MULTICS TECHNICAL BULLETIN MTB-012

DATE: November 7, 1973

TO: Distribution

FROM: B. Wolman

SUBJ: New procedure tracing command

Attached is the draft documentation for the MPM section for the trace debugging command. Your comments are solicited. Send written comments to Barry Wolman, Honeywell Information Systems, 575 Technology Square, Cambridge, MA 02139 or mail comments to Wolman.Multics.

Command
Development System
11/1/73

Name: trace

The command trace is a debugging tool which lets the user monitor all calls to a specified set of external procedures. trace modifies the standard Multics procedure call mechanism so that whenever control enters or leaves one of the procedures specified by the user, a debugging procedure is invoked. The arguments given to the debugging procedure by trace enable it to obtain the arguments and return point of the procedure being called.

The user may start tracing a procedure at any time, even if it has already been executed. Tracing may be removed at any time; subsequent calls of the procedures will execute normally. Any PL/I or FORTRAN procedure may be traced without interfering with other users who may be sharing the segment.

Unless the user specifies otherwise, trace uses a default debugging procedure trace_print_. The action taken by trace_print_ is to print a message on the stream "user_i/o" whenever control enters or leaves one of the procedures being traced. There are a number of options which the user can specify to request such actions as printing the arguments (at entry, exit, or both), or stopping (at entry, exit, or both). The user can control the frequency with which the tracing message is

printed, e.g., every 100 calls after the 1000th call, or only if recursion depth is less than 5. trace_print_ "stops" the execution of the user's program by calling the debug command; this makes all of the facilities of debug available to the user.

The message printed by trace_print_ when control enters a procedure may appear in any one of several formats, depending on the user-settable brief switch and the status of the procedure making the call. If the procedure making the call is unbound or occurs in a bound segment containing a bind-map, the message takes the form

Call 4.1 of alpha from beta|127, ap = 204|10746

This is the fourth call of procedure alpha which is at recursion level 1; the call comes from location 127 in component beta, and the argument list is at 204|10746. If the procedure making the call is in a bound segment which does not contain a bind-map, the message takes the form

Call 4.1 of alpha from bound_gamma|437 (beta), ap = ...

The name in parentheses may not always be available and may be omitted in some cases. If the user has requested the brief output mode the message is shortened to

Call 4.1 of alpha

The trace command maintains an internal static table with an entry for each procedure being traced. The table entry holds a

number of parameters which are used to control when and how the debugging procedure is to be called. When an entry is added to the table (or an existing entry is changed), the trace control parameters in the trace table entry are set to the current values in an internal static trace control template (TCT). If the user does not alter the values in the TCT, the initial default values are used (see below). The initial values in the TCT specify that every call should be monitored.

Usage

The commands

```
trace n1 n2 n3 ...
```

or

```
trace -start n1 n2 n3
```

are used to start tracing procedures n1, n2, n3, ... using the current values in the TCT. Tracing will begin with the next call of the procedure. If a procedure is already being traced, its trace parameters will be altered and tracing will continue. The abbreviation for "-start" is "-sr".

The names n_i may be path names or reference names. If n_i does not contain a "$", the segment name is taken to be the same as the entry name. If the first character of n_i is "<" or ">", the segment with the given path name is initiated with the indicated segment name. If n_i is a reference name and the

specified procedure is not known to the process, the standard search rules are used to find it. An error message will be printed and tracing will not take place if a procedure name is not found or any of the restrictions (see below) are violated.

The name that is actually stored in the trace table is derived from the name n_i specified by the user. If the specified name is a path name, the name stored in the trace table is the segment name with the path stripped off; otherwise, the name n_i is used. For example, the command

```
trace >path>routine
```

would initiate the segment having the specified path name and add the name "routine" to the trace table. In subsequent requests to debug, the user could use either the original path name or just the segment name "routine".

trace recognizes a number of control arguments which change trace's actions upon encountering a procedure name appearing in the argument list after the control argument. If the control argument appears at the end of the argument list (i.e. no names follow it), the action is applied to all names currently in the trace table. An error message is printed if any of the specified procedures are not in the trace table.

The command

trace -remove n1 n2 ...

causes trace to remove entries n1, n2, ... from the trace table; any future calls will not be traced. Tracing may be removed at any time, even while a procedure is active or is being entered. The abbreviation for "-remove" is "-rm".

The command

trace -print n1 n2 ...

causes trace to print the number of calls and current recursion depth of all procedures named afterwards in the argument list. The abbreviation for "-print" is "-pr".

The command

trace -reset n1 n2 ...

causes trace to reset to zero the number of calls and recursion level of all procedures named afterwards in the argument list. The abbreviation for "-reset" is "-rs".

The command

trace -off n1 n2 ...

causes trace to stop calling the debugging procedure for all procedures named afterwards in the argument list. The specified procedures remain in the trace table and control continues to go through the trace package at entry to the procedures. Calls to

the specified procedures will continue to be counted, but no other action will be taken.

The command

trace -on n1 n2 ...

causes trace to resume calling the debugging procedure for all procedures specified afterwards in the argument list.

Several control arguments may be used in a single invocation of the trace command. The appearance of a control argument terminates the action of any control argument preceding it on the command line. For example, the command

trace alpha -remove beta -print gamma delta -reset

would start tracing procedure alpha, stop tracing beta, print the counts of procedures gamma and delta, and reset the counts of all procedure being traced.

The Trace Control Template

As mentioned earlier, the trace table entry holds a number of parameters whose values are determined by the contents of the TCT at the time the table entry is filled in. These parameters are used in conjunction with N, the number of calls of the traced procedure in this process, and R, the current recursion depth, to control when and how the procedure should be monitored. The execution count N is set to zero when tracing is first started

and is incremented by one every time the traced procedure is called. The recursion depth R is set to zero when tracing is first started and is incremented by one every time control enters the traced procedure and is decremented by one every time control leaves the traced procedure.

Let

D = the maximum recursion depth to be monitored,

F = the number of the first call to be monitored,

L = the number of the last call to be monitored,

E = how often monitoring should occur,

H = when the debugging program should halt at entry to the traced procedure,

S = when the debugging program should stop at exit from the traced procedure,

A = when the debugging program should print the arguments of the traced procedure,

I = a bit which is "1"b if the debugging procedure should print the arguments of the traced procedure when control goes in to the traced procedure,

and O = a bit which is "1"b if the debugging procedure should print the arguments of the traced procedure when control goes out of the traced procedure.

Then, a call will be monitored and the debugging procedure will

be called if and only if

F <= N <= L

R <= D

and mod(N,E) = 0.

If A ^= 0, mod(N,abs(A)) = 0, and I = "1"b, trace_print_ will print after the call message the values of the arguments (if any) being passed to the traced procedure. All of the arguments will be listed when A > 0. If A < 0, the procedure is assumed to be a function and the value of the last argument is printed after the procedure returns.

If H ^= 0 and mod(N,H) = 0, trace_print_ will call debug before returning to trace; the call to debug occurs before the procedure being traced has created its stack frame. Whenever trace_print_ calls debug, it first prints "DB " (without a new-line) as a cue to the user.

After control leaves the traced procedure, trace_print_ will print a line of the form

Return N.R from alpha

If A ^= 0 and mod(N,abs(A)) = 0 then all of the arguments of the traced procedure will be printed if 0 = "1"b; otherwise, if A < 0 the value of the last argument (assumed to be the value of the function) will be printed.

Finally, trace_print_ will call debug before returning to trace if S ^= 0 and mod(N,S) = 0. This call to debug occurs after the stack frame of the procedure being traced has been destroyed.

The following control arguments may be used to alter the values in the trace control template. Since the values in the template are copied when an entry is added to the trace table, the template should be updated before an entry is added if the entry requires new values. A change to a tracing parameter remains in effect until changed by another request, (n denotes a decimal integer).

-depth n
-dn n — set D = n; initially D = 9999999999.

-first n
-ft n — set F = n; initially F = 1.

-last n
-lt n — set L = n; initially L = 9999999999.

-every n
-ev n — set E = n; initially E = 1.

-halt n
-ht n — set H = n; initially H = 0.

-stop n set S = n; initially S = 0.

-sp n

-argument n set A = n; initially A = 0.

-ag n

-in set I = "1"b and O = "0"b; these are the initial values.

-out set I = "0"b and O = "1"b.

-inout set I, O = "1"b.

-io

Metering

The trace command may be used to meter the execution of a specified set of procedures. When the metering feature is being used, trace does not call the debugging procedure when control enters a procedure being traced; instead, it determines the current time and the virtual cpu time and number of page faults used by the user's process before control enters and after control leaves the traced procedure. This information is used to compute the real time, cpu time, and number of page faults used by the traced procedure on a local and global basis. The global cpu time is the time spent in the procedure including the time spent in any traced procedures which it calls; the local cpu

time does not include the time spent in any traced procedure called by the procedure. The local and global versions of real time and page faults are calculated in a similar manner. Note that metering is only done when the first, last, every, and depth tracing conditions are satisfied.

The control argument pair

-meter on

sets the metering switch in the TCT; any procedures added to the trace table or who have their table entries updated after this argument is used will be metered. The control argument pair

-meter off

turns off the metering switch in the TCT; any procedures currently being metered will continue to be metered. "-meter" may be abbreviated "-mt".

The control argument

-total

causes trace to print the metering statistics of all procedures in the trace table. The output gives the number of calls (#CALLS), global cpu time (GCPU), global real time (GREAL), global page waits (GPWS), local cpu time (LCPU), local real time (LREAL), local page waits (LPWS), and usage based on local real time (%USAGE, which is 100 * LCPU divided by total LCPU of all procedures being metered). The metering statistics are set to

zero after they are printed. The control argument

-subtotal

prints the same information as "-total", but does not clear the statistics. The abbreviation for "-total" is "-tt"; the abbreviation for "-subtotal" is "-stt".

Recursion Limiting

The control argument pair

-govern on

sets a bit in the TCT which causes recursion limiting to be in effect for any procedure subsequently added to the trace table. When the governing feature is used, the depth control parameter is ignored and trace_print_ will print the call message only when the recursion depth of the traced procedure reaches a new, maximum depth. Each call message will have a recursion depth one greater than the previous call message. In addition, trace_print_ will call debug whenever the recursion depth is a multiple of 10. Return messages are not printed. This feature enables the user to find and limit uncontrolled recursion; it can be very useful in finding the procedure(s) responsible for fatal process errors. The control argument pair

-govern off

turns off the governing switch in the TCT; any procedure currently being governed will continue to be governed. The

abbreviation for "-govern" is "-gv".

The Watch Facility

The trace command has an optional watch facility in which trace_print_ watches the contents of a set of previously specified memory cells. As long as the values in the locations being watched remain the same, no action is taken and no tracing messages are printed. The tracing message will be printed as soon as trace_print_ finds that any of the locations being watched has had its value changed; this may be found either at entry to or exit from the traced procedure. When any value changes, the tracing message will be preceded by lines giving the new values of all of the locations that have changed and debug will be called (even if the H or S conditions are not met). When execution continues, the locations that have changed will be watched with the new value being used in subsequent checks. This feature can be very useful in determining which of the user's procedures has incorrectly modified a word of storage.

The control argument pair

-watch location

causes all procedures being traced to watch for a change in the current contents of the memory word(s) specified by the string "location". "location" may consist of a single address specification or a series of address specifications separated by

blanks and surrounded by quotes. If an address specification does not contain a "|", it is taken to be an octal number giving a location in the stack; otherwise, it is taken to be an octal segment number and octal word offset in the standard form 204|1276. The control argument pair

-watch off

turns off the watch facility. The abbreviation for "-watch" is "-wt".

The watch facility differs from other trace facilities because there is a single table of locations being watched which is used by all procedures being traced. When the watch control argument pair is processed, the new location(s) specified replace any locations currently in the watch table. There is no provision made for removing a single location from the watch table; the user must re-issue a watch request which omits the location to be removed from the table.

Changing the Debugging Procedure

The debugging procedure used by trace may be changed by the control argument pair

-call proc

where proc is the path name or reference name of the new debugging procedure. An error message is printed and no action is taken if the procedure proc cannot be initiated. As with

other changes to the TCT, procedures already being traced are not affected. If proc is the zero length string "", trace_print_ will become the debugging procedure again.

When it calls the debugging procedure, trace uses the calling sequence

```
call proc(tp,inout,from,args,space);
```

where:

tp (ptr) points at the trace table entry.

inout (fixed bin) is zero if entering and non-zero if leaving the traced procedure.

from (ptr) points at the return location of the procedure making the call.

args (ptr) points at the argument list being passed to the traced procedure.

space (dimension(16) fixed bin) is a work area that the debugging procedure may use for any purpose it wishes. This work area is in the stack frame that trace creates before calling the debugging procedure.

trace_print_ uses this same calling sequence; it may be called from the user's debugging procedure.

The number of times the traced procedure has been called is found in two different ways, depending on whether proc is called for an entry to or an exit from the traced procedure. When inout = 0, the execution count is given by the count field of the trace table entry. When inout ^= 0, the count is given by the value of inout. The declaration of the trace table entry may be found in the include file tracetable.incl.pl1.

Command Execution

The command execution facility of trace allows the user to specify a Multics command line to be executed whenever trace_print_ is called. trace_print_ calls the command processor with the specified string after printing the tracing message, but before calling debug. The control argument pair

-execute string

sets the execution string parameter in the TCT. string is a single argument, so if it is to contain any spaces it should be enclosed in quotes. The execution parameter in the TCT will be turned off if string is the zero length string "". The abbreviation for "-execute" is "-ec".

Changing Output Stream

All of the messages from the trace command that may be generated while actually monitoring procedures are normally written on the stream "user_i/o" so that trace may conveniently

be used with procedures that change the attachment of the normal stream "user_output". The control argument pair

-stream name

causes trace to write further output on stream name which must already be suitably attached.

Miscellaneous Control Arguments

The control argument

-status

causes trace to print the current status of the trace control table. The abbreviation for "-status" is "-st".

The control argument sequence

-info n1 n2 ...

causes trace to print all of the tracing parameters used for procedures n1, n2, ... If "-info" appears at the end of the command argument list, the tracing parameters of all entries in the table are printed. The abbreviation for "-info" is "-if".

The control argument

-brief

causes trace_print_ to use a shortened form of call message. The abbreviation for "-brief" is "-bf".

The control argument

-long

restores the normal verbose output mode of trace_print_. The abbreviation for "-long" is "-lg".

Restrictions

The following restrictions apply to the use of trace:

1. Only external procedures compiled by Version II PL/I or FORTRAN may be traced.
2. Ring 0 or gate entries may not be traced.
3. Incorrect execution will result if the traced procedure looks back a fixed number of stack frames, e.g. cu_$arg_ptr may not be traced.
4. Only 100 procedures may be traced at one time. Up to 16 locations may be watched at one time.
5. The procedure being traced and the trace package itself must share the same combined linkage segment.
6. If a procedure in a bound segment is to be tracable, its entry point must be externally available.
7. The recursion limiting, location watching, and automatic command execution facilities are only available when the default debugging procedure trace_print_ is used.