One of the most useful mainframe-style features in SCOPE/Hustler that I still miss is the dayfile. A dayfile was a file associated with each job that recorded all commands issued by that job, and certain system messages. Each entry was time-stamped. Not only was each control statement logged, but by default, after each command had completed, the resources the job had used so far was also logged. By subtracting consecutive entries, you could figure out how much each job step cost.

Implementation-wise, there was a system dayfile, and a separate dayfile for each job. A batch job's dayfile was appended to its printed output. A DAYFILE command was provided primarily to allow interactive users to view their dayfiles, but it was rare for interactive users to need that output.

A Sample Dayfile

The following dayfile is from an actual typical simple Computer Science programming project. The batch job that produced this dayfile was about as simple as they get. It consisted of the following cards:

- Sequence number card
- Problem Number Card (PNC)
- Jobcard
- Password card
- (control statements)
- End-of-Record card, signifying end of control statements (7-8-9 multipunch in column 1).
- (FORTRAN and COMPASS source)
- End-of-Record card, signifying end of input to compiler.
- (Data to be read by program)
- End-of-Information card, signifying end of job (6-7-8-9 multipunch in column 1).

Lines of the dayfile are shown in bold fixed font. Dayfile lines that echo control statements are in blue. My annotations are indented, in normal text. Notice the odd formatting of some of the messages. To ease programming, variable portions of messages were often aligned on word boundaries (that's 10 characters).

03/06/75 +MSU HUSTLER 2 L239 LSD 36.30 01/19/75

The first date is the date the job was run. I can't remember the significance of the +; I think it had something to do with whether a recovery deadstart has been done since the last normal deadstart. Hustler was at version 2 for almost its entire life. L239 refers to the CDC PSR (Program Summary Report) level of the operating system; Hustler was at level 239 for most of its life because we stopped applying CDC fixes to the OS itself in the early 70's. (We did apply selected fixes and upgrade to “dependent products” such as the compilers, and maybe even to selected bits and pieces of the OS.) LSD means Latest System Description: the version of SCOPE/Hustler. LSD numbers were stored internally as two 6-bit numbers (major, minor version numbers). When we started to get close to minor version number 63, we had to think of a major change to make to the OS to justify going to the next major number so we could reset the minor number. The second date was the date this version of the operating system was generated.
This is the job's sequence number. The first character indicates the source of the job. In the case of
jobs submitted from a central site card reader--and I think from remote batch stations--the sequence
number came from the first card in the deck. These sequence number cards were provided free of
charge, and were not supposed to be reused. (In the early days, blank Hollerith cards were provided
free of charge to all users, but this practice was discontinued before I arrived at MSU.) Jobs
submitted from a central site card reader had sequence numbers starting with “I”. Jobs submitted
from the Engineering remote batch station started with M; from Chemistry, with O, and so on. Other
jobs had their sequence numbers assigned automatically. Jobs submitted from other jobs (typically
via the DISPOSE control statement) had sequence numbers starting with T. Interactive jobs always
started with S. The second letter of a sequence number indicated where the job would print. B
meant a central site printer; A meant the special remote batch station at the central site; V meant
Engineering, X meant Chemistry, and so on. By default, jobs printed at the site from which they were
submitted, so “I” jobs would print at “B”, “M” jobs would print at “V”, and so on. This could be
changed with the DISPOSE control statement. Interactive jobs had S as their second letter and
produced no printed output unless the interactive user explicitly issued a DISPOSE command to
create a print job.

Note that this job was read by the card reader almost 2 hours before it began execution. This wasn't
that unusual in the mid-70's, when usage was increasing. Be glad you are computing today and not
in 1975; the 6500 wasn't upgraded until about three and a half years after the time of this job!

This is the “job card”. It was the third card of the deck, the first two being the sequence number card
and the problem number card (PNC), which were not shown in the dayfile. Accounts had a two-level
hierarchy consisting of problem numbers (PNs) and userids within problem numbers. Userids could
be duplicated across PNs; that is, there could have been multiple unrelated userids named
RIORDAM312 in different problem numbers. However, generally PN managers tried to create userids
that were unique throughout the system. The job card required only the userid, but could also
contain limits to apply to the job, overriding default limits contained in the authorization file. In this
example, the job is to run at Rate Group 2. This specifies both the priority of the job and the cost
factor. RG2 was normal priority; RG1 was low priority (with a 50% monetary discount), and RG3 was
high priority (50% markup). RG1 jobs would run only evenings and weekends. JC80 meant the
system should abort the job if its cost exceeded $0.80. L25 meant that only the first 25 pages of
output should be printed.

Computer Science student's accounts were set to expire at the end of the term, and warnings
started well before expiration. Hence, this warning was seen a lot by CPS students.

I had last run a job at 5:34 pm that day. I'm not sure whether B stood for Batch (vs. Interactive), or
for the print disposition of the last job (B being the central site printers).

This warning was given when the dollar balance for this userid was less than $10, which was usually
the case for poor CPS students.
I had run 114 jobs, and I had $5.52 left in my account.

621 cards were read, costing me $0.62 in supplies costs. Reading and punching cards, and printing output, were considered supplies costs and were calculated and accounted for independently from compute costs.

I've used 0.083 seconds of CPU time so far, and 0.013 seconds of pseudo-PPU time. PP seconds were fictitious numbers computed from other measures. I recall that PP seconds were a function of PP requests and PRUs transferred, but that's inconsistent with the RP line below.

FTN was the Fortran Extended compiler, by far the most commonly-used language translator on the system. By default, FTN read a program from the file INPUT. For a batch job, INPUT was a disk file that contained an image of the card deck. When the job began, the file was positioned at the first card after the EOR card that marked the end of the control cards.

"New Length": the system has changed the amount of memory allocated to this job to 45000 octal words, based on information in the system table entry for FTN.

"Requests Processed". The two numbers are number of PP requests so far, and number of Physical Record Units (PRUs; disk sectors or tape blocks) transferred so far.

Attach the permanent file CPS312ASSIGN6 to this job as local file TAPE10. In a FORTRAN program, I/O unit xx by default would refer to a local file named TAPExx. This had nothing to do with whether the file was actually on tape. Note the use of "(" rather than "." to terminate the control statement. This was unusual but permissible.

There could be up to 63 cycles, or versions, of a file under the same name. VAX/VMS implemented a similar capability years later. Since by default the highest-numbered cycle was attached, cycles were convenient for publishing a new version of a file without disturbing jobs using the current version. But this capability was not frequently used outside of the Computer Laboratory.
By default, compilers put their relocatable binary output on a file named LGO, which stood for Load and GO. Specifying such a file on a control statement caused the file to be loaded, and execution was begun at the "transfer address" (main program entry point). On many operating systems, before a program can be executed, an executable (in MSDOS terminology, .EXE) file must be made from the relocatables (.OBJs) and libraries (.LIBs) via a command like LINK. SCOPE/Hustler's loader did not require this; it could link and load in one step. If you were going to run a program often, though, you could save money by creating an “overlay” file consisting of the linked relocatables and library routines.

22.31.06.NL   15500

Memory size was reduced to run the program. Hustler didn't bother to reduce memory to run ATTACH, because a system table entry for ATTACH told it that ATTACH was a quick-running program and that it wouldn't be worth shuffling memory around.

22.31.06.RP 000000000135 000000001174

I don't know why there are a different number of leading zeroes here than on the initial RP message. Perhaps the initial RP message for a job was hard-coded and the coder made a minor error.

22.31.06.EXEC BEGUN.22.31.06.

This was a standard message issued by the startup code in the FORTRAN runtime. Looking for an EXEC BEGUN message at startup was a clue as to whether the programmer was cool. Authors of respectable system utilities, even those written in FORTRAN, typically suppressed the EXEC BEGUN message in various ways. Often, a shell assembler main program was written to call the FORTRAN subroutines.

22.31.07.EXIT

For some reason, I terminated the program via CALL EXIT rather than STOP (or less legally, by falling through to END in the main program). I believe that calling EXIT prevented the program from displaying a verbose termination message on the terminal when run interactively.

22.31.07.MAX FILES 0007 MAX PRUS 000700B.

The job is ending. I used a maximum of 7 files at once, with most of those files probably being temporary ones used by FTN. My maximum disk space utilization was 700 octal 64-word PRUs.

22.31.07.PP 005.972 SEC.
22.31.07.RP 0000000000150 000000001207
22.31.07.CP USE 004.828 SEC VALUE$ 000.21
22.31.07.PP USE 011.047 SEC VALUE$ 000.03
22.31.07.CM USE 001.193 W-H VALUE$ 000.32
22.31.07.TOTAL COMPUTE VALUE AT RG2 $ 000.56

A listing of charges for this job. In this job, as with most, the lion's share of the cost comes from memory usage, computed as an integral in word-hours. Memory was a valuable commodity on a system with an 18-bit address space. (17 bits on the 6500.) Note that you got socked twice for CPU use: once for the raw number of seconds, and again for its role in computing the memory integral (CPU seconds * memory size).
This final line came when the job printed just a minute and a half later. The $0.81 was a supplies charge. As I recall, it was unusual for a normal-priority job to languish 2 hours in the input queue, but be completed printing only a minute and a half after ending.