**KDB AND GDB**

**-BY URJA PATEL (121058)**

**...................................................................................................................................................................**

**INDEX PAGE**

1. **Debugger 1**
2. **Why there is need of debugger? 1**
3. **KDB( kernel debugger) 1**
4. **GDB ( GNU kernel debugger) 2**
5. **How KDB is different from GDB 2**
6. **Pros of KDB and GDB 2**
7. **Cons of KDB and GDB 2**
8. **Current technology in KDB and GDB 2**
9. **Future technnolgy in KDB and GDB 3**
10. **References 3**

**.................................................................................................................................................................**

1. **Debugger**

A debugger runs your program, just similar to, when you type the name of your program. The basic distinction is a debugger can step through your source code, line by line, executing each line only when you want it to. At any point, you can inspect and even change the value of any variable at run-time. If your program crashes, a debugger tells you where and why the program crashed so you can see what went wrong. You can go through the program and see what source code lines get executed and in what order. KDB and GDB are types of kernel debugger.

1. **Why there is need of debugger?**

* Bugs in a kernel code leads to reboot or lockup of the whole system.
* Difficult to locate the problem- break points.
* Debuggers help collect information.
* Debuggers help to look at the source level in an easy way, helps to understand the code.
* Debuggers make kernel internals more transparent. You can single step through instructions, disassemble instructions, display and modify kernel variables, and look at stack traces.

1. **KDB( kernel debugger)**

* Kdb is an instruction-level debugger used for debugging kernel code and device drivers. Before you can use it, you need to patch your kernel sources with Kdb support and recompile the kernel
* Provides a simple debugger via a system console with no high level source.
* Kernel debugger is integrated into the kernel and allows full control of the system but before using kernel debugger user need to enable the kernel debugger.
* Assembly level debugger
* Accessible through console
* Kernel debugger allows setting breakpoints and single steeping through code.
* It allows determining the process state without the use of any external system or serial console.
* Requires understanding of instruction set because it performs at the machine instruction level but it is less convenient than a source level debugging.
* Analysis of kernel state- registers, variables
* Kdb provides the service of examining the data structures and memory in a user friendly format.
* When a kernel debugger is invoked by the condition then it is the only program running in the system. All the other processes are stopped and interrupts are disabled. One of the processor is designated as the debugging processor which is arbitrarily chosen and after that it does the working of the kernel debugger KDB.

1. **GDB ( GNU kernel debugger)**

* The GNU Debugger, usually called just GDB and named gdb as an executable file, is the standard debugger for the GNU operating system. However, its use is not strictly limited to the GNU operating system; it is a portable debugger that runs on many Unix-like systems and works for many programming languages, including Ada, C, and C++, Objective-C, Free Pascal, FORTRAN, Java and partially others.
* The purpose of a GDB is to allow you to see what is going on inside another program while it executes- or what another program was doing at the moment it crashed.
* GDB offers a 'remote' mode often used when debugging embedded systems. Remote operation is when GDB runs on one machine and the program being debugged runs on another.
* Works only with uart based console driver also GDB can communicate to the remote 'stub' which understands GDB protocol via Serial or TCP/IP.
* Source level debugger
* Requirement of two machines – host and target
* Live analysis – single step, break points
* Analysis of kernel state- registers, variables
* GDB can be used to debug C, C++, Objective-C, Fortran, Java and Assembly programs

1. **How KDB is different from GDB**

* To use GDB one must have a serial cable and one more system where GDB will be executed while in KDB everything is done only in one system. In Linux systems, for KDB systems should be equipped with Linux 2.2 while for GDB it should be equipped with Linux 2.2 or 2.4 which is not all environments are equipped with these versions.

1. **Pros of KDB and GDB**

* The main advantage of Kdb is that’s easy to set up, since you don’t need an additional machine to do the debugging
* In GDB it’s easier to use than Kdb, since you don’t have to spend time correlating assembly code with your source code
* KDB is "lightweight," meaning it does not interfere or change the normal flow and timing of the kernel code during runtime

1. **Cons of KDB and GDB**

* Kdb’ s main disadvantage is that you need to manually and mentally correlate your source code with disassembled assembly code
* In GDB it’s more difficult to set up since an additional machine is needed to do debugging.
* In GDB, The kernel has to be informed about the identity and baud rate of the serial port via command-line parameters.

1. **Current technology in KDB and GDB**

* **KDB**
* For KDB, VMware Workstation 7 also enables a powerful technique that lets you record system execution deterministically and then replay it as desired, even backwards. So as soon as the system crashes you can go backwards and see what was happening then (and even try changing something and see if it still crashes).
* Kprobes by IBM, Kernel Dynamic Probes (Kprobes) provides a lightweight interface for kernel modules to implant probes and register corresponding probe handlers. A probe is an automated breakpoint that is implanted dynamically in executing (kernel-space) modules without the need to modify their underlying source.
* The Windows NT family includes a kernel debugger named KD, which can act as a local debugger with limited capabilities (reading and writing kernel memory, but not setting breakpoints) and can attach to a remote machine over a serial line, IEEE 1394 connection, USB 2.0 or USB 3.0 connection. The WinDbg GUI debugger can also be used to debug kernels on local and remote machines
* **GDB**
* January 15th, 2015: GDB 7.8.2 Released
* The Portland Group sells an excellent high-quality GUI debugger named pgdbg. Pgdbg specializes in debugging all kinds of parallel code on many different kinds of clusters (distributed memory, SMP servers, etc). While pgdb is a very high-powered debugger, it's also expensive
* GDB target processors (as of 2003) include: Alpha, ARM, AVR, H8/300, Altera Nios/Nios II, System/370, System 390, X86 and its 64-bit extension X86-64, IA-64 "Itanium", Motorola 68000, MIPS, PA-RISC, PowerPC, SuperH, SPARC, and VAX. Lesser-known target processors supported in the standard release have included A29K, ARC, ETRAX CRIS, D10V, D30V, FR-30, FR-V, Intel i960, 68HC11, Motorola 88000, MCORE, MN10200, MN10300, NS32K, Stormy16, and Z8000. (Newer releases will likely not support some of these.) GDB has compiled-in simulators for even lesser-known target processors such like M32R or V850.

1. **Future technnolgy in KDB and GDB**

* GDB is still actively being developing. As of version 7.0 new features include support for Python scripting and as of version 7.8 GNU Guile scripting as well. Since version 7.0, support for "reversible debugging" — allowing a debugging session to step backward, much like rewinding a crashed program to see what happened — is available.

1. **References**

* <https://sourceware.org/systemtap/kprobes/>
* <http://stackoverflow.com/questions/1762253/current-linux-kernel-debugging-techniques>
* <http://en.wikipedia.org/wiki/GNU_Debugger>
* <http://www.linux-mag.com/id/2103/>
* <http://en.wikipedia.org/wiki/Kernel_debugger>
* <http://oss.sgi.com/projects/kdb/faq.html>
* <http://gneerajgupta.blogspot.in/2011/12/kgdb-and-gdb.html>
* <http://www-01.ibm.com/support/knowledgecenter/ssw_aix_53/com.ibm.aix.cmds/doc/aixcmds3/kdb.htm?cp=ssw_aix_53%2F1-2-0-10-0>
* <https://sourceware.org/gdb/current/onlinedocs/gdb/Summary.html#Summary>
* <https://sourceware.org/gdb/current/onlinedocs/gdb/index.html#Top>
* <http://www.drdobbs.com/open-source/linux-kernel-debugging/184406318>