DSORG1 PIB1. @90 RECFM0 PIB1. 
  @92 BLKSIZE PIB2. LRECL PIB2. KEYLENG PIB1. 
  @100 ALO PIB1. SECALOC PIB3. TT PIB2. R PIB1. ;
  IF YC>0 THEN CREATED=DATEJUL(1000*YC+DAYC); 
  IF YE>0 THEN EXPIRES=DATEJUL(1000*YE+DAYE); 
  IF YU>0 THEN LASTUSE=DATEJUL(1000*YU+DAYU); 
 FORMAT CREATED EXPIRES LASTUSE DATE7.;
  *-------- COUNT AND LAST USE ARE TUEC SPECIALS------;
  IF NOT (IND=0 | IND=64) THEN COUNT=.; 
  IF NOT (IND=0 | IND=64) THEN LASTUSE=.;

*-------- DSORG RECODING--------*
  IF MOD(INT(DSORG1/2),2) THEN DSORG='PO';
  IF DSORG1>32 THEN DSORG='DA'; IF DSORG1>64 THEN DSORG='PS';
  IF DSORG1>128 THEN DSORG='IS';
*-------- RECFM RECODING --------*

RECFM1='I'; Z=INT(RECFM0/64);
IF Z=1 THEN RECFM1='V'; IF Z=2 THEN RECFM1='F';
IF Z=3 THEN RECFM1='U';
RECFM2='F'; IF MOD(INT(RECFM0/16),2) THEN RECFM2='B';
RECFM3='A'; IF MOD(INT(RECFM0/4),2) THEN RECFM3='A';
RECFM4='S'; IF MOD(INT(RECFM0/8),2) THEN RECFM4='S';

*-------- ALLOC TYPE RECODING -------*

ALOC=' '; Z=INT(ALO/64);
IF Z=1 THEN ALOC='BLK';
IF Z=2 THEN ALOC='TRK';
IF Z=3 THEN ALOC='CYL';

*-------- ALLOCATED SPACE --------*

INPUT CCHR $5.; OUTPUT FMT1; RETURN;
OUTD: OUTPUT DSN; RETURN;

*----- ROUTINES TO COLLECT EXTENT INFORMATION-----*

EXT4: LINK EXL; LINK EXL; LINK EXL;
EXL: INPUT INP PIB1. +1 (CCHHI HHLO CCHI HHHI) (PIB2.);
IF IND=0 THEN RETURN;
TRACKS=TRACYL *(CC8I-CCL0)+HH8I-HHLO+1;
RETURN;

FORMAT2: TRACKS=0;
INPUT @1 CCHR $CHAR5.@10@;
LINK EXL4; INPUT +10;
IF IND>0 THEN LINK EXL4;
IF IND=0 THEN LINK EXL4;
IF IND<0 THEN LINK EXL;
ALLOC3=TRACKS; OUTPUT FMT3; RETURN;

FORMAT5: INPUT @10@; RECN+1;

EXT: INPUT LOC PIB2. CYL PIB2. TRAC PIB2.8;
IF LOC=0 THEN RETURN;
IF LOC<1 THEN INPUT +10;
TOTAL=20*CYL+TRACK;
OUTPUT FREE;
OUTD: OUTPUT DSN; RETURN;

PROC PRINT DATA=FREE; TITLE3 FREE SPACE EXTENTS;
PROC MEANS N SUM MEAN MAX MIN DATA=FREE; VAR TOTAL TRACK CYL;
PROC SORT DATA=DSN; BY DSNAME;
TITLE3 ALPHABETICAL LISTING OF DATA SETS AND THEIR ATTRIBUTES;
PROC SORT DATA=FMT1; BY CCHR;
PROC SORT DATA=FMT3; BY CCHR;
DATA M; MERGE FMT1(IN=II) FMT3; BY CCHR;
IF II; TRACKS=TRACKS+ALLOC3;
PROC SORT DATA=M; BY DSNAME;
DATA DSN; SET DSN M; BY DSNAME;
DATA NULL; SET DSN END=EOF; FILE PRINT HEADER=H; M=-1;
PUT DSNAM $44. +1 DSORG $2. +2
(RECM1-RECFM4) ($1. +M)
$I1 (LREC BLKSIZE) (5. +1) +1
(CREATED EXPIRES LASTUSE) (DATE7. +1) +1
COUNT 5. +2 ALOC $3. +1 SECALOC 4. +2
EXTENTS 5. +1 TRACKS 6. TT 5. R 4. ;
TOTAL-TRACKS;
IF EOF THEN PUT // 'TOTAL NUMBER OF DATASETS = ' N,
// 'TOTAL NUMBER OF TRACKS = ' TOTAL;
RETURN;

H: PUT 'VOL-SER=' VOLSER $6.;
PUT /'DATA SET NAME'
@43 'DSORG RECFM LRECL BLKSZ CREATED EXPIRES LASTUSE' 
' USES SECONDARY';
@10 'EXTENTS TRACKS TT- -R' /; RETURN;
This sample called "LOADMAP" reads in a load module and produces the described reports.

*-------------------------------- LOADMAP --------------------------------*  
| THIS SAMPLE CODE READS IN THE ESD AND RLD RECORDS OF A LOAD MODULE,  
| AND CREATES: (1) A MODULE MAP ALMOST IDENTICAL TO THE ONE PRODUCED  
| BY THE LINKAGE EDITOR, (2) THE OVERLAY STATEMENTS NECESSARY TO  
| RECREATE THE LOAD MODULE, AND (3) CROSS REFERENCES BY CALLER AND  
| CALLEE. TO USE THIS CODE FOR ANY PARTICULAR MODULE, JUST CHANGE  
| THE INFILE STATEMENT.  
*--------------------------------------------------------------------* 

DATA 
ESD  (RESEP~NAME TYPE ESDID ADDR SEGMENT LENGTH HEXLENG ALIGN ID) 
RLD (RESEP~RESID PESID RLDTYPE OFFSET LENG SIGN); 
INFILE STEPLIB(MATRIX2) COLUMN=COL; 
INPUT RECTYPE PIB1. +3 ESDID PIB2. LREC PIB2. 0; 
   IF RECTYPE=32 THEN GOTO ESD; 
   IF RECTYPE=1 THEN GOTO CNTL; 
   IF RECTYPE=2 THEN GOTO RLD; 
   IF RECTYPE=6 THEN GOTO RLD; 
   IF RECTYPE=14 THEN GOTO RLD; 
   RETURN; 
CNTL: INPUT; INPUT; RETURN; * SEQUENCE PAST TEXT RECORD; 
ESD: MAXCOL=LREC+8; * EXTERNAL SYMBOL DICTIONARY RECORD; 
   LENGTH IB3. 0; 
   HEXLENG=LENGTH; ESDTYPE=MOD(ESDTYPE,32); 
   TYPE=''; 
   IF ESDTYPE=0 THEN TYPE='SD'; IF ESDTYPE=1 THEN TYPE='LI'; 
   IF ESDTYPE=2 THEN TYPE='ER'; IF ESDTYPE=3 THEN TYPE='LR'; 
   IF ESDTYPE=4 THEN TYPE='PC'; IF ESDTYPE=5 THEN TYPE='CM'; 
   IF ESDTYPE=6 THEN TYPE='PR'; IF ESDTYPE=10 THEN TYPE='WX'; 
   IF ESDTYPE=7 THEN ESDTYPE>15 THEN GOTO ESDN; 
   IF TYPE='LR' THEN ID=LENGTH/256; 
   IF TYPE='PR' THEN ALIGN=SEGMENT; 
   IF TYPE='PR' THEN SEGMENT=; 
   OUTPUT ESO; 
ESDN: ESDID=ESDID+1; 
   IF COL<=MAXCOL THEN GOTO LOOP; 
   RETURN; 
RLD: MAXCOL=LREC+16; * RELATION DICTIONARY RECORD; 
INPUT 017 0; 
ES: INPUT (RESID PESID) (PIB2.) 0; 
RPT: INPUT FLG PIB1. OFFSET PIB3. 0; 
   RLDTYPE=' '; Z=INT(FLG/16); 
   IF Z=0 THEN RLDTYPE='A'; IF Z=1 THEN RLDTYPE='V'; 
   IF Z=2 THEN RLDTYPE='D'; IF Z=3 THEN RLDTYPE='C'; 
   LENG=MOD(INT(FLG/4),4)+1; 
   CNTL: OUTPUT RLD; 
   IF COL<=MAXCOL THEN GOTO RPT; 
   IF COL=MAXCOL THEN GOTO ES; 
   IF MOD(RECTYPE,2) THEN LINK CNTL; 
FORMAT ADDR HEXLENG OFFSET HEX6.; 
DATA; SET ESD; IF TYPE='SD' | TYPE='LR' | TYPE='PR'; 
   IF TYPE='PR' THEN SEGMENT=16; * TO SORT HIGH; 
PROC SORT OUT=ESDS; BY SEGMENT ADDR DESCENDING TYPE;
*---------------- PRINT THE MODULE MAP --------------*
DATA NULL; SET ESDS; BY SEGMENT; RETAIN;
FILE 'PRINT HEADER=HEAD NOTITLES; OPTIONS PS=56;
IF FIRST SEGMENT & N >1 THEN PUT /;
IF TYPE='PR' THEN GOTO PR;
IF TYPE='LR' THEN GOTO LR;

*-------- PROCESS SD TYPE ENTRIES (CSECTS);
IF COL>41 THEN PUT;
PUT +2 NAME $8. @13 ADDR HEX6. @21 LENGTH HEX6. SEGMENT 6.;
COL=41; RETURN;

*-------- PROCESS LR TYPE ENTRIES (ENTRYPOINTS);
LR: PUT COL NAME $8. +2 ADDR HEX6. @; COL=COL+2;
IF COL<=41 THEN RETURN; COL=41; PUT; RETURN;

*-------- PROCESS PR TYPE ENTRIES (PSEUDO REGISTERS);
PR: NPR=SUM(NPR,1); IF NPR>1 THEN GOTO PRNXT;
PUT 'PSEUDO REGISTERS'; LINK PRHEAD;
PRNXT: PUT COL NAME $B. +5 ADDR HEX3. +6 LENGTH HEX2. @;
COL=COL+30; IF COL<=110 THEN RETURN; PUT; COL=3;
PUT; RETURN;

*-------- ROUTINE TO PRINT PAGE HEADINGS;
HEAD: IF NPR>0 THEN GOTO PRHEAD;
IF N<=1 THEN PUT @41 'ENTRY';
PUT / ' NAME ORIGIN LENGTH SEG. NO. ';
COL=41; RETURN;

*-------- PAGE HEADINGS FOR PSEUDO REGISTER PRINT OUT;
PRHEAD: PUT 4* ' NAME ORIGIN LENGTH ';
COL=3; RETURN;

*----------------- CREATE OVERLAY STATEMENTS ---------------*
DATA NULL; SET ESDS; BY SEGMENT;
FILE 'PRINT COLUMN=COL HEADER=HH;
IF SEGMENT=1 | TYPE='SD' THEN RETURN;
TITLE 'OVERLAY STATEMENTS FOR LOAD MODULE';
IF 'FIRST SEGMENT Then GOTO INSERT;
PUT;
PUT @7 'OVERLAY ' @16 'X ADDR HEX6. '
/ @7 'INSERT ' @16 @6;
INSERT: IF COL<=68 THEN GOTO INS;
PUT @72 ' ' @16 @6;
INS: PUT NAME @6; C=COL-1; PUT @C @6;
IF 'LAST SEGMENT THEN PUT ' @6;
RETURN;
HH: PUT // 8*'1234567890 ' /; RETURN;
*------------------- DO THE CROSS REFERENCE ------------------*
DATA ESDD; SET ESDS; IF TYPE='SD' THEN SD=NAME;
IF TYPE='PR' THEN SD=' ';
RETAIN;
PROC SORT DATA=ESDD OUT=ESDD; BY ESDID;
DATA RLD; SET RLD; IF RLDTYPE='V';
IF PESDID*RESIDID;
PROC SORT DATA=RLD OUT=RLDP; BY PESDID;
DATA RLDP;
MERGE ESDD (KEEP=ESDID NAME SEGMENT)
   RENAME=(ESDID=PESDID NAME=PRNAME SEGMENT=PSEGMENT))
   RLDP (IN=R); BY PESDID; IF R THEN OUTPUT;
PROC SORT DATA=RLDP OUT=RLDR; BY RESIDID;
DATA RLDR;
MERGE ESDD (KEEP=ESDID NAME SEGMENT SD
   RENAME=(ESDID=RESIDID NAME=RNAME SEGMENT=RSEGMENT))
   RLDR (IN=R); BY RESIDID; IF R THEN OUTPUT;
PROC SORT OUT=RLDR1; BY RSEGMENT SD PSEGMENT PNAME;
DATA NULL ; SET; BY RSEGMENT SD PSEGMENT PNAME;
IF ~FIRST.PNAME THEN RETURN;
FILE PRINT COLUMN=COL HEADER=H;
IF FIRST.SD THEN PUT / SD $8. PSEGMENT 3. +2 @;
IF COL>122 THEN PUT / $14 @;
PUT +1 PNAME $8. PSEGMENT 3. +1 @; RETURN;
H: PUT 'CSECT SEG'/ 'WAS CALLED BY' $* CSECT SEG' /; RETURN;
TITLE CROSS REFERENCE BY CALLEE;
PROC SORT DATA=RLDR1; BY PSEGMENT PNAME RSEGMENT SD;
DATA NULL ; SET; BY PSEGMENT PNAME RSEGMENT SD;
IF ~FIRST.SD THEN RETURN;
FILE PRINT COLUMN=COL HEADER=H;
IF FIRST.PNAME THEN PUT / PNAME $8. PSEGMENT 3. +2 @;
IF COL>122 THEN PUT / $14 @;
PUT +1 SD $8. RSEGMENT 3. +1 @; RETURN;
H: PUT 'CSECT SEG'/ 'DOES CALLS TO' $* CSECT SEG' /; RETURN;
TITLE CROSS REFERENCE BY CALLER;