Paper DV-06-2014

How to Create a UNIX Space Management Report Using SAS®

Thomas Lehmann, MBA, Truven Health Analytics, Ann Arbor, MI
Matthew Shevrin, MM, Truven Health Analytics, Ann Arbor, MI

ABSTRACT

Storage space on a UNIX platform is a costly - and finite - resource to maintain, even under ideal conditions. By regularly monitoring and promptly responding to space limitations that might occur during production, an organization can mitigate the risk of wasted expense, time and effort caused by this problem. SAS programmers at Truven Health Analytics have designed a reporting tool to measure space usage by a number of distinct factors over time. Using tabular and graphical output, it provides a full picture of what often contributes to critical reductions of available hardware space. It enables managers and users to respond appropriately and effectively whenever this occurs. It also helps identify ways to encourage more efficient practices, thereby minimizing the likelihood of this from occurring in the future. Operating System: RHEL 5.4 (Red Hat Enterprise Linux), Oracle Sun Fire X4600 M2 SAS 9.3 TS1M1 Skill level: Intermediate

INTRODUCTION

The authors work in a SAS-oriented production environment using very large healthcare claims data files on a shared UNIX server. Dozens of users rely on a series of mounted directories, known as volumes, to store data, programs and other files. To monitor space use both retrospectively and in real time, a repeatable reporting tool that uses UNIX commands inserted into SAS and produces tables and graphics was developed. This paper describes the report works in detail; it explains the key UNIX commands executed to calculate space usage; how to embed them within a SAS program; and how to insert the results into a report using the Output Delivery System (ODS) and SAS/GRAPH®. The sections of the reports include volume usage at the root level; project-level usage by volume and at user-defined levels; appropriate date thresholds for data sets to be archived or removed; files ranked by size to identify unusually large data sets for attention; and cyclical variations of space usage over time.

KEY UNIX SPACE USAGE COMMANDS

There are some key commands to execute in UNIX to determine how resources are being used. The 'find' command allows one to report on a range of details on selected areas of the server. By including the '-size' command, the find command shows how much space each file is taking up at the time. To run this within a SAS program, the two commands must be preceded by the 'x' command. This tells SAS to start a shell in order to execute a given command. Below are examples where running the x and find commands, executed from SAS, generates results to a text file.

UNIX commands to report on old data sets:

```
x find &path. -size +000000001c -ls > $HOME/&printo._&Runtype..txt;
```

```
clin-info     drwxrws---   1024 Jul     8       2004     /clin-info/informatics/hpp/2004
hcia          -rwxrwx---   5103 Jun     8       2004     /clin-info/informatics/hpp/2004/hpp.sas
hcia          -rwxrwx---   1665 Mar     12      2004     /clin-info/informatics/hpp/2004/hpphosp.lst
hcia          -rwxrwx---   2355 Mar     12      2004     /clin-info/informatics/hpp/2004/hpphosp.sas
```

UNIX commands to report on large data sets:

```
x find &path. -size +100000000c -ls > $HOME/&printo._&Runtype..txt;
```

```
24932  132136 -rwxrwx---   1 tlehmann hcia     135306574  Jun 20 04:17
     /clin-info/informatics/msdrg_12/msdrg27_10/pidb/data/dbridge/cms10q3_disch.txt.gz

24933  133439 -rwxrwx---   1 tlehmann hcia     136641427  Jun 20 03:39
     /clin-info/informatics/msdrg_12/msdrg27_10/pidb/data/dbridge/cms10q2_disch.txt.gz

24934  139217 -rwxrwx---   1 tlehmann hcia     142557267  Jun 20 02:36
     /clin-info/informatics/msdrg_12/msdrg27_10/pidb/data/dbridge/cms10q1_disch.txt.gz

24787 1033327 -rwxrwx---   1 tlehmann hcia     1058118067 Jul 30 12:38
     /clin-info/informatics/msdrg_12/msdrg27_10/pidb/data/msdrg_27.sas7bdat.gz
```

Two other UNIX commands of interest are ‘du’ and ‘df’, which summarize disk usage and report disk free space, respectively.

UNIX commands to report project sums:

```
x du -sk * -ls > $HOME/&printo._&Runtype..txt;
```

```
1720          medpar
3495          msdrg_11
16403908      msdrg_12
12400124      msdrg_13
```

UNIX commands to report volume sums:

```
x df -k . > /&path./&Runtype./&printo._&Runtype..txt;
```

```
Filesystem                    1K-blocks      Used  Available Use%  Mounted on
/dev/vx/dsk/g1/clininformatica 314572800 288046973  24872877   93%  /clin-info/informatics
```

The next section explains how these commands, in combination, are called within a single macro-driven program.

CREATING SPACE MANAGEMENT REPORT TABLES IN SAS

The following steps describe how SAS code is structured to produce the tabular portions of a space report. See Figures 1-3 for report samples and the Appendix for examples of the log. The entire SAS program is posted on sasCommunity.org at http://www.sascommunity.org/wiki/File:SpaceReportV4.2_mloop.sas.

1. Organize UNIX commands that generate a text-formatted layout template for input data by the following dimensions
    - Date (by age of last modification)
    - Size (in kilobytes)
    - Volume (directory)
    - Project within volume (user-defined)

2. Build a macro to break apart a directory structure from its root
    - Use macro assignments to define directory levels by delimiting with spaces as individual words
    - Use decomposed words to assign the macro variable ‘printo’ as the name of the output text file (in between underscores)
    - Use the sub-routine ‘path’ to build the directory structures by run type by using underscores

3. Use the macro elements to define distinct file names and paths and produce text files to import into SAS. Figure 1 is an example of how the log and list files are named based on the macro variable ‘printo’. Figure 2 shows how the text files are named by project.
    - Use the ‘x’ command with unique path names replete with underscores to access the root where an ‘inventory’ is to be taken
    - Use the infile statement with the ‘printo’ name to import text files into SAS
    - Separate data into distinct reporting groups, such as SizeAnalysis, TimeCut, or ProjectCentric run types
    - For ProjectSums, data is read with a colon identifier. For VolumeSums, it is read with a colon identifier in a block.

4. Insert ODS commands and appropriate tagsets to make Excel the target object. Figure 3 displays a sample of the set of tables exported into Excel.
    - Use a firstlast flag to define appropriate ODS resources, such as specifying tagsets and file names
    - Use the SHEET_NAME option in ODS to define the WorkSheet name (‘printo’)
    - Use the PRINT procedure in ODS to specify a WorkSheet with an appropriate ‘printo’ name
    - Format and summarize file sizes in numeric format with a TAGATTR command to display results in Excel
    - Use a firstlast flag to close ODS tagsets

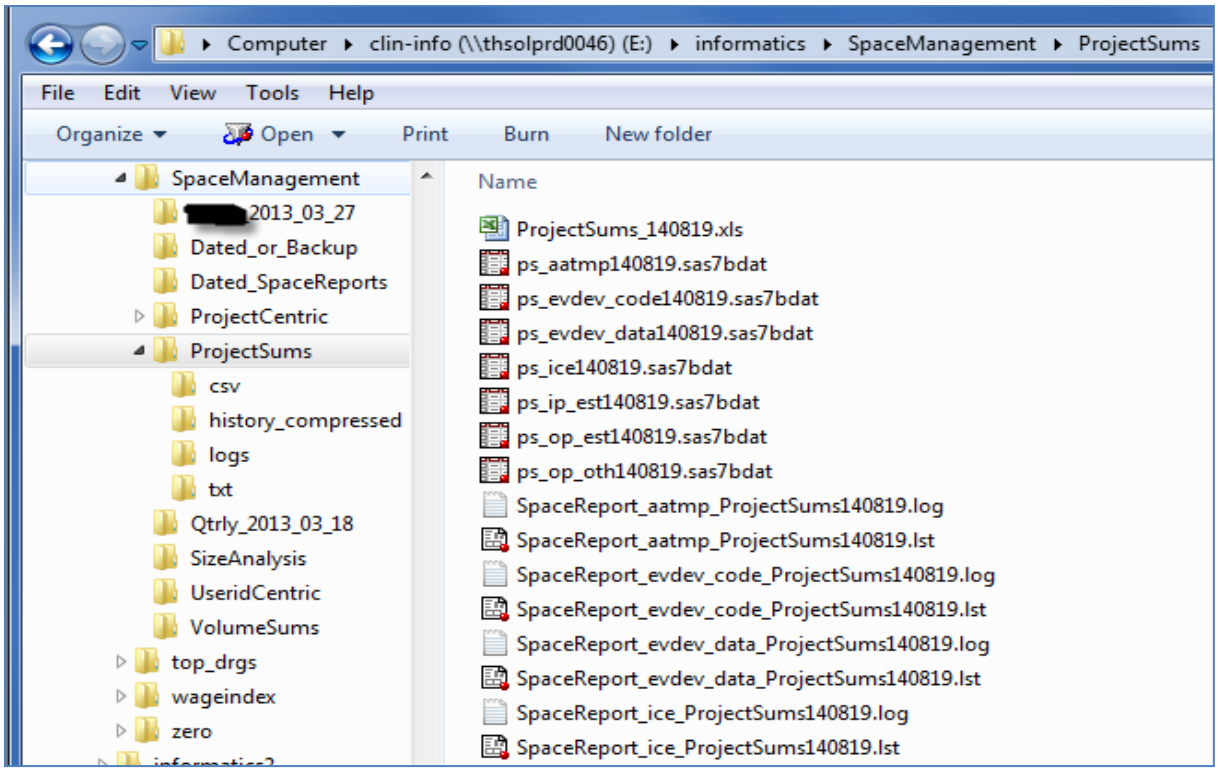

Figure 1. Example of the macro variable 'PRINTTO' with pieces of the names of the final outputs for log and list files. For ProjectSums, the SAS data sets are used as the monthly inputs for the graphics.

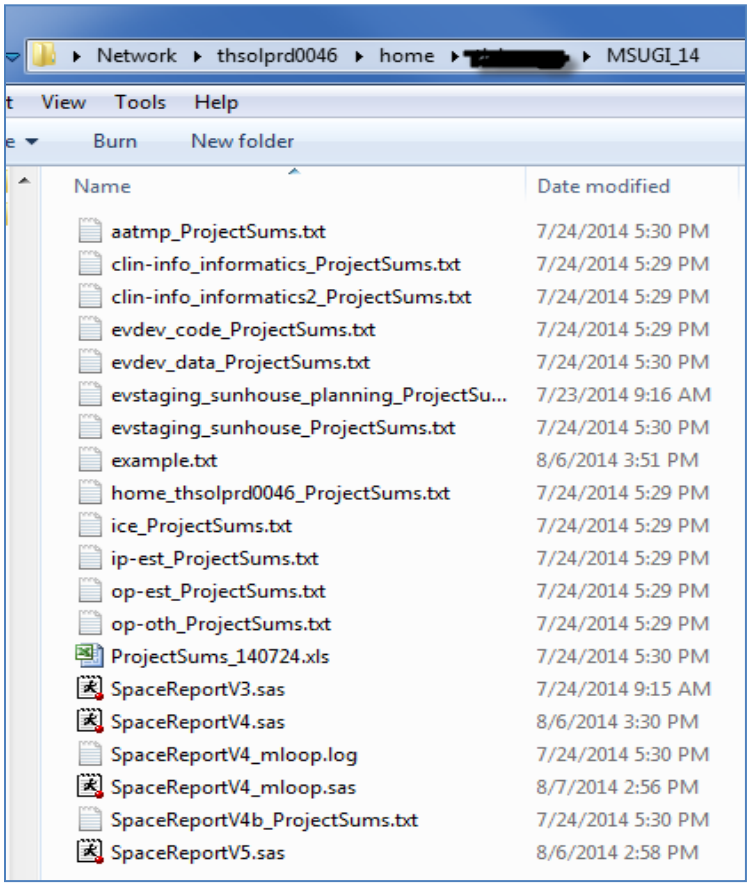

Figure 2. Example of intermediate text files written to the home directory with the resolution of the project names

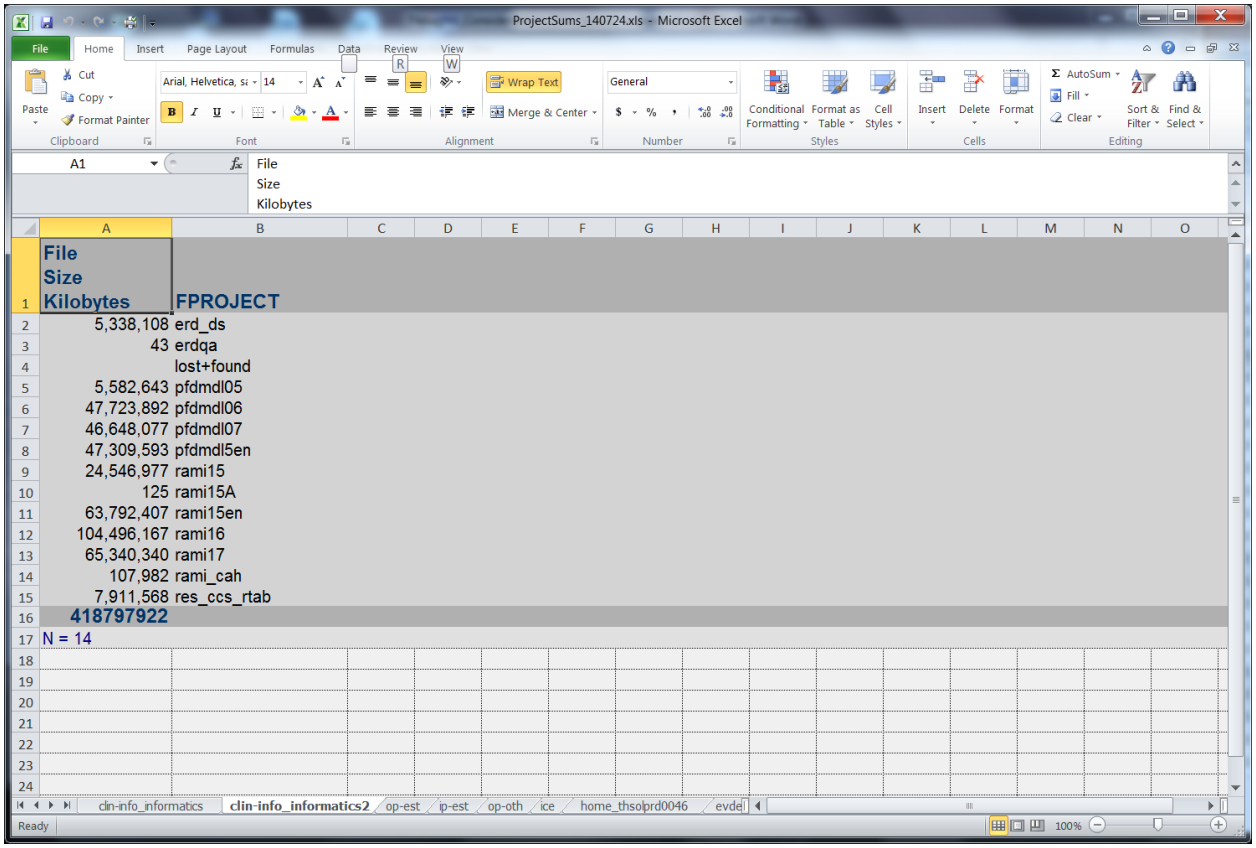

Figure 3. Example of multiple sheet report in Excel

CREATING SPACE MANAGEMENT REPORT GRAPHICS IN SAS

The following steps describe how SAS code is structured to produce the graphical portions of a space report. This requires having SAS/GRAPH version 9.2 or above installed. Figure 4 displays a bar chart of space usage by volume. Figure 5 shows space usage by project and month.

1. Select the desired report output types from the tabular section described above
2. Open ODS graphics component to output results

```
ODS graphics on;
ODS trace on;
ODS pdf file="&path.\adhoc\mwsug14\Space Management Report
       &file_date..pdf";
```

3. Generate plot of current space usage by volume using SGPLOT

```
title "Space Report: Usage by Volume on UNIX Server thsolprd0046 as of &rptdte.";

ODS PROCLABEL 'Usage by Volume';

PROC SGPLOT Data=work.all_&file_date._used_pct;
     hbar directorysimple / response=pct_used;
     yaxis display =(nolabel) discreteorder=rdata;
run;
```

4. Generate panel plots of usage by project and month using SGPANEL

```
title "Space Report: Monthly Usage by Project on &full_volume as of &rptdte.";

ODS PROCLABEL "Monthly Usage by Project on &volume. ";

PROC SGPANEL Data=&volume._combined (where=(proj ne .));
     panelby proj / rows=3 columns=5;
     hbar  month_n / response=fsize_gb barwidth=.5;
```

```
        colaxis values= (0 to 100 by 25);
        format  proj projdesc. month_n mdesc_tst.;
        label   fsize_gb = 'In Gigabytes (1 = 1,000,000 KB)'
                month_n = 'Month';

run;
```

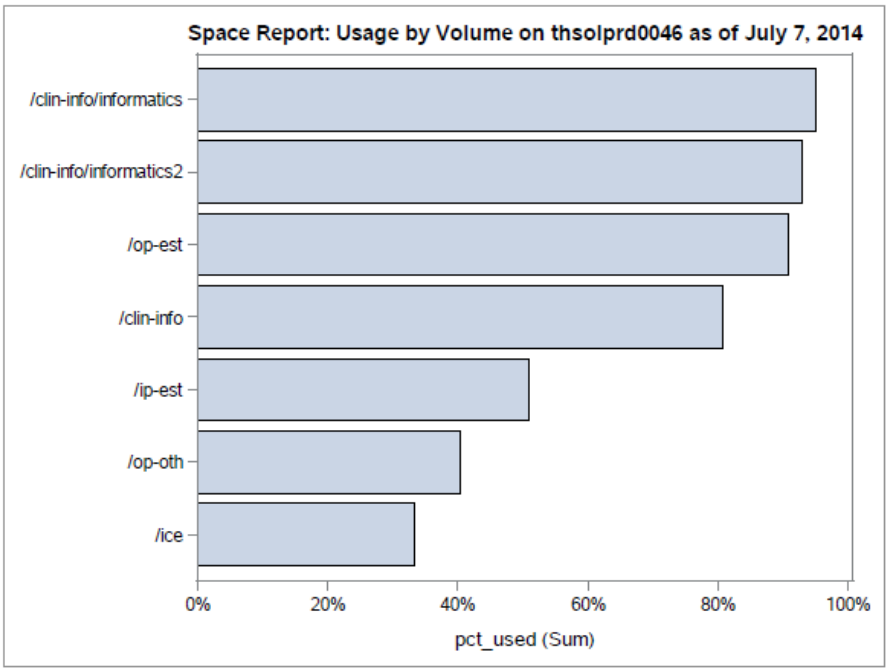

Figure 4. PROC SGPLOT: UNIX space by volume

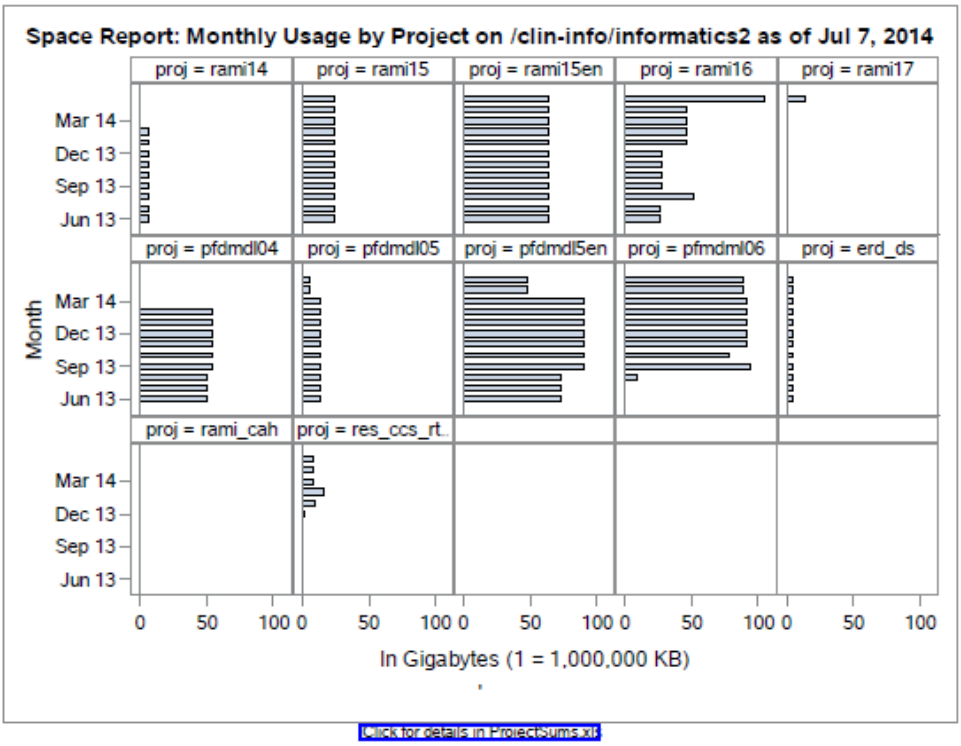

Figure 5. PROC SGPANEL: UNIX space by project and month

ADDITIONAL SPACE REPORTING SUGGESTIONS

There are other resources not discussed in this paper to consider when setting up a space management report. For instance, UNIX octal permissions can be reported by project with user details. Note that '660' is considered the minimum necessary so other users can change permissions. Also, users can be notified to compress files on completed projects and automated alerts can be sent via e-mail when usage reaches a certain level. A space management tool guide is recommended for users to increase their knowledge of acceptable space usage practices and policies. It could explain how to employ such UNIX commands as 'MAN' and 'UMASK"; show users how to get disk usage on their own; find and recursively change permission levels for a project; identify and compress files; and identify permission settings that aren't high enough for non-owners to compress.

CONCLUSION

The space management report described here helps UNIX administrators and users monitor space usage that occurs in real time and retrospectively. By executing UNIX commands within SAS and generating tables and graphs with ODS and SAS/GRAPH, one can produce a broad range of information about the resources being used at many levels and over time. As a result, targeted action can be taken to avoid reduced productivity when space capacity is at risk. There are many other things one can do with UNIX and SAS to support efficient use of space that go beyond the scope of this paper. The authors have provided some examples in detail and encourage the reader to consult the references and recommended reading shown below for more on this topic.

REFERENCES

Shelley Powers, et al, 2002, UNIX Power Tools 3rd Edition, O'Reilly Media, Inc, Sebastopol, CA.

Airaha Chelvakkanthan Manickam, "SAS® UNIX-Space Analyzer – A Handy Tool for UNIX SAS® Administrators"

http://www.lexjansen.com/pharmasug/2012/PO/PharmaSUG-2012-PO11.pdf

Adeline J Wilcox, "Disk Space Management Topics for SAS@ in the Solaris™ Operating Environment"

http://www.lexjansen.com/nesug/nesug04/as/as04.pdf

RECOMMENDED READING

- Output Delivery System: The Basics and Beyond, by Lauren E Haworth, et al, 2009.
- Statistical Graphics in SAS® , by Warren F Kuhfeld, 2010.
- Sample 25271: Automated Storage Monitoring Routine for UNIX Servers
  http://support.sas.com/kb/25/271.html

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Thomas Lehmann, MBA
Truven Health Analytics
777 E Eisenhower Bl
Ann Arbor, MI 48108
734-913-3782
Thomas.lehmann@truvenhealth.com

Matthew Shevrin, MM
Truven Health Analytics
777 E Eisenhower Bl
Ann Arbor, MI 48108
734-913-3410
Matthew.shevrin@truvenhealth.com

APPENDIX: LOG SAMPLE

Note: This log snippet explains the macro build for variable 'PRINTO'. Two macro variables are created in CNTVAR because there is a list, or Inventory, with just two words: clin-info informatics. See OneInv1 and OneInv2 in the resolution of macro sub-routine PRINTTO below. The entire SAS program is posted on sasCommunity.org at http://www.sascommunity.org/wiki/File:SpaceReportV4.2_mloop.sas .

```
SYMBOLGEN:  Macro variable I resolves to 1
SYMBOLGEN:  Macro variable ONEINV1 resolves to clin-info
MLOGIC(PRINTTO):  %PUT _user_
PRINTTO I 1
PRINTTO UNDERSCRE _
MAJORLOOP FL first
MAJORLOOP INVENTORY clin-info informatics
GLOBAL PRINTO clin-info
GLOBAL RUNTYPE ProjectSums
GLOBAL ONEINV1 clin-info
GLOBAL ONEINV2 informatics
GLOBAL DTE 140724
GLOBAL CNTINV 2
MLOGIC(PRINTTO):  %DO loop index variable I is now 2; loop will iterate again.
SYMBOLGEN:  Macro variable I resolves to 2
MLOGIC(PRINTTO):  %IF condition &i.=1 is FALSE
MLOGIC(PRINTTO):  %LET (variable name is ADJ)
SYMBOLGEN:  && resolves to &.
SYMBOLGEN:  Macro variable I resolves to 2
SYMBOLGEN:  Macro variable ONEINV2 resolves to informatics
MLOGIC(PRINTTO):  %LET (variable name is PRINTO)
SYMBOLGEN:  Macro variable PRINTO resolves to clin-info
SYMBOLGEN:  Macro variable UNDERSCRE resolves to _
SYMBOLGEN:  Macro variable ADJ resolves to informatics
MLOGIC(PRINTTO):  %PUT _user_
PRINTTO I 2
PRINTTO UNDERSCRE _
PRINTTO ADJ informatics
MAJORLOOP FL first
MAJORLOOP INVENTORY clin-info informatics
GLOBAL PRINTO clin-info_informatics
GLOBAL RUNTYPE ProjectSums
GLOBAL ONEINV1 clin-info
GLOBAL ONEINV2 informatics
GLOBAL DTE 140724
GLOBAL CNTINV 2
MLOGIC(PRINTTO):  %DO loop index variable I is now 3; loop will not iterate again.
SYMBOLGEN:  Macro variable RUNTYPE resolves to ProjectSums
SYMBOLGEN:  Macro variable PRINTO resolves to clin-info_informatics
SYMBOLGEN:  Macro variable RUNTYPE resolves to ProjectSums
SYMBOLGEN:  Macro variable DTE resolves to 140724
MPRINT(PRINTTO):   PROC PRINTTO LOG=
"/clin-info/informatics/SpaceManagement/ProjectSums/SpaceReport_clin-
info_informatics_ProjectSums140724.log" new;
MPRINT(PRINTTO):   RUN;
```