Subject: Re: idl6.3 bug on Macintel?
Posted by Karl Schultz on Wed, 18 Oct 2006 15:08:43 GMT
View Forum Message <> Reply to Message

On Tue, 17 Oct 2006 11:47:18 -0500, Christopher Thom wrote:

> Quoth Kenneth Bowman:
>
>>> I have no idea even how to begin tracing down the problem, or reporting
>>> the bug to RSI...so any advice most welcome!
>>>
>>> cheers
>>> chris
>>
>> There is a known problem. See this tech note
>>
>> http://www.ittvis.com/services/techtip.asp?ttid=4081
>
> Ahhhhh. Thanks for the pointer. I searched the tech tips for my error
> message, but didn't turn up anything.

There is no error message listed in the tech tip because an OS-induced memory corruption error can manifest itself in many ways:

1) Application crashes. No error messages, only a crash log.
2) IDL could issue any one of dozens of error messages.
3) No error message at all, only silent data corruption. This is probably the worse case situation because you may not know there's a problem and your app could give you plausible, yet incorrect results. In fact, this is how we found the problem here at ITTVIS - noticed that an image had a few incorrect pixels in the ENVI application.

Faithful followers of this newsgroup may recall a similar issue with the ppc architecture back in the OS X 10.1 or 10.2 days. The memcpy() function is implemented with Altivec instructions to speed it up. The OS X signal handler mechanism didn't save/restore the Altivec registers when handling the signal. If the signal handler code used memcpy(), it would change the state of the Altivec registers and when the main thread resumed execution of the memcpy, the wrong stuff was in the Altivec registers, and plop!

(IDL uses a couple of signal handlers for various things, one of them being repairing graphics windows periodically)

Memcpy() in OS X for the intel architecture uses the MMX registers for performance. And the signal handler was failing to save/restore the MMX registers. The situation is slightly worse here because the MMX registers and the floating point stack regs share the same register file on these chips. So, if the signal hander does a memcpy() or executes an FP

operation, the MMX register state changes, and the memcpy operation in the main thread is compromised.

Here's a minimal C program that demonstrates the problem.

```
/*
  Compile with:

  cc -o pgm pgm.c

  If the return value from memcmp is not zero, then the problem exists.
*/
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <strings.h>
#include <sys/time.h>

int nTimerPops = 0;
float f = 1;
static void timer_sigalrm_handler(int signo)
{
  nTimerPops++;
  f = 1.000001 * f;  // delete this line and bcopy works ok.
}


int main(int argc, char **argv)
{
  char *src, *dst;
  int n = 10 * 4096 * 4096;
  int i, rc;
  struct itimerval interval;
  /* init data that we will bcopy repeatedly */
  src = (char*)malloc(n);
  dst = (char*)malloc(n);
  bzero(src,n);
  /* set up timer and signal handler */
  interval.it_value.tv_sec = 0;
  interval.it_value.tv_usec = 10000;
  interval.it_interval.tv_sec = 0;
  interval.it_interval.tv_usec = 10000;
  signal(SIGALRM, timer_sigalrm_handler);
  rc = setitimer(ITIMER_REAL, &interval, NULL);
  /* start bcopy operations */
  for(i=0; i<50; i++) {
    bcopy(src,dst,n);
    rc = memcmp(src,dst,n);
```

```
    printf("iter=%d memcmp=%d num pops=%d\n", i, rc, nTimerPops);
  }
  return 0;
}
```

It's both entertaining and disturbing to run this on 10.4.7 (intel) and watch the failures. The good news is that it is fixed in 10.4.8.

> Now my dilemma...I just spent 2days re-installing 10.4.7 after discovering
> all my X apps suddenly didn't work. The proverbial rock and a hard place
> :-(

Understood. The tech tip does a good job of explaining your options. We reported the X app problem to Apple and they said it was a duplicate, so others have also reported the same problem. All we can do is hope that 10.4.9, or a patch for this specific problem, comes out soon.

Karl
