Subject: [PATCH] mark read_crX() asm code as volatile
Posted by Kirill Korotaev on Tue, 02 Oct 2007 14:04:42 GMT
View Forum Message <> Reply to Message

Some gcc versions (I checked at least 4.1.1 from RHEL5 & 4.1.2 from gentoo)
can generate incorrect code with read_crX()/write_crX() functions mix up,
due to cached results of read_crX().

The small app for x8664 below compiled with -O2 demonstrates this
(i686 does the same thing):

```
------------------ cut ----------------------
static inline unsigned long read_cr3(void)
{
	unsigned long cr3;
	asm("movq %%cr3,%0" : "=r" (cr3));
	return cr3;
}

static inline void write_cr3(unsigned long val)
{
	asm volatile("movq %0,%%cr3" :: "r" (val) : "memory");
}

void main()
{
	unsigned long c;
	c = read_cr3();
	write_cr3(c | 0x80);
	c = read_cr3();
	write_cr3(c | 0x100);
}
------------------ cut ----------------------

# objdump -dr tst
                ....
0000000000400430 <main>:
  400430:       0f 20 d8                mov    %cr3,%rax
  400433:       48 89 c2                mov    %rax,%rdx
  400436:       80 ca 80                or     $0x80,%dl
  400439:       0f 22 da                mov    %rdx,%cr3
  40043c:       80 cc 01                or     $0x1,%ah
  40043f:       0f 22 d8                mov    %rax,%cr3
  400442:       c3                      retq
                ....
```

As one can notice, cr3 value is incorrectly read only once,
and finally updated with only one bit set.

So better be on the safe side and mark all asm statements
in read_crX() functions as volatile which helps.
i686 already has most of these functions marked as volatile already.

I faced this bug myself in i686 arch code when did code
rearrangement in 2.6.18.

Signed-Off-By: Kirill Korotaev <dev@openvz.org>
Acked-By: Pavel Emelianov <xemul@openvz.org>

---

 asm-i386/system.h | 2 +-
 asm-x86_64/system.h | 8 ++++----
 2 files changed, 5 insertions(+), 5 deletions(-)

```
--- ./include/asm-i386/system.h.ve4321 2007-10-02 17:09:53.000000000 +0400
+++ ./include/asm-i386/system.h 2007-10-02 17:39:40.000000000 +0400
@@ -141,7 +141,7 @@ static inline unsigned long native_read_
 {
  unsigned long val;
  /* This could fault if %cr4 does not exist */
- asm("1: movl %%cr4, %0  \n"
+ asm volatile("1: movl %%cr4, %0  \n"
   "2:    \n"
   ".section __ex_table,\"a\" \n"
   ".long 1b,2b   \n"
--- ./include/asm-x86_64/system.h.ve4321 2007-09-18 12:42:19.000000000 +0400
+++ ./include/asm-x86_64/system.h 2007-10-02 17:40:17.000000000 +0400
@@ -85,7 +85,7 @@ static inline void write_cr0(unsigned lo
 static inline unsigned long read_cr2(void)
 {
  unsigned long cr2;
- asm("movq %%cr2,%0" : "=r" (cr2));
+ asm volatile("movq %%cr2,%0" : "=r" (cr2));
  return cr2;
 }

@@ -97,7 +97,7 @@ static inline void write_cr2(unsigned lo
 static inline unsigned long read_cr3(void)
 {
  unsigned long cr3;
- asm("movq %%cr3,%0" : "=r" (cr3));
+ asm volatile("movq %%cr3,%0" : "=r" (cr3));
  return cr3;
 }
```

```
@@ -109,7 +109,7 @@ static inline void write_cr3(unsigned lo
 static inline unsigned long read_cr4(void)
 {
  unsigned long cr4;
- asm("movq %%cr4,%0" : "=r" (cr4));
+ asm volatile("movq %%cr4,%0" : "=r" (cr4));
  return cr4;
 }

@@ -121,7 +121,7 @@ static inline void write_cr4(unsigned lo
 static inline unsigned long read_cr8(void)
 {
  unsigned long cr8;
- asm("movq %%cr8,%0" : "=r" (cr8));
+ asm volatile("movq %%cr8,%0" : "=r" (cr8));
  return cr8;
 }
```