

## [PATCH] fix x86-64 compile errors in 2.6.0-test2

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2003-07/3398.html>

---

**From:** Mikael Pettersson (*mikpe\_at\_csd.uu.se*)

**Date:** 07/31/03

Date: Thu, 31 Jul 2003 23:27:06 +0200 (MEST)  
To: ak@suse.de

x86-64 needs these two fixes to build in 2.6.0-test2:

- Add include/asm-x86\_64/local.h which is now required.  
This is a copy of i386' local.h with s/long/int/ since local\_t should be consistent with atomic\_t, which is 32 bits.
- Updated x86-64's ia32\_binfmt.c:elf\_core\_copy\_task\_fpregs() to match its updated prototype. The new struct pt\_regs\* parameter should (as far as I can tell from inspecting patch-2.6.0-test2 and x86-64's dump\_fpu()) simply be ignored.

/Mikael

```
diff -ruN linux-2.6.0-test2/arch/x86_64/ia32/ia32_binfmt.c
linux-2.6.0-test2.x86_64-fixes/arch/x86_64/ia32/ia32_binfmt.c
--- linux-2.6.0-test2/arch/x86_64/ia32/ia32_binfmt.c 2003-07-11 00:08:27.000000000 +0200
+++ linux-2.6.0-test2.x86_64-fixes/arch/x86_64/ia32/ia32_binfmt.c 2003-07-31 23:02:29.000000000
+0200
@@ -202,7 +202,7 @@
 }
```

```
static inline int
-elf_core_copy_task_fpregs(struct task_struct *tsk, elf_fpregset_t *fpu)
+elf_core_copy_task_fpregs(struct task_struct *tsk, struct pt_regs *ignored, elf_fpregset_t *fpu)
{
```

```
    struct _fpstate_ia32 *fpstate = (void*)fpu;
    struct pt_regs *regs = (struct pt_regs *) (tsk->thread.rsp0);
diff -ruN linux-2.6.0-test2/include/asm-x86_64/local.h
linux-2.6.0-test2.x86_64-fixes/include/asm-x86_64/local.h
--- linux-2.6.0-test2/include/asm-x86_64/local.h 1970-01-01 01:00:00.000000000 +0100
+++ linux-2.6.0-test2.x86_64-fixes/include/asm-x86_64/local.h 2003-07-31 23:03:28.000000000 +0200
@@ -0,0 +1,72 @@
+#ifndef _ARCH_X86_64_LOCAL_H
+#define _ARCH_X86_64_LOCAL_H
+
+/* copy of include/asm-i386/local.h with s/long/int/ for consistency with atomic_t */
+
+#include <linux/percpu.h>
+
```

```

+typedef struct
+{
+ volatile unsigned int counter;
+} local_t;
+
+#define LOCAL_INIT(i) { (i) }
+
+#define local_read(v) ((v)->counter)
+#define local_set(v,i) (((v)->counter) = (i))
+
+static __inline__ void local_inc(local_t *v)
+{
+ __asm__ __volatile__(
+ "incl %0"
+ : "=m" (v->counter)
+ : "m" (v->counter));
+}
+
+static __inline__ void local_dec(local_t *v)
+{
+ __asm__ __volatile__(
+ "decl %0"
+ : "=m" (v->counter)
+ : "m" (v->counter));
+}
+
+static __inline__ void local_add(unsigned int i, local_t *v)
+{
+ __asm__ __volatile__(
+ "addl %1,%0"
+ : "=m" (v->counter)
+ : "ir" (i), "m" (v->counter));
+}
+
+static __inline__ void local_sub(unsigned int i, local_t *v)
+{
+ __asm__ __volatile__(
+ "subl %1,%0"
+ : "=m" (v->counter)
+ : "ir" (i), "m" (v->counter));
+}
+
+/* On x86, these are no better than the atomic variants. */
+#define __local_inc(l) local_inc(l)
+#define __local_dec(l) local_dec(l)
+#define __local_add(i,l) local_add((i),(l))
+#define __local_sub(i,l) local_sub((i),(l))
+
+/* Use these for per-cpu local_t variables: on some archs they are
+ * much more efficient than these naive implementations. Note they take
+ * a variable, not an address.

```

Linux-Kernel: [PATCH] fix x86-64 compile errors in 2.6.0-test2

```
+ */
+#define cpu_local_read(v) local_read(&__get_cpu_var(v))
+#define cpu_local_set(v, i) local_set(&__get_cpu_var(v), (i))
+#define cpu_local_inc(v) local_inc(&__get_cpu_var(v))
+#define cpu_local_dec(v) local_dec(&__get_cpu_var(v))
+#define cpu_local_add(i, v) local_add((i), &__get_cpu_var(v))
+#define cpu_local_sub(i, v) local_sub((i), &__get_cpu_var(v))
+
+#define __cpu_local_inc(v) cpu_local_inc(v)
+#define __cpu_local_dec(v) cpu_local_dec(v)
+#define __cpu_local_add(i, v) cpu_local_add((i), (v))
+#define __cpu_local_sub(i, v) cpu_local_sub((i), (v))
+
+#endif /* _ARCH_X86_64_LOCAL_H */
-
```

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/>