ZEN
AND THE ART OF
SOFTWARE MAINTENANCE

MARC COVITT
SYSTEMS ENGINEER
HEWLETT-PACKARD

PURPOSE:

This paper and the associated presentation will discuss the issues of Software Maintenance. The author will present an opinion that many types of software problems can be prevented by exercising regular preventative steps. There is an analogy to the steps taken as part of hardware maintenance. The author will also discuss several utilities available to perform the maintenance function. Specific areas covered will include:

Programming Techniques
Disc Space Management
Data Base Checking

OVERVIEW:

All of us are aware of the concept of hardware maintenance. On your computer system, your Customer Engineer will perform two types of maintenance. One is Remedial Maintenance--- in other words, something is broken and it needs to be fixed. The other is called Preventative --- nothing is broken (hopefully) and we take the time to perform routine custodial care. The purpose of preventative maintenance is to minimize the occurence of remedial maintenance and to prolong the life of the equipment.

In addition to your computer system, consider the case of your car. (Yes, I know Eugene that it will be several days before you have one--- but consider it anyway.) When something goes wrong, you take it in to be remedied. Periodically, you should also take it in for routine preventative care--- at least to change the oil and give it a tune-up.

Having introduced the simple terminology, I now move on to my first point: --

PREVENTATIVE MAINTENANCE IS NOT JUST FOR HARDWARE.

It is logical and reasonable to extend the PM philosophy to our software as well. Periodically we should make routine checks on our programs and systems as well as the hardware. After all, if an application fails to perform and is inoperative for the user, it may not matter to the user if the cause was hardware or software--- the result is the same.

With regular PM's on our hardware, we do things like change the filters, clean the equipment, and verify that the equipment performs within certain known tolerances. It is not that easy to change the filters on software. First of all, it's not that easy to change anything on software. Secondly, most application software has not been designed to use any kind of filter, so there is nothing to check or change. So what do I mean by performing PM on our software and systems.

First, lets look at the way software breaks. In other words what kind of remedial care do we provide our systems. Then we can look at ways of preventing these breaks and prolonging the running life (UP-TIME) of our systems.

WHAT CAN GO WRONG?

Murphy's Law says that whatever can go wrong, will! The correlary to Murphy's Law is "Murphy was an optimist".

In the complex world of systems almost anything can go wrong. This can include:

HUMAN ERROR - This includes error at any
from level and it can be any area, whether it
oper is programming, ations or the user.

HARD WARE - This includes hardware-huma induced errors or the n errors that arose from when we tried to recover an actual hardware problem.

DISC SPACE - Who out there has never had syst their program or em fail due to a lack incl of disc space. This also udes those cases beca where the program aborts use the al-loted file size was not big enough.

PROG RAM ERROR - This can include a this variety of things, but in instance, I am assu talking about program mptions such as inte "my program will never be rrupted".

WHAT IF IT'S NOT THE HARDWARE?

When something catastrophic happens (or even on minor interruption) it is almost always easier on the operations and programming personnel when the problem turns out to be hardware. After all, then "it's not my fault". You call the CE and wait until the problem is resolved. How many of us have been in a position where the machine went down because a room overheated due to an air-conditioning failure. Then not only do your operations personnel have someone else to blame, but so does your harware service representative.

On the other hand, if it turns out to be a specific operational problem --- such as the operator mounted the wrong tape or didn't follow the new procedures (that you just wrote neatly on the back of an envelope)--- then the operator suffers a tremendous ego loss. The same can occur to a programmer, although we adjust to that differently. You notice that we call any error a "BUG" --- as though some how it just crept in. We rarely say that there was an error on the part of the designer/developer/programmer. Instead we remark that we just "found a bug" as if we should be congratulated for our detective work instead of castigated for our failures in the development process. Ah well, the ego is a wonderful thing.

In any event, if we have an error it still will cause some loss of productive time on the part of the user, operations, programming, or all of them.

It is not my contention that we can eliminate all software errors though some magical process I call the PM. We don't solve all hardware problems though PM's. Besides the subject of producing better or error-free code is a whole separate topic that I have neither the interest nor inclination to address here.

Instead, let's focus on some simple steps to avoid the late night phone calls. After all, who wants to be interrupted from an amorous evening with your date by a phone call from the second shift operator telling you that the weekly MRP run blew off in the middle and he/she doesn't have a clue as to how to fix it. This leads me to my next thesis statement:

AN OUNCE OF PREVENTION IS WORTH A POUND OF LATE NIGHT PHONE CALLS

Actually there are many ways to avoid these calls. One of my customers simply leaves his phone off the hook at night. However there may be some other things that can be done to brighten up the picture. However, before I get into ways to fix things, I should point out that I do not advocate that you go back and apply these techniques to all existing jobstreams. After all, everyone has a million things that they would like to do to older systems but there is never the time or resources to accomplish that. Moreover, as Casey Stengel is reputed to have said --- "IF IT'S NOT BROKEN, DON'T FIX IT!"

BUILD MAINTAINABILITY INTO THE PRODUCT

Rather than try to develop procedures ex post facto, why not design them in at the start. Some examples of this include:

* CHECKPOINTS Consider designing some checkpoints into your programs and jobstreams. Checkpoints are places where the program/job identifies that it has completed a certain portion of the task and COULD be interrupted. This is very useful (and needed) on systems that will run a long time. Consider the frustration that occurs when a long program is running: Usually no one knows when it will be done but they don't wish to abort it because they will just have to start again. Say, for example, a DBLOAD. This is an example of a program from HP that probably should be checkpointed.

A good example from HP of this concept is the MM jobstreams. Many of these have checkpoints and a checkpoint/restart procedure to go with them.

* ASSUME THE WORST Remember Murphy. Programmers should not assume that everything is OK, especially if it would be very easy to make a verification check.

* PROGRAMMATICALLY CHECK FILE/DB CAPACITY Programmers should get in the habit of developing a simple routine to check the database capacity (use DBINFO) at the start of the program. Most of us have seen some application program that runs along until -- OOPS, the data base capacity is reached and the next record causes the program to abort. Usually there is little the operator/programmer can do except RESTORE the database to before the program ran and do some recovery. This involves expanding the database and then rerunning the job. It would be a lot simpler, if the program stopped before that problem

occurred and notified the operator to take appropriate action and then the program would pick up where it left off.

* HEURISTIC JCL There are examples of where the JCL learns what to do based on previous runs. For example, the MM/3000 MRP maintains a record of the size of the file the last few times the program was run. Based on that number and based on the transaction file, it establishes the file size requirements for all new files at run-time. This gets around the problem of creating a file that is too small as well as avoiding the problems created by always specifying a very large file and then only using a few records.

* GOOD DOCUMENTATION I shouldn't have to mention this, but the best way to make sure that your operators can solve problems is to provide quality documentation in a standardized format. Still, it always surprises me that many shops have little to no documentation.

* MANAGE YOUR DISC SPACE Sooner or later some program or system that you have will go "down the tubes" because you didn't have enough disc space. Moreover, it is really frustrating for programmers when they get blamed because the system ran out of disc space for a spooled report. Because this is such an important topic, I will get into this in more detail.

MANAGE YOUR DISC SPACE BEFORE IT MANAGES YOU

Like Zen, there are gurus of disc space management. Every one has his/her own unique mystical ways of addressing the problem. Usually, the problem arises in crisis. For example, it's a new month for the General Ledger and there is not enough room. All the players scurry around until someone astutely says "Let's purge Bill's account; he's on vacation this week."

Notwithstanding the shotgun approach, there are other ways to monitor and control your disc space. I would like to introduce some of the techniques of -

THE GENTLE ART OF DISC SPACE MANAGEMENT

LIMIT - Use the ;FILES=filespace parameter on the NEW/ALTACCT or NEW/ALTGROUP commands. If you are able to limit all groups and accounts so that the sum of all the limits is less than the total amount of disc space, you almost home free. Limiting still has its problems. First of all, you may be using a large amount of spool space, or there may be a lot of lost disc space (see LOST AND FOUND) so that you could still run out of available disc space. But the most obvious problem with limits is that sooner or later, some user will have his/her job aborted because they were the unlucky sole to run up against the limit. Because of this reason, I do not advocate the use of LIMITS except for accounts/groups which are not under central control and this is the only way you can keep these people in line. It is also good for the accounts you set up for users who want to "play" or use the system to do their homework for the programming course they are taking at the local night school.

MONITOR - A much more practical method is to begin to monitor the disc space so that you may take preventative action at an early stage. There are several ways to accomplish this. The quickest recap is to run FREE2 and monitor the amount of available space. By charting this data on a regular (at least once a week) basis you can get a good estimate as to when you are going to run into trouble. In conjunction with this, it is usually necessary to determine the answer to the question - "Who's using all the disc space?" or "Where the ** did all of our disc space go?" To determine who is using the disc space, you can use the REPORT command. This can be used in several ways.

1) REPORT of @.@ to a disc file and write a program to analyze the data and show you who has changed since the last report. (I think there is a contributed program that does this, but I can't remember which one. Also there are one or two third-party packages on the market that are supposedly capable of performing this monitoring function.) 2) REPORT to a hard copy device and analyze it by hand. Here's a simple trick to make that process easier. To get a quick recap at the account level just enter -

:REPORT ZZ@.@

This will get you a one line summary for each account as well as a breakdown for any groups that start with ZZ. Assuming that you have no groups that start with ZZ, you just get a summax for all the accounts. It is then reasonably easy to add these totals. Also if you have TDP you can use the TOTAL command as follows to have the system produce the total:

```
:RUN TDP.PUB.SYS
/:PURGE TOTALRPT
/:FILE TOTALRPT;SAVE;DEV=DISC;REC=-72,3,F,ASCII;NOCCTL
/:REPORT ZZ@.@,*TOTALRPT
/T TOTALRPT,UNN
/SET LEFT=13,RIGHT=22 << sets left/right to proper column >>
```

```
/TOTALQ 3/LAST,APPEND << ignore all the warning messages
about the line would be too long>>
/L LAST << last line now contains the total >>
```

LOST AND FOUND - (Or WHAT IS LOST DISC SPACE)

On a COOLSTART, you may have seen the prompt - RECOVER LOST DISC SPACE ? and wondered what it meant. When the system is running there are two file domains---system (permanent) and temporary. Temporary includes not only each user's temporary files but also the system spooled files. The system disc (or private volume) directory maintains entries only for permanent files. Temporary files for your session are maintained as part of the JDT (Job Directory Table) which resides in memory during the execution of your job/session. The system also maintains a separate table (IDD/ODD) for keeping track of spooled input and output files. As long as your session terminates and the spooled files are printed or deleted, you will not lose any file space.

When the system places a file on the disc, whether it is temporary or permanent, the disc's free space bit map is updated to reflect that the space is now used. When the file is purged, the free space map is also updated. However, if you have a system failure, OR if you RESTART THE SYSTEM (with other than a WARMSTART) and you had spooled files to be printed you will have lost disc space. In other words, the disc free space bit map indicates that the space is in use, but the system has no idea of what that space is used for and no way to get to it. It is therefore "LOST DISC SPACE". The only way to reclaim this is to answer YES to RECOVER LOST DISC SPACE? on a COOLSTART, or do a complete RELOAD. Depending on how long it has been since your last reload (or last RECOVERy of LOST DISC SPACE) this can amount to many sectors of lost space.

One way to tell how much lost disc space you have is to use the following formula:

LOST = TOTALAVAIL - (FREESPACE + INUSE + OVHD) where

TOTALAVAIL = Amount of usable formatted disc space based on the number and type of disc drives you have.

Use the following numbers:

DRIVE TYPE	# AVAIL SECTORS	FORMATTED CAP IN BYTES	LOST SECTOR DEFECTIVE *
7906	76,800	19,660,800	48
7911	109,824	28,114,944	0 (64)*
7912	256,256	65,601,536	0 (64)*
7914	516,096	132,120,576	0 (64)*
7920	195,600	50,073,600	48
7925	469,440	120,176,640	64
7933	1,579,916	404,458,496	0 (92)*

* CS80 drives have built in sparing for the first defective track. Otherwise use the number in parenthesis. This column shows how many sectors will be lost if you delete a specific track. If a track is reassigned, you do not lose any capacity---only the time spent for extra head movement.

FREESPACE = Total available as shown by FREE2.

INUSE = Sum of DISC space used as shown by REPORT.

OVHD = Constant for your specific configuration. This number allows for the system directory, virtual memory space, system disc tables, disc labels and free space maps. You have to compute this number after a RELOAD or COOLSTART (with RECOVER LOST DISC SPACE) so that you will be sure that LOST = 0.

Once you have computed the OVHD number it will remain constant as long as you don't change your configuration. The addition of a few IO devices will not affect this number significantly, but a change in the size of your directory or in the amount of virtual memory will have an impact.

If you now monitor your disc space, you can always make a quick check to determine if you have any lost disc space.

In addition to RECOVER LOST DISC SPACE and complete RELOAD, you can regain space on a private volume with the VINIT subsystem and the > COND ldn;RECOVER command.

OUT WITH THE OLD

The last option (besides buying more discs from your friendly HP sales representative) is to get rid of some of the unused files. Unfortunately it is usually hard to get a consensus as to what is "unused". There are several techniques to determine which files have not been accessed since a certain date. These include the :STORE command with the ;DATE <= parameter, and several utilities (LISTFXX from your friendly S.E. or MPEX from your friendly independent third-party vendor). These allow you to identify "old" files.

I recommend to my customers that they get in the habit of storing to tape all files which have not been used in 6 months and purging these from the system. Somehow everyone understands

the process, but no one seems to implement it on a regular basis. Oh well, I'm not really hurt--- I guess they just keep on buying ! more discs.

This brings us to Covitt's law of disc space which states:

"The amount of disc space used rapidly climbs until it reaches the total amount of usable disc space... AND STAYS THERE FOREVER. (Until more discs are added. Then it rapidly climbs until it reaches the total amount of usable disc space and stays THERE forever. Until ....)"

MAKE THE MOST OF WHAT YOU'VE GOT

Assuming that you cannot add more disc space (due to lack of budget or other critical resource), your best option is to try and use less disc space to hold your existing data. Assuming you have already tried to remove the old files. REMEMBER - (OLD FILES NEVER DIE --- THEY JUST TAKE UP DEAD DISC SPACE)

There are several ways to reduce the amount of space used by your existing files. However some of the methods have limitations or other operational drawbacks. But here's a sampling of things you can try:

REBLOCK existing files.
This allows you to continue your applications without change and may return some disc space. LISTFXX gives you an indication of how much space you could reclaim if you reblocked. However, this may not make a lot of sense if you have a job stream that recreates the file with the old block factors.

PURGE those dead files.
This has already been mentioned, but I want to emphasize that are some other files (like EDITOR/TDP K - files) that can be purged without even worrying about backing them up.

SCRUNCH files.
This is a utility that moves the LIMIT on the file back to the current EOF. It can be very helpful on files that change in size from run to run. Create the files with enough records (;DISC= nnnn,32,32) and then run SCRUNCH on the file immediately after it is closed. If you do your own FCLOSE on the file, you can do the same thing by making sure that the DISP parameter is set to return space beyond the EOF. However, this does not work well with KSAM files and any file that is to be later used with APPEND access.

COMPRESS files.
This utility stores the data in a compressed format, usually by doing blank compression. There is a corresponding utility to UNPRESS the files. This will usually result in a significant disc space saving, but will require that the file be expanded before it is accessed by almost any program. You might do just as well to ARCHIVE the file to tape.

Along with COMPRESS, you can save space by saving files in the VARIABLE format instead of FIXED format within the EDITOR. This will save a little space for JCL files by simply truncating the trailing blanks.

ARCHIVE files.
As mentioned above you can archive files to tape. Several utilities exist in the contributed library to manage this feature. These will typically be data files that won't be accessed for a while. Good candidates for this are the history files ---like the last seven years of sales data that marketing wants to keep ONLINE.

LIMIT YOUR DATABASE CAPACITY

Because the IMAGE database manager reserves all your allowable space immediately, you may have some wasted or at least available space within your databases. Because of the way that the space is originally allocated, the MPE file system thinks that the LIMIT and EOF are the same and that 100% of the space is in use. However, if you look within the database, you will often find that only a small portion of a particular dataset is actually in use.

It is not an easy task, however, to constantly play with the database capacities. If you have no outside utilities, you would be forced into a complete DBUNLOAD/DBLOAD. I would recommend that you waste disc space rather than your time in this case. However if you happen to have ADAGER or some of the utilities in the contributed library, you might try to allocate your disc space more conservatively and assume that you will be able to increase it if you need to. However, that may create some other problems down the road. This requires that you monitor your data base capacity and brings us to the next topic:

WATCH YOUR DATABASE CAPACITY

Even if you are not taking conservative approaches to allocate your data base capacity, sooner or later you will have to increase it. The question is not "IF" but "WHEN". In order to monitor this you can periodically run QUERY and do a FORM SETS on each database, or have a program perform this task. I have written a simple COBOL program (and contributed it to the library) called DBCAPCHK.

DBCAPCHK simply calls DBINFO for each database that you pass to it. It provides a simple report that shows percentage full for each dataset. It can easily be enhanced to start tracking some statistical data and develop trend analysis. Those modifications are left to the reader as an exercise. (I always hated that caveat in the math textbooks, but I thought I'd put it in here anyway!)

A RETURN TO SOFTWARE P.M.'s

Having digressed for a while onto the subject of disc space management, let's return to the generic subject of Preventative Maintenance for software. Actually watching your disc space is probably the single most important aspect of software PM's.

I would like to suggest to the reader that you implement some periodic jobstreams to perform the PM. These should be at least on a weekly basis. I would like you to establish a Job that includes

1) FREE2
2) REPORT ZZ@.@
3) MEMLOGAN
4) LOGERR (or IOERR) (formerly LOGUTIL)
5) DBCAPCHK
6) LISTFXX
7) PURGEK (JANITOR or something else that PURGES K-Files)

Recently HP has been introducing to most customers the TELESUP program and associated files. Many of the old-time "SE contributed" programs that were "unsupported" but essential have now been identified as supported utilities through the TELESUP program. Also included with these is a jobstream that includes FREE2, MEMLOGAN, TUNER5, and IOERR. The hardware folks have found that keeping a regular watch on your system will alert them to any significant problems.

My hope is that those of us concerned with the software side will extend this philosophy and also watch disc space and data base capacities. Also I'd like to reiterate my request that you look at archiving old files on a monthly basis.

SUMMARY

Yes, we've finally come to the end. I hope that the reader will take the thoughts and ideas included in this paper and develop a regular preventative jobstream for his/her shop. The long-term rewards from preventing major disasters and eliminating the late night phone calls will more than compensate for the start-up effort. So start doing your software PM's NOW!

For additional material on this topic, I suggest you read the article in these proceedings on the topic of managing your system written by my fellow S.E., John Vega.

BIOGRAPHY FOR

Marc Covitt Systems Engineer Hewlett-Packard 9606 Aero Drive San Diego, CA 92123 (619) 279-3200

Marc has been with Hewlett-Packard for 9 1/2 years in various positions. He has been a Systems Engineer for 2 1/2 years. Marc holds a bachelors degree from M.I.T. and a Masters in Computer Science from West Coast University. He was awarded the C.D.P. certificate in 1980.

He has been a regular contributor at International, Regional and Local User Group Meetings. He has had articles published in the HP User's Group Journal. Marc is an accomplished speaker and gives great talks.

In 1980, he was named by the International Users Group as the outstanding contributor to the Users Group within Hewlett-Packard.
