Subject: utrace regressions (was: -mm merge plans for 2.6.21)
Posted by adobriyan on Mon, 12 Feb 2007 12:47:06 GMT
View Forum Message <> Reply to Message

> utrace-utrace-tracehook.patch
> utrace-utrace-tracehook-ia64.patch
> utrace-utrace-tracehook-sparc64.patch
> utrace-utrace-tracehook-s390.patch
> utrace-utrace-regset.patch
> utrace-utrace-regset-ia64.patch
> utrace-utrace-regset-sparc64.patch
> utrace-utrace-regset-s390.patch
> utrace-utrace-core.patch
> utrace-utrace-ptrace-compat.patch
> utrace-utrace-ptrace-compat-ia64.patch
> utrace-utrace-ptrace-compat-sparc64.patch
> utrace-utrace-ptrace-compat-s390.patch

> utrace just got added to -mm.

We're aware of two regressions compared to mainline if ptrace is utrace:

1) zero holes for PTRACE_PEEKUSR vanished.

strace(1) when sees unknown syscall, does something like:

```
for (i = 0; i < MAX_ARGS; i++)
  if (ptrace(PTRACE_PEEKUSR, i * 4, ...) < 0)
 return -1;
```

since THREAD_SIZE is 17 on x86 which is less that MAX_ARGS (32 on x86), you'll get -EIO(utrace) or 0(ptrace) when i=0.

No new syscalls stracing for developers.

Proposed patch:

```
--- a/arch/i386/kernel/ptrace.c
+++ b/arch/i386/kernel/ptrace.c
@@ -728,6 +728,7 @@ EXPORT_SYMBOL_GPL(utrace_i386_native);
 #ifdef CONFIG_PTRACE
 static const struct ptrace_layout_segment i386_uarea[] = {
  {0, FRAME_SIZE*4, 0, 0},
+ {FRAME_SIZE*4, offsetof(struct user, u_debugreg[0]), -1, 0},
  {offsetof(struct user, u_debugreg[0]),
   offsetof(struct user, u_debugreg[8]), 4, 0},
  {0, 0, -1, 0}
```

Looks like x86_64 also needs zero holes added, haven't checked in runtime.

---------- -------

2. The following proggie renders box unusable in ~10 seconds (but not
   mainline kernel where Ctrl+C will kill process).

```
#include <stdio.h>
#include <unistd.h>
#include <pthread.h>
#include <sys/ptrace.h>
#include <signal.h>

static void *thread_func(void *arg)
{
 execl("/proc/self/exe", NULL);
 return NULL;
}

int main(int argc, const char *argv[])
{
 pthread_t thread;
 int pid, n;

 if (argv[0] && (pid = fork()))
  for (n = 1; /*n < 1000000*/; ++n) {
   ptrace(PTRACE_ATTACH, pid, NULL, 0);
   ptrace(PTRACE_DETACH, pid, NULL, 0);
   if (!(n % 100000))
    printf("passed: %d\n", n);
  }

 if (pthread_create(&thread, NULL, thread_func, NULL))
  perror("pthread_create");

 while (1)
  pause();
 return 1;
}
```