Managing Disk Space With SAS®

Greg Mast, HealthCare COMPARE Corp.

Abstract

I have been involved with managing a growing number of SAS users in a VMS environment. While the number of users has increased, the amount of available disk space has not increased proportionally. I created a BASE SAS reporting tool which has helped me monitor disk space utilization by user and by the type of file. The code is not very sophisticated, but has helped avoid numerous frustrations and delays due to full drives.

Introduction

I manage a group of 6 programmer analysts and provide SAS operational support for roughly 40 others. Two years ago, this number was only 20. As the number of SAS users has continued to grow, so has the demand for disk space. There were a few other aspects which served to make effective space management more challenging:

1. Our department implemented data warehousing on some of the same disk drives. While this served to reduce the propagation of duplicate copies of various files across the disks, the net impact was the addition of many new files.

2. Some of the disk drives were shared with SAS users in other departments. Obviously, these colleagues were busy with their own projects, and we began to find ourselves in situations where multiple projects were competing for the same disk space.

3. Although disk space has been decreasing in cost over time, additional drives were difficult to obtain. Further, one of the key factors in justifying additional space has been the department's track record in effectively managing the storage resources it already had.

Clearly, the stage had been set for disk space shortages, competing priorities, and frustrated users.

After reviewing several alternatives, a SAS routine which used embedded DCL (Digital Control Language) commands was developed. When paired with an existing DCL routine described below, the results were most effective.

The DCL routine was a command file which gave users an instant summary of disk space usage across several drives. This routine was written by another department, and was shared with other SAS users. This will be elaborated upon a bit later on.

The SAS program was designed as a series of macros which contained embedded DCL directory commands. The output from these commands was then parsed into a SAS data set. A series of short reports were generated.

When the DCL command shows disk space is dropping on one or more drive, I submit the SAS program, which quickly tells me which users are accountable for using most of the space. A phone call or e-mail message to the users is usually sufficient to resolve the problem before it causes the drive to fill up and jobs to crash.

This approach has proved to be quite simple, effective, and has required very little time. By averting numerous potential space problems, the productivity of several departments in addition to my own has been enhanced.

The DCL Routine

Again, this command was written in another department of the company. Typing the word “space” on the command line resulted in the following display:

Utilization as of 20-SEP-1996 18:25

Disk	Total Space	Used	Free	% Free
ASRESTEMP	11808768	6094514	5714254	48%
ASRESSCR	11808768	5159744	6649024	56%
OTHSAS	4178385	3708865	469520	11%
OTHSAS2	12328410	10870098	1458312	11%
OTHSAS3	**5904423**	**5674623**	**229800**	**3%**
OTHSASO1	4109470	3226438	883032	21%
OTH	26520688	23480024	3040664	11%
TOTAL	50138224	34734282	15403942	30%

The name of the disk drive has been shown on the leftmost column. The total space in blocks, the number of blocks used, the number of free blocks, and the percentage of free blocks have also been displayed (1 block equals 512 bytes in the VMS environment).

Note the boldfaced line. When free space on any drive dips below 10%, the command file has been designed to display the line in boldface. Due to the way I have configured my terminal session, the boldfaced line would be red, and the other lines yellow against a black background. This arrangement ensured that a problematic drive would stand out very clearly on the display.

[The DCL code for this command file is quite lengthy, and rather unwieldy for inclusion into this paper. Interested readers may obtain it upon request from the author.]

The SAS Routine

When space is getting tight, I submit a SAS program simply called “disk-space.sas” to one of our batch queues. The logic has in this program been constructed out of a series of macros which run in sequence several times.

The first macro was designed to run the “space” command to make its output available for the program to use:

```
*-routine to check global disk space--;
*this calls the space command and outputs it to
a flat file which SAS can read.;

%macro foxinsox(drive);
x "@othsas:[mastgr]space.com /
out=asresscr:[mastgr]temp.asc";

data space1 ;
  infile "asresscr:[mastgr]temp.asc"
       missover;
       *lrecl=86, variable length;

  input drive $7-15 maximum 22-29
       used 38-45 free 53-60;
run;
```

The routine continued by creating a series of macro variables:

```
*---here is a routine which writes the code to
call the macros which are defined later on in
this program.  After much experimentation with
symput statements and the like, this routine
works best.;

data _null_;
  file 'asresscr:[mastgr]macro.sas'   noprint;
  set space1;
  if drive="&drive";
  x = length(drive) + 11;
  xx = x-3;
  put@1 '%catinhat(' @11 drive $9.
       @x ');'
   /  @1 '%seuss(' @8 drive $9. @xx ','
       @xx+1 maximum 8. @xx+9 ','
       @xx+10 used 8. @xx+19 ');' ;
run;
%mend foxinsox;
```

The point of the macro was to get the total amount of space available on each drive, key information which was used later on in the program for calculations.

The next macro was run once for each of the drives shown by the “space” command.

```
%MACRO CATINHAT(DRIVE);
x "dir/size=allocation/date/own/nohead/
  notrail/width=(filename=100,
  display=195,size=10)
  /out=asresscr:[mastgr]&DRIVE..asc
    &DRIVE.:[000000...]";
%MEND CATINHAT;
```

Sample output from the “catinhat” macro has been shown below:

```
DRIVE:[DIRECTORY]SAMPLE.SAS;9             20
24-MAR-1994 15:33:00.41  [USERGROUP,USERID]
```

The full name of the file, including the drive and directory where it has been written, was obtained. The number of blocks, file creation date and time, as well as the userid and the user’s group were also obtained. The numeral “9” just to the right of the semicolon was the version of the file, discussed below. By running this file through an input statement, a great deal of useful information could be generated.

A final macro follows which, due to its length, has been placed at the end of this document. It has been summarized as follows:

1. An input statement created a SAS data set from the output of the “catinhat” macro.

2. Additional data step logic created variables which depicted the name of the file, its size, who created it, and when it was created.

3. Array logic then created “buckets” for the number of files and the space they consumed based upon several age ranges: files 48 hours old or less, files up to 3 months old, and files over 3 months old. These categories were designed to be mutually exclusive. There was nothing magic about the boundaries of these categories.

It would be very easy, for example, to adjust one of these categories to reflect files written within the last few hours. With some additional reworking of the code, one could quickly develop a version of this program which could be run interactively against a specified drive. For my own situation, however, I have found that looking at the total usage of the drive helps me manage the drives a bit more proactively.

4. The data set was then fed to routines consisting of PROC SUMMARY, data steps, PROC SORT, and PROC PRINT. The macro variables created in the first macro were employed in these reports by calculating the difference between the space usage as measured by SAS and usage as measured by the DCL “space” command. The difference between them furnished an estimate of how much space was being used by various system files which the program does not have the security clearance (known in the DEC world as “privileges”) to read.

The results were tabled in a series of 3 reports. A brief discussion of each of the reports follows; since each report has a linesize of 180, it was not practical to paste report images into this paper. Please refer to the handout.

Report 1
Report 1 was developed to list disk space utilization by user . A PROC FORMAT routine was used to take the userid and display their name and phone number on the report.

I have found that one of the most important uses of this report relates to purging. In the VMS environment, a new file will not overwrite an old one bearing the same

name. Instead, a new version is created. DEC has provided a feature which limits the number of file versions retained to a user-specified value, but for several reasons, our user group has opted not to use this feature. Most veteran DEC users in my group have long known to purge rigorously and regularly, but the same is not true for DEC novices or sporadic users. Left unmanaged, multiple copies of old versions of files can seriously interfere with the productivity of an entire department Fortunately, it is also one of the simplest issues to resolve, by using the DCL "purge" command.

Among other useful information, the first report has also been invaluable by showing who has been hoarding a "museum" of old reports and SAS logs in their home directory.

Report 2
The second report has been designed to list files with over 50,000 blocks and all SAS work directories, regardless of their size. Our group's SAS.CFG files have been set to point work files to a "scratch" drive, where files are automatically deleted after 48 hours. Sometimes, however, a new user's account may not have been set up appropriately, or an experienced user may have decided to end run normal routines due to pressing circumstances. By watching SAS work directories, these situations have been much easier to identify and correct.

Report 3
The third report has been set up to profile disk space by file type across all directories. Our group has designated certain drives for holding either SAS data sets and indexes, or for SAS code, logs, reports, DCL command files, and the like. As such, the final report has been instrumental in systematically spotting files written to the wrong place.

Invoking the Macros

Once all the macros have been compiled, the code below has been used to execute them:

```
%foxinsox(OTHSAS);
%include 'asresscr:[mastgr]macro.sas';
x 'del asresscr:[mastgr]OTHSAS.ASC;*';
```

Again, while there are more elegant ways to approach this, the code above has run reliably for over 2 years.

Conclusions

One side benefit from this approach to disk management has been what one might call the sentinel effect. Once users realized they were being watched, several of the largest space users began to police their own work.

The program has also pointed out users who needed coaching in their use of disk space. Since sloppy disk space use has frequently been the result of excessive sorting and other inefficient coding practices, we have seen opportunities to help these colleagues develop sounder approaches to writing good SAS code.

Finally, this basic approach to managing disk space with SAS could be applied to numerous computing platforms by altering the embedded operating system commands, file references, and so on. The unit of measure could be changed from blocks to bytes, cylinders, or some other appropriate unit. A variation of this code, for example has also been run on one of our company's local area networks (LAN). The LAN version, however, was modified to read an extract from a Microsoft Access database maintained by the LAN manager instead of issuing DOS subroutines to gather the data.

Future steps for this code could include writing a window with DATA _NULL_ which would query the user for the disk drives they were interested in. If one had a large number of disk drives to manage, this could save a lot of time. Another idea would be to add a new routine which would issue a list of files, per user, which have exceeded a certain age threshold. Since many of the older files could be from completed projects, a routine like this could serve as a follow-up mechanism reminding users to archive aging data or code. One could even embed DCL commands to automatically send a Vaxmail message containing a list of the old files to the user. One final possibility might be to edit the DCL "space" command to automatically launch the SAS "disk-space" program if any drive were to dip below a specified space threshold. The SAS code could then provide reports only for the drives needing attention.

Effective disk space management needs to answer the same basic questions as any well-written newspaper article: it must answer who, what, when, where, why and how. The solution discussed above answers the first four of these questions, and good, routine communication with the SAS users in my group answers the last two.

References

Anagnostopoulos, Paul C. (1989), *Writing Real Programs in DCL*. Burlington, MA: Digital Press.

Digital Press (1989), *VMS User's Manual*. Burlington, MA: Digital Press.

SAS Institute Inc. (1990), *SAS Companion for the VMS Environment, Version 6, First Edition.* Cary, NC: SAS Institute, Inc.

Acknowledgments

I would like to thank Martin Haase for his feedback and suggestions in developing this article. I would also like to acknowledge Karsten Self for all of his great work on developing the initial "space" command.

The author may be reached at :

Greg Mast
Manager Data Analysis & Reporting
HealthCare COMPARE Corp.
750 Riverpoint Drive
West Sacramento, CA 95605
Greg_Mast@hccompare.com
(916) 374-4757

The Final Macro

The code below was the final macro referred to in the "SAS Routine" section.

```
%MACRO SEUSS(DRIVE,BIG,USED);
data a;
  infile "asresscr:[mastgr]&DRIVE..asc"
       lrecl=195 missover;
  input blob $ 1-100  size 103-112
       @115 L date11. @115 dd 2. @118 mmyy $8.;

*---count files where users have implemented
tight file protection. A high number of  them
warrants inquiry. ;

  if size=. then protectd=1; else protectd=0;

*--identify the directory and the user--;
length dir $ 50 userid $ 8 filetype $ 10;

*first, get the directory;

a = index(blob,"[") + 1;
b = index(blob,"]") - 1;
c = (b-a)+1;
dir = substr(blob,a,c);

*--now we can identify the userid--;
d = index(dir,".") - 1;
if d > 1 then userid = substr(dir,1,d);
else userid = dir;

*----identify the file  name and extension---;
e = length(blob);
h = index(blob,";") - 1;

filename = substr(blob,(b+2),(e-(b+1)));
dot = index(filename,".");
fun = length(filename);

nuf = substr(filename,(dot+1),(fun-dot));
*this is the extension + version number;

ufn = index(nuf,";") - 1;
*this is the byte where extension ends;

fufu = length(nuf);
filetype = substr(nuf,1,ufn);

if filetype='SYS' then  userid='SYSTEM';

*--identify the version for need-to-purge
detection--;

i = e - (h+1);
*h+1 is where the semicolon is in the filename;

version = input(substr(blob,(h+2),4),5.);
blob2 = substr(blob,1,h);
*this is the filename without the version
number;


*keep the filesize data of files we could purge;
if  lag(blob2) = blob2
    then purge = lag(size);
else purge=0;

*--edit an artifact from incomplete journal
files;

if purge > (size*5000) then purge = size;
*[potential underreporting or other minor
inaccuracy may result,  but this is better than
an occassional wildly high number];

*-----flag old files----------------;
*create current date;
 ahora=today();

*measure the age of the file in days;
age = ahora - l;

*--now do age buckets - 1 for space and 1 for
the number of files.  create one set of vars
for:  2 days,  3-90 days, and over 90 days;

array ned{6} mo6_cnt mo6_sp mo3_cnt mo3_sp
       x48_cnt x48_sp;

   do n=1 to 8;
     ned{n}=0;
   end;

 if age le 2 then do;
     x48_cnt = 1;
     x48_sp  = size;
  end;

  else if (2 < age < 91) then do;
     mo3_cnt = 1;
     mo3_sp  = size;
  end;

 else if age > 90 then do;
    mo6_cnt = 1;
    mo6_sp = size;
  end;

keep dir freq size protectd purge userid
filetype mmyy mo6_cnt mo6_sp mo3_cnt mo3_sp
x48_cnt x48_sp;

run;

*---calculate the size of system files;
proc summary data=a nway;
  var size;
  output out=xxx(drop=_type_) sum=sizex;
run;

data marvin;
  set xxx;
  sizez = ( &used - sizex);
*total used space minus all space SAS can
account for due to priveleges;
  call symput("SYSX",sizez);
run;
```

```
*-------now apply the calculated system file
space, being careful not to doublecount;

proc sort data=a;
  by userid;
run;

data marco;
  set a;
  by userid;
  length freq size  mo6_cnt mo6_sp 8;
  if first.userid and  userid='SYSTEM' then do;
    size = ( &SYSX / 1);
    mo6_sp = size;
    freq = 1;
    mo6_cnt = 1;
  end;

  else if userid='SYSTEM' then do;
     size=0;
     mo6_sp=0;
     freq=1;
     mo6_cnt=1;
   end;

  else freq = 1;
run;

*----summarize-----------;
proc summary data=marco missing;
  class userid filetype dir;
  var freq size protectd purge mo6_cnt mo6_sp
       mo3_cnt mo3_sp x48_sp x48_cnt;
  output out=b sum=;
run;

*----create report---;
data c;
  set b;

  if _type_=0 then do;
    userid='ZZZZZZZZ';
    filetype='ZZZ';
    dir='ZZZ';
  end;

  if _type_=1 then do;
    userid='ZZZZZZZZ';
    filetype='ZZZ';
   end;

  if _type_=2 then do;
    userid='ZZZZZZZZ';
    dir = 'ZZZ';
  end;

  if _type_=3 then userid='ZZZZZZZZ';

  if _type_=4 then do;
    filetype='ZZZ';
    dir = 'ZZZ';
  end;

  if _type_=5 then filetype='ZZZ';
  if _type_=6 then dir='ZZZ';

  pct = (size /  &big ) * 100;
  pct2= (purge / size) * 100;

  length whom $ 8;
  whom = userid;
run;

*-----userid level summary-------------;
proc sort data=c out=d;
  by descending _type_ descending size;
  where _type_=0 or _type_=4;
run;

proc print data=d noobs  split="*";
  var userid whom freq protectd size pct purge
       pct2 x48_cnt x48_sp  mo3_cnt mo3_sp
       mo6_cnt mo6_sp;

  format size                          comma10.
         purge mo6_sp  mo3_sp x48_sp   comma9.
         freq                          comma6.
         mo6_cnt mo3_cnt  x48_cnt      comma5.
         pct pct2 protectd             3.
         whom                          $whom.  ;

*($whom is a format which takes a userid and
returns the user name and telephone extension);

  label userid  = 'User Id'
        whom    = 'Name & Phone Extension'
        freq    = '# Files'
        protectd= 'Pro-*tected'
        size    = 'Blocks'
        pct     = 'Pct'
        purge   = 'Blocks*To*Purge'
        pct2    = 'Pct'
        mo6_cnt = 'Files*Over 3*Mos'
        mo6_sp  = 'Space*Over 3*Mos'
        mo3_cnt = 'Files*Up To 3*Mos'
        mo3_sp  = 'Space*Up To 3*Mos'
        x48_cnt = 'Files*2 Days'
        x48_sp  = 'Space*2 Days'     ;
  title "&DRIVE SPACE USAGE REPORT BY USER ID";
run;

*------------summary by directory--------------;
*screen out little stuff...look for large files
and sas work directories;

data xx;
  set c;
  if _type_=1;
  if (size > 49999)
   or (index(dir,"SAS$WORK") > 0);
  pct = (size / &big ) * 100;
  pct2= (purge / size) * 100;
run;

proc sort data=xx;
  by dir;
run;

proc print data=xx  noobs split="*";
  var dir freq protectd size pct purge pct2
       x48_cnt x48_sp mo3_cnt mo3_sp mo6_cnt
       mo6_sp;

  format size                          comma10.
         purge  mo6_sp  mo3_sp x48_sp  comma9.
         freq                          comma6.
         mo6_cnt mo3_cnt  x48_cnt      comma4.
         pct pct2 protectd             3.
         whom                          $whom.  ;


  label dir     = 'Directory'
        freq    = '# Files'
        protectd= 'Pro-*tected'
        size    = 'Blocks'
        pct     = 'Pct'
        purge   = 'Blocks*To*Purge'
        pct2    = 'Pct'
        mo6_cnt = 'Files*Over 3*Mos'
        mo6_sp  = 'Space*Over 3*Mos'
        mo3_cnt = 'Files*Up To 3*Mos'
        mo3_sp  = 'Space*Up To 3*Mos'
        x48_cnt = 'Files*2 Days'
        x48_sp  = 'Space*2 Days' ;
```

```
  title "&DRIVE SPACE USAGE REPORT BY DIRECTORY
- SAS$WORK DIRECTORIES OR OTHER FILES OVER
49,999 BLOCKS";
run;

*------------summary by file type--------------;

*summarize again, and recode to simplify
results.;

proc summary data=marco missing nway;
  class filetype;
  var size protectd purge mo6_cnt mo6_sp mo3_cnt
       mo3_sp x48_sp x48_cnt;
  output out=bb sum=;
run;

data xxx;
 set bb;
  if filetype in
('JOU','SAS','DAT','LIS','LOG','TXT','TDS',
 'SSD','COM','MAI','SASEB$DATA','SASEB$CATA',
 'SASEB$VIEW','SYS','TPU$JOURNAL') then do;
     *absolutely nothing;
  end;
  else filetype='ALL OTHERS';
  rename _freq_ = freq;
run;

proc summary data=xxx MISSING nway;
  class filetype;
  var size protectd purge mo6_cnt mo6_sp mo3_cnt
       mo3_sp x48_sp x48_cnt freq;
  output out=bb sum=;
run;

proc sort data=bb;
  by descending _type_ filetype;
run;

data cc;
  set bb;
  by descending _type_ filetype;
  if _type_=0 then
    filetype='*** GRAND TOTAL ***';

  pct = (size / &big ) * 100;
  pct2= (purge / size) * 100;
run;

proc print data=cc  noobs split="*";
  var filetype freq protectd size pct purge pct2
       x48_cnt x48_sp  mo3_cnt mo3_sp mo6_cnt
       mo6_sp;

format size                           comma10.
      purge yr_sp mo6_sp mo3_sp x48_sp comma9.
            freq                      comma6.
            mo6_cnt mo3_cnt  x48_cnt  comma4.
            pct pct2 protectd         3.
            whom                      $whom.  ;

  label filetype= 'File*Type'
        freq    = '# Files'
        protectd= 'Pro-*tected'
        size    = 'Blocks'
        pct     = 'Pct'
        purge   = 'Blocks*To*Purge'
        pct2    = 'Pct'
        mo6_cnt = 'Files*Over 3*Mos'
        mo6_sp  = 'Space*Over 3*Mos'
        mo3_cnt = 'Files*Up To 3*Mos'
        mo3_sp  = 'Space*Up To 3*Mos'
        x48_cnt = 'Files*2 Days'
        x48_sp  = 'Space*2 Days' ;
  title "&DRIVE SPACE USAGE REPORT BY FILETYPE";
run;

proc datasets library=work nolist;
  delete a b c d bb cc xx;
run;
%MEND   SEUSS;
```