CANONICAL

Ubuntu Kernel Debug Tips

December 8th 2011

Pete Graner

Ubuntu Kernel Engineering Manager

Contents

- Ubuntu Kernel Team Knowledge Base
- Tools to Help Identify & Characterize Issues
- Mainline Kernel Builds
- Suspend/Resume
- Debugfs
- Helpful Code Snippets

Ubuntu Kernel Team Knowledge Base

- https://wiki.ubuntu.com/Kernel/Debugging
  - Backlight -- how to debug issues related to backlight brightness control
  - HighTemperatures -- how to debug thermal issues and/or fan related problems
  - Hotkeys -- How to debug non-working hotkeys.
  - Scheduling While Atomic -- How to gather a trace to help debug "scheduling while atomic" bugs
  - DebuggingKernelSuspendHibernateResume -- Help debugging suspend/resume and hibernate/resume problems.
  - LinuxWireless -- Testing and reporting bugs to the upstream linux wireless developers
  - DebuggingKernelOops -- Analyzing Kernel Oopses
  - DebuggingKernelBoot -- debugging Kernel boot issues
  - Debugging Guides
  - BIOSandUbuntu -- Debugging BIOS issues on Ubuntu (DSDT, Reboot, Suspend/Resume)
  - DebuggingSoundProblems -- Basic troubleshooting tips for debugging sound problems
  - DebuggingIRQProblems -- debugging interrupt related issues
  - DebuggingACPI -- debugging ACPI (Advanced Configuration and Power Interface)
  - DebuggingSystemCrash -- Tips for gather information regarding system crashes and lockups

Tools

- Hardware & System Interactions
- Network Measuring:
  - tcpdump, ethereal, speedometer, nethogs
- Intel tools:
  - LatencyTop, PowerTop
- I/O Measuring:
  - iotop,iostat, speedometer
- Process Monitoring:
  - ps, vmstat, top, atop, htop, strace

Kernel Mainline Builds

- Ubuntu Mainline kernel builds, built against Linus' git repository:
  - Checking if a fix is in an upstream kernel
  - Checking if the bug occurs in the Ubuntu Kernel and not in mainline (or vice-versa)
  - Quick bisects between kernel versions.
  - Downloadable kernel debs
  - 100% Linus code with Ubuntu config
  - No patches
  - https://wiki.ubuntu.com/Kernel/MainlineBuilds

Debugging Suspend/Resume

- Suspend/Resume issues fall into 3 basic categories:
  - BIOS Problems
  - Buggy Resume Code in Drivers
  - Kernel Panics
- BIOS – need to look at the DSDT
- Useful Links:
  - https://wiki.ubuntu.com/DebuggingKernelSuspend
  - https://wiki.ubuntu.com/DebuggingKernelSuspendHibernateResume
  - https://wiki.ubuntu.com/BIOSandUbuntu
  - https://wiki.edubuntu.org/DebuggingKernelSuspend
  - http://smackerelofopinion.blogspot.com/search/label/resume

Kernel Panics during Suspend

- Ubuntu is configured to hide console messages during the suspend/resume phase
- Sometimes the kernel panics late into suspend and no console message appear
- Trick: force console not to suspend
  - **`no_console_suspend=1`** via grub or **`/etc/sysctl.conf`**
- Change to console #1 and suspend:
  - **`chvt 1 && pm-suspend`**
- Non KMS 80x24 (pre-Karmic) console is too small to capture a full panic message. To get more lines on the console:
  - **`consolechars -f /usr/share/consolefonts/Uni1-VGA8.psf.gz`**
  - Take a photo of the panic

Slowing down printk() on boot

- Boot messages often scroll off the console before they can be read
- Useful to sometimes slow down the console message before a panic on early boot
  - **`CONFIG_PRINTK_DELAY=y`**
  - **`boot_delay=N`** (where N=millisecs)
- Kernel can be instrumented to delay at anytime, modify **`boot_delay_mesg()`** in **`kernel/printk.c`**:
  - **`if (boot_delay == 0 || system_state != SYSTEM_BOOTING) return;`**

Console & prink()

- printk() is the kernel dev's friend
- Pros:
  - Easy to insert debug messages
  - Use **`dmesg`** to view the messages
- Cons:
  - Timing – sensitive to race issues with various hardware
  - Default log is too small, console is limited in size (e.g. pains OOPs messages or stack dumps)
  - Buffer size: **`log_bug_len=N`** (N must be a power of 2) via grub
  - Change debug levels:
  - **`echo 7 > /proc/sys/kernel/prink`** -or- **`ignore_loglevel`**

USB Serial Console

- Console over plain old serial tty
- Slow max speed 115200 baud
- H/W flow control may not be available for serial console tty – can drop chars at high speed
- Many new laptops/netbooks don't have external serial port
- Use a USB->Serial dongle
- Ubuntu kernels are per-configured to use serial console just pass:
  - **`console=tty console=ttyUSB0,115200n8`**
  - Pass via the kernel command line
  - Replace ttyUSB0 with appropriate device name

debugfs

- A virtual filesystem to help in debugging
- Allows the developer to read/write u8, u16, u32 and boolean values
- Allows for the modification or examination of driver internals and hardware
- Examples:
  - rt73usb driver
  - I915 driver
  - drm driver
- Stock Ubuntu has debugfs mounted by default
- Can be mounted with:
  - **`sudo mount -t debugfs debug /sys/kernel/debug`**
- More info at:
  - http://lwn.net/Articles/115405/
  - http://docs.huihoo.com/linux/kernel/2.6.26/filesystems/ch04.html

Virtual Machine Method

- Virtualisation is ideal to reproduce bugs that are not hardware specific
- Host can capture huge amounts of console log messages
- Use **minicom** to capture console messages to a log file
- Configure **minicom** to read a named pipe i.e. **`unix#/tmp/vbox`** instead of a serial tty port
- Boot kernel with:
  - **`console=ttyS0,9600`**

Useful git repositories

- Test Scripts:
  - http://kernel.ubuntu.com/git?p=cking/scripts/.git;a=summary
  - Suspend/Resume, Battery, USB persist, wifi/network measurement & more
- Example Debug Code:
  - http://kernel.ubuntu.com/git?p=cking/debug-code/.git;a=summary
  - LED Flash, PC speaker beep, net constant bitrate test, reboot code & more
- Feel free to contribute!

POST code debugging

- Power On Self Test
  - http://en.wikipedia.org/wiki/Power-on_self-test
- Requires a POST code plug-in board or logic analyzer
- Write a value to port 0x80
- Linux uses writes to port 0x80 as a delay, so you must change the delay value:
  - **`io_delay=0xed`** or **`io_delay=udelay`** via kernel command line

Keyboard LED to debug

- Useful where console not available (i.e. late suspend, early resume)
- Kernel must be instrumented
- LED on – indicates we've reached some code
- LED flash patterns – different states
- Example code at:
- http://kernel.ubuntu.com/git?p=cking/debug-code/.git;a=summary
  - C example
  - Assembly example – for early resume

CANONICAL

Thank you

Questions ?