MPE log files for performance, security and debugging.

by Denys Beauchemin

1) Introduction

This paper will cover the different logging done by MPE on your behalf. We will review these log records, what they mean, and how to use the log records to analyse performance, help with system security and also on some small aspects of debugging.

We will examine a few true-to-life cases and try to set ourselves up for more control on the HP3000.

The following is a layout of the paper:
i) Log records, different types of.
ii) Impact of logging.
iii) Performance analysis with MPE logging.
iv) Enhanced security with MPE logging.
v) Debugging with MPE logging.
vi) Conclusion.

A final note before we get into the subject matter, you will find numerous references to a program called LOGANAL. This program is on the swap tape and is the primary tool used in the analysis of log files. The program was written by the author of the paper and contributed to the IUG thru the swap tape mechanism, and as such no implied or explicit guarantees and/or responsability is assumed by either the IUG and/or the author and his employer.

I) LOG RECORDS, DIFFERENT TYPES OF.

As of the CIPER mit, the following events could be logged by MPE, provided that these events were 'turned on'. For those of you who may not know, you 'turn on' the events at COLDLOAD time either from a tape created by SYSDUMP or on prompts from INITIAL.

Event	Description	
0	LOG FAILURE	Log failure record
1	HEAD RECORD	First record when system comes up
2	JOB INITIATION	Logon record
3	JOB TERMINATION	Logoff record
4	PROCESS TERMINATION	Process termination record
5	FILE CLOSE	File closure record
6	SYSTEM SHUTDOWN	System shutdown
7	POWER FAIL	Power failure record
8	SPOOLING	Spoolfile creation, mod, destruction
9	LINE DISCONNECTION	DSN line disconnect, DS, RJE, MRJE etc
10	LINE CLOSE	DSN line closes, DS, RJE, MRJE etc
11	I/O ERROR	Any error on any devices
12	VOLUME MOUNT	Physical private volumes mount/dismount
13	VOLUME SET MOUNT	Logical private volumes mount/dismount
14	TAPE LABELS	Tapes labels recognized by AVR
15	CONSOLE	Console log event record
16	PROGRAM FILE EVENT	Underflow simulation record
17	CALL PROGRESS SGNLS	
18	DCE PROVIDED INFO	

0- LOG FAILURE.

This record is written as soon as MPE logging is reeabled. It contains the following information. No of log records, no of logons & no of logoffs missing. MPE logging can be stopped for various reasons, but when it is to be restarted, the following command should be issued: RESUMELOG. You should consult the console operator guide for the causes of log failures.

1- SYSTEM UP.

When the system is brought up, this record is written to the log file. It contains come coldload information such as:
Update level
Fix level
Core size, main memory
size expressed in Kwords.
CST size, number of entries in CST.
DST size, number of entries in DST.
PCB size, number of entries in PCB
IOQ size, number of entries in IOQ.
TRL size, number of entries in TRL.
IDC size, number of entries in ICS.
Maximum number of running jobs/sessions.

2- JOB INITIATION.

Every time a session or job is started, this record is created. It contains the following data.
Job type, Job number.
User name, account name,
job name and logon group name.
Input device number and
output device number.
Logon queue, B,C,D,E.
CPU time limit, for the job/session.
INPRI and OUTPRI.
LOGANAL makes use of this record, especially for security purposes. It will also be usefull to you for performance recording.

3- JOB TERMINATION.

When a session/job is terminated, this record is created. It contains the following info.
Job type, job number.
Maximum priority, ever attained
by any process, usually 0.
Number of creations, of processes.
Ie of many programs runned.
CPU time, in seconds.
CONNECT time, in minutes.
LOGANAL will report on this. It is very usefull when you are tracking a specific session/job.

4- PROCESS TERMINATION

This record is created for 3 events. First, when you terminate a program, second, if you have a program that has launched a process, and this process terminates. And third when you log off, don't forget that a session/job is a process. Here is the data contained in the record:
# of program file segments.
number of sl segments,
non-mpe this impacts your CST.
maximum stack size ever, expressed in words.
maximum data segment size ever,
expressed in words. This is for any
extra data segment.
cumulative total of virtual storage,
ammount of disc in sectors
requested for data both stack
and extra data segments.

5- FILE CLOSES

Every time you open a file, it will be closed either by yourself or by MPE. This record is created at file closure time. This record is usefull in security tracking and also is used for performance analysis. It is also used for debugging, but more on that later. The information is the record is as follows:
File name, fully qualified.
Disposition, of file at fclose, ie delete, save, etc.
Domain, of file at fopen,
ie new, temp, perm etc.
Number of sectors allocated,
sectors reserved on disc.
Device type, on which file resides.
Device number, on which file resides.
Records processed, since the fopen for that file.
Blocks processed, since the fopen for that file.

6- SHUTDOWN

When a =SHUTDOWN command is executed, this record will be written and contains the following info.:
number of jobs, on system at shutdown.
number of sessions, on system at shutdown.

7- POWER FAIL

This is self-explanatory, and the record contains a power fail recovery status flag (it should be set at 0, but I have seen it a %177777 (-1)) (documentation problem?).

8- SPOOLING

Every time a spoolfile is created, modified or deleted, this record is created. Since I have not used this feature, I will not discuss it any more than to say what information is recorded:

User name, account name, job name, file name.
Job type, comes from either a job or a session on this, or another system.
Job number, of original creator.

I/O, 0=input and 1=output.
Devicefileid.
Device type, 0 for input and 32 for output.
Spoolee number, input=disc number on which file resided,
output=device number printer upon.
Numcopies, number of copies yet to be printed.
Number of records processed, input-number of lines input
output-number of lines printed.
Number of sectors used, or occupied on disc by the spoolfile.
Subtype, input-0, output=subtype of printer used.
Func, is the last operation of the spooler
normal completion, deletespoolfile, defer spoolfile, relink.
Number of physical pages, only for 2680a laser.
Number of Logical pages per physical pages, only for 2680a laser.

9- LINE DISCONNNECT

When you have a DSN link that disconnects, this record is created. Again, I have not used this event so here is just the information.
Logical device number.
Time of connection.
Number of output data transfers.
Number of input data transfers.
Number of recoverable line errors.
Number of irrecoverable line errors.
Local ID sequence.
Remote ID sequence.
Phone number of remote, if local system dialed.

10- LINE CLOSES

This record is created when the line is closed. I have the same things to say about it as in number 9. Here is the record format:
Logical device number.
Time stamp of open.
Driver name, max 8 ascii digits.

11- I/O ERRORS

This record is very important when time comes for a PM. It is created for every occurance of an I/O error on any device on the system. I will not cover the data at all.

12- VOLUME MOUNT

This record is created every time you physically mount a pack on a private volume drive. Since I have never used private volumes, I will not cover the data at all.

13- VOLUME SET MOUNT

This record is created when a user requests the use of a pack. See 12 above.

14- TAPE LABEL

This record is created every time a tape is mounted on a drive and AVR (automatic volume recognition) identifies it as a label tape. I will not cover the data.

15- CONSOLE LOG

This record is created by many different events occuring at the console. It may record action from or action directed to, the console. I will try to present a list of the events recorded. But first, here is a layout of the information contained. Byte length, of console line, length is negative for input, positive for output. Console line.

16- PROGRAM FILE EVENTS

This record came about with the advent of MPE IV. It records the underflow situation simulated by MPE. The most common example is for BASIC programs compiled under pre-MPEIV MITs that contain the CHAIN or INVOKE command.

17- CALL PROGRESS SGNLS

I have no information on this event.

18- DCE PROVIDED INFO

I have no information on this event.

There have also been at one time, and may still be 2 other events that could be logged: 46-MPE maintenance request log and 47-DCU (diagnostic control unit) log. I do not know exactly what these are for.

II) IMPACT OF MPE LOGGING

In order to analyse log files to get information on performance and security, you must enable the following logging:

Type	Reason
2	Job/Session logon records
3	Job/Session termination records
4	Process termination records.
5	File close records

Warning: When type-5 (file close) is enabled, your log files will jump in size very rapidly.

You should dump these log files on a very regular basis and then put them on tape for later analysis, then purge them.

III) PERFORMANCE ANALYSIS

If you were to look at the logfiles with LISTLOG2, you would rapidly see that you are unable to get a clear picture of the events easily. This is why I created the program LOGANAL which you will find on the swap tape.

From now on, we will only look at logfiles as seen thru LOGANAL. We will not bother with LISTLOG2 except to mention that a number of events are disregarded by LOGANAL (ie, power fails, I/O errors etc.), I feel that LISTLOG2 does a good job of reporting those.

Now, here is an example of a typical short session:

INSERT EXAMP1 HERE (*)

Here is also another small view of a file used during that session:

INSERT EXAMP2 HERE (*)

As you can see, this is a blow-by-blow account of the session from beginning to end. Let's look closer:

We have the console logon message, followed by the type-2 job init. record. Next we have a type-5, file close on COMMAND.PUB.SYS, this file is used to find out which UDC's are to be set for the session.

Then I went in to TDP, and you see that the program file TDP.PUB.SYS has two fcloses, a system fclose and a session fclose. I suspect that the loader does strange things to a program file when you run a program. Then the file TDPPARMS.TDPDATA.HPOFFICE is closed, this file is accessed by TDP when you first run the program, it sets up all sorts of parameters. It then closes PARMSET in my account because I have my own parameters.

I then texted in a file called PAPERC and you can see the fclose for that file after TDP opened for some checking thru FGETINFO I suppose. TDP also close the keep file K3480924 as new permanent file. And then it closes PAPERC as and old permanent file after copying the contents over to the keep file.

Then I keep the file back as PAPERC, and you can see the file PAPERC closed with delete option. Then PAPERC is again closed as a new permanent file after TDP has copied the data from the keep file. And finally K3480924 is also closed with delete option.

I then exit TDP and you see the type-4, process termination record, and a series of fcloses for $STDIN, STDINX and 2 $STDLIST.

The next program that I ran was a BASIC compiled program XDPROG. At the end of the program, you see the type-4 record, followed by the standard $STDIN, STDLIST and 2 new ones BASLIST and BASIN. These last files are used only with BASIC compiled programs.

And then I terminate my session, so you see a type-4 record for the session. Don't forget that a session is an MPE process! If you don't believe that, just do a SHOWQ and look at all the Mxxx PINs. You then see the fcloses for my UDC files, and finally the console logoff message and the type-3, job termination record.

I have added an extra line of report in LOGANAL, this tells you how many files the session used and the maximum and average values for the following: Stack, Data segment and Virtual storage.

Another feature of the program is that you can specify a certain file, and it will look for each and every occurance of a file close for that file. This is a good tool for security purposes, since there is no way that a standard user can access a file without a log record being produced. Note that if one were to use DISKED2.PUB.SYS, the logging can be circumvented.

System load is defined as who is doing what to which file and at what time. This means that we want to know if there are certain times in the day when the system is more active or less active than at other times.

As an example, we could see that the response time during lunch is much better than say 10:00am. These you can determine by yourself, or you can get these thing out of the log files.

One method is as follows:

You pick a certain day and you take the corresponding log file.

You then do a listing of the logons and logoffs for that period.

You then produce a listing of the activities of each session.

You can then plot on paper which programs & which files were accessed and by whom during that period.

Now you, being fairly knowledgable about your system, can then be in a position to determine which processes were using the most resources, which processes were superflous, which ones could have been rescheduled and which ones could have be deleted outright.

Of course, if you were to do this for any extended period, you would start being flooded by information, but one thing you may do is to modify the program to do the above for you.

We will look at a specific example at the conference.

We can easily see the periods of intense activity and the slow or slack periods.

We can also see that if we have UDC's, that just by logging on and off, we generate a series of fopens and closes.

We can also see some limitations with this method, First, it does not give us a perfectly exact picture of the system activity for a very specific period of time, but if does give us an overall look into the workings of the system. It provides wonderfull tracking. Second, all this data can only be analysed after the fact, but again, if we do this exercise for an extended period on a typical environment (mix), one will be able to ascertain the peak and slack periods very easily.

IV) ENHANCED SECURITY WITH LOGGING

I will relate a true-to-life incident that occurred some months back, and at another company.

We had on a system a 'GOD'-like or SM program, a utility that, temporarly, gives the user all capabilities. Now, to be useful, this program had been renamed, released and lockworded.

Only a select, ie very small group, knew that it existed and were supposed to access it. At one point, I heard that some other users knew about and had been using this program.

So, I decided to find out for sure. I ran LOGANAL and got a list of all fcloses on that program for a number of weeks in the past. There were something like 12 fcloses. Then I ran a list of logons & logoffs for the same period. I then compared the session numbers of the fcloses to the logons. Rapidly I ascertained that 11 or the 12 fcloses were from legitimate users, (namely myself). But the last one, I could not remember executing. So I took a listing of the session in question, and came up with a very interesting listing. The session initiated after hours, and on a secluded terminal. I knew who the user was. The user ran TDP, did a few things to his/her files, then terminated TDP. Then the user ran the 'GOD' or SM program 15 minutes into the session. The user proceeded to look at a few innocent files, then went into FCOPY. At which point he/she proceeded to look at some very confidential files.

You can, with this method, follow any job or sessions, you can also find out all the users who accessed a specific file or file range. This can be a great help in documentation.

V) DEBUGGING WITH MPE LOGGING

Again, the best way to explain this feature is to relate a true experience. One day, while near the console, I saw the following message: LOGxxxx 1/2 FULL. That's quite common, but what followed is not, because 30 seconds later I saw: LOGxxxx 3/4 FULL, and soon after: LOGxxxx FULL, LOGxxxx+1 IS ON. Something was wrong, some event was being logged at an incredible rate. I started LISTLOG2.PUB.SYS, thinking that it might be an I/O problem of some kind, and LOGANAL does not report on those. On the laser printer I see a long series of file closes coming from a job, and all on the same file. 'Abort job so and so' I shout to the operator. This done, I recognized the file name, it was used only in a small SPL routine that places a program in a wait state for data coming back thru MRJE. I had tested that routine time and time again, and it always worked perfectly. But the data coming back thru MRJE was always relatively short, except on that day. There was a small bug in the routine; If the file checked was busy, the procedure would start over again without going thru the waiting stage. This never impacted on a short transmission, but wreaked havoc on long transmissions. This is the kind of bug that will usually leave no trace. The fix took all of 4 seconds wall time. Without proper MPE logging, we might never have spotted this bug, and we might have looked elsewhere for the reason of very increased activity while waiting for MRJE data.

With MPE logging and LOGANAL, you can track a complex program of going thru many files, and thus assure yourself of proper functionnality.

VI) CONCLUSION

In this paper, we have looked at three different uses of MPE logging, Performance analysis, security and debugging. MPE logging also gives the C.E.s an idea about which devices may cause trouble. I have also seen MPE logging used for a job accounting environment. There are many uses one can make of the data contained in the log files. After all the data is there and the mechanism to capture it is part of MPE so let's use it!

The material on logfile record layouts was taken from the system manager/system supervisor manual, from Hewlett-Packard Ltd.

(*) There were no insert examples supplied.
