

# [PATCH 1/2] FRV: Fix FRV signal handling

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-11/9512.html>

---

**From:** David Howells (*dhowells\_at\_redhat.com*)

**Date:** 11/29/05

Date: Tue, 29 Nov 2005 21:18:00 GMT  
To: torvalds@osdl.org, akpm@osdl.org, dalomar@serrasold.com

The attached patch makes FRV signal handling work properly:

- (1) After `do_notify_resume()` has been called, the work flags must be checked again (there may be another signal to deliver or the process might require rescheduling for instance).
- (2) After the signal frame is set up on the userspace stack, `ptrace()` should be given an opportunity to single-step into the signal handler.
- (3) The error state from setting up a signal frame should be passed back up the call chain.
- (4) The segfault handler shouldn't be preemptively reset in the arch if we fail to deliver a SEGV signal: `force_sig()` will take care of that.

Signed-Off-By: David Howells <dhowells@redhat.com>

```
---
warthog>diffstat -p1 frv-signal-2615rc2.diff
 arch/frv/kernel/entry.S |    2 -
 arch/frv/kernel/signal.c |   73 ++++++-----
 2 files changed, 46 insertions(+), 29 deletions(-)
diff -uNrp linux-2.6.15-rc2-frv-shmem/arch/frv/kernel/entry.S linux-2.6.15-rc2-frv-signal/arch/frv-shmem/arch/frv/kernel/entry.S
--- linux-2.6.15-rc2-frv-shmem/arch/frv/kernel/entry.S 2005-03-02 12:07:44.000000000 +0000
+++ linux-2.6.15-rc2-frv-signal/arch/frv/kernel/entry.S 2005-11-29 16:01:40.000000000 +0000
@@ -1076,7 +1076,7 @@ __entry_work_notifysig:
     LEDS                0x6410
     ori.p               gr4,#0,gr8
     call                do_notify_resume
-    bra                 __entry_return_direct
+    bra                 __entry_resume_userspace

     # perform syscall entry tracing
__syscall_trace_entry:
diff -uNrp linux-2.6.15-rc2-frv-shmem/arch/frv/kernel/signal.c linux-2.6.15-rc2-frv-signal/arch/frv-shmem/arch/frv/kernel/signal.c
--- linux-2.6.15-rc2-frv-shmem/arch/frv/kernel/signal.c 2005-11-01 13:18:57.000000000 +0000
+++ linux-2.6.15-rc2-frv-signal/arch/frv/kernel/signal.c 2005-11-29 16:41:11.000000000 +0000
@@ -297,7 +297,8 @@ static inline void __user *get_sigframe(
 /*
 *
 */
-static void setup_frame(int sig, struct k_sigaction *ka, sigset_t *set, struct pt_regs * regs)
```

## Linux-Kernel: [PATCH 1/2] FRV: Fix FRV signal handling

```
+static int setup_frame(int sig, struct k_sigaction *ka, sigset_t *set,
+                        struct pt_regs *regs)
+
+  {
+    struct sigframe __user *frame;
+    int rsig;
@@ -362,26 +363,30 @@ static void setup_frame(int sig, struct
+
+    set_fs(USER_DS);
+
+    /* the tracer may want to single-step inside the handler */
+    if (test_thread_flag(TIF_SINGLESTEP))
+        ptrace_notify(SIGTRAP);
+
+    #if DEBUG_SIG
+        printk("SIG deliver %d (%s:%d): sp=%p pc=%lx ra=%p\n",
-            sig, current->comm, current->pid, frame, regs->pc, frame->precode);
+            sig, current->comm, current->pid, frame, regs->pc,
+            frame->precode);
+    #endif
-
-    return;
+
+    return 1;
+
+    give_sigsegv:
-    if (sig == SIGSEGV)
-        ka->sa.sa_handler = SIG_DFL;
-
+    force_sig(SIGSEGV, current);
+    return 0;
+
+  } /* end setup_frame() */
+
+  /*
+   *
+   */
-static void setup_rt_frame(int sig, struct k_sigaction *ka, siginfo_t *info,
-                          sigset_t *set, struct pt_regs * regs)
+static int setup_rt_frame(int sig, struct k_sigaction *ka, siginfo_t *info,
+                          sigset_t *set, struct pt_regs * regs)
+
+  {
+    struct rt_sigframe __user *frame;
+    int rsig;
@@ -457,17 +462,21 @@ static void setup_rt_frame(int sig, stru
+
+    set_fs(USER_DS);
+
+    /* the tracer may want to single-step inside the handler */
+    if (test_thread_flag(TIF_SINGLESTEP))
+        ptrace_notify(SIGTRAP);
+
+    #if DEBUG_SIG
+        printk("SIG deliver %d (%s:%d): sp=%p pc=%lx ra=%p\n",
-            sig, current->comm, current->pid, frame, regs->pc, frame->precode);
+            sig, current->comm, current->pid, frame, regs->pc,
+            frame->precode);
+    #endif
-
-    return;
+
+    return 1;
+
+    give_sigsegv:
```

## Linux-Kernel: [PATCH 1/2] FRV: Fix FRV signal handling

```

-     if (sig == SIGSEGV)
-         ka->sa.sa_handler = SIG_DFL;
-         force_sig(SIGSEGV, current);
+     return 0;

} /* end setup_rt_frame() */

@@ -475,10 +484,12 @@ give_sigsegv:
/*
 * OK, we're invoking a handler
 */
-static void handle_signal(unsigned long sig, siginfo_t *info,
-                          struct k_sigaction *ka, sigset_t *oldset,
-                          struct pt_regs *regs)
+static int handle_signal(unsigned long sig, siginfo_t *info,
+                          struct k_sigaction *ka, sigset_t *oldset,
+                          struct pt_regs *regs)
{
+    int ret;
+
+    /* Are we from a system call? */
+    if (in_syscall(regs)) {
+        /* If so, check system call restarting.. */
@@ -493,6 +504,7 @@ static void handle_signal(unsigned long
+        regs->gr8 = -EINTR;
+        break;
+    }

+    /* fallthrough */
+    case -ERESTARTNOINTR:
+        regs->gr8 = regs->orig_gr8;
@@ -502,16 +514,22 @@ static void handle_signal(unsigned long

+    /* Set up the stack frame */
+    if (ka->sa.sa_flags & SA_SIGINFO)
-        setup_rt_frame(sig, ka, info, oldset, regs);
+        ret = setup_rt_frame(sig, ka, info, oldset, regs);
+    else
-        setup_frame(sig, ka, oldset, regs);
+        ret = setup_frame(sig, ka, oldset, regs);
+
+    if (ret) {
+        spin_lock_irq(&current->sighand->siglock);
+        sigorsets(&current->blocked, &current->blocked,
+                 &ka->sa.sa_mask);
+        if (!(ka->sa.sa_flags & SA_NODEFER))
+            sigaddset(&current->blocked, sig);
+        recalc_sigpending();
+        spin_unlock_irq(&current->sighand->siglock);
+    }

+    return ret;

-    spin_lock_irq(&current->sighand->siglock);
-    sigorsets(&current->blocked, &current->blocked, &ka->sa.sa_mask);
-    if (!(ka->sa.sa_flags & SA_NODEFER))
-        sigaddset(&current->blocked, sig);
-    recalc_sigpending();
-    spin_unlock_irq(&current->sighand->siglock);
} /* end handle_signal() */

/*****/

```

## Linux-Kernel: [PATCH 1/2] FRV: Fix FRV signal handling

```
@@ -542,12 +560,10 @@ int do_signal(struct pt_regs *regs, sigs
    oldset = &current->blocked;

    signr = get_signal_to_deliver(&info, &ka, regs, NULL);
-   if (signr > 0) {
-       handle_signal(signr, &info, &ka, oldset, regs);
-       return 1;
-   }
+   if (signr > 0)
+       return handle_signal(signr, &info, &ka, oldset, regs);

- no_signal:
+no_signal:
    /* Did we come from a system call? */
    if (regs->syscallno >= 0) {
        /* Restart the system call - no handlers present */
@@ -565,6 +581,7 @@ int do_signal(struct pt_regs *regs, sigs
    }

    return 0;
+
+ } /* end do_signal() */

/*****/
-
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/
```