Linux Kernel Debugger (kdb)

 <kaos@ocs.com.au> December 2003

What is kdb designed for?

- A low level, in kernel, debugger
- Can be used directly on the machine being debugged
- Can also be used from a second machine over a serial console
- It will debug in places where higher level debuggers cannot function, including when interrupts are disabled
- Supports ix86, ia64, sparc64, xscale, ppc (2.4 kernels), ix86, ia64 (2.6-test kernels)

What is kdb not designed for?

- Source level debugging
- Debugging user programs

Applying kdb patches

- ftp://oss.sgi.com/projects/kdb/download
- Current version is v4.3
- You need a -common- patch plus the architecture specific patch for your platform
- You may also need architecture specific kernel patches, from the kernel arch maintainers
- Read the README

Building with kdb

- New config options
  - KDB, build the kernel with kdb
  - KDB_MODULES, include additional debug commands
  - KDB_OFF, whether kdb is off by default
  - KDB_CONTINUE_CATASTROPHIC, what to do after a catastrophic error
  - KDB_USB, support a USB keyboard
- You need kallsyms support from modutils. This can be a problem when cross compiling on 2.4 kernels.
- kdb/kdb_cmds defines your preset kdb commands

Running with kdb

- Boot options
  - kdb=on, activate kdb if you compiled with KDB_OFF
  - kdb=off, boot without kdb being active
  - kdb=early, enter kdb very early in the boot sequence, typically when you want to set breakpoints in the hardware probing code
- **echo "0" > /proc/sys/kernel/kdb** turns kdb off, **echo• "1"** to activate it
- Enter kdb with the Pause key on the PC keyboard, control-A on the serial console. See kdbmain.c, kdb_serial_str if you want a different sequence

Kernel problems

- The kernel is patched to enter kdb for panics, oops, deadlocks that are detected by the NMI watchdog, and all the other ills that the kernel is subject to.
- You can add your own code to enter kdb

```
if (something_unexpected_happened())
    KDB_ENTER();
```

Documentation

- Documentation, noun: information created by the developer and ignored by all the users
- Man pages in Documentation/kdb for the main kdb commands
- Read the source of any extra kdb modules for the commands that they add to kdb

Single machine debugging

- Use the PC keyboard
- Output is limited to a screen at a time
- No scroll up/down for the screen, that requires kernel support and the kernel is not moving when kdb is in control
- Write everything down by hand

Two machine debugging

- Build the kernel with serial console support
- In lilo, add serial=0,38400
- Boot with console=ttyS0,38400
- Capture the kdb output from a second machine using a null modem, using your favourite terminal emulator
- Use the terminal emulator to scroll up and down and to save output for later analysis
- Keep the source code on the second machine for review
- Keep the binary on the second machine for objdump