Subject: [PATCH 1/2] ia64: add TIF_RESTORE_SIGMASK
Posted by adobriyan on Tue, 23 Jan 2007 15:59:58 GMT
View Forum Message <> Reply to Message

ia32 compat code not tested. :-(
please, point fingers if I broke something. --adobriyan

[PATCH] ia64: add TIF_RESTORE_SIGMASK

From: Alexey Kuznetsov <kuznet@ms2.inr.ac.ru>

Signed-off-by: Alexey Kuznetsov <kuznet@ms2.inr.ac.ru>
Signed-off-by: Alexey Dobriyan <adobriyan@openvz.org>

```
 arch/ia64/ia32/ia32_entry.S    |   39 ----------------------
 arch/ia64/ia32/ia32_signal.c   |   59 +++++-----------------------------
 arch/ia64/kernel/entry.S       |   26 ---------------
 arch/ia64/kernel/process.c     |    6 +--
 arch/ia64/kernel/sigframe.h    |    2 -
 arch/ia64/kernel/signal.c      |   71 ++++++++++++++----------------------------
 include/asm-ia64/thread_info.h |    4 +-
 include/asm-ia64/unistd.h      |    2 +
 8 files changed, 43 insertions(+), 166 deletions(-)

--- a/arch/ia64/ia32/ia32_entry.S
+++ b/arch/ia64/ia32/ia32_entry.S
@@ -52,43 +52,6 @@ ENTRY(ia32_clone)
  br.ret.sptk.many rp
 END(ia32_clone)

-ENTRY(sys32_rt_sigsuspend)
- .prologue ASM_UNW_PRLG_RP|ASM_UNW_PRLG_PFS, ASM_UNW_PRLG_GRSAVE(8)
- alloc loc1=ar.pfs,8,2,3,0  // preserve all eight input regs
- mov loc0=rp
- mov out0=in0    // mask
- mov out1=in1    // sigsetsize
- mov out2=sp    // out2 = &sigscratch
- .fframe 16
- adds sp=-16,sp    // allocate dummy "sigscratch"
- ;;
- .body
- br.call.sptk.many rp=ia32_rt_sigsuspend
-1: .restore sp
- adds sp=16,sp
- mov rp=loc0
- mov ar.pfs=loc1
- br.ret.sptk.many rp
```

```
-END(sys32_rt_sigsuspend)
-
-ENTRY(sys32_sigsuspend)
- .prologue ASM_UNW_PRLG_RP|ASM_UNW_PRLG_PFS, ASM_UNW_PRLG_GRSAVE(8)
- alloc loc1=ar.pfs,8,2,3,0  // preserve all eight input regs
- mov loc0=rp
- mov out0=in2    // mask (first two args are ignored)
- ;;
- mov out1=sp    // out1 = &sigscratch
- .fframe 16
- adds sp=-16,sp    // allocate dummy "sigscratch"
- .body
- br.call.sptk.many rp=ia32_sigsuspend
-1: .restore sp
- adds sp=16,sp
- mov rp=loc0
- mov ar.pfs=loc1
- br.ret.sptk.many rp
-END(sys32_sigsuspend)
-
 GLOBAL_ENTRY(ia32_ret_from_clone)
  PT_REGS_UNWIND_INFO(0)
 { /*
@@ -389,7 +352,7 @@ ia32_syscall_table:
  data8 sys_rt_sigpending
  data8 compat_sys_rt_sigtimedwait
  data8 sys32_rt_sigqueueinfo
- data8 sys32_rt_sigsuspend
+ data8 compat_sys_rt_sigsuspend
  data8 sys32_pread   /* 180 */
  data8 sys32_pwrite
  data8 sys_chown /* 16-bit version */
--- a/arch/ia64/ia32/ia32_signal.c
+++ b/arch/ia64/ia32/ia32_signal.c
@@ -452,59 +452,20 @@ sigact_set_handler (struct k_sigaction *
   sa->sa.sa_handler = (__sighandler_t) (((unsigned long) restorer << 32) | handler);
 }

-long
-__ia32_rt_sigsuspend (compat_sigset_t *sset, unsigned int sigsetsize, struct sigscratch *scr)
+asmlinkage long
+sys32_sigsuspend (int history0, int history1, old_sigset_t mask)
 {
- extern long ia64_do_signal (sigset_t *oldset, struct sigscratch *scr, long in_syscall);
- sigset_t oldset, set;
-
- scr->scratch_unat = 0; /* avoid leaking kernel bits to user level */
- memset(&set, 0, sizeof(set));
```

```
-
- memcpy(&set.sig, &sset->sig, sigsetsize);
-
- sigdelsetmask(&set, ~_BLOCKABLE);
-
+ mask &= _BLOCKABLE;
  spin_lock_irq(&current->sighand->siglock);
- {
-  oldset = current->blocked;
-  current->blocked = set;
-  recalc_sigpending();
- }
+ current->saved_sigmask = current->blocked;
+ siginitset(&current->blocked, mask);
+ recalc_sigpending();
  spin_unlock_irq(&current->sighand->siglock);

- /*
-  * The return below usually returns to the signal handler.  We need to pre-set the
-  * correct error code here to ensure that the right values get saved in sigcontext
-  * by ia64_do_signal.
-  */
- scr->pt.r8 = -EINTR;
- while (1) {
-  current->state = TASK_INTERRUPTIBLE;
-  schedule();
-  if (ia64_do_signal(&oldset, scr, 1))
-   return -EINTR;
- }
-}
-
-asmlinkage long
-ia32_rt_sigsuspend (compat_sigset_t __user *uset, unsigned int sigsetsize, struct sigscratch
*scr)
-{
- compat_sigset_t set;
-
- if (sigsetsize > sizeof(compat_sigset_t))
-  return -EINVAL;
-
- if (copy_from_user(&set.sig, &uset->sig, sigsetsize))
-  return -EFAULT;
-
- return __ia32_rt_sigsuspend(&set, sigsetsize, scr);
-}
-
-asmlinkage long
-ia32_sigsuspend (unsigned int mask, struct sigscratch *scr)
```

```
-{
- return __ia32_rt_sigsuspend((compat_sigset_t *) &mask, sizeof(mask), scr);
+ current->state = TASK_INTERRUPTIBLE;
+ schedule();
+ set_thread_flag(TIF_RESTORE_SIGMASK);
+ return -ERESTARTNOHAND;
 }

 asmlinkage long
--- a/arch/ia64/kernel/entry.S
+++ b/arch/ia64/kernel/entry.S
@@ -1202,32 +1202,6 @@ (pNonSys) mov out2=0    // out2==0 => no
  br.ret.sptk.many rp
 END(notify_resume_user)

-GLOBAL_ENTRY(sys_rt_sigsuspend)
- .prologue ASM_UNW_PRLG_RP|ASM_UNW_PRLG_PFS, ASM_UNW_PRLG_GRSAVE(8)
- alloc loc1=ar.pfs,8,2,3,0 // preserve all eight input regs in case of syscall restart!
- mov r9=ar.unat
- mov loc0=rp    // save return address
- mov out0=in0    // mask
- mov out1=in1    // sigsetsize
- adds out2=8,sp    // out2=&sigscratch->ar_pfs
- ;;
- .fframe 16
- .spillsp ar.unat, 16
- st8 [sp]=r9,-16    // allocate space for ar.unat and save it
- st8 [out2]=loc1,-8   // save ar.pfs, out2=&sigscratch
- .body
- br.call.sptk.many rp=ia64_rt_sigsuspend
-.ret17: .restore sp
- adds sp=16,sp    // pop scratch stack space
- ;;
- ld8 r9=[sp]    // load new unat from sw->caller_unat
- mov rp=loc0
- ;;
- mov ar.unat=r9
- mov ar.pfs=loc1
- br.ret.sptk.many rp
-END(sys_rt_sigsuspend)
-
 ENTRY(sys_rt_sigreturn)
  PT_REGS_UNWIND_INFO(0)
  /*
--- a/arch/ia64/kernel/process.c
+++ b/arch/ia64/kernel/process.c
@@ -155,7 +155,7 @@ show_regs (struct pt_regs *regs)
 }
```

```
 void
-do_notify_resume_user (sigset_t *oldset, struct sigscratch *scr, long in_syscall)
+do_notify_resume_user (sigset_t *unused, struct sigscratch *scr, long in_syscall)
 {
  if (fsys_mode(current, &scr->pt)) {
   /* defer signal-handling etc. until we return to privilege-level 0.  */
@@ -170,8 +170,8 @@ #ifdef CONFIG_PERFMON
 #endif

  /* deal with pending signal delivery */
- if (test_thread_flag(TIF_SIGPENDING))
-  ia64_do_signal(oldset, scr, in_syscall);
+ if  (test_thread_flag(TIF_SIGPENDING)||test_thread_flag(TIF_REST ORE_SIGMASK))
+  ia64_do_signal(scr, in_syscall);
 }

 static int pal_halt        = 1;
--- a/arch/ia64/kernel/sigframe.h
+++ b/arch/ia64/kernel/sigframe.h
@@ -22,4 +22,4 @@ struct sigframe {
  struct sigcontext sc;
 };

-extern long ia64_do_signal (sigset_t *, struct sigscratch *, long);
+extern void ia64_do_signal (struct sigscratch *, long);
--- a/arch/ia64/kernel/signal.c
+++ b/arch/ia64/kernel/signal.c
@@ -41,47 +41,6 @@ # define PUT_SIGSET(k,u) __put_user((k)-
 # define GET_SIGSET(k,u) __get_user((k)->sig[0], &(u)->sig[0])
 #endif

-long
-ia64_rt_sigsuspend (sigset_t __user *uset, size_t sigsetsize, struct sigscratch *scr)
-{
- sigset_t oldset, set;
-
- /* XXX: Don't preclude handling different sized sigset_t's.  */
- if (sigsetsize != sizeof(sigset_t))
-  return -EINVAL;
-
- if (!access_ok(VERIFY_READ, uset, sigsetsize))
-  return -EFAULT;
-
- if (GET_SIGSET(&set, uset))
-  return -EFAULT;
-
- sigdelsetmask(&set, ~_BLOCKABLE);
```

```
-
- spin_lock_irq(&current->sighand->siglock);
- {
-  oldset = current->blocked;
-  current->blocked = set;
-  recalc_sigpending();
- }
- spin_unlock_irq(&current->sighand->siglock);
-
- /*
-  * The return below usually returns to the signal handler.  We need to
-  * pre-set the correct error code here to ensure that the right values
-  * get saved in sigcontext by ia64_do_signal.
-  */
- scr->pt.r8 = EINTR;
- scr->pt.r10 = -1;
-
- while (1) {
-  current->state = TASK_INTERRUPTIBLE;
-  schedule();
-  if (ia64_do_signal(&oldset, scr, 1))
-   return -EINTR;
- }
-}
-
 asmlinkage long
 sys_sigaltstack (const stack_t __user *uss, stack_t __user *uoss, long arg2,
    long arg3, long arg4, long arg5, long arg6, long arg7,
@@ -478,10 +437,11 @@ handle_signal (unsigned long sig, struct
  * Note that `init' is a special process: it doesn't get signals it doesn't want to
  * handle.  Thus you cannot kill init even with a SIGKILL even by mistake.
  */
-long
-ia64_do_signal (sigset_t *oldset, struct sigscratch *scr, long in_syscall)
+void
+ia64_do_signal (struct sigscratch *scr, long in_syscall)
 {
  struct k_sigaction ka;
+ sigset_t *oldset;
  siginfo_t info;
  long restart = in_syscall;
  long errno = scr->pt.r8;
@@ -493,9 +453,11 @@ # define ERR_CODE(c) (IS_IA32_PROCESS(&s
   * doing anything if so.
   */
  if (!user_mode(&scr->pt))
-  return 0;
+  return;
```

```
- if (!oldset)
+ if (test_thread_flag(TIF_RESTORE_SIGMASK))
+  oldset = &current->saved_sigmask;
+ else
   oldset = &current->blocked;

  /*
@@ -558,8 +520,15 @@ # define ERR_CODE(c) (IS_IA32_PROCESS(&s
    * Whee!  Actually deliver the signal.  If the delivery failed, we need to
    * continue to iterate in this loop so we can deliver the SIGSEGV...
    */
-  if (handle_signal(signr, &ka, &info, oldset, scr))
-   return 1;
+  if (handle_signal(signr, &ka, &info, oldset, scr)) {
+   /* a signal was successfully delivered; the saved
+    * sigmask will have been stored in the signal frame,
+    * and will be restored by sigreturn, so we can simply
+    * clear the TIF_RESTORE_SIGMASK flag */
+   if (test_thread_flag(TIF_RESTORE_SIGMASK))
+    clear_thread_flag(TIF_RESTORE_SIGMASK);
+   return;
+  }
  }

  /* Did we come from a system call? */
@@ -585,5 +554,11 @@ # define ERR_CODE(c) (IS_IA32_PROCESS(&s
    }
   }
  }
- return 0;
+
+ /* if there's no signal to deliver, we just put the saved sigmask
+  * back */
+ if (test_thread_flag(TIF_RESTORE_SIGMASK)) {
+  clear_thread_flag(TIF_RESTORE_SIGMASK);
+  sigprocmask(SIG_SETMASK, &current->saved_sigmask, NULL);
+ }
 }
--- a/include/asm-ia64/thread_info.h
+++ b/include/asm-ia64/thread_info.h
@@ -84,6 +84,7 @@ #define TIF_SIGPENDING  1 /* signal pend
 #define TIF_NEED_RESCHED 2 /* rescheduling necessary */
 #define TIF_SYSCALL_TRACE 3 /* syscall trace active */
 #define TIF_SYSCALL_AUDIT 4 /* syscall auditing active */
+#define TIF_RESTORE_SIGMASK 5 /* restore signal mask in do_signal() */
 #define TIF_POLLING_NRFLAG 16 /* true if poll_idle() is polling TIF_NEED_RESCHED */
 #define TIF_MEMDIE  17
```

```
 #define TIF_MCA_INIT  18 /* this task is processing MCA or INIT */
@@ -94,6 +95,7 @@ #define _TIF_SYSCALL_TRACE (1 << TIF_SYS
 #define _TIF_SYSCALL_AUDIT (1 << TIF_SYSCALL_AUDIT)
 #define _TIF_SYSCALL_TRACEAUDIT (_TIF_SYSCALL_TRACE|_TIF_SYSCALL_AUDIT)
 #define _TIF_NOTIFY_RESUME (1 << TIF_NOTIFY_RESUME)
+#define _TIF_RESTORE_SIGMASK (1 << TIF_RESTORE_SIGMASK)
 #define _TIF_SIGPENDING  (1 << TIF_SIGPENDING)
 #define _TIF_NEED_RESCHED (1 << TIF_NEED_RESCHED)
 #define _TIF_POLLING_NRFLAG (1 << TIF_POLLING_NRFLAG)
@@ -102,7 +104,7 @@ #define _TIF_DB_DISABLED (1 << TIF_DB_DI
 #define _TIF_FREEZE  (1 << TIF_FREEZE)

 /* "work to do on user-return" bits */
-#define TIF_ALLWORK_MASK
(_TIF_NOTIFY_RESUME|_TIF_SIGPENDING|_TIF_NEED_RESCHED|_TIF_S
YSCALL_TRACE|_TIF_SYSCALL_AUDIT)
+#define TIF_ALLWORK_MASK
(_TIF_NOTIFY_RESUME|_TIF_SIGPENDING|_TIF_NEED_RESCHED|_TIF_S
YSCALL_TRACE|_TIF_SYSCALL_AUDIT|_TIF_RESTORE_SIGMASK)
 /* like TIF_ALLWORK_BITS but sans TIF_SYSCALL_TRACE or TIF_SYSCALL_AUDIT */
 #define TIF_WORK_MASK
(TIF_ALLWORK_MASK&~(_TIF_SYSCALL_TRACE|_TIF_SYSCALL_AUDI T))

--- a/include/asm-ia64/unistd.h
+++ b/include/asm-ia64/unistd.h
@@ -298,6 +298,7 @@ #ifdef __KERNEL__
 #define NR_syscalls   279 /* length of syscall table */

 #define __ARCH_WANT_SYS_RT_SIGACTION
+#define __ARCH_WANT_SYS_RT_SIGSUSPEND

 #ifdef CONFIG_IA32_SUPPORT
 # define __ARCH_WANT_SYS_FADVISE64
@@ -308,6 +309,7 @@ # define __ARCH_WANT_SYS_OLD_GETRLIMIT
 # define __ARCH_WANT_SYS_OLDUMOUNT
 # define __ARCH_WANT_SYS_SIGPENDING
 # define __ARCH_WANT_SYS_SIGPROCMASK
+# define __ARCH_WANT_COMPAT_SYS_RT_SIGSUSPEND
 # define __ARCH_WANT_COMPAT_SYS_TIME
 #endif
```