548

AT887-0356 MDS

K. A. Bohrer and W. J. Cheney

IBM Technical Disclosure Bulletin Vol. 33 No. 2 July 1990

90A062061

METHOD TO PROVIDE CHANGES TO THE AIX OPERATING SYSTEM KERNEL FOR MULTI-PROCESS DEBUGGING

A method of changing the AIX* operating system is described which will allow AIX to support any method of debugging a program that is already running or the child of a program that is in debug mode. It will also support any method of debugging a program that performs an exec (overlays itself with a new program).

Since AIX supports debuggers only when the program being debugged is a child of the debugger, in order to provide debuggers that can debug a program that is already running, or programs that are not a child process of the debugger, changes are made to AIX operating system by the new method.

The method provides the following new system functions and changes.

proc.h: Add new fields.

a. p_dpid - Debugger process id. The debugger process id may or may not be the parent of the program being debugged. A new ptrace(a) call will be made to set the p_dpid. This field will be used by the kernel to know that an attached debugger is running and allows a non-parent to be a program debugger. A parent may still be the debugger.

b. p_dext - Debug extension flag. A new ptrace() call will turn this flag on and off. This flag will indicate to the kernel that multi-process debugging functions, such as fork() and exec...() require special action. When the flag is set, then the debugger expects to debug the forked or execed program.

ptrace - Letters represent new ptrace calls. Numbers will be assigned later.

a. ptrace(a): Add function that debugger is attached. This function is similar to ptrace(0) except an external program makes the ptrace(a) call as the program debugger.

b. ptrace(b): Add function that debugger is to detach from a process.

c. ptrace(c): Add a function to get the program name. Used when attaching a debugger to a program or when an exec has occurred.

d. ptrace(d): Add request for new debugger to attach (re-attach). Used to switch debugger. Normally used when a program forks and a different debugger is to be attached.

e. ptrace(e): Add call to set multi-process debugging active or not active. This function sets flag p_dext (debug extended) in proc.h. May require two calls (set & clear).

f. ptrace(0): Clear p_dpid and p_dext.

issig():

a. When stopping for the debugger to process a signal, send SIGCHLD to the process associated with p_dpid instead of always sending to the parent process.

fork():

a. If multi-task debugger is set, then

   1. Set most proc.h information for child like parent.
   2. Put both parent and child process to sleep.
   3. Wake up debugger (p_dpid). At this time the parent and child should have the same debugger process id. See "wait" for details.

stop():

a. If program has a attached debugger (p_dpid), then wake up their debugger as well as parent.

exit():

a. If program has a attached debugger (p_dpid), then wakeup() the debugger and parent.

exec...():

a. If program has multi-process debugging set, then.

   1. Put the process to sleep.
   2. Send SIGTRAP to the new process.
   3. wakeup() the debugger (p_dpid).

wait():

a. Change return status (stat_loc) for trace mode.

   If program has the multi-process debugging set, then:

1. The low order 8 bits of stat_loc will be set as follows:

   0x7F = normal trace mode
   0x7E = Program forked. Child pid will be returned.
   0x7D = Program execed.

With the described changes, debuggers can be written that may start debugging processes that are already running. This helps in debugging a program that is looping, or a program that remains resident.

In addition, the debugger can be coded to provide the user with the option of debugging all processes that an application forks and/or execs. The user would know that his application has a child and could debug the child as well as the parent, without losing the original parent-child relationship.

* Trademark of IBM Corp.