Application Note

CPE

Linux 3.4 Migration Guide

BROADBAND CARRIER ACCESS GROUP

CPE-AN300-R

5300 California Avenue • Irvine, CA 92617 • Phone: 949-926-5000 • Fax: 949-926-5203

November 25, 2013

Revision History

Revision	***Date***	***Change Description***
CPE-AN300-R	11/25/13	Initial release

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

Table of Contents

Overview

This document describes some of the common issues faced when upgrading drivers from the 2.6.30 kernel to the 3.4 kernel. It is intended as a resource for developers who are upgrading proprietary drivers, and meant as a reference. It is in no way complete, and only covers some of the more common issues encountered when upgrading internal drivers.

Common Issues

ioctl's: Big Kernel Lock Was Deprecated

The old kernel had a concept of a Big Kernel Lock (BKL), which was automatically grabbed by the kernel when a driver attempted to access the kernel space via an ioctl. It was impossible for two ioctls to be processed at the same time. This led to inefficiencies and real-time issues, which the Linux team sought to eliminate.

The new kernel no longer supports the big kernel lock, and ioctls are now called in an unlocked context. To make this very apparent, the new kernel has replaced the `ioctl` member of `file_operations` with `unlocked_ioctl`. This `unlocked_ioctl` works the same, except, as the name implies, it is called without any locks.

As long as the original ioctl does not share resources with another driver/ioctl call, the solution is simple: create a wrapper function which grabs a local mutex, then calls the old ioctl function. It is also possible that the existing ioctl function is concurrently safe, in which case the existing function can be used to populate `unlocked_ioctl` instead of `ioctl`. If unsure, use the local mutex approach.

Several parameters have been modified as well. The user now gets a file instead of the inode (the inode can be retrieved from `filep->f_dentry->d_inode`).

Refer to `spu_ioctl_unlocked` in spudrv.c for an example of an upgraded driver.

lock_kernel: Big Kernel Lock Was Deprecated

`lock_kernel` is no longer available. Users must use local locks and be careful to not introduce race conditions with other modules when doing so.

New Location of autoconf.h

autoconf.h is now in the 'generated' folder. To include it, the user must include `<generated/autoconf.h>` instead of `<linux/autoconf.h>`. Note that `LINUX_VERSION_CODE` has not moved, and can still be accessed using `include <linux/version.h>.` It can be used to determine which directory to read autoconf from.

```
#include <linux/version.h>
#if (LINUX_VERSION_CODE >= KERNEL_VERSION(2, 6, 33))
#include <generated/autoconf.h>
#else
#include <linux/autoconf.h>
#endif
```

Broadcom Convention for Including Kernel Header Files from Userspace

Broadcom has changed its convention for including files from userspace. Typically, if a userspace file requires inclusion of a linux file, and it includes `<linux/types.h>` for example, it will grab this file from the toolchain directory by default. If a file needs to include a file from the Broadcom's kernel directory, it should use the `bcm_local_kernel_include` link (which is generated during make under the kernel directory). Thus, it would look like:

```
include <bcm_local_kernel_include/linux/types.h>
```

To make this work, the userspace makefile should add the `$(KERNEL_LINKS_DIR)` directory to its `PATH` setting. A userspace program should never add any `kernel/linux-3.4/` subdirectory directly to its `PATH`, as this can lead to GPL contamination and compile issues going forward.

Spinlock Initialization

Users can no longer initialize a spinlock using `SPIN_LOCK_UNLOCKED`. Instead, they must use `DEFINE_SPINLOCK(sl)` from within global scope or `spin_lock_init(&sl)` from within an initialization function. When allocating a spinlock as a part of a structure, `__SPIN_LOCK_UNLOCKED` can be used as follows:

```
struct fscache_cookie struct = {
 .usage = ATOMIC_INIT(1),
 .lock = __SPIN_LOCK_UNLOCKED(fscache_fsdef_index.lock),
};
```

DECLARE_MUTEX

`DECLARE_MUTEX` actually created a semaphore instead of a mutex. As such, the latest kernel has renamed it to `DEFINE_SEMAPHORE` to better reflect its function. Use `DEFINE_SEMAPHORE` in its place, or if an actual mutex is needed, users can use `DEFINE_MUTEX` and change all of the `up`/`down` calls to `mutex_lock`/`mutex_unlock`.

I^{2}C

I^{2}C has changed significantly. See http://www.mjmwired.net/kernel/Documentation/i2c/upgrading-clients for information on upgrading I^{2}C clients.

Interrupts

Some interrupt handling has changed between the kernels, and an analysis should be done to rule out any new race conditions. The BCA CPE Reference Software configures the kernel to send most interrupts to only one CPU (CPU 0). Only the Inter-Processor Interrupt (IPI) and Timer interrupts are sent to both CPUs. See make menuconfig, under Kernel Configuration selection, Enable compiled-in kernel cmdline, CMDLINE: "irqaffinity=0". We strongly recommend that users do not send driver's interrupts to both CPUs. A race condition exists in handle_level_irq where the irq is left masked and no bottom half is scheduled. This makes it impossible for new interrupts to be processed.

To confirm which CPU the device driver's interrupts are sent to, users must go to the Busybox shell and type "cat /proc/interrupts" and look at the interrupt counts for the driver under CPU0 and CPU1. Only one of the columns should be non-zero, and the other should remain at zero. Alternatively, users can go to cat /proc/irq/<irq_num>/smp_affinity. This shows the CPU bitmask of the interrupt routing: 0x1 means CPU0 only, 0x2 means CPU1 only, and 0x3 means both CPU0 and CPU1.

mtd

mtd function handles should not be called directly anymore (and have thus been renamed). To replace these, several `mtd_` functions have been created. `mtd_read`, `mtd_write`, `mtd_block_isbad`, `mtd_read_oob`, `mtd_erase` and `mtd_write_oob` should now be used.

The `MTD_OOB_XXX` macros have been replaced with `MTD_OPS_XXX_OOB`. This is to better conform to existing Linux naming conventions, and there are no functional changes associated with this name change.

Miscellaneous Structures

Several structures have changed form. Most of the time when this happens, macros are created to access the moved fields. Rather than accessing the structure fields directly, users should try to find the correct macros.

Methods of Investigating Other Issues

One of the most useful resources for investigating changes is using the identifier search in http://lxr.free-electrons.com/ident?v=3.4;a=mips. This website allows users to find all instances of a symbol, browse through the Linux history to see when this symbol was introduced or deprecated, and see how existing built-in drivers adapted to that change.

Note: 2.6.30 and 2.6.31 are no longer supported on this website.

RT-Preempt Patch, SoftIrq Threads, BCM Custom Threads

In the previous 2.6.30 Linux kernel, only a portion of the RT-Preempt patch was applied. In the new 3.4 Linux kernel, the entire RT-Preempt patch is applied. The kernel tarball at kernel/src-linux-3.4.11-rt19.tar.bz2 contains kernel source from kernel.org with the RT preempt patch applied. However, just because the RT preempt patch is applied, it does not mean it is activated in the kernel. The RT preempt patch has three major operating modes:

- Preempt LL (Low Latency)
- Preempt RTB (Real Time Basic)
- Preempt RT_FULL ("All and everything")

The BCA CPE Reference Software configures the 3.4rt kernel to operate in the "Preempt LL" mode. No other mode is tested or supported at this time. We strongly recommend that users stay in the Preempt LL mode.

SoftIRQs

In the 2.6.30 Linux kernel with the partial RT-Preempt patch applied, softirqs (which includes NAPI) are executed in a thread. When users type ps on the 2.6.30 kernel, they may notice the softirq threads with names such as sirq-net-rx/0, sirq-net-tx/0, sirq-timer/0, sirq-net-rx/1, sirq-net-tx/1, sirq-timer/1. In the Linux 3.4 kernel and the full RT Preempt patch, these threads have been eliminated. Therefore, softirq processing (including NAPI) are now done in the "bottom half" context instead of in a thread context. Doing processing in the bottom half context has two undesirable effects:

1. The bottom half processing is triggered after the hardware interrupt processing is done, but before the execution returns to the interrupted process. This means that the bottom half processing, which may be processing low-priority packets, could still delay the processing of a high-priority thread.
2. In theory, if this bottom half processing takes too long, the kernel will defer the work to a thread called ksoftirqd/0 or ksoftirqd/1. However, these ksoftirqd threads run at normal priority. So if high-priority packet processing takes too long, that processing could be deferred to ksoftirqd/0 or ksoftirqd/1, which means they will be processed at normal priority. In practice, it seems like ksoftirqd/0 and ksoftirqd/1 are still running at very high priority. This area needs more investigation.

To overcome these issues, the packet processing of the bcmenet driver has been moved to a custom thread called bcmsw_rx. This thread roughly corresponds to the sirq-net-rx/0 thread in the 2.6.30 kernel. Therefore, bcmenet bottom half processing does not happen in the softirq context, it happens in the context of the bcmsw_rx thread. Also, the bcmsw_rx thread has been assigned the same Real Time priority and CPU binding of the previous sirq-net-rx/0 thread.

In future releases, Broadcom will be creating more custom kernel threads to replace the functionality of the obsoleted sirq threads. In the interim period, if users notice that top or mpstat is reporting high levels of CPU usage in softirq processing, Broadcom recommends that users keep the above issues in mind.

Workqueue Changes and WLAN Threads

The implementation of the Linux Workqueue subsystem has changed between 2.6.30 and 3.4. See kernel/linux3.4rt/Documentation/workqueue.txt for details.

In the 2.6.30 kernel, the WLAN driver used the generic Linux workqueue threads (events/0 and events/1) to do the bulk of the wlan processing. However, due to the new workqueue changes, using the generic linux workqueue threads was no longer possible. So two new threads have been created, wl0-kthrd (for the first WLAN adapter) and wl1-kthrd (for the second WLAN adapter, if present). These threads can be given a Real Time priority and be “bound” to a particular CPU, just like the events/0 and events/1 threads in the previous 2.6.30 kernel. See the WLAN documentation/release notes for more details.

Connecting everything®

BROADCOM®

BROADCOM CORPORATION
5300 California Avenue
Irvine, CA 92617

Phone: 949-926-5000
Fax: 949-926-5203
E-mail: info@broadcom.com
Web: www.broadcom.com

CPE-AN300-R November 25, 2013