Linux Kernel Patches: The good, the bad, and the ugly

Wolfram Sang (wsa)

Hacker & Maintainer & Consultant
General remarks

About me
- Linux Kernel hacker → talks about Linux Kernel
- pretty much everything applies to patches for other projects as well

About you
- assuming you are more experienced with hacking and patching than the average user
- hope you can read unified diffs

About your devices
- please mute them for the talks
Patches ≠ Authors

Bad and ugly patches are not (necessarily) created by bad and ugly people!

Reasons for bad patches:

- too high workload
- patch not in main focus
- change of interest
- ...
- (okay, bad coders are also one reason)
I have Motorola-branded prolific usb cable. This cable, once inserted in a port, claims to be a 0307 chipset adapter, and Linux kernel doesn't recognize it: not working.

... To make it work you have to recompile the module pl2303, with this patch (my kernel is 2.6.35.10-74.fc14.i686). Be careful to setup the right values in the kernel root Makefile in order to fit your running kernel.
The already existing patch

--- a/drivers/usb/serial/pl2303.c
+++ b/drivers/usb/serial/pl2303.c
@@ -50,6 +50,7 @@ static const struct usb_device_id id_table[] = {
     { USB_DEVICE(PL2303_VENDOR_ID, PL2303_PRODUCT_ID_MMX) },
     { USB_DEVICE(PL2303_VENDOR_ID, PL2303_PRODUCT_ID_GPRS) },
     { USB_DEVICE(PL2303_VENDOR_ID, PL2303_PRODUCT_ID_HCR331) },
+    { USB_DEVICE(PL2303_VENDOR_ID, PL2303_PRODUCT_ID_MOTOROLA) },
     { USB_DEVICE(IODATA_VENDOR_ID, IODATA_PRODUCT_ID) },
     { USB_DEVICE(IODATA_VENDOR_ID, IODATA_PRODUCT_ID_RSAQ5) },
     { USB_DEVICE(ATEN_VENDOR_ID, ATEN_PRODUCT_ID) },

--- a/drivers/usb/serial/pl2303.h
+++ b/drivers/usb/serial/pl2303.h
@@ -21,6 +21,7 @@
 #define PL2303_PRODUCT_ID_MMX 0x0612
 #define PL2303_PRODUCT_ID_GPRS 0x0609
 #define PL2303_PRODUCT_ID_HCR331 0x331a
+  #define PL2303_PRODUCT_ID_MOTOROLA 0x0307

 #define ATEN_VENDOR_ID 0x0557
 #define ATEN_VENDOR_ID2 0x0547
From: Dario Lombardo <dario.lombardo@libero.it>
Subject: [PATCH] drivers: update to pl2303 usb-serial to support Motorola cables

Added 0x0307 device id to support Motorola cables to the pl2303 usb serial driver. This cable has a modified chip that is a pl2303, but declares itself as 0307. Fixed by adding the right device id to the supported devices list, assigning it the code labeled PL2303_PRODUCT_ID_MOTOROLA.

Signed-off-by: Dario Lombardo <dario.lombardo@libero.it>
Two tools you should know

No typical flaws?

- scripts/checkpatch.pl $YOUR_PATCH

Whom to send it to?

- scripts/get_maintainer.pl $YOUR_PATCH
Project 2: Ziplt Z2 kernel
From: Vasily Khoruzhick <anarsoul@gmail.com>
Subject: [PATCH] ARM: PXA27x: CPUFREQ: Don't use fastbus mode

PXA27x does not like fastbus for some reason, it can hang in random places when it's enabled. So don't use it to make cpufreq stable.

Signed-off-by: Vasily Khoruzhick <anarsoul@gmail.com>
The abandoned(?) patch

--- a/arch/arm/mach-pxa/cpufreq-pxa2xx.c
+++ b/arch/arm/mach-pxa/cpufreq-pxa2xx.c
@@ -156,13 +156,13 @@ MODULE_PARM_DESC(pxa255_turbo_table, "Selects the frequency table (0 = run table
((T) ? CCLKCFG_TURBO : 0))

static pxa_freqs_t pxa27x_freqs[] = {
- {104000, 104000, 0, PXA27x_CCCR(1, 8, 2), 0, CCLKCFG2(1, 0, 1), 900000, 1705000 },
- {156000, 104000, PXA27x_CCCR(1, 8, 3), 0, CCLKCFG2(1, 0, 1), 1000000, 1705000 },
- {104000, 104000, PXA27x_CCCR(1, 8, 2), 0, CCLKCFG2(0, 0, 1), 900000, 1705000 },
+ {104000, 104000, PXA27x_CCCR(1, 8, 2), 0, CCLKCFG2(0, 0, 1), 900000, 1705000 },
+ {156000, 104000, PXA27x_CCCR(1, 8, 3), 0, CCLKCFG2(0, 0, 1), 1000000, 1705000 },
+ {104000, 104000, PXA27x_CCCR(1, 8, 2), 0, CCLKCFG2(0, 0, 1), 900000, 1705000 },
+ {208000, 208000, PXA27x_CCCR(0, 16, 2), 1, CCLKCFG2(0, 0, 1), 1180000, 1705000 },
- {312000, 208000, PXA27x_CCCR(1, 16, 3), 1, CCLKCFG2(1, 0, 1), 1250000, 1705000 },
- {416000, 208000, PXA27x_CCCR(1, 16, 4), 1, CCLKCFG2(1, 0, 1), 1350000, 1705000 },
- {520000, 208000, PXA27x_CCCR(1, 16, 5), 1, CCLKCFG2(1, 0, 1), 1450000, 1705000 },
- {624000, 208000, PXA27x_CCCR(1, 16, 6), 1, CCLKCFG2(1, 0, 1), 1550000, 1705000 }
+ {312000, 208000, PXA27x_CCCR(1, 16, 3), 1, CCLKCFG2(0, 0, 1), 1250000, 1705000 },
+ {416000, 208000, PXA27x_CCCR(1, 16, 4), 1, CCLKCFG2(0, 0, 1), 1350000, 1705000 },
+ {520000, 208000, PXA27x_CCCR(1, 16, 5), 1, CCLKCFG2(0, 0, 1), 1450000, 1705000 },
+ {624000, 208000, PXA27x_CCCR(1, 16, 6), 1, CCLKCFG2(0, 0, 1), 1550000, 1705000 }
};

#define NUM_PXA27x_FREQS ARRAY_SIZE(pxa27x_freqs)
What should be done

- check if still needed for recent kernel
- read mail thread for further details
- extend patch to to make disabling fastbus optional
- test on real hardware
- update patch header
- send upstream
- be around to answer questions and to do follow-ups
in general, they have become very good at upstreaming

- they simply have a *lot* of patches
--- a/drivers/mtd/devices/m25p80.c
+++ b/drivers/mtd/devices/m25p80.c
@@ -1071,6 +1071,7 @@ static int m25p_probe(struct spi_device
     if (info->flags & M25P_NO_ERASE)
         flash->mtd.flags |= MTD_NO_ERASE;
+
     memset(&ppdata, '\0', sizeof(ppdata));
     ppdata.of_node = spi->dev.of_node;
     flash->mtd.dev.parent = &spi->dev;
     flash->page_size = info->page_size;
What should be done 1

- check if still needed for recent kernel
- look for improvements
- (test on real hardware)
- add patch header (description + Signed-off-by)
- send upstream
Ugly patch 2

--- a/drivers/mtd/chips/jedec_probe.c
+++ b/drivers/mtd/chips/jedec_probe.c
@@ -115,6 +115,10 @@
     #define UPD29F064 0x221C

 /* PMC */
+    #define PM39LV512 0x001B
+    #define PM39LV010 0x001C
+    #define PM39LV020 0x003D
+    #define PM39LV040 0x003E
    #define PM49FL002 0x006D
    #define PM49FL004 0x006E
    #define PM49FL008 0x006A
@@ -1259,6 +1263,54 @@ static const struct amd_flash_info jedec
             ERASEINFO(0x02000,2),
             ERASEINFO(0x04000,1),
         },
+            .mfr_id = CFI_MFR_PMC,
+            .dev_id = PM39LV512,
+            .name = "PMC Pm39LV512",
+            .devtypes = CFI_DEVICETYPE_X8,
+            .uaddr = MTD_UADDR_0x0555_0x02AA,
+            .dev_size = SIZE_64KiB,
+            .cmd_set = P_ID_AMD_STD,
+            .nr_regions = 1,
+            .regions = {
+                ERASEINFO(0x01000,16),
+            }
+        },
+    },

What should be done 2

- check if still applies to recent kernel
- check if it is the best solution
- add patch header
- test on real hardware
- send upstream
- be around to answer questions and to do follow-ups
--- a/drivers/net/ppp/pppoe.c
+++ b/drivers/net/ppp/pppoe.c
@@ -850,7 +850,7 @@ static int pppoe_sendmsg(struct kiocb *
      goto end;

- skb = sock_wmalloc(sk, total_len + dev->hard_header_len + 32,
+ skb = sock_wmalloc(sk, total_len + dev->hard_header_len + 32 + NET_SKB_PAD,
     0, GFP_KERNEL);
     if (!skb) {
         error = -ENOMEM;
@@ -858,7 +858,7 @@ static int pppoe_sendmsg(struct kiocb *
      }

       /* Reserve space for headers. */
- skb_reserve(skb, dev->hard_header_len);
+ skb_reserve(skb, dev->hard_header_len + NET_SKB_PAD);
     skb_reset_network_header(skb);

     skb->dev = dev;
check if still applies to recent kernel
check if hacky workaround, custom case, or important bugfix
check for dependencies
add patch header
(test on real hardware)
send upstream
be around to answer questions and to do follow-ups
Final remarks

It’s not all or nothing

- even partial steps are worthwhile, they add up
- yes, adding a patch header only is worth a commit

Great learning opportunity

- for testing, sometimes a whole patch series needs to be ported to a recent kernel
- can be some effort, but great way to start kernel hacking

There are nightmare patches, too…

- huge chunks which need to be separated
- useless changes (whitespace, personal style adaptions)
- ...
Thank you for your attention!

Happy Hacking! :)