

[PATCH 1/4] pfnmap: fix 2.6.15-rc3 driver breakage

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-11/9447.html>

From: Hugh Dickins (hugh_at_veritas.com)

Date: 11/29/05

Date: Tue, 29 Nov 2005 16:53:02 +0000 (GMT)
To: Linus Torvalds <torvalds@osdl.org>

I've not tested any of them, but believe 2.6.15-rc3 will suffer from premature freeing of pages in the case of the following drivers:

arch/ia64/kernel/perfmon.c
drivers/char/ftape/lowlevel/ftape-ctl.c
drivers/media/video/cpia.c
drivers/media/video/em28xx/em28xx-video.c
drivers/media/video/meye.c
drivers/media/video/planb.c
drivers/media/video/vino.c
drivers/media/video/zoran_driver.c
drivers/media/video/zr36120.c
drivers/usb/class/audio.c
drivers/usb/media/ov511.c
drivers/usb/media/pwc/pwc-if.c
drivers/usb/media/se401.c
drivers/usb/media/sn9c102_core.c
drivers/usb/media/stv680.c
drivers/usb/media/usbvideo.c
drivers/usb/media/vicam.c
drivers/usb/media/w9968cf.c
drivers/video/gbafb.c
drivers/video/sbuslib.c
net/packet/af_packet.c

About twenty drivers fill in their mmap vma by applying `remap_pfn_range` repeatedly, many of them using `vmalloc_to_pfn`: `vm_pgoff` will match only the last range or page mapped in, and the others will be considered normal by `vm_normal_page` – thus although they were not counted in by `remap_pfn_range`, they will be counted out by `zap_pte_range`, so freed well before the driver has finished with them (and frees them again).

A temporary hack (checking for `PageAnon` and `ZERO_PAGE`) sure to disgust Linus, but keep those drivers safe until the proper solution arrives.

Signed-off-by: Hugh Dickins <hugh@veritas.com>

Linux-Kernel: [PATCH 1/4] pfnmap: fix 2.6.15-rc3 driver breakage

```
mm/memory.c | 15 ++++++-----
1 files changed, 14 insertions(+), 1 deletion(-)
--- 2.6.15-rc3/mm/memory.c      2005-11-29 08:40:07.000000000 +0000
+++ linux/mm/memory.c          2005-11-29 15:59:34.000000000 +0000
@@ -372,6 +372,7 @@ void print_bad_pte(struct vm_area_struct
 struct page *vm_normal_page(struct vm_area_struct *vma, unsigned long addr, pte_t pte)
 {
     unsigned long pfn = pte_pfn(pte);
+
     struct page *page;

     if (vma->vm_flags & VM_PFNMAP) {
         unsigned long off = (addr - vma->vm_start) >> PAGE_SHIFT;
@@ -399,7 +400,19 @@ struct page *vm_normal_page(struct vm_ar
     * The PAGE_ZERO() pages and various VDSO mappings can
     * cause them to exist.
     */
-
- return pfn_to_page(pfn);
+
+ page = pfn_to_page(pfn);
+
+ /*
+  * Temporary hack to avoid driver breakage.
+  * About twenty drivers fill in their mmap vma by applying
+  * remap_pfn_range repeatedly: vm_pgoff will match only the
+  * last range mapped; so also check if page is COWed or ZERO.
+  */
+ if ((vma->vm_flags & VM_PFNMAP) &&
+     !PageAnon(page) && page != ZERO_PAGE(addr))
+     return NULL;
+
     return page;
 }

/*
-
To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@vger.kernel.org
More majordomo info at http://vger.kernel.org/majordomo-info.html
Please read the FAQ at http://www.tux.org/lkml/
```