Using and Configuring Printk

CPE-AN800-R

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

February 11, 2014

Revision History

Revision	*Date*	*Change Description*
CPE-AN800-R	02/11/14	Initial release

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

Table of Contents

About This Document

Purpose and Audience

This document explains best practices when using printk functions to output diagnostic information from the kernel, while preventing performance and quality of service problems, especially with applications/services that are sensitive to delay.

This document is intended for software and system engineers.

Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use.

For a comprehensive list of acronyms and other terms used in Broadcom documents, go to: http://www.broadcom.com/press/glossary.php.

Document Conventions

The following conventions may be used in this document:

Convention	*Description*
Bold	User input and actions: for example, type **exit**, click **OK,** press **Alt+C**
`Monospace`	Code: `#include <iostream>` HTML: `<td rowspan = 3>` Command line commands and parameters: `wl [-l] <command>`
< >	Placeholders for *required* elements: enter your <username> or `wl <command>`
[ ]	Indicates *optional* command-line parameters: `wl [-l]` Indicates bit and byte ranges (inclusive): [0:3] or [7:0]

References

The references in this section may be used in conjunction with this document.

Note: Broadcom provides customer access to technical documentation and software through its Customer Support Portal (CSP) and Downloads and Support site (see Technical Support).

For Broadcom documents, replace the "xx" in the document number with the largest number available in the repository to ensure that you have the most current version of the document.

Document Name	*Number*	*Source*
Broadcom Documents		
[1] Application Debugging Using GDB - V4.2 FOR BCA LINUX® ROUTERS	CPE-AN5xx-R	CSP

Technical Support

Broadcom provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates through its customer support portal (https://support.broadcom.com). For a CSP account, contact your Sales or Engineering support representative.

In addition, Broadcom provides other product support through its Downloads and Support site (http://www.broadcom.com/support/).

Overview

The kernel print function, printk, is a very common and easy way to output debug diagnostics and error messages from the kernel. However, its ease of use can lead to overuse and inappropriate usage. In the default configuration, printk's are output on the console, which is a relatively slow serial port, and hardware interrupts and scheduling/context switching are disabled during printk output. So this function can cause performance and quality-of-service (QoS) problems, especially with applications/services that are sensitive to delay, such as IPTV, VoIP, and DECT. Symptoms of such problems include:

- Lower throughput
- Macro blocking on IPTV
- Low voice quality
- DECT subsystem shutting down due to high interrupt service latency

Note: Because printk's can cause problems in the system, it is important to use and configure them correctly.

Background Information

When kernel or driver code calls printk, the line is first written to a 16 KB circular log buffer. This buffer can be read by userspace applications such as klogd or dmesg. Both klogd and dmesg are described in the sections below. If the importance level of the printk line is greater than the system printk log level, it is sent to all registered console drivers. Specifying the importance level of a printk line and setting the system printk log level are also described in the following sections.

In theory, the console driver can be a serial port, a USB port, a virtual console device, and so on. In practice, the only console driver on our system is the serial port.

Our serial port is configured for 115200 baud, so it takes about 70 µs to transmit a single character. So to output a line of only 10 characters, it takes about 1 ms (700 µs + printk overhead), during which no hardware interrupts will be serviced and no scheduler context switching will occur on the CPU where the printk is called. This happens even if the serial port is not connected to a host.

Using Printk

There are two major forms of printk. The first form specifies the importance level of the line. The second form does not specify the importance level of the line, but instead allows the system to select the level. The two forms are shown below:

- printk(KERN_INFO "mydriver: initializing, buffers=%d\n", num_buffers)
- printk("mydriver: initializing, buffers=%d\n", num_buffers);

Broadcom recommends the first form, where the importance level is specified. Although, one could make the argument that the second form is more flexible and allows users to configure the importance level of the printk lines. In any case, the existing code contains many printk's in the second form, so their presence must be managed.

The various log levels and meanings are defined in kernel/linux/include/linux/kernel.h. The values are listed below:

- KERN_EMERG (0)
- KERN_ALERT (1)
- KERN_CRIT (2)
- KERN_ERR (3)
- KERN_WARNING (4)
- KERN_NOTICE (5)
- KERN_INFO (6)
- KERN_DEBUG (7)

Note: A lower numerical value means greater importance.

Control and Management

Using /proc

The /proc files, which control printk behavior, are listed below:

```
/proc/sys/kernel/printk
```

When you type `cat /proc/sys/kernel/printk` you will see four numbers. In the default system configuration, they are: 8 4 1 8.

The first number means to send printk's with higher importance (lower numerical value) than this number to the console. Since the least important log level is KERN_DEBUG (7), an 8 means to send all printk's to the console.

The second number means assign this importance level to all printk's which do not specify their importance level. So a 4 here means all printk's which do not specify their importance level are treated as KERN_WARNING(4).

The third number is the minimum allowed log level.

The fourth number is the boot-time default log level.

To set the first number, simply echo the number into /proc/sys/kernel/printk. For example, `echo 4 > /proc/sys/kernel/printk`. By setting the first number to 4, you are telling the kernel to only send printk's with higher importance (lower numerical value) than 4 to the console. Since all printk's which do not specify their log level are assigned to level 4, this setting prevents all of those printk's from going to the console.

To set the second number, you need to run **make menuconfig**, go to the Debug Selection section, and set the default console printk level. Then you need to run a **make clean** and **make again**.

Using /proc/sys/kernel/printk_with_interrupts_enabled

By default, the kernel will call printk with hardware interrupts and scheduler context switching disabled. This ensures that the printk message is fully displayed on the console before the kernel executes the next instruction (on the CPU where the printk is being called). When the debugging kernel crashes with printk's, it is useful to have this timing guarantee.

Note: On SMP systems, this timing guarantee is not as useful since the other CPU could still cause a crash before the first CPU is able to output the first character of its printk line. So the relative timing of when something happened on one CPU and when the crash happened on the other is not clear.

For services which are sensitive to hardware interrupt service latency, such as DECT, the default behavior can cause problems. By doing `echo 1 > /proc/sys/kernel/prink_with_interrupts_enabled`, hardware interrupt service routines are allowed to run in the middle of a printk. However, scheduler context switching is still disabled.

Using syslogd/klogd

By default, the syslogd/klogd subsystem is disabled. To enable this subsystem, use the WebUI and go to the Management section and select the **System Log** item. Set the drop-down menu's appropriately. Note that on the WebUI page, the levels select messages which are "more important or equally important" as the level selected, whereas when writing to /proc/sys/kernel/printk, the levels select messages which are strictly “more important” than the specified level/number.

Note: Configuring a remote syslog server to receive syslog messages is beyond the scope of this document.

Background Information: syslogd is the main logging daemon. It is responsible for receiving log messages from a variety of sources, including userspace applications. syslogd will then log the message locally and/or send the messages to a remote syslog server. klogd is a helper daemon which has the specific purpose of getting kernel printk messages from the kernel log buffer and sending them to syslogd for processing.

Using dmesg

If you do not want to run syslogd/klogd, you can simply type **dmesg** to dump the contents of the kernel circular buffer. While using dmesg is simple, it suffers from two drawbacks:

- One, if the kernel is generating a lot of printk's and you are not running dmesg often enough, you will miss some messages because they will be overwritten.

- Two, dmesg does not keep track of what messages have been dumped before, so if you type dmesg twice, the second dmesg will output many of the same lines from the first dmesg because those lines are still in the circular buffer (they have not yet been overwritten). To solve this problem, you can run dmesg with the -c option, which will clear the kernel log buffer after it has been read.

Enhancing klogd Redirect (altconsole)

There is a Broadcom internal patch, available on request, which will enhance klogd to send the kernel messages to a Telnet TTY.

Proper Use of ioctl

Some Broadcom drivers inappropriately use prink's to output information in response to an ioctl command. This is bad for several reasons:

- printk's can cause performance problems.
- The output will not be seen if the ioctl is executed on a Telnet or SSH window.
- This is not the standard Linux way of handling ioctl's.

The correct way to output information in response to an ioctl is to copy the requested information back to the userspace buffer, and then let the userspace application print out the information. See bcmdrivers/opensource/char/board/bcm963xx/impl1/board.c and search for BOARD_IOCTL_GET_PSI_SIZE or BOARD_IOCTL_FLASH_READ for examples of the correct way to handle an ioctl.

If the information to be returned is large or variable in size, there are two ways to handle it. The first approach is to implement two separate ioctl's. One ioctl returns the size of the userspace buffer needed to store the requested information, then a separate ioctl to actually get the information.

The second approach is to allow the userspace application to make a ‘best guess’ of the buffer size. If the buffer is too small, the driver can return an error with an indication of how much buffer space is actually needed.

Note: Both approaches can be used in combination with each other.

Proper Use of /proc Output

Some Broadcom drivers inappropriately use printk's to output information from a "cat /proc/some_proc_filename" operation for the same reasons as listed above.

There are two correct approaches to output information from a "cat /proc/some_proc_filename" operation.

First, for proc files that will output less than 4 KB of information, the information can be written to the 4 KB buffer provided to the proc read callback function. See bcmdrivers/opensource/net/enet/shared/bcmproc.c:proc_get_dma_summary or kernel/linux/fs/proc/proc_brcm.c:softirq_proc_read for examples of this approach.

Second, for proc files that can output more than 4 KB of information, the sequence file approach must be used. There is an excellent three-part tutorial on the web for how to use sequence files. Just search for "kernel newbie sequence files part" and you should see links for all three parts of the tutorial. Also see bcmdrivers/opensource/net/enet/shared/bcmproc.c:rxbd_fopts and rxbd_seq_ops and kernel/linux/fs/proc/proc_brcm.c:proc_kernel_config_operations for examples of this approach.

Connecting everything®

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