Content ▶

User: Password: Log in |

Subscribe | Register

[PATCH] Fix mmap_kmem (was: [question] What's the difference between /dev/kmem and /dev/mem)

[Posted August 16, 2005 by corbet]

From:	Steven Rostedt <rostedt-AT-goodmis.org>
To:	LKML <linux-kernel-AT-vger.kernel.org>
Subject:	[PATCH] Fix mmap_kmem (was: [question] What's the difference between /dev/kmem and /dev/mem)
Date:	Thu, 11 Aug 2005 21:15:02 -0400
Cc:	Andrew Morton <akpm-AT-osdl.org>, Linus Torvalds <torvalds-AT-osdl.org>

```
On Thu, 2005-08-11 at 17:36 -0400, Steven Rostedt wrote:
> OK, I thought I use to know this. But what is the difference
> between /dev/kmem and /dev/mem.  I thought that with /dev/kmem you could
> use the actual kernel addresses to read from.
>
> For example, if I wanted to read the current variable X in the kernel, I
> could look up the address of X in System.map, then mmaping to /dev/kmem
> I could get to that variable using the address that I got from
> System.map.  But this doesn't seem to work.
>
> I'm getting an IO error on read. And looking at this I see:
>
>
> static int mmap_kmem(struct file * file, struct vm_area_struct * vma)
> {
>         unsigned long long val;
>       /*
>        * RED-PEN: on some architectures there is more mapped memory
>        * than available in mem_map which pfn_valid checks
>        * for. Perhaps should add a new macro here.
>        *
>        * RED-PEN: vmalloc is not supported right now.
>        */
>       if (!pfn_valid(vma->vm_pgoff))
>               return -EIO;
>       val = (u64)vma->vm_pgoff << PAGE_SHIFT;
>       vma->vm_pgoff = __pa(val) >> PAGE_SHIFT;
>       return mmap_mem(file, vma);
> }
>
> I printed out the value in vma->vm_pgoff, and it still has the
> 0xc0000000 (but shifted >> 12). Isn't this suppose to also remove the
> 0xc?  Or am I just totally off here?
>
```

> Thanks,
>
> -- Steve
>

Found the problem. It is a bug with mmap_kmem. The order of checks is wrong, so here's the patch. Attached is a little program that reads the System map looking for the variable modprobe_path. If it finds it, then it opens /dev/kmem for read only and mmaping it to read the contents of modprobe_path.

Without this fix I get:

```
# ./tmap /boot/System.map
found modprobe_path at (0xc03647e0) c03647e0
mmap: Input/output error
```

On a machine with the patch, I now get:

```
# ./tmap /boot/System.map
found modprobe_path at (0xc03aa900) c03aa900
/sbin/modprobe
```

Note that the attached program does not handle the case of the string crossing over a page.

-- Steve

Here's the simple patch:

Signed-off-by: Steven Rostedt <rostedt@goodmis.org>

```
--- linux-2.6.13-rc6-git1/drivers/char/mem.c.orig       2005-08-11 20:48:34.000000000 -0400
+++ linux-2.6.13-rc6-git1/drivers/char/mem.c    2005-08-11 20:48:48.000000000 -0400
@@ -269,10 +269,10 @@ static int mmap_kmem(struct file * file,
         *
         * RED-PEN: vmalloc is not supported right now.
         */
-       if (!pfn_valid(vma->vm_pgoff))
-               return -EIO;
        val = (u64)vma->vm_pgoff << PAGE_SHIFT;
        vma->vm_pgoff = __pa(val) >> PAGE_SHIFT;
+       if (!pfn_valid(vma->vm_pgoff))
+               return -EIO;
        return mmap_mem(file, vma);
 }
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdarg.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>

#include <sys/types.h>
#include <sys/stat.h>
#include <sys/poll.h>
#include <sys/mman.h>

int page_size;
#define PAGE_SIZE page_size
#define PAGE_MASK (~(PAGE_SIZE-1))
```

```
void get_var (unsigned long addr) {
        off_t ptr = addr & ~(PAGE_MASK);
        off_t offset = addr & PAGE_MASK;
        int i = 0;
        char *map;
        static int kfd = -1;

        kfd = open("/dev/kmem",O_RDONLY);
        if (kfd < 0) {
                perror("open");
                exit(0);
        }

        map = mmap(NULL,PAGE_SIZE,PROT_READ,MAP_SHARED,kfd,offset);
        if (map == MAP_FAILED) {
                perror("mmap");
                exit(-1);
        }
        printf("%s\n",map+ptr);

        return;
}

int main(int argc, char **argv)
{
        FILE *fp;
        char addr_str[11]="0x";
        char var[51];
        unsigned long addr;
        char ch;
        int r;

        if (argc != 2) {
                fprintf(stderr,"usage: %s System.map\n",argv[0]);
                exit(-1);
        }

        if ((fp = fopen(argv[1],"r")) == NULL) {
                perror("fopen");
                exit(-1);
        }

        do {
                r = fscanf(fp,"%8s %c %50s\n",&addr_str[2],&ch,var);
                if (strcmp(var,"modprobe_path")==0)
                        break;
        } while(r > 0);
        if (r < 0) {
                printf("could not find modprobe_path\n");
                exit(-1);
        }
        page_size = getpagesize();
        addr = strtoul(addr_str,NULL,16);
        printf("found modprobe_path at (%s) %08lx\n",addr_str,addr);
        get_var(addr);
}
```

(Log in to post comments)