I Haven't Got a Lot of Time - I Haven't Got a Lot of Money
(Food For Thought For Penniless System Managers)

George T. Blessing
City of Pasadena
100 N. Garfield Ave, B-29
Pasadena, CA 91109-7215

There is a seemingly steady flow of new products designed to help the System Manager take care of his system. Some of us, however, don't have the funds to pay the big-ticket prices associated with some of these items. For those of us in this situation, this paper suggests a few simple (and relatively cheap) methods to help us do our jobs more effectively without spending a lot of money or time.

More specifically, I will deal briefly with three major concerns: Backup Strategies, Disc Space Management, and File Capacity Management. We've all been told repeatedly that attention to details is the most important concern a System Manager can have. But paying attention to details can be time-consuming. What I have done at our site is to try to develop strategies for system management which require the least amount of time and give the largest benefits. In this paper, I will outline a few of these strategies and provide the listings for processes I've written where applicable.

BACKUP STRATEGIES

One of the problems facing us was a shrinking window for our overnight processing. At our site, we do a Full Backup on Monday morning, and a Partial Backup on Tuesday-Friday mornings. The backup took 3 to 3.5 hours for the Full, and 1.5 - 2 hours for the Partial. We also had only one operator who was working a split shift: coming in as early as 4:00 in the morning and then coming in later to do other work at night. Since additional staffing was out of the question and we needed more open time at night and better use of the operator we had, I decided something would have to be done.

I decided that the first thing to do was shorten the backup time, by cutting it into two pieces. We had two 6250bpi tape drives, but TurboSTORE was not yet released. I split the backup in half by defining two files with explicit account names in them, and doing a backup to each tape drive of different accounts. The format of the file containing the account names looks like this:

```
@.@.ALDON
@.@.BUDGET
@.@.CENTRAL
@.@.CLERK
    :
```

```
    :
@.@.INFOSYS
@.@.LICENSE
```

I stored the lists in two files called ACCT7 and ACCT17 (7 and 17 are the LDEV numbers of our tape drives), and we began splitting the backup by running one on each of two terminals. The backup commands for the backup to the LDEV 17 tape drive are:

```
:FILE SYSLIST;DEV=LP
:FILE TP17;DEV=17
:SYSDUMP *TP,*SYSLIST
Enter dump date: 10/1/87
Enter Subset of files:
!ACCT17
List files dumped to printer: Y
```

This change produced terrific results. The full backup was split into two halves which took similar amounts of time. The length of the full backup dropped to 1:50. The partial the next day dropped to about 0:55. Then I realized that the moment's paradise was slipping away fast. I now had two files which had to be maintained manually, and if they got out of whack, part of the system would not be backed up.

So I thought to myself "I need a way for the machine to maintain these, or I'll get burned for certain." I decided it wouldn't take me long to build an account and forget to put it in. So I picked an HP command which provides a list of all the accounts on the system which can be dumped into a file, and edited. I wrote JCL to create this file, text it into the editor, change it to the format I needed, and split it into two pieces, saving them as ACCT7 and ACCT17. I ran the job, and after a few tries, the output was correct. I went home with the satisfaction of a job I thought was well done.

I returned to the office to find trouble. The partial backup had taken too long and users were about to start complaining. The problem proved the theory that most things that look simple are actually not. The split which worked so well for the Full Backup, had soon proven to be unbalanced for the Partial. Though it was not so obvious on Tuesday, by Wednesday, the difference between the two sets

was severe. Some accounts had more activity than others, and they happened to be on the same tape drive. So, I went back to the drawing board.

I modified the JCL for the job which sets up the files (ACCTJOB), changing it so that it did two splits, the first one for the expected balance for a full into FACCT7 and FACCT17, and the second one for partial backups into PACCT7 and PACCT17. I told the operator of the change. He was not thrilled that he would have to change his already typed documentation, and prepared for the worst. The changes worked well anyway, and after a couple tweaks to the partial side, it got even better. Here is the final listing of the JCL for ACCTJOB, with comments on what different commands are doing:

```
!JOB ACCTJOB,MANAGER.SYS
!OCSSTART
!COMMENT    THIS JOB EXAMINES THE ACCOUNTING STRUCTURE
!COMMENT    OF THE SYSTEM AND PREPARES EXPLICIT BACKUP
!COMMENT    LISTS, INSURING THAT NO ACCOUNT GETS LEFT
!COMMENT    OUT OF A BACKUP.  IT SHOULD BE RUN EACH DAY
!COMMENT    IN THE EVENING TO CAPTURE ANY ACCOUNTS BUILT
!COMMENT    DURING THE DAY.
!COMMENT
!COMMENT    THE FOLLOWING LINES PURGE THE CURRENT SPLITS.
!CONTINUE
!PURGE ACCTLIST.SYSOP.SYS
!CONTINUE
!PURGE PACCT7.SYSOP.SYS
!CONTINUE
!PURGE PACCT17.SYSOP.SYS
!CONTINUE
!PURGE FACCT7.SYSOP.SYS
!CONTINUE
!PURGE FACCT17.SYSOP.SYS
!COMMENT
!COMMENT    NOW, A REPORT IS DONE TO A TEMPORARY FILE.
!COMMENT
!FILE RLIST=RLIST;REC=-80,,F,ASCII;NOCCTL
!REPORT BUGGER.@,*RLIST
!COMMENT
!COMMENT    THE FILE IS TEXTED INTO THE EDITOR AND MODIFIED
!COMMENT    TO PRODUCE A FULL ACCOUNT LIST (ACCTLIST)
!COMMENT
!ED
 T RLIST
```

```
SET TIME=9000
CQ 9 TO "-" IN ALL
FQ 1
WHILE
  FQ "-"
  DQ *(*)/*(LAST)
CQ 1 TO "@.@." IN ALL
CQ " " TO "" IN ALL
DQ 1/2
:COMMENT   AT THIS POINT THE FULL LIST IS FINISHED.
:COMMENT
:COMMENT   MODIFY FAMISX REFERENCE TO EXCLUDE THE
:COMMENT   GROUP @.DATA.FAMISX.
CQ "@.@.FAMISX" TO "@.@.FAMISX-@.DATA.FAMISX" IN ALL
:COMMENT   SAVE THE FULL LIST
K ACCTLIST.SYSOP.SYS,UNN
:COMMENT   THE FIRST SPLIT IS FOR PARTIAL BACKUPS
:COMMENT   (PACCT7,PACCT17).
:COMMENT   THE ACCOUNT MENTIONED BELOW - FQ "acctname"
:COMMENT   WILL BE THE LAST ACCOUNT IN THE FIRST SET.
FQ FIRST
FQ "FAMTEST"
K PACCT7.SYSOP.SYS(FIRST/*),UNN
FQ FIRST
FQ "FAMTEST"
K PACCT17.SYSOP.SYS(*+1/LAST),UNN
:COMMENT   THE SECOND SPLIT IS FOR FULL BACKUPS
:COMMENT   (FACCT7,FACCT17).
FQ FIRST
FQ "LICENSE"
K FACCT7.SYSOP.SYS(FIRST/*),UNN
FQ FIRST
FQ "LICENSE"
K FACCT17.SYSOP.SYS(*+1/LAST),UNN
:COMMENT   CLEAR OUT THE BUFFER
DQ ALL
YES
:COMMENT   TEXT IN SET 2 OF THE PARTIAL (PACCT17)
T PACCT17.SYSOP.SYS
:COMMENT   ADD ONE LINE TO THE BEGINNING TO GET
:COMMENT   @.PUB.SYS AND @.DBLOG.@ ONTO THE FIRST
:COMMENT   TAPE.
A .5
@.PUB.SYS,@.DBLOG.SYS
//
:COMMENT   KEEP SET 2 OF THE PARTIAL (PACCT17)
K
YES
```

```
:COMMENT
:COMMENT    TEXT IN SET 2 OF THE FULL (FACCT17)
 T FACCT17.SYSOP.SYS
:COMMENT    ADD ONE LINE TO THE BEGINNING TO GET
:COMMENT    @.PUB.SYS AND @.DBLOG.@ ONTO THE FIRST
:COMMENT    TAPE.
 A .5
@.PUB.SYS,@.DBLOG.SYS
//
:COMMENT    KEEP SET 2 OF THE FULL (FACCT17)
 K
 YES
:COMMENT
:COMMENT    CLEAR OUT THE BUFFER
 DQ ALL
 YES
 EXIT
!COMMENT
!EOJ
```

For those of you who have a contributed library, I recommend the use of BULDACCT (sometimes in pub.telesup) at the end of this jobstream to build current files with the accounting/udc structures in them to be backed up in the morning. This eliminates another one of those things you always want to do but never get around to.

This change was a solution, but not a complete one. I still wanted to get more mileage from my operator on Partial days. I examined the listings from the Partial backup, and after a few minutes, several items drew my immediate attention. Big datasets. Big datasets that got backed up every partial. I mean, datasets over 200,000 sectors a piece. So I picked up my expensive analytical tools (calculator and highlighter) and went to work. It shouldn't have been a surprise, but it was. Most of the backup for the first day or so was made up of datasets from the major applications.

Well, if you've been thinking about Image/TurboImage logging, and data recovery hasn't been enough of an issue to get you to do it, here are two more reasons to do it. First, if you're going to log a database and create that extra overhead, you may as well turn on Autodefer, since the log is written to disk whether Autodefer is on or not. Also, most people who didn't pay attention to the manuals or classes haven't noticed that performance with logging and Autodefer enabled is actually <u>better</u> than without it.

Second (and I must admit that this was my primary motivation), if you backup the log files on the partial, you don't need to backup the database itself. Now if your databases aren't much of your system, this will be meaningless, but if they are, think of the difference it would make. I did, twice, because I couldn't believe it the first time. Think of it. A 400,000 sector dataset receives 10 DBPUT's to it on Monday afternoon, and you're backing up the whole thing on Tuesday's partial. On a 6250bpi tape drive that's about 2/3 of one tape! Well, maybe I'm not a genius, but it didn't take me long to figure out that this would be the answer I was looking for.

So we tried it on two databases, then four, then nine. It was a hit. Response time on some database operations got faster, the backup got even shorter, and the night processing window grew to a fat 13 hours. We developed templates for different tasks to make the logging easier to manage and prevent us from being unprepared in case of corruption.

First, we developed a job stream to do a roll-forward recovery for one database. For each one of the databases which is logged, there is a copy of this job stream with the information for that database filled in. This enables us to recover by restoring the the database from the last full backup tapes and then applying the best available copy of the logfile against it by streaming the recovery job. Here is the template of the recovery file:

```
!JOB dbname,dbcreator.dbaccount,dbgroup
!COMMENT    THIS JOB RECOVERS THE DATABASE FROM
!COMMENT    THE LAST FULL BACKUP BY APPLYING TRANSACTIONS
!COMMENT    CONTAINED IN THE CURRENT LOG FILE.
!COMMENT
!COMMENT    FIRST SET FLAGS IN THE DATABASE
!RUN DBUTIL.PUB.SYS
 DISABLE dbname FOR ACCESS
 ENABLE dbname FOR RECOVERY
 EXIT
!COMMENT
!COMMENT    NOW, PERFORM THE RECOVERY
!RUN DBRECOV.PUB.SYS
 CONTROL NOSTAMP,NOSTORE
 RECOVER dbname
 RUN
```

```
!COMMENT
!COMMENT    THEN SET FLAGS BACK IN THE DATABASE
!RUN DBUTIL.PUB.SYS
 ENABLE dbname FOR ACCESS
 DISABLE dbname FOR RECOVERY
 EXIT
!COMMENT    DATABASE IS READY TO BE LOGGED AGAIN
!EOJ
```

We prepared jobstreams to handle the starting, restarting, and stopping of the logging processes, in order to reduce the error possibility. You will notice references in all three to the "existence" of a particular file. To allow the jobs to start logging processes, they need to be granted the capability of using the LOG command. To do this we have the jobs start, send a message to the console, and then wait for the existence of a particular file. When the operator sees the message, he enters the command SYSALLOW, which allows the job the LOG command and then builds the file. Here is are the listings of the three jobs, LOGSTRT, LOGRSTRT, and LOGSTOP:

```
!JOB LOGSTRT,OPRSM.SYS
!PURGE SYSALLOW.DBLOG
!COMMENT    THIS JOB STARTS ALL LOGGING PROCESSES
!COMMENT    IT MUST BE RUN ONLY AFTER THE FULL BACKUP!
!COMMENT
!COMMENT
!COMMENT    THERE MUST BE TWO LINES PER LOGGING PROCESS
!COMMENT    INSERTED BELOW IN THE FOLLOWING FORMAT:
!COMMENT     !FILE P=dbname.DBLOG.dbaccount;SAVE
!COMMENT     !PURGE *P
!COMMENT
!COMMENT ***** Account: CLERK *****
!FILE P=recdc.DBLOG.clerk;SAVE
!PURGE *P
!COMMENT
!COMMENT ***** Account: CP *****
!FILE P=cpdl.DBLOG.cp;SAVE
!PURGE *P
!COMMENT
!COMMENT ***** Account: ENG *****
!FILE P=pwrdf.DBLOG.eng;SAVE
!PURGE *P
!COMMENT
!COMMENT ***** Account: LICENSE *****
```

```
!FILE P=fal1db.DBLOG.license;SAVE
!PURGE *P
!FILE P=fbl1db.DBLOG.license;SAVE
!PURGE *P
!FILE P=fbl2db.DBLOG.license;SAVE
!PURGE *P
!COMMENT
!COMMENT ***** Account: PASADENA *****
!FILE P=citsdb.DBLOG.pasadena;SAVE
!PURGE *P
!COMMENT
!COMMENT ***** Account: PMSPAS *****
!FILE P=bsprdb.DBLOG.pmspas;SAVE
!PURGE *P
!FILE P=bssidb.DBLOG.pmspas;SAVE
!PURGE *P
!COMMENT ***** End of File Maintenance *****
!COMMENT
!TELLOP   ************************************§
!TELLOP   *   YOU HAVE 3 MINUTES TO ISSUE THE   *
!TELLOP   *   FOLLOWING COMMAND OR THE LOGGING  *
!TELLOP   *   WILL NOT START.......             *
!TELLOP   *           :SYSALLOW                 *
!TELLOP   ************************************
!COMMENT   THE FOLLOWING PROGRAM CHECKS FOR EXISTENCE
!COMMENT   OF THE FILE SYSALLOW.DBLOG.SYS EVERY 15 SECS
!COMMENT   FOR 3 MIN.   IF FOUND THE PROCESS CONTINUES.
!RUN FILEINQP.COMP.OCS;PARM=15;INFO="12,SYSALLOW.DBLOG"
!PURGE SYSALLOW.DBLOG
!COMMENT   THERE MUST BE ONE LINE PER LOGGING PROCESS
!COMMENT   INSERTED BELOW IN THE FOLLOWING FORMAT:
!COMMENT         !LOG dbname,START
!COMMENT
!LOG recdc,START
!LOG fal1db,START
!LOG fbl1db,START
!LOG fbl2db,START
!LOG citsdb,START
!LOG bsprdb,START
!LOG bssidb,START
!LOG cpdl,START
!LOG pwrdf,START
!EOJ

!JOB LOGRSTRT,OPR.SYS
!PURGE SYSALLOW.DBLOG
!COMMENT   THIS JOB RESTARTS ALL LOGGING PROCESSES
!COMMENT   IT MUST BE RUN AFTER THE PARTIAL BACKUP!
```

```
!COMMENT
!TELLOP   ************************************§
!TELLOP   *   YOU HAVE 3 MINUTES TO ISSUE THE   *
!TELLOP   *   FOLLOWING COMMAND OR THE LOGGING  *
!TELLOP   *   WILL NOT RESTART.....             *
!TELLOP   *          :SYSALLOW                  *
!TELLOP   ************************************
!COMMENT   THE FOLLOWING PROGRAM CHECKS FOR EXISTENCE
!COMMENT   OF THE FILE SYSALLOW.DBLOG.SYS EVERY 15 SECS
!COMMENT   FOR 3 MIN.   IF FOUND THE PROCESS CONTINUES.
!RUN FILEINQP.COMP.OCS;PARM=15;INFO="12,SYSALLOW.DBLOG"
!PURGE SYSALLOW.DBLOG
!COMMENT
!COMMENT   THERE MUST BE ONE LINE PER LOGGING PROCESS
!COMMENT   INSERTED BELOW IN THE FOLLOWING FORMAT:
!COMMENT         !LOG dbname,RESTART
!COMMENT
!LOG recdc,RESTART
!LOG fal1db,RESTART
!LOG fbl1db,RESTART
!LOG fbl2db,RESTART
!LOG citsdb,RESTART
!LOG bsprdb,RESTART
!LOG bssidb,RESTART
!LOG pwrdf,RESTART
!LOG cpdl,RESTART
!EOJ

!JOB LOGSTOP,OPR.SYS
!PURGE SYSALLOW.DBLOG
!COMMENT   THIS JOB STOPS ALL LOGGING PROCESSES
!COMMENT   IT MUST BE RUN BEFORE FULL OR PARTIAL BACKUP!
!COMMENT
!TELLOP   ************************************§
!TELLOP   *   YOU HAVE 3 MINUTES TO ISSUE THE   *
!TELLOP   *   FOLLOWING COMMAND OR THE LOGGING  *
!TELLOP   *   WILL NOT SHUT DOWN...             *
!TELLOP   *          :SYSALLOW                  *
!TELLOP   ************************************
!COMMENT   THE FOLLOWING PROGRAM CHECKS FOR EXISTENCE
!COMMENT   OF THE FILE SYSALLOW.DBLOG.SYS EVERY 15 SECS
!COMMENT   FOR 3 MIN.   IF FOUND THE PROCESS CONTINUES.
!RUN FILEINQP.COMP.OCS;PARM=15;INFO="12,SYSALLOW.DBLOG"
!PURGE SYSALLOW.DBLOG
!COMMENT
!COMMENT   THERE MUST BE ONE LINE PER LOGGING PROCESS
!COMMENT   INSERTED BELOW IN THE FOLLOWING FORMAT:
!COMMENT         !LOG dbname,STOP
```

```
!COMMENT
!LOG recdc,STOP
!LOG fa11db,STOP
!LOG fb11db,STOP
!LOG fb12db,STOP
!LOG citsdb,STOP
!LOG bsprdb,STOP
!LOG bssidb,STOP
!LOG pwrdf,STOP
!LOG cpd1,STOP
!EOJ
```

To keep the logged databases themselves from being backed up during a partial backup, I created a jobstream which prevents them from being picked up by sysdump. Basically, the jobstream requests a store of the chosen files. The operator streams this job before doing the real partial, but never replies to the request for LDEV number for the tape. This keeps the files from being backed up by the other processes as they are in use by STORE. When the partial is completed, the operator replies to the request with an LDEV number of 0, and the jobstream terminates. The list of files to be kept from backup is kept in a separate file which can be edited. For those who have HPDESKMANAGER on their system, if you do a MAILMAINT each night, you may choose not to backup the mail databases in the morning (the MAILMAINT job does a store of the databases before the mailmaint takes place). Here is the listing of the JCL for the lockdown job, DBLLOCK, and its configuration file, DBLLIST:

```
!JOB DBLLOCK,OPR.SYS
!TELLOP   ***********************************************
!TELLOP   *  THIS HAD BETTER BE A PARTIAL!§ BACKUP...    *
!TELLOP   *  IT IS OK TO BEGIN THE REAL PARTIAL BACKUP   *
!TELLOP   *  ONCE THE REQUEST FOR "LOCKTAPE" HAS COME    *
!TELLOP   *  FROM THIS JOB...                            *
!TELLOP   ***********************************************
!TELLOP   *  IF THIS ISN'T A PARTIAL, REPLY '0' TO THE   *
!TELLOP   *  REQUEST FOR "LOCKTAPE" NOW.                 *
!TELLOP   ***********************************************
!TELLOP   *  AFTER THE PARTIAL BACKUP HAS COMPLETED,     *
!TELLOP   *  REPLY '0' TO THE REQUEST FOR "LOCKTAPE".    *
!TELLOP   ***********************************************
!FILE LOCKTAPE;DEV=TAPE
!STORE !DBLLIST;*LOCKTAPE;SHOW
```

```
!EOJ


RECDC@.PUB.CLERK
FAL1DB@.DATA.LICENSE
FBL1DB@.DATA.LICENSE
FBL2DB@.DATA.LICENSE
CITSDB@.PUB.PASADENA
BSSIDB@.DATA.PMSPAS
BSPRDB@.DATA.PMSPAS
@.@.HPOFFICE
```

So, with the implementation of the split backup, image logging, and no backups of logged databases on partial days, the full backup now takes about 1:45 to complete, and the partials vary from 20 - 30 minutes. We do a full backup now on Monday, since it has the largest processing window in front of it, and partial backups on Tuesday through Friday.

DISC SPACE MANAGEMENT

The next problem I tackled is a difficult one to deal with: The use/abuse of disc space. This is particularly tough in a development environment where people are leaving their sometimes not-so-little test files around for posterity. It's not possible for you to watch everything that goes on, so let the machine do everything possible for you. I don't want to plug anyone's software, but I confess that a great deal of this function is being handled by MPEX from Vesoft.

I use two jobstreams to monitor the appearance of files on the system: one that alerts me to new files being built, and a second one that lists files not accessed. The first job logs on at 2AM on Saturday morning, and does a special LISTF of all files created during the past week. You should try it just to see what shows up. Huge files with no records often appear with misspelled names that seem to elude their creator's dustpan.

```
!JOB NEWFILES,MGR.OCS
!COMMENT  THIS JOB LISTS ALL FILES CREATED IN THE LAST
!COMMENT  WORK WEEK WHICH ARE LARGER THAN 500 SECTORS.
!FILE PP;DEV=PP;ENV=LP4.HPENV.SYS
!CONTINUE
!RUN MPEX.PUB.VESOFT
```

```
SET DEFAULT,LISTF,!
LISTF @.@.@(CREDATE>=*-7 & DISCSPACE>500),2;*PP
EXIT
!EOJ
```

Then at around 3AM the second job logs on. It also uses MPEX and does a listf of all files that have not been accessed in the last 90 days. This listing often reveals only items which need to be archived, but occasionally a choice find appears. It's a good practice to get into, but again, one which takes time to run and time to review. So let the machine at least take care of running it while you're not around.

```
!JOB OLDFILES,MGR.OCS
!COMMENT  THIS JOB LISTS ALL FILES NOT ACCESSED IN THE
!COMMENT  PAST 90 DAYS.
!FILE PP;DEV=PP;ENV=LP2.HPENV.SYS
!CONTINUE
!RUN MPEX.PUB.VESOFT
SET DEFAULT,LISTF,!
LISTF @.@.@(ACCDATE<*-90),2;*PP
EXIT
!EOJ
```

I also believe that it's important to watch for sudden changes in freespace on your system. The Hewlett-Packard utility FREE5.PUB.SYS (as we all know) is supplied with the fundamental operating system and is quite capable of creating its own instant I/O bottleneck when running. The output of this program can be useful in determining how fragmented the freespace on your discs is or the amount of total freespace on the system. I have found that having a record of the total freespace can help determine what has been happening on the system. Obviously, if the amount of free sectors drops by 1.2 million, someone is building some hefty files. To facilitate having the operator record the freespace on a daily basis, I wrote a jobstream to supply him with only the information I was interested in.

```
!JOB FREEJOB,MGR.OCS
!CONTINUE
!PURGE FREE5OUT
!BUILD FREE5OUT;REC=-80,,F,ASCII;NOCCTL
```

```
!RUN FREE5.PUB.SYS;STDLIST=FREE5OUT
!IF JCW < FATAL THEN
!  CONTINUE
!  ED
T FREE5OUT
F FIRST
WHILE
  FQ "LARGEST"
  DQ */*+7
DQ 1
CO LAST-1 TO 1
A .5
          SYSTEM FREE SPACE
//
CQ 1 TO ":TELLOP " IN ALL
K TEMPUSE,UNN
USE TEMPUSE
:PURGE TEMPUSE
DQ ALL
YES
EXIT
!ENDIF
```

This jobstream runs FREE5.PUB.SYS and directs the output to a file. It then runs EDITOR.PUB.SYS and massages the discspace information until it becomes a series of :TELLOP commands. The altered file is saved in the file TEMPUSE, then USEd by the editor to send messages to the console showing only the total freespace for each disc drive and the system as a whole. Although this output is rather rudimentary and was intended for a written record, if you have access to a PC with a spreadsheet application, time comparisons can be made. I am hoping to write an application later in this year to go a little further and evaluate the condition of the discs based on the fragmentation information as well as the freespace.

FILE CAPACITY

The last thing I'd like to discuss is database capacities. If you are involved in a small shop, it's very likely that as the System Manager you work many different jobs. Probably you are also acting as the Database Administrator. There is very little (with the possible exception of a system failure) that annoys me as much as having a dataset fill up unexpectedly. Our system has a great many small

applications running on it, producing an endless supply of databases. Keeping track of how full each of them is could be a full-time job in itself. In fact, on our system just keeping track of where they are could be a full-time job. So while I was wearing the DBA hat one day I decided it would be wonderful if the HP3000 could just tell me when sets were filling up.

Evidently, other people were thinking the same thing because I've noticed at least one commercial product that seems to do this to some degree. But my no-cost procedure at least gives me a list of the sets which have passed a particular threshold. Hewlett-Packard provided the means to accomplish this quite a while back, perhaps without even knowing it. When they introduced TurboIMAGE, they also introduced a special group in the SYS account called CONVALL. For those of you who haven't checked it out, I suggest that you do. It contains several goodies like TMPCONVP, a program which lets you change your session log-on without losing your capabilities or UDC's. Aside from that, it also contained a utility which searches the system, looking for databases, and creates a JCL from a supplied template file for each one it finds. Its original use was to convert each DB to TurboIMAGE. The CONVALL programs were well-documented and is able to insert variables such as the creator, passwords, etc. into the template you supply. I imagine that this utility is still available with some coaxing from your S.E. or may be on one of your old tapes.

I decided to use this utility in combination with a contributed program called DBANALYS. DBANALYS can provide a list similar to FORM SETS in QUERY, but also shows the percentage of each dataset in use. I know there are other contributed programs floating around which provide similar output. I've been using this one for some time now and have stuck with it. I wrote a sequence of jobstreams which essentially performs three tasks: 1) find all of the databases on the system and produce JCL to run DBANALYS against each database, appending the output from DBANALYS to a single file 2) launch the new JCL 3) massage the output from DBANALYS and produce a report. Below are the listings of DBCAPJ1, DBCAPJ3, and a sample report from the process.

```
!JOB DBCAPJ1,MANAGER.SYS,SYSOP
!OCSSTART
!COMMENT  ********************************************
!COMMENT  *  THIS JOB IS THE FIRST OF THREE JOBS USED  *
```

```
!COMMENT  *  TO CHECK FOR DATASETS OVER 85% FULL ON     *
!COMMENT  *  THE SYSTEM...                             *
!COMMENT  ********************************************
!COMMENT
!COMMENT  ********************************************
!COMMENT  *  FIRST, PURGE EXPENDABLE FILES...          *
!COMMENT  ********************************************
!PURGE DBCAPX.SYSOP
!PURGE DBCAPJ2.SYSOP
!PURGE JCONVERT.SYSOP
!PURGE TEMPLATE.SYSOP
!COMMENT  ********************************************
!COMMENT  *  THEN, BUILD NEW FILES...                  *
!COMMENT  ********************************************
!BUILD DBCAPX;REC=-80,128,F,ASCII;DISC=100000,32,1
!RELEASE DBCAPX.SYSOP
!COMMENT  ********************************************
!COMMENT  *  BUILD THE TEMPLATE FILE...               *
!COMMENT  ********************************************
!ED
A
|RUN TMPCONVP.CONVALL.SYS
\FILE,\LOGON
|FILE DBCAPX=DBCAPX.SYSOP.SYS;ACC=APPEND
|CONTINUE
|RUN DBANALYS.UTIL.SYS;STDLIST=*DBCAPX
BASE
\FNAME
;
5
SETS
EXIT
//
CHANGE "|","!" IN ALL
:COMMENT  *  NOW KEEP THE TEMPLATE FILE...            *
K TEMPLATE,UNN
EXIT
!COMMENT  ********************************************
!COMMENT  *  FIND ALL OF THE DB'S...                   *
!COMMENT  ********************************************
!BUILD JCONVERT;REC=-80,128,F,ASCII;DISC=100000,32,1
!FILE TEMPLATE.CONVALL.SYS=TEMPLATE.SYSOP.SYS
!RUN CONVALL.CONVALL.SYS
@.@
!FILE NEWCON=JCONVERT,OLD
!FILE OLDCON=JCONVERT,OLDTEMP
!FCOPY FROM=*OLDCON;TO=*NEWCON
!PURGE JCONVERT,TEMP
```

```
!COMMENT  ********************************************
!COMMENT  *  CREATE THE JOBSTREAM DBCAPJ2...          *
!COMMENT  ********************************************
!ED
T JCONVERT
A.5
|JOB DBCAPJ2,MANAGER.SYS,SYSOP
|OCSSTART
//
A
|RUN TMPCONVP.CONVALL.SYS
DBCAPJ2,MANAGER.SYS,SYSOP
|IF JCW < FATAL THEN
|SCHEDULE "DBCAPJ3"
Y
|OCSNORMAL
|SET STDLIST=DELETE
|ELSE
|OCSABNORM
|ENDIF
|EOJ
//
CHANGE "|" TO "!" IN ALL
K DBCAPJ2,UNN
EXIT
!COMMENT  ********************************************
!COMMENT  *  READY FOR DBCAPJ2 TO EXECUTE...         *
!COMMENT  ********************************************
!IF JCW < FATAL THEN
!SCHEDULE "DBCAPJ2"
Y
!OCSNORMAL
!SET STDLIST=DELETE
!ELSE
!OCSABNORM
!ENDIF
!EOJ

!JOB DBCAPJ3,MANAGER.SYS,SYSOP
!OCSSTART
!ED
 T DBCAPX
 SET TIME=9000
 CQ 67 TO "|" IN ALL
:COMMENT ************************************************
:COMMENT ** THE FOUR LINES FOLLOWING ENSURE REPORTING OF **
:COMMENT ** THE CAPACITY OF THE ITEM-HEADER SET IN HPDESK *
:COMMENT ** WITHOUT RESTRICTION BY PERCENTAGE           **
```

```
FQ FIRST
F "LOCAL"
F "ITEM-HEADER"
CQ "|","" IN *
:COMMENT ************************************************
 CQ "|8 ","|0" IN ALL
 CQ "|9 ","|0" IN ALL
 CQ "|80","|0" IN ALL
 CQ "|81","|0" IN ALL
 CQ "|82","|0" IN ALL
 CQ "|83","|0" IN ALL
 CQ "|84","|0" IN ALL
 CQ "|8","8" IN ALL
 CQ "|9","9" IN ALL
 CQ "| ","  " IN ALL
 FQ FIRST
 WHILE
   FQ "|"
   DQ *
 FQ FIRST
 FQ "EXIT"
 COPY *-1/* TO 1
 COPY 1 TO .5
 CHANGE "->" TO "REPORT OF DATASETS WITH LOADING >85% OF CAPACITY"
 FQ FIRST
 WHILE
   FQ "->"
   DQ */*+13
:COMMENT NEW STUFF ADDED 12/16/87
 C 26 TO "|" IN ALL
 FQ FIRST
 WHILE
   FQ " |"
   DQ *
 C "|" TO "" IN ALL
 C 5 TO "|" IN ALL
 FQ FIRST
 WHILE
   FQ " |"
   BEGIN
     CQ "SET|S" TO "|SET|S" IN *-1
     FQ *+2
     END
 FQ FIRST
 WHILE
   FQ " SET|S"
   DQ *
 C "|" TO "" IN ALL
```

```
 L ALL,UNN,OFFLINE
 EXIT
!IF JCW < FATAL THEN
!OCSNORMAL
!SET STDLIST=DELETE
!ELSE
!OCSABNORM
!ENDIF
!EOJ

REPORT OF DATASETS WITH LOADING >85% OF CAPACITY
 SETS  OF DATABASE : OBV.FGLSRC.BUDGET          MAY 26, 1988
  11  PERSONNEL        DETAIL      3000        2920       97
 SETS  OF DATABASE : PHONDB.DATA.CENTRAL
   6  ODX'M-IRN        MANUAL      809         726        90
   8  ODX'M-IRN-TREE   DETAIL      62          58         94
 SETS  OF DATABASE : SURP.DATA.CENTRAL
   5  RDETAIL          DETAIL      18984       16101      85
                              .
                              .
                              .
```

This process now runs at night once a week and enables us to review quickly what datasets are reaching the threshold and need attention without having to wade through a stack of printouts.

I have also felt somewhat remiss in the amount of attention which I can pay to the loading of the disc drives. As files tend to move if they are built on class DISC, it can be difficult to follow them around. I was particularly interested in datasets and the problems that develop when two related or heavily traveled sets reside on the same LDEV. When I started to evaluate them after the last reload, I stopped almost immediately. It is difficult to remember that in database AIDB the critical sets are 10, 14, and 15, and not 12, 16, and 17. After creating a paper nightmare and still not effectively solving the problem, I called on my old friend EDIT/3000 one more time. Since datasets are built on one LDEV (unless you mess with them), I reasoned that the output from LISTDIR5 with the MAP option would provide me with the LDEV number for any file I chose to list.

So, I began to work on this problem. I built a small editor file in which I placed the names of several datasets (e.g. AIDB04.PUB.SO) and a brief description of each set (to

remind me of their contents). I wrote a jobstream which texts in that file, and changes it into a series of commands for LISTDIR5. It then runs LISTDIR5 using the new file of commands as its $STDIN. The output from LISTDIR5 is captured in a new file, which is brought into the editor again. The editor massages the data, passes it through a sort by LDEV number and filename, and then on to a printed report. The report lists the files and their respective LDEV numbers and shows at a glance which files are on which disc drives. Below is a listing of the files DLCONFIG, DLJOB, and a sample report.

```
CITSDB01.PUB.PASADENA          M,CITATION
CITSDB02.PUB.PASADENA          A,LICENSE
CITSDB04.PUB.PASADENA          D,VEHICLE-ID-NO
CITSDB07.PUB.PASADENA          D,BAIL-DISPOSITION

FMMASTER.DATA.FAMIS            KSAM, FAMIS MASTER
FMMASKEY.DATA.FAMIS            KSAMK, FAMIS MASTER KEYFILE

CPDL05.PUB.CP                  A,M-CITE-KEY
CPDL07.PUB.CP                  A,M-COURT-KEY
CPDL13.PUB.CP                  A,M-IRN
CPDL20.PUB.CP                  D,CDISPO
CPDL21.PUB.CP                  D,CHARGES
CPDL23.PUB.CP                  D,CHISTORY
CPDL26.PUB.CP                  D,COURT
CPDL28.PUB.CP                  D,MASTER
```

```
!JOB DLJOB,MANAGER.SYS,SYSOP
!PURGE DLIN
!PURGE DLOUT
!PURGE DLUSE
!ED
SET TIME = 9000
T DLCONFIG.SYSOP.SYS
CQ 2 TO "|" IN ALL
FQ FIRST
WHILE
  FQ " |"
  DQ *
CQ "|","" IN ALL
CQ 31/72 TO "" IN ALL
CQ 1 TO "LISTF " IN ALL
A
EXIT
```

```
//
K DLIN,UNN
EXIT
!COMMENT
!BUILD DLOUT;REC=-72,7,F,ASCII;DISC=1000
!RUN LISTDIR5.PUB.SYS;STDIN=DLIN;STDLIST=DLOUT
!COMMENT
!ED
SET TIME=9000
T DLOUT
CQ 2 TO "|" IN ALL
FQ FIRST
WHILE
  FQ " |"
  DQ *
CQ "|" TO "" IN ALL
FQ FIRST
WHILE
  FQ "FCODE"
  DQ * / "DISC DEV"
FQ FIRST
WHILE
  FQ "DISC TYPE"
  DQ * / ">"
FQ FIRST
WHILE
  FQ "LISTF"
  DQ * / *+1
CQ "*"," " IN ALL
CQ 3 TO "|" IN ALL
FQ FIRST
WHILE
  FQ " |"
  DQ *
CQ "|","" IN ALL
CQ " #"," " IN ALL
FQ FIRST
WHILE
  FQ "ACCESSED:"
  DQ *(*) / *(72)
CQ " :","                                    " IN ALL
CQ "FILE: ","+" IN ALL
G ALL
DQ 1/2
DQ LAST
K
EXIT
!ED
```

```
SET TIME=9000
T DLOUT
FQ FIRST
WHILE
  FQ "+"
  DQ *
CQ 32 TO \C 31 TO "\ IN ALL
CQ 43 TO \" IN *+1\ IN ALL
K DLUSE,UNN
T DLOUT
CQ 2 TO "|" IN ALL
FQ FIRST
WHILE
  FQ " |"
  DQ *
CQ "+|","" IN ALL
A .5

//
FQ FIRST
USE DLUSE
K
EXIT
!FILE DLLIST;DEV=LP
!RUN SORT.PUB.SYS
I DLOUT
O *DLLIST
K 32,1
K 31,1
K 1,8
V
END
!TELL MANAGER.SYS;DLJOB COMPLETED!
!EOJ


FMMASTER.DATA.FAMIS             1
CITSDB04.PUB.PASADENA           3
FMMASKEY.DATA.FAMIS             3
                .
                .
                .
CITSDB01.PUB.PASADENA           9
CITSDB07.PUB.PASADENA           12
CITSDB08.PUB.PASADENA           12
CITSDB02.PUB.PASADENA           13
```

I realize that my methods are no panacea, and could be easily expanded upon. I'm certain that those who pick up where I have stopped will continue to evolve some of these ideas into even more efficient and practical tools, and I expect mainly to spark the curiosity and imagination of others.