ORACLE

# Firmware and Bootloader Logging Specification

**Alec Brown**

Oracle, Software Developer

**Daniel Kiper**

Oracle, Software Developer, GRUB upstream maintainer

# Project Background

- More information was needed on how the platform was setup during boot for TrenchBoot
- Developed an early version of the logs that worked on the GRUB2 bootloader
  - https://lkml.org/lkml/2020/5/29/428
- Found that this could be useful for other cases, such as firmware or shim logs

# Firmware / Bootloader Log Structure

```
bf_log_header:
        +-------------------+
u32     | version           |
u32     | size              |
u8[64]  | producer          |
U8[64]  | log_format        |
u64     | flags             |
u64     | next_bflh_addr    |
u64     | log_addr          |
u32     | log_size          |
        +-------------------+
```

```
bf_log_buffer:
                +-------------------+
u32             | version           |
u32             | size              |
u8[64]          | producer          |
u32             | next_msg_off      |
bf_log_msg[l]   | msgs              |
                +-------------------+
bf_log_msg:
                +-------------------+
u32             | size              |
u64             | ts_nsec           |
u32             | level             |
u32             | facility          |
u32             | msg_off           |
u8[n]           | type              |
u8[m]           | msg               |
                +-------------------+
```

# Firmware / Bootloader Log Structure

## Description (bf_log_header)

```
bf_log_header:
        +------------------+
u32     | version          |
u32     | size             |
u8[64]  | producer         |
u8[64]  | log_format       |
u64     | flags            |
u64     | next_bflh_addr   |
u64     | log_addr         |
u32     | log_size         |
        +------------------+
```

- version: the firmware and bootloader log header version number, 1 for now
- size: the size of the bf_log_header to allow for backward compatibility if other fields are added
- producer: the producer/firmware/bootloader/… entity, NUL terminated string;  the length allows for ASCII UUID storage
- log_format: the format used to record the log messages, NUL terminated string; various producers may generate logs in various formats if needed
- flags: bit field used to store information about the log state, if bit 0 is set then the log was truncated

# Firmware / Bootloader Log Structure

## Description (bf_log_header cont.)

```
bf_log_header:
        +------------------+
u32     | version          |
u32     | size             |
u8[64]  | producer         |
u8[64]  | log_format       |
u64     | flags            |
u64     | next_bflh_addr   |
u64     | log_addr         |
u32     | log_size         |
        +------------------+
```

- next_bflh_addr: the physical address of the next bf_log_header structure, none if it is zero
- log_addr: the physical address of where the log buffer is stored
- log_size: the total size of the log buffer

# Firmware / Bootloader Log Structure

## Description (bf_log_buffer)

```
bf_log_buffer:
                +--------------------+
u32             | version            |
u32             | size               |
u8[64]          | producer           |
u32             | next_msg_off       |
bf_log_msg[1]   | msgs               |
                +--------------------+
```

- version: the firmware and bootloader log version number, 1 for now
- size: the total allocated space for the bf_log_buffer including the log messages stored in msgs
- producer: the producer/firmware/bootloader/… entity, NUL terminated string;  the length allows for ASCII UUID storage
- next_msg_off: the byte offset from the beginning of the allocated space for bf_log_buffer to the next byte after the last bf_log_msg in msgs
- msgs: the array of log messages

# Firmware / Bootloader Log Structure

## Description (bf_log_msg)

```
bf_log_msg:
        +-------------------+
u32     | size              |
u64     | ts_nsec           |
u32     | level             |
u32     | facility          |
u32     | msg_off           |
u8[n]   | type              |
u8[m]   | msg               |
        +-------------------+
```

- size: the total size of the bf_log_msg entry
- ts_nsec: timestamp in nanoseconds starting from 0 (zero); the producer using this log format defines the meaning of 0
- level: similar to the syslog meaning; used to differentiate normal log messages from debug log messages, but the exact interpretation depends on the producer
- facility: similar to the syslog meaning; used to differentiate the sources of the log messages, but the exact interpretation depends on the producer

# Firmware / Bootloader Log Structure

## Description (bf_log_msg)

```
bf_log_msg:
        +-------------------+
u32     | size              |
u64     | ts_nsec           |
u32     | level             |
u32     | facility          |
u32     | msg_off           |
u8[n]   | type              |
u8[m]   | msg               |
        +-------------------+
```

- msg_off: the byte offset which the msg field starts in bf_log_msg
- type: the log message type; similar to facility but NUL terminated string instead of integer
- msg: the log message, NUL terminated string

# How logging works in GRUB2

- The log reads messages from `grub_*printf*()` functions and stores them in a temporary buffer which gets dynamically (re)allocated.
- The log is copied to a final spot before being passed to the Linux Kernel
- Controlled via `grub_log`, `grub_log_debug`, and `grub_log_debug_fl` enviorment variables
  - `grub_log`: allows messages to be recorded to the log
  - `grub_log_debug`: allows debug messages to be recorded
  - `grub_log_debug_fl`: allows file and line numbers to be recorded

# Linux kernel boot_params/zero_page for legacy BIOS and TrenchBoot

```
diff --git a/arch/x86/include/uapi/asm/bootparam.h b/arch/x86/include/uapi/asm/bootparam.h
index 13093c7..278c947 100644
--- a/arch/x86/include/uapi/asm/bootparam.h
+++ b/arch/x86/include/uapi/asm/bootparam.h
@@ -142,7 +142,9 @@ struct boot_params {
        __u32 ext_ramdisk_image;                /* 0x0c0 */
        __u32 ext_ramdisk_size;                 /* 0x0c4 */
        __u32 ext_cmd_line_ptr;                 /* 0x0c8 */
-       __u8 _pad4[116];                        /* 0x0cc */
+       __u64 bootloader_log_addr;              /* 0x0cc */
+       __u32 bootloader_log_size;              /* 0x0d4 */
+       __u8 _pad4[104];                        /* 0x0d8 */
        struct edid_info edid_info;             /* 0x140 */
        struct efi_info efi_info;               /* 0x1c0 */
        __u32 alt_mem_k;                        /* 0x1e0 */
```

# Specification Placement

- UEFI Spec
- ACPI Spec
- Multiboot2 Spec
- OASIS Standards

# Passing the log to the OS

- On UEFI platforms, we can use configuration tables to pass the log
  - `Bootloader →`

    `EFI_BOOT_SERVICES.InstallConfigurationTable(EFI_GUID *bl_guid, void *data)`
  - The Linux Kernel looks for `bl_guid` in the configuration tables
  - The kernel exposes the log through `/sys/firmware/efi/bootlogs/<producer>`
- On ACPI and Device tree platforms, we can use the `next_log_addr` field in `bf_log_header` to pass the log

# Discussions and Existing Code

- Firmware and bootloader log discussions:
  - https://lists.gnu.org/archive/html/grub-devel/2020-11/msg00100.html
  - https://lists.gnu.org/archive/html/grub-devel/2020-12/msg00021.html
- GRUB2 patch:
  - https://github.com/rossphilipson/travail/blob/master/misc/grub-booloader-log-support.patch
- Linux kernel patch:
  - https://github.com/rossphilipson/travail/blob/master/misc/0001-Bootloader-log-support.patch
- sl-stat – the bootloader log parser:
  - https://github.com/TrenchBoot/sltools/tree/master/sl-stat

# Thank You!

**Alec Brown**
alec.r.brown@oracle.com