Foswiki > Main Web > KGDB (08 Oct 2014, AdminUser)

Kernel Masters

http://www.kernelmasters.org

KGDB

Introduction to KGDB

Kgdb is intended to be used as a source level debugger for the Linux kernel. It is used along with gdb to debug a Linux kernel. The expectation is that gdb can be used to "break in" to the kernel to inspect memory, variables and look through call stack information similar to the way an application developer would use gdb to debug an application. It is possible to place breakpoints in kernel code and perform some limited execution stepping.

Two machines are required for using kgdb. One of these machines is a development machine and the other is the target machine. The kernel to be debugged runs on the target machine. The development machine runs an instance of gdb against the vmlinux file which contains the symbols (not boot image such as bzImage, zImage, uImage...). In gdb the developer specifies the connection parameters and connects to kgdb.

KGDB vs KDB

Kernel Debugger (kdb): 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. The main advantage of kdb is that it's easy to set up, because you don't need an additional machine to do the debugging (unlike kgdb). The main disadvantage is that you need to correlate your sources with disassembled code (again, unlike kgdb).

Kernel GNU Debugger (kgdb): Kgdb is a source-level debugger. It is easier to use than kdb because you don't have to spend time correlating assembly code with your sources. However it's more difficult to set up because an additional machine is needed to front-end the debugging.

Compiling a kernel

Enable the CONFIG options given below for debugging and build the kernel.

CONFIG_DEBUG_RODATA is not set

CONFIG_FRAME_POINTER=y

CONFIG_KGDB=y

CONFIG_KGDB_SERIAL_CONSOLE=y

CONFIG_KGDB_KDB=y

CONFIG_KDB_KEYBOARD=y

NOTE: Otherwise you can add the following patch to automatically set the above configuration parameters.

.config.old 2014-05-17 12:22:39.964139767 +0530

+++ .config 2013-05-17 13:57:40.984901530 +0530

@@ -1393,6 +1393,7 @@

CONFIG_SERIAL_MAX3107 is not set

CONFIG_SERIAL_CORE=y

CONFIG_SERIAL_CORE_CONSOLE=y

+CONFIG_CONSOLE_POLL=y

CONFIG_SERIAL_OF_PLATFORM is not set

CONFIG_SERIAL_OMAP=y

CONFIG_SERIAL_OMAP_CONSOLE=y

@@ -2789,7 +2790,11 @@

CONFIG_ATOMIC64_SELFTEST is not set

CONFIG_SAMPLES is not set

CONFIG_HAVE_ARCH_KGDB=y

-# CONFIG_KGDB is not set

+CONFIG_KGDB=y

+CONFIG_KGDB_SERIAL_CONSOLE=y

+# CONFIG_KGDB_TESTS is not set

+CONFIG_KGDB_KDB=y

+CONFIG_KDB_KEYBOARD=y

CONFIG_TEST_KSTRTOX is not set

CONFIG_STRICT_DEVMEM is not set

CONFIG_ARM_UNWIND=y

Using KGDB/ GDB on Desktop Systems: DESKTOP <==========> DESKTOP

In order to use kgdb you must activate it by passing configuration information to one of the kgdb I/O drivers. If you do not pass any configuration information kgdb will not do anything at all.. Connecting with gdb to a serial port

Boot Time

Guest Side

In the Grub Screen, goto Advanced Options for Ubuntu, select your compiled kernel and press 'e' for editing the boot arguments. Goto the specific line shown below and add the arguments as highlighted.

linux /boot/vmlinuz-3.5.1 root=UUID=dfb38612-9a88-4c9d-b1a6-c6de87008173 ro quiet splash $vt_handoff kgdboc=ttyS0,115200 kgdbwait

This will stop the kernel from booting and your screen stops and shows as

Loading linux X.X.X

Loading Initial ramdisk ...

Host Side:

Connect from gdb

gdb ./vmlinux

(gdb) set remotebaud 115200

(gdb) target remote /dev/ttyS0

This makes your host system to connect to the target system provides information as

kgdb_breakpoint () at kernel/debug/debug_core.c:987

in kernel/debug/debug_core.c

(gdb)

Run Time

Guest Side

After the kernel gets booted, configure the kgdboc as given below

echo ttyS0 > /sys/module/kgdboc/parameters/kgdboc

echo g > /proc/sysrq-trigger ----> This command will hang your guest system

Host Side

Connect from gdb

gdb ./vmlinux

(gdb) set remotebaud 115200

(gdb) target remote /dev/ttyS0

This makes your host system to connect to the target system provides information as

kgdb_breakpoint () at kernel/debug/debug_core.c:987

in kernel/debug/debug_core.c

(gdb)

Using KGDB/ GDB on Embedded Platform: DESKTOP <==========> Embedded Board

Boot Time

Guest Side

Setting boot arguments

In order to add boot arguments, you have to edit the boot.txt file from the boot partition of the SD card as highlighted below

setenv bootargs "console=ttyO2,38400n8 root=/dev/mmcblk0p2 rootwait rw loglevel=8 kgdb=ttyO2,38400 kgdboc=ttyO2,38400n8 kgdbwait earlyprintk fixrtc nocompcache vram=48M omapfb.vram=0:24M,1:24M mem=456M@0x80000000 mem=512M@0xA0000000 init=/init androidboot.console=ttyO2"

Now, Create the boot.scr file by running the following command.

mkimage -A arm -T script -O linux -C none -a 0 -e 0 -n "boot.scr" -d boot.txt boot.scr

Now, replace the boot.txt and boot.scr files of "boot partition" of SD card with the respective edited files and insert the SD card into the Embedded Board and start the board. The Embedded board will halt and enter into KDB mode as shown below

kgdb: Registered I/O driver kgdboc

kdb:

Now, issue the command as "kgdb" to enter to shift to KGDB mode.

kdb: kgdb

Entering please attach debugger or use $D#44+ or $3#33

when the debugger gets connected from the host side, The systems will share some packets to connect and you will see them as

+$OK#9a+$QC01#f+$QC01#f-c0081d8f$00d084f+$ffdeff+$ffdeff+$ffdeff+$ffdeff

Host Side

Connect from gdb

arm-linux-gnueabihf-gdb vmlinux

(gdb) set remotebaud 38400

(gdb) target remote /dev/ttyS0

This makes your host system to connect to the target system provides information as

kgdb_breakpoint () at kernel/debug/debug_core.c:987

in kernel/debug/debug_core.c

(gdb)

KDB Logs

kdb ps A

Task Addr	Pid	Parent	[*]	cpu	State	Thread	Command
0xeffdfbc0	1	0	1	0	R	0xeffdfdb8	*swapper/0
0xeff92060	0	0	1	1	R	0xeff92258	swapper/1
0xeffdfbc0	1	0	1	0	R	0xeffdfdb8	*swapper/0
0xeffdf8e0	2	0	0	1	M	0xeffdfad8	kthreadd
0xeffdf600	3	2	0	0	M	0xeffdf7f8	ksoftirqd/0
0xeffdf320	4	2	0	0	M	0xeffdf518	kworker/0:0
0xeffdf040	5	2	0	0	M	0xeffdf238	kworker/u:0
0xeff92be0	6	2	0	0	M	0xeff92dd8	migration/0
0xeff92900	7	2	0	1	M	0xeff92af8	migration/1
0xeff92620	8	2	0	1	M	0xeff92818	kworker/1:0
0xeff92340	9	2	0	1	M	0xeff92538	ksoftirqd/1
0xeffaec00	10	2	0	1	M	0xeffaedf8	khelper
0xeffae920	11	2	0	1	M	0xeffaeb18	kdevtmpfs
0xeffae640	12	2	0	1	M	0xeffae838	netns
0xeffae360	13	2	0	1	M	0xeffae558	kworker/u:1
0xeff26940	16	2	0	1	M	0xeff26b38	suspend
0xeff2d680	22	2	0	1	M	0xeff4fe58	kworker/1:1
0xeff4f980	60	2	0	0	M	0xeff4fb78	irq/104-serial
0xeff4f6a0	62	2	0	0	M	0xeff4f898	irq/105-serial

kdb: rd ---> read the registers

r0: f084d000 r1: a0000113 r2: 00000000 r3: c093ee3c r4: c0965540

r5: c0965544 r6: c08e6368 r7: c08f1d08 r8: 00000000 r9: c08ad190

r10: eff88000 fp: 00000000 ip: 00000001 sp: eff89f58 lr: c0025458

pc: c0083604 f0: ?? f1: ?? f2: ?? f3: ?? f4: ?? f5: ?? f6: ?? f7: ??

fps: 00000000 cpsr: 60000113

KDB Commands

Command	Description
bc	Clear Breakpoint
bd	Disable Breakpoint
be	Enable Breakpoint
bl	Display breakpoints
bp	Set or Display breakpoint
bph	Set or Display hardware breakpoint
bpa	Set or Display breakpoint globally
bpha	Set or Display hardware breakpoint globally
bt	Stack backtrace for current process
btp	Stack backtrace for specific process
bta	Stack backtrace for all processes
btc	Cycle over all live cpus and backtrace each one
cpu	Display or switch cpus
dmesg	Display system messages
defcmd	Define a command as a set of other commands
ef	Print exception frame
env	Show environment
go	Restart execution
handlers	Control the display of IA64 MCA/INIT handlers
help	Display help message
id	Disassemble Instructions
kill	Send a signal to a process
ll	Follow Linked Lists
lsmod	List loaded modules
md	Display memory contents
mdWcN	Display memory contents with width W and count N.
mdp	Display memory based on a physical address
mdr	Display raw memory contents
mds	Display memory contents symbolically
mm	Modify memory contents, words

mmW	Modify memory contents, bytes
per_cpu	Display per_cpu variables
pid	Change the default process context
ps	Display process status
reboot	Reboot the machine
rd	Display register contents
rm	Modify register contents
rq	Display runqueue for one cpu
rqa	Display runqueue for all cpus
set	Add/change environment variable
sr	Invoke SysReq commands
ss	Single step a cpu
ssb	Single step a cpu until a branch instruction
stackdepth	Print the stack depth for selected processes
summary	Summarize the system

Kernel Masters

http://www.kernelmasters.org

-- KishoreBoddu - 07 Sep 2013

Edit | Attach | Print version | History: r5 < r4 < r3 < r2 | Backlinks | View wiki text | Edit wiki text | More topic actions

Topic revision: r5 - 08 Oct 2014, AdminUser