
Subject: [PATCH -mm 2/2] x86_64: semi-rewrite of PTRACE_PEEKUSR, PTRACE_POKEUSR

Posted by [Alexey Dobriyan](#) on Wed, 20 Jun 2007 15:08:44 GMT

[View Forum Message](#) <> [Reply to Message](#)

Refactor x86_64 PTRACE_PEEKUSR, PTRACE_POKEUSR implementations through regset code.

Signed-off-by: Alexey Dobriyan <adobriyan@sw.ru>

arch/x86_64/kernel/ptrace.c | 203 ++++++-----
1 file changed, 110 insertions(+), 93 deletions(-)

--- a/arch/x86_64/kernel/ptrace.c

+++ b/arch/x86_64/kernel/ptrace.c

@@ -305,9 +305,114 @@ static unsigned long getreg(struct task_struct *child, unsigned long regno)

}

+static struct regset x86_64_gp_regset = {

+ .getreg = getreg,

+ .setreg = putreg,

+};

+

+static unsigned long x86_64_db_getreg(struct task_struct *tsk, unsigned long addr)

+{

+ switch (addr) {

+ case offsetof(struct user, u_debugreg[0]):

+ return tsk->thread.debugreg0;

+ case offsetof(struct user, u_debugreg[1]):

+ return tsk->thread.debugreg1;

+ case offsetof(struct user, u_debugreg[2]):

+ return tsk->thread.debugreg2;

+ case offsetof(struct user, u_debugreg[3]):

+ return tsk->thread.debugreg3;

+ case offsetof(struct user, u_debugreg[6]):

+ return tsk->thread.debugreg6;

+ case offsetof(struct user, u_debugreg[7]):

+ return tsk->thread.debugreg7;

+ default:

+ return 0;

+ }

+}

+

+static int x86_64_db_setreg(struct task_struct *tsk, unsigned long addr, unsigned long val)

+{

```

+ int i, dsize = test_tsk_thread_flag(tsk, TIF_IA32) ? 3 : 7;
+
+ switch (addr) {
+ case offsetof(struct user, u_debugreg[4]):
+ case offsetof(struct user, u_debugreg[5]):
+ return -EIO;
+ /* Disallows to set a breakpoint into the vsyscall */
+ case offsetof(struct user, u_debugreg[0]):
+ if (val >= TASK_SIZE_OF(tsk) - dsize)
+ return -EIO;
+ tsk->thread.debugreg0 = val;
+ return 0;
+ case offsetof(struct user, u_debugreg[1]):
+ if (val >= TASK_SIZE_OF(tsk) - dsize)
+ return -EIO;
+ tsk->thread.debugreg1 = val;
+ return 0;
+ case offsetof(struct user, u_debugreg[2]):
+ if (val >= TASK_SIZE_OF(tsk) - dsize)
+ return -EIO;
+ tsk->thread.debugreg2 = val;
+ return 0;
+ case offsetof(struct user, u_debugreg[3]):
+ if (val >= TASK_SIZE_OF(tsk) - dsize)
+ return -EIO;
+ tsk->thread.debugreg3 = val;
+ return 0;
+ case offsetof(struct user, u_debugreg[6]):
+ if (val >> 32)
+ return -EIO;
+ tsk->thread.debugreg6 = val;
+ return 0;
+ case offsetof(struct user, u_debugreg[7]):
+ /*
+  * See arch/i386/kernel/ptrace.c for an explanation of this
+  * awkward check.
+  */
+ val &= ~DR_CONTROL_RESERVED;
+ for (i = 0; i < 4; i++)
+ if ((0x5554 >> ((val >> (16 + 4 * i)) & 0xf)) & 1)
+ return -EIO;
+ tsk->thread.debugreg7 = val;
+ if (val)
+ set_tsk_thread_flag(tsk, TIF_DEBUG);
+ else
+ clear_tsk_thread_flag(tsk, TIF_DEBUG);
+ return 0;
+ default:

```

```

+ BUG();
+ }
+}
+
+static struct regset x86_64_db_regset = {
+ .getreg = x86_64_db_getreg,
+ .setreg = x86_64_db_setreg,
+};
+
+static struct ptrace_usr_area x86_64_usr_area[] = {
+ {
+ .start = 0,
+ .end = sizeof(struct user_regs_struct),
+ .regset = &x86_64_gp_regset,
+ }, {
+ .start = sizeof(struct user_regs_struct),
+ .end = offsetof(struct user, u_debugreg[0]),
+ .hole = 1,
+ }, {
+ .start = offsetof(struct user, u_debugreg[0]),
+ .end = offsetof(struct user, u_debugreg[8]),
+ .regset = &x86_64_db_regset,
+ }, {
+ .start = offsetof(struct user, u_debugreg[8]),
+ .end = sizeof(struct user),
+ .hole = 1,
+ },
+ {}
+};
+
long arch_ptrace(struct task_struct *child, long request, long addr, long data)
{
- long i, ret;
+ long ret;
  unsigned ui;

  switch (request) {
@@ -318,43 +423,9 @@ long arch_ptrace(struct task_struct *child, long request, long addr, long
data)
    break;

    /* read the word at location addr in the USER area. */
- case PTRACE_PEEKUSR: {
-   unsigned long tmp;
-
-   ret = -EIO;
-   if ((addr & 7) ||
-       addr > sizeof(struct user) - 7)

```

```

- break;
-
- switch (addr) {
- case 0 ... sizeof(struct user_regs_struct) - sizeof(long):
- tmp = getreg(child, addr);
- break;
- case offsetof(struct user, u_debugreg[0]):
- tmp = child->thread.debugreg0;
- break;
- case offsetof(struct user, u_debugreg[1]):
- tmp = child->thread.debugreg1;
- break;
- case offsetof(struct user, u_debugreg[2]):
- tmp = child->thread.debugreg2;
- break;
- case offsetof(struct user, u_debugreg[3]):
- tmp = child->thread.debugreg3;
- break;
- case offsetof(struct user, u_debugreg[6]):
- tmp = child->thread.debugreg6;
- break;
- case offsetof(struct user, u_debugreg[7]):
- tmp = child->thread.debugreg7;
- break;
- default:
- tmp = 0;
- break;
- }
- ret = put_user(tmp,(unsigned long __user *) data);
+ case PTRACE_PEEKUSR:
+ ret = ptrace_peekusr(child, x86_64_usr_area, addr, data);
  break;
- }

/* when I and D space are separate, this will have to be fixed. */
case PTRACE_POKETEXT: /* write the word at location addr. */
@@ -363,63 +434,9 @@ long arch_ptrace(struct task_struct *child, long request, long addr, long
data)
  break;

case PTRACE_POKEUSR: /* write the word at location addr in the USER area */
- {
- int dsize = test_tsk_thread_flag(child, TIF_IA32) ? 3 : 7;
- ret = -EIO;
- if ((addr & 7) ||
-     addr > sizeof(struct user) - 7)
- break;
- }

```

```

- switch (addr) {
- case 0 ... sizeof(struct user_regs_struct) - sizeof(long):
- ret = putreg(child, addr, data);
- break;
- /* Disallows to set a breakpoint into the vsyscall */
- case offsetof(struct user, u_debugreg[0]):
- if (data >= TASK_SIZE_OF(child) - dsize) break;
- child->thread.debugreg0 = data;
- ret = 0;
- break;
- case offsetof(struct user, u_debugreg[1]):
- if (data >= TASK_SIZE_OF(child) - dsize) break;
- child->thread.debugreg1 = data;
- ret = 0;
- break;
- case offsetof(struct user, u_debugreg[2]):
- if (data >= TASK_SIZE_OF(child) - dsize) break;
- child->thread.debugreg2 = data;
- ret = 0;
- break;
- case offsetof(struct user, u_debugreg[3]):
- if (data >= TASK_SIZE_OF(child) - dsize) break;
- child->thread.debugreg3 = data;
- ret = 0;
- break;
- case offsetof(struct user, u_debugreg[6]):
- if (data >> 32)
- break;
- child->thread.debugreg6 = data;
- ret = 0;
- break;
- case offsetof(struct user, u_debugreg[7]):
- /* See arch/i386/kernel/ptrace.c for an explanation of
- * this awkward check.*/
- data &= ~DR_CONTROL_RESERVED;
- for(i=0; i<4; i++)
- if ((0x5554 >> ((data >> (16 + 4*i)) & 0xf)) & 1)
- break;
- if (i == 4) {
- child->thread.debugreg7 = data;
- if (data)
- set_tsk_thread_flag(child, TIF_DEBUG);
- else
- clear_tsk_thread_flag(child, TIF_DEBUG);
- ret = 0;
- }
- break;
- }

```

```
+ ret = ptrace_pokeusr(child, x86_64_usr_area, addr, data);
  break;
- }
+
  case PTRACE_SYSCALL: /* continue and stop at next (return from) syscall */
  case PTRACE_CONT: /* restart after signal. */
```
