KGDB: Kernel Source Level Debugger

Swapnil Pimpale
(swapnil@geeksofpune.in)

foss.in
2012
Nov 29 - Dec 01
Bangalore, India

GEEP
GEEKS OF PUNE

Agenda:

- Need for kernel debugging
- Kernel debugging techniques and their usage
- KGDB overview
- KGDB internals
- KGDB in action
- References

Need for kernel debugging

- Bugs in kernel code frequently result in a lockup or a reboot
- Kernel code is difficult to execute under a debugger
- Kernel code errrors can be hard to reproduce
- Debugging techniques help monitor kernel code, trace errors and collect useful information

printks

- Easy, printf like
- Lets you classify messages according to their severity by associating different *loglevels* with the messages
- Can be used from most kernel code
- Can be turned on/off and can also be rate-limited *(printk_ratelimited)*

foss.in 2012 Nov 29 - Dec 01 Bangalore, India

GEEP GEEKS OF PUNE

/proc filesystem

- Plenty of information exported
- */proc/[pid]:* Numerical sub-directory for each running process containing a lot of process related info
- */proc/cpuinfo:* Information about the CPUs
- */proc/slabinfo:* Information about kernel caches
- */proc/interrupts:* Number of interrupts per CPU
- */proc/vmstat:* Virtual memory statistics

strace

- Shows a log of system calls, arguments to the calls and their return values in symbolic form
- Works on programs regardless of whether or not compiled with debugging support and stripping
- Locate cause to user or kernel land
- Compare strace log with expected set of system calls
- *strace -o /tmp/log /bin/ls /*

gdb

- *gdb /usr/src/linux/vmlinux /proc/kcore*
- *kcore* represents the kernel executable in core file format
- Can print variables, structures, follow pointers
- Cannot modify kernel data
- Cannot set breakpoints / watchpoints
- Cannot single step through kernel functions
- Needs to be taught how to examine LKM

gdb

- core-file <core_file_name>
- add-symbol-file <sym_file_name> <.text base address> -s .bss <sec_addr> -s .data <sec_addr>

kdb

- Can set breakpoints
- Query/Change kernel data
- Single stepping (by instructions, not lines of C source code)
- Disassembling code
- Analysis of kernel state – registers, variables, stack traces
- 2.6.35 – *kdb* merged & uses same backend as *kgdb*

kgdb

- Source level debugger – gdb interface
- Analysis of kernel state – registers, variables, stack traces
- Live Analysis – single step (C source code), breakpoints, threads
- Module debugging
- 2.6.26 – *kgdb* merged into mainline

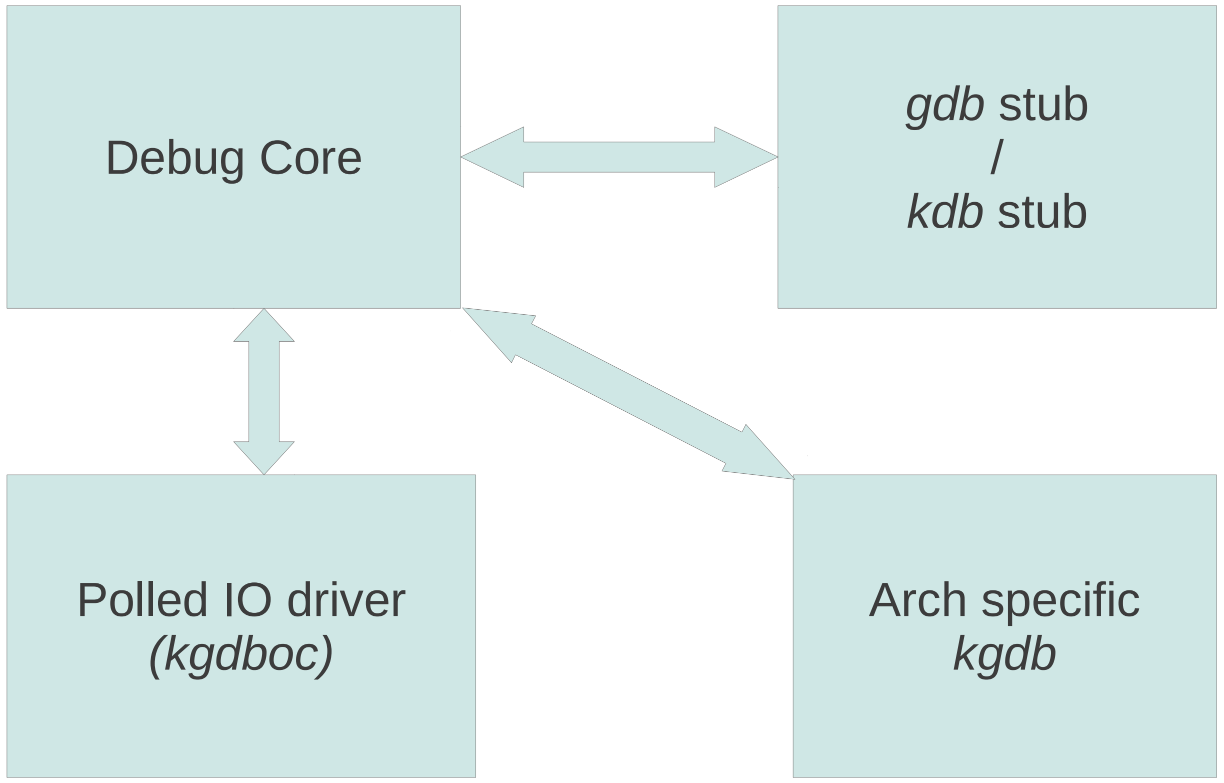
kgdb Setup

- Require two machines – development and test (host and target)
- Requires serial line between the development and test machines
- Test machine runs a *kgdb* enabled kernel
- Development machine runs a copy of *gdb*

kgdb config options

- *CONFIG_DEBUG_INFO=y*
- *CONFIG_FRAME_POINTER=y*
- *CONFIG_DEBUG_RODATA=n*
- *CONFIG_KGDB_SERIAL_CONSOLE=y*
- *CONFIG_HAVE_ARCH_KGDB=y*
- *CONFIG_KGDB_LOW_LEVEL_TRAPS=y*
- *CONFIG_MAGIC_SYSRQ=y*

kgdb Architecture



Debug core

- *kernel/debug/debug_core.c*
- Generic OS exception handler
- API to talk to *kgdb* IO drivers
- API to talk to arch-specific *kgdb*
- Logic to perform safe memory read/write while using the debugger
- Weak Implementation of software breakpoints
- API to invoke kdb/kgdb frontend to debug core

foss.in 2012
Nov 29 - Dec 01
Bangalore, India

Architecture specific *kgdb*

- *arch/*/kernel/kgdb.c*
- Arch specific trap catcher which invokes *kgdb_handle_exception()*
- *gdb_regs_to_pt_regs(), pt_regs_to_gdb_regs()*
- Registration/unregistration of arch specific handlers
  - Die notifier handling/cleanup
- Hardware breakpoints (optional)

kgdb IO driver

- Configuration via *built-in* or *module*
- Read and write character interface
- Cleanup handler for unconfiguring from the *kgdb* core
- Early debug methodology (optional)
- *kgdb* core repeatedly “polls” *kgdb* IO driver for characters
- *kgdboc, kgdb_8250, kgdboe*

foss.in 2012
Nov 29 - Dec 01
Bangalore, India

kgdbwait

- *kgdbwait* as a kernel command line argument will stop as early as the IO driver supports
- Useful mainly if you want to set breakpoints in early boot stages
- Can be used only if the IO driver is compiled into the kernel and driver config. is specified as kernel command line argument
  - *kgdboc=ttyS0,115200 kgdbwait*

kgdbcon

- Allows you to see *printk()* messages inside gdb while gdb is connected to the kernel
- Kgdb supports using gdb serial protocol for this
- Config:
  - Kernel Command Line: ***kgdbcon***
  - Using sysfs: ***echo 1 > /sys/module/debug_core/parameters/kgdb_use_con***
- Needs to done before configuring kgdb IO driver

foss.in 2012
Nov 29 - Dec 01
Bangalore, India

GEEP

kgdb – Living with optimizations

- Kernel compiled with optimizations
- Each C source line spread over instructions
- Control may appear to go backward in *gdb*
- Line numbers in inline functions make life difficult
- Disable some of the optimizations *(man gcc)*
- Run *objdump -S* on *vmlinux* or *module.ko* to find _exact_ line numbers from instruction pointer

kgdb in action

- *kgdb demo*

References

- kgdb.wiki.kernel.org/index.php/Main_Page
- kernel.org/pub/linux/kernel/people/jwessel/dbg_webinar/
- kgdb.geeksofpune.in/index.html
- geeksofpune.in/files/kerneldebugging-1.pdf
- geeksofpune.in/files/kerneldebugging-2.pdf

foss.in 2012
Nov 29 - Dec 01
Bangalore, India

GEEP
GEEKS OF PUNE

About *GEEP (GEEks of Pune)*

- GEEP is a non-profit group intended to create a platform for system software programmers in Pune.
- Founded in 2006 by a few kernel hackers in Pune. Since then it has grown to a more than 350 members group.
- GEEP organizes workshops for software professionals and engineering students. The workshops focus on kernel developments, embedded systems, networking, module programming, kernel debugging and more.
- geeksofpune.in