```
request_firmware() hotplug interface:
------------------------------------
       Copyright (C) 2003 Manuel Estrada Sainz

Why:
---

Today, the most extended way to use firmware in the Linux kernel is linking
it statically in a header file. Which has political and technical issues:

 1) Some firmware is not legal to redistribute.
 2) The firmware occupies memory permanently, even though it often is just
    used once.
 3) Some people, like the Debian crowd, don't consider some firmware free
    enough and remove entire drivers (e.g.: keyspan).

High level behavior (mixed):
============================

kernel(driver): calls request_firmware(&fw_entry, $FIRMWARE, device)

userspace:
       - /sys/class/firmware/xxx/{loading,data} appear.
       - hotplug gets called with a firmware identifier in $FIRMWARE
         and the usual hotplug environment.
               - hotplug: echo 1 > /sys/class/firmware/xxx/loading

kernel: Discard any previous partial load.

userspace:
               - hotplug: cat appropriate_firmware_image > \
                                        /sys/class/firmware/xxx/data

kernel: grows a buffer in PAGE_SIZE increments to hold the image as it
        comes in.

userspace:
               - hotplug: echo 0 > /sys/class/firmware/xxx/loading

kernel: request_firmware() returns and the driver has the firmware
        image in fw_entry->{data,size}. If something went wrong
        request_firmware() returns non-zero and fw_entry is set to
        NULL.

kernel(driver): Driver code calls release_firmware(fw_entry) releasing
                the firmware image and any related resource.

High level behavior (driver code):
==================================

        if(request_firmware(&fw_entry, $FIRMWARE, device) == 0)
               copy_fw_to_device(fw_entry->data, fw_entry->size);
        release(fw_entry);

Sample/simple hotplug script:
```

```
============================

        # Both $DEVPATH and $FIRMWARE are already provided in the environment.

        HOTPLUG_FW_DIR=/usr/lib/hotplug/firmware/

        echo 1 > /sys/$DEVPATH/loading
        cat $HOTPLUG_FW_DIR/$FIRMWARE > /sysfs/$DEVPATH/data
        echo 0 > /sys/$DEVPATH/loading

Random notes:
============

- "echo -1 > /sys/class/firmware/xxx/loading" will cancel the load at
  once and make request_firmware() return with error.

- firmware_data_read() and firmware_loading_show() are just provided
  for testing and completeness, they are not called in normal use.

- There is also /sys/class/firmware/timeout which holds a timeout in
  seconds for the whole load operation.

- request_firmware_nowait() is also provided for convenience in
  user contexts to request firmware asynchronously, but can't be called
  in atomic contexts.


about in-kernel persistence:
---------------------------
Under some circumstances, as explained below, it would be interesting to keep
firmware images in non-swappable kernel memory or even in the kernel image
(probably within initramfs).

Note that this functionality has not been implemented.

- Why OPTIONAL in-kernel persistence may be a good idea sometimes:

        - If the device that needs the firmware is needed to access the
          filesystem. When upon some error the device has to be reset and the
          firmware reloaded, it won't be possible to get it from userspace.
          e.g.:
                - A diskless client with a network card that needs firmware.
                - The filesystem is stored in a disk behind an scsi device
                  that needs firmware.
        - Replacing buggy DSDT/SSDT ACPI tables on boot.
          Note: this would require the persistent objects to be included
          within the kernel image, probably within initramfs.

  And the same device can be needed to access the filesystem or not depending
  on the setup, so I think that the choice on what firmware to make
  persistent should be left to userspace.
```