TO: Distribution

FROM: D. Kayden and R. Roach

DATE: January 2, 1974

SUBJECT: Proposed Command Name Usage Monitor

I. GOALS

The purpose of this proposal is to provide a method to monitor the usage of commands in terms of number of calls and optionally list the users. This is necessary to guarantee that a particular command is not being used before it is deleted or to determine which users are still using it so they may be contacted. This could also be used to determine the users of a particular command should this be desired in conjunction with proposed changes.

It is desired that this feature be easy to turn off and on and minimize overhead when not in use. The routine "record_command_usage_" is not adequate for several reasons:

a. It is not easy to implement. Every command to be monitored has to have calls added to it and to do this for all commands would be extremely costly.

b. It does not have the ability to record who is using the command. This is necessary unless we are to withdraw commands which are currently in use by people who somehow did not get the word.

c. There is no easy way to turn it off and on. There is no way to be selective on what is being monitored.

d. It is relatively expensive as there is at a minimum two extra calls to be made in each command even if record_command_usage_ is a null program.

This proposal does not intend to replace record_command_usage_ since that routine supplies command usage information not provided by the proposed monitor (e.g. processor time used and number of page faults).

II. METHOD

The basic idea is to provide two additional data bases to be used by this monitoring. The first is a control segment which is edited by a special command and specifies what, if any, commands are to be monitored. This segment is normally read-only except for the user permitted to edit it. The second segment is a usage segment which contains the counts for each command listed in the control segment. An additional segment would be used for each command for which the users are being recorded.

The control segment is a fixed format segment with entries giving 1) the name to be monitored, 2) a flag to indicate whether or not the users are to be recorded and 3) an index into the usage segment showing where the counter is maintained. This allows multiple names to be assigned to the same counter (i.e. for dprint and dp).

The recording procedure would be called only by find_command_ and not by full_find_command_, eliminating explicit pathname references and name$entry references. The find_command_ routine would check a flag in the control segment to see if monitoring is on before each call to the recording procedure. If any condition is raised while accessing the control segment or calling the recording procedure (such as linkage_fault or access_violation), an internal static switch would be set to bypass checking the control segment flag and calling the recording procedure.

If the recording procedure finds the command name in its list and if the "record users" flag is on, a segment called command_name.usage (where command_name is one of the command names being monitored collectively with the invoked command) is searched and if the user is already in the list, the counter for that user is incremented by one. Otherwise, the user is added and the counter initialized to one.

A program would be provided to edit the control segment and to selectively print out the usage segments.

III. RESTRICTIONS

The following restrictions and/or problems have been identified and deemed to be insignificant:

a. For efficiency, only the names are used in the control table, not the absolute path names. This means that usage of private commands with the same names as system commands would be counted. Since this errs on the side of addition rather than omission, we chose to use only the name. If desired, it is easy to also add the absolute path name in the control file at the cost of a larger control file and an extra call to hcs_$fs_get_path_name in the recording program.

b. The initial version did not plan for a locking system for the usage files. This was done as it was designed to monitor suspected obsolete command usage and it seemed unlikely that two updates for the same command would interfere with each other. This is especially true as the adding of the user is done in one operation and would take only a few instructions.

c. The usage segments, being writable in the user ring, are not hack proof. (i.e. a user could store into the segment thus confusing the results.) We see no easy way to avoid this other than to provide a gate into ring 1 and this would be too costly as far as we are concerned. Incidentally, record_command_usage_ is likewise subject to this type of hacking.

Please send comments on the above proposal to:

Dave Kayden (MIT 39-458) or Kayden.HSED

or

Roger Roach (MIT 39-445) or Roach.PDO

Special Command
Administrative/User Ring
01/07/74

Name: command_usage_count, cuc

The Multics command processor contains the general purpose command name usage monitor, command_usage_count_. When find_command_ is called to locate a command prior to its invocation, it calls the monitor to search for the command name in the system wide data base, command_usage_list_. If the name is found, then its usage information is updated. The usage information includes a count of the number of invocations of the command by all users, and optionally the process group id and number of invocations for the first 200 users of the command.

The use of commands with multiple names can be monitored separately for each name. This allows dprint to be monitored separately from dpunch, even though dprint and dpunch are names on the same command. The use of commands with multiple names can also be monitored collectively. For example, dprint1, dp1, dprint2, dp2, and dp are all names on the dprint command which can be monitored collectively to measure the use of the dprint command. In addition, several different commands can be monitored collectively to measure the use of logical groupings of commands. For example, the pl1, fortran, and ft commands can be monitored collectively to measure compiler usage.

The command_usage_count command provides facilities for dynamically adding or deleting entries from the monitoring list and selectively printing current monitoring data.

Usage

```
     command_usage_count   key   -command1-    ...    -commandn-
           -ctl_arg1- ... -ctl_argn-
```

1) key — is one of the following functions:

add — adds a command name group to the monitoring list. command1 must be present and will be added to the monitoring list as a command name group. Monitoring will occur for all uses of each commandi, but data will be accumulated for the group only.

delete, dl	deletes command name groups from usage monitoring. Either command*i*, or -all must be specified. For any command*i* specified, the command name group containing command*i* will be removed from the monitoring list.
print, pr	prints current monitoring data. For each command*i* specified, the command names and usage counts of its command name group are printed. If no command*i* are specified, -all is assumed.
2) command*i*	is any character string not containing any of the characters <, >, or $.
3) control_arg*i*	may be selected from the following:
-all, -a	specifies that the function should be applied to all command name groups presently in the monitoring list. command*i* must not be present. Not applicable to the add function.
-clear, -cl	specifies, for the print function only, that usage counts for the command name groups being printed should be reset after printing.
-first *n*, -ft *n*	specifies, for the print function only, that only the *n* most frequent users of each command should be listed.
-total, -tt	specifies, for the add function, that per user usage counts should not be accumulated; and specifies, for the print function, that per user usage counts should not be printed (whether they were accumulated or not).

Notes

The system wide data base, command_usage_totals_, contains the number of invocations of each command name group by all users. If per user usage counts have been requested, they are contained in the system wide data base, command_name.usage (where command_name is one of the names in the command name group).

In order for a user's command invocations to be monitored, he must have read access to command_usage_list_, and read and write access to command_usage_totals_. In addition, these two segments must be found in his search rules. The command_name.usage segments will be created in the directory containing command_usage_totals_ with an Access Control List (ACL) copied from command_usage_totals_. They will also be accessed directly from that directory.

(END)

<u>Name</u>: command_usage_count_

This procedure is called by find_command_ to record information about command name usage. command_usage_count_ searches for the command name in the system wide data base, command_usage_list_. If the name is found, the number of invocations of the command by all users is updated in the system wide data base, command_usage_totals_, and optionally the user's process group Id and number of invocations is updated in the system wide data base, command_name.usage (where command_name is one of the command names being monitored collectively with the invoked command).

<u>Usage</u>

```
     declare command_usage_count_ entry (char (*) aligned);

     call command_usage_count_ (com_name);
```

1) com_name   is the name of the command to be monitored.

<u>Notes</u>

The data bases command_usage_list_ and command_usage_totals_ are located via the search rules. The command_name.usage data bases are located in the directory containing command_usage_totals_.

If any condition is raised during the invocation of this routine (such as access_violation or linkage_fault), find_command_ will set an internal static switch turning off usage monitoring.

(END)