INTERDEPARTMENTAL

MASSACHUSETTS INSTITUTE OF TECHNOLOGY *CAMBRIDGE, MASS. 02139*

from the office of

November 10, 1970

To: Those called upon to produce impromptu demonstrations of Multics

From: J. H. Saltzer

Subject: A Multics Demonstration

It is possible to give a demonstration of Multics which is very revealing to an experienced computer professional, merely by using available system commands, with no preparation, and avoiding artificial "game-playing' exhibits. The following selection of exhibits takes me about half an hour to perform on a fully loaded system; I describe them because you may wish to borrow some ideas or suggest others:

1. Log in, explaining login ritual, message of the day, and the ready message.
2. Type who$long to get a list of users
3. create xyz

   fo xyz; who$long; co

   print xyz (Quit after a few lines)

   dprint xyz (Pick up the output after the demonstration)

   (Point out that any console output may be so handled, also console input)
4. help runoff

   print runoff.info (full tree name is obtained from the help file)

   Quit after a few lines, then

   dprint runoff.info (pick up later)
5. help (no arguments - go through a step or two to show top of information tree).
6. list your directory with

   list -a
7. cwd to a subdirectory and list it.
8. cwd to the root and list it.

9. cwd to system_control_dir, list it. Compare the access modes of the accounting files, the "who" table, ("whotab") and your own files. (Extra activity for real pros: list some acls and cacls.)
10. Attempt to copy one accounting file. (Shows protection.)
11. mail (Empty your mailbox, then have someone send you mail while you are logged in.)
12. Demonstrate a gate error: type "hcs_$terminate_file alpha" as a command. (Shows a very helpful system.)
13. Demonstrate Source Language Debugging in PL/I:

type in with edm:

```
edm blowup.pl1
blowup:  procedure
declare (i,j) fixed;
declare a(10) fixed;
          do  i = -1 to -10000 by -1;
              j = a(i);
              end;
```

Pl1 blowup table

When you compile this, PL/I will give 2 warning diagnostics:

a. Too few end statements
b. A(i) is referenced but not set.

Ignore these and proceed: (Be sure you got a symbol table)

blowup

(causes out of bound fault on stack)

debug

/blowup/&x+25,s ← insert here the offset given in the out of bound diagnostic.

debug will respond by printing "j = a(i);", the statement which caused the trouble. type "i", and debug will print the value of "i", for you.

type "|q" to leave debug and

go on to the next demonstration.

14. Demonstrate programming in BASIC:

```
bsys foo
10  do i = 1 to 10
20      print i, i*i
30      next i
40      end
run
```

This will get you two diagnostics: 10 is ill formed, and 30 steps a variable without a "for" statement.

Fix program with

```
10      for i = 1 to 10
run
```

and get your answers. Compare compilation time of PL/I and basic.

Leave basic with "quit".

15. (Optional) If you have a Multics number, and the visitor is looking for more, try

```
tcm -all
ttm
fsm -all
```

You can then spend the rest of the morning explaining the meaning of all the data printed by the metering commands.

16. Check the status of your two print requests:

```
list -dtm -ℓ -p >DDD>idd>q3
```

17. Log out.

Hints:

1. Leave your visitor with
   a. A copy of "Highlights of the Multics System", MPM section 1.1.1.
   b. A copy of the Multics Bibliography, MPM section 1.1.2.
   c. The printed console session output of the demo.
   d. The output from the printer daemon.

2. Use full command and file names rather than abbreviations wherever possible, since your visitor may wish to reexamine the console output later and he won't remember the abbreviations.
3. Be careful not to get deeply into impromptu side tracks unless you have carefully thought them out in advance. The tricks one frequently uses to get around minor problems are often very obscure to an outsider, and are not worth the explanation they require. It takes a resonable amount of planning to make sure that a sequence of operations does not include some obscure trick.
4. Respect your visitor's time: make the demonstration brief but telling. Don't allow long printouts to waste time.
5. Point out after the demonstration that some of Multics most important features are not apparent in a short demonstration. For example,
   - 22 hours/day, 7 days/week service
   - dynamic reconfiguration of cpu and memory
   - decentralized project administration
   - sharing of files while building up a large subsystem of programs.
   - programming ease of a fully supported PL/I language.
6. Assuming that the demonstration was successful, it is appropriate to point out that one of the most significant features of Multics is that a single system has all of the demonstrated features in one place. The same tuned human interface which permits this demo to be completed in 30 minutes on a loaded system permits any programmer a wide range of tools at his fingertips.
7. Be sure to practice the demonstration once by yourself before trying it on a live visitor. Undoubtedly you will discover at least one item that I have explained poorly enough that the demonstration comes out differently than you expected.