



Etnus TotalView on the Cray X1

Bob Moench
May 2, 2003



Presentation Overview



- Etnus TotalView Background
- Debugging issues unique to Cray X1
- Current capabilities
- Next release 4Q03
- Still to Come
- Slide show demonstration
- Summary



Etnus TotalView Background



- Common ancestor with Cray TotalView (CTV)
- Cray purchased software rights to Etnus TotalView (ETV) from Etnus LLC
- Periodic updates included
- Command Line Interface (CLI) version, `totalviewcli`, released December 2002
- Graphical User Interface (GUI) version, `totalview`, in final exposure
 - Based on ETV version 6.1



Debugging Issues Unique to Cray X1



- Vector registers
- Multi-stream processors (MSPs)
- Distributed Memory (DM) machine
- Remote Translation Table (RTT) shared memory core file sets
- `aprun` launcher



Current Capabilities



- Debugs command mode executables
- Debugs MSP mode, non-DM executables
- Performs `-G0/-g` live and core file debugging
- Supports C-Language, C++, Fortran, Assembly Language
- Supports X1 registers, NV1 instruction set, Unicos/mp
- Supports `-Ostream0` with streamed libraries



Next Release 4Q03



- Compensate for relocation of DM processes
- Support RTT shared memory core file debugging
- Support `aprun` launching of DM jobs in concert with the debugger
- Access to information from SSPs 1-3
- Upgrade to latest Etnus TotalView source
- Improve debugger group's regression test suite



Still to Come



- Direct support for Distributed Memory models
 - MPI, CoArrays, UPC
- Direct support for Shared Memory models
 - PThreads, OpenMP
- Support `-G1 / -gp`
- Support for Vector registers
- Support for Watchpoints
- Provide DebugView like capability



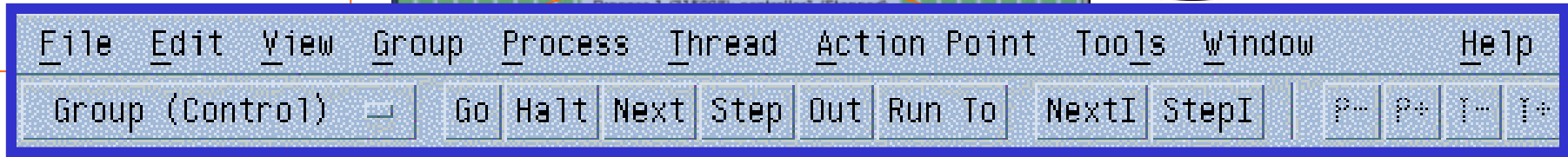
Demo Overview



- A “live” demonstration via slides
 - GUI demo
 - CLI demo



Process/Thread Window



Source code

Current location

```

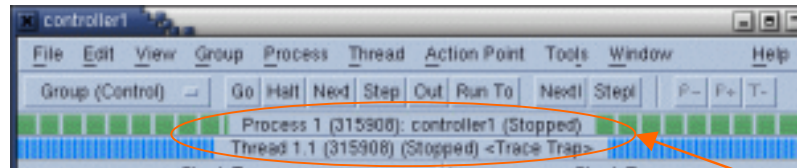
322 |-----
323 |      program main
324 |      this module uses Controller related classes, as described in
325 |      Scientific and Engineering C++, by John J. Barton and Lee K. Nackaa
326 |      [Addison-Wesley, Reading, MA, 1994], pp. 226-249.
327 |      use Voltmeter_class
328 |      use VoltageSupply_class
329 |      use IVTester_class
330 |      type (GPiBcontroller_Stub) :: gpib
331 |      type (AcmeI30), target :: volt_supply, supply1
332 |      type (VoltyMetrics), target :: meter
333 |      type (VoltOn59), target :: supply2
334 |      type (IVTester) :: iv
335 |      real :: result, v_step = 1.0
336 |      | first example, p. 228
337 |      call new(volt_supply, gpib, 12)    | Supply at GPiB address 12.
338 |      call set(volt_supply, 3.6)
339 |      | second example, p. 232
340 |      call new(meter, gpib, 14)
341 |      call new(supply1, gpib, 12)
342 |      result = orig checkCalibration(supply1, meter, 1.0)
343 |      Print *, 'AcmeI30 relative error at 1 volt is: ', result
344 |      | third example, p. 235
345 |      call new(voltOn59, gpib, 13)
    
```

Arguments, locals, and registers

Action points



Process Status



Process 1 (315908): controller1 (Stopped)
 Thread 1.1 (315908) (Stopped) <Trace Trap>

```

322 |-----|
323 |   program main
324 |   | this module uses Controller related classes, as described in
325 |   | Scientific and Engineering C++, by John J. Barton and Lee R. Hackaa
326 |   | [Addison-Wesley, Reading, MA, 1994], pp. 226-249.
327 |   use Voltmeter_class
328 |   use VoltageSupply_class
329 |   use IVTester_class
330 |   type (GPBController_Stub) :: gpib
331 |   type (AcmeI30), target :: volt_supply, supply1
332 |   type (VoltMetrics), target :: meter
333 |   type (VoltOn59), target :: supply2
334 |   type (IVTester) :: iv
335 |   real :: result, v_step = 1.0
336 |   | first example, p. 228
337 |   call new(volt_supply, gpib, 12)    | Supply at GPB address 12.
338 |   call set(volt_supply, 3.6)
339 |   | second example, p. 232
340 |   call new(meter, gpib, 14)
341 |   call new(supply1, gpib, 12)
342 |   result = orig checkCalibration(supply1, meter, 1.0)
343 |   Print *, 'AcmeI30 relative error at 1 volt is: ', result
344 |   | third example, p. 235
345 |   call new(meter2, mib, 13)
    
```

Thread (1) | Action Points

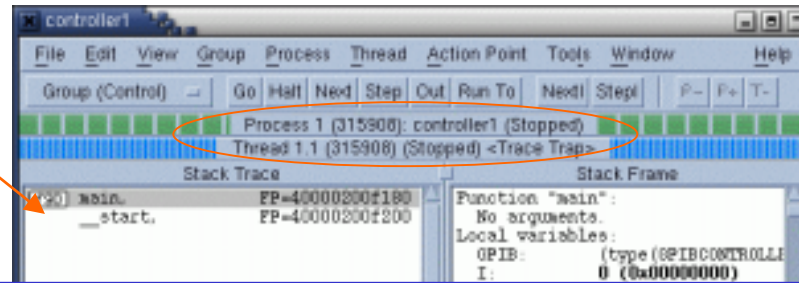
1.1	T	in main	STOP	1 controller1.f#337 main
-----	---	---------	------	--------------------------



Stack Trace Pane

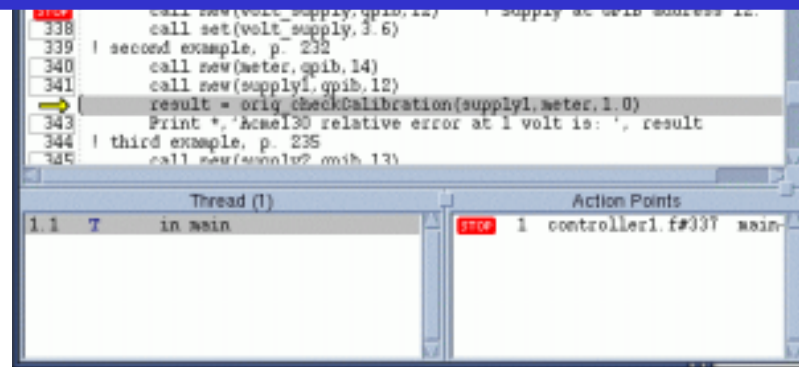


Stack trace



Stack Trace

f90	main,	FP=40000200f180
	__start,	FP=40000200f200





Source Pane



Source code

```

controller1
File Edit View Group Process Thread Action Point Tools Window Help
Group (Control) Go Halt Next Step Out Run To Next Step P- P+ T-

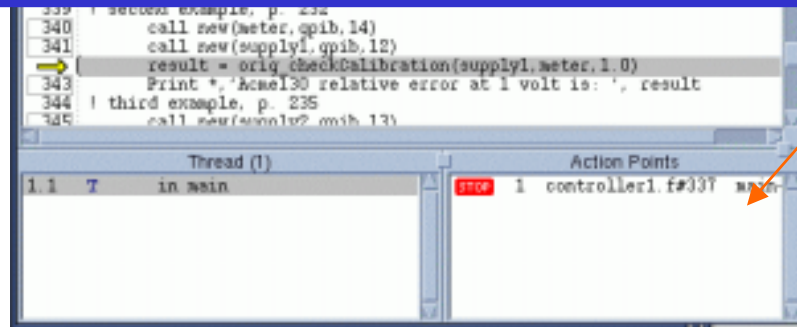
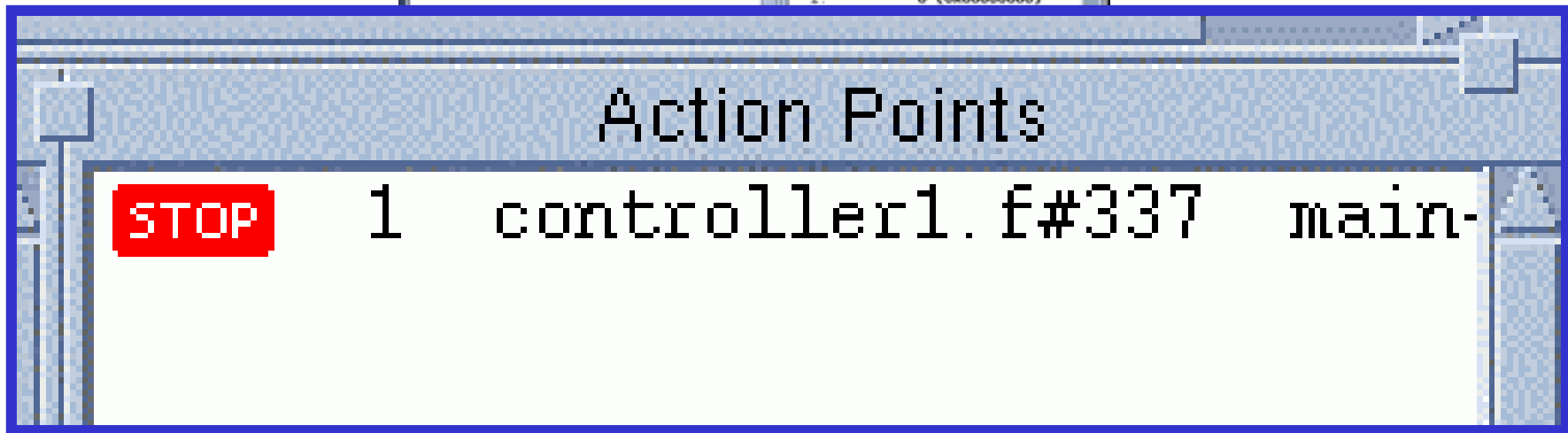
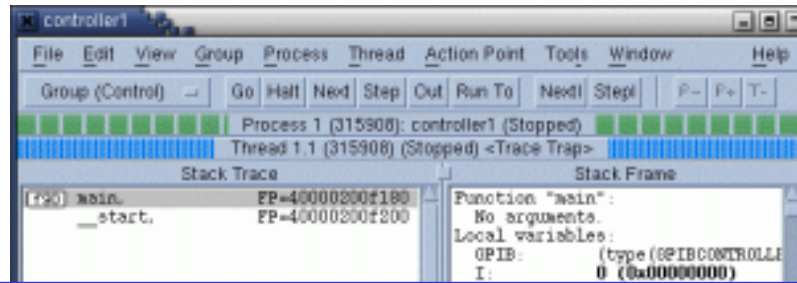
332 type (voltmetrics), target
333 type (VoltOn59), target ::
334 type (IVTester) :: iv
335 real :: result, v_step = 1
336 ! first example, p. 228
337 call new(volt_supply, gpib,
338 call set(volt_supply, 3.6)
339 ! second example, p. 232
340 call new(meter, gpib, 14)
341 call new(supply1, gpib, 12)
342 result = orig_checkCalibra
343 Print *, 'Acme130 relative
344 ! third example, p. 235
345 call new(sensor2, mib, 13)

Thread (1)
1.1 T in main

Action Points
STOP 1 controller1.f#337 main
    
```



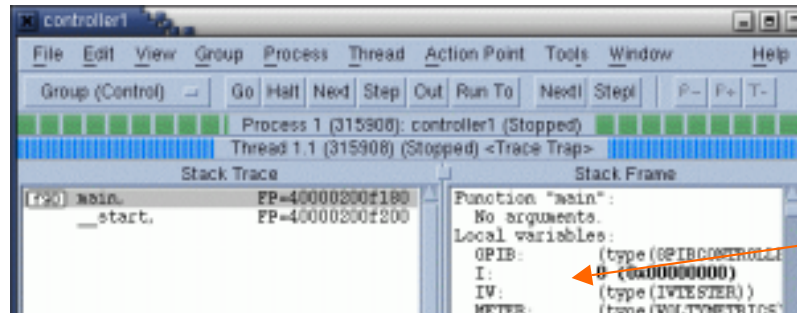
Action Points Pane



Action points



Stack Frame Pane



Arguments,
locals, and
registers

```
opped) <Trace Trap>
Stack Frame
Function "main":
  No arguments.
Local variables:
  GPIB:      (type (GPIBCONTROLLI
  I:         0 (0x00000000)
  IV:        (type (IVTESTER) )
  METER:     (type (VOLTYMETRICS)
  RESULT:   0
```




Stepped into New Function



The screenshot shows the Etnus TotalView debugger interface. The 'Out' button in the 'Action' menu is circled in orange. The 'Stack Trace' window shows the current function 'ORIG_CHECKCALIBRATION' and its callers 'main' and '__start'. The 'Registers for the frame' window shows register values.

New function

Out of function button

Stack Trace

f90	ORIG_CHECKCALIBRATION, FP=40000200	
f90	main,	FP=40000200f0c0
	__start,	FP=40000200f140

ers



X1 Registers



The screenshot shows the Etnus TotalView debugger interface. The 'Stack Trace' pane on the left lists the call stack, with the current function 'ORIG_CHECKCALIBRATION' selected. The 'Stack Frame' pane on the right shows the function name and local variables. The 'Registers for the frame' section displays the values of registers a0, a1, and a2. The 'Out' button in the 'Action Point' menu is circled in red. A callout box labeled 'New function' points to the function name in the stack trace. Another callout box labeled 'Out of function button' points to the 'Out' button.

New function

Out of function button

Registers for the frame:

```

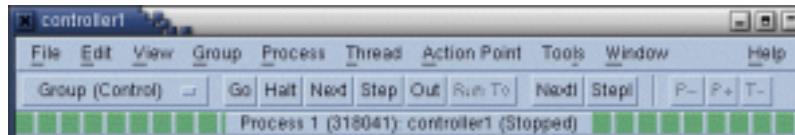
a0: 0x00000000 (0)
a1: 0x40000200ee98 (70368777779317
a2: 0x4000000000000000 /70360777770310
    
```

Function ORIG_CHECKCALIBRATION in controller1.f

rs

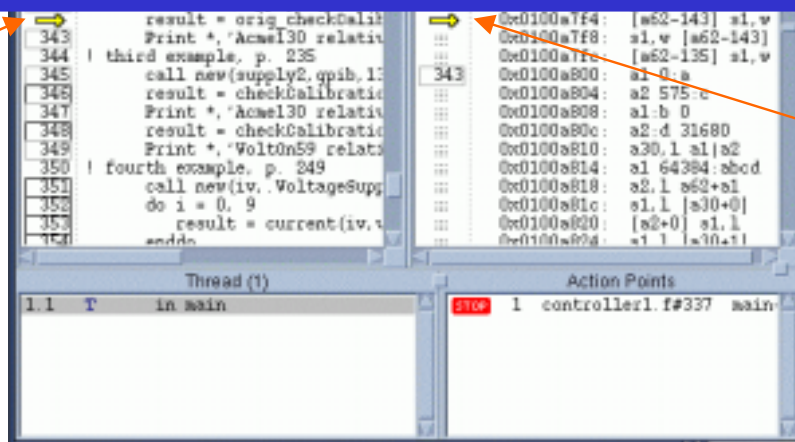


Stepping out to Assembly View



```

333      ! second example, p. 232
340      call new(meter, gpib, 14)
341      call new(supply1, gpib, 12)
342      result = orig_checkCalib
343      Print *, 'Acme130 relativ
344      ! third example, p. 235
345      call new(supply2, gpib, 12)
    
```

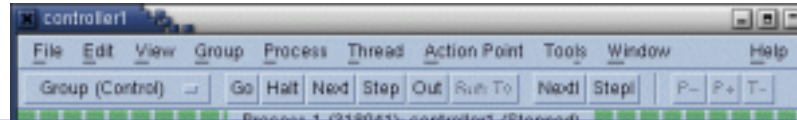


Still in call?

Just after jsr, but not yet at line 343



Assembly



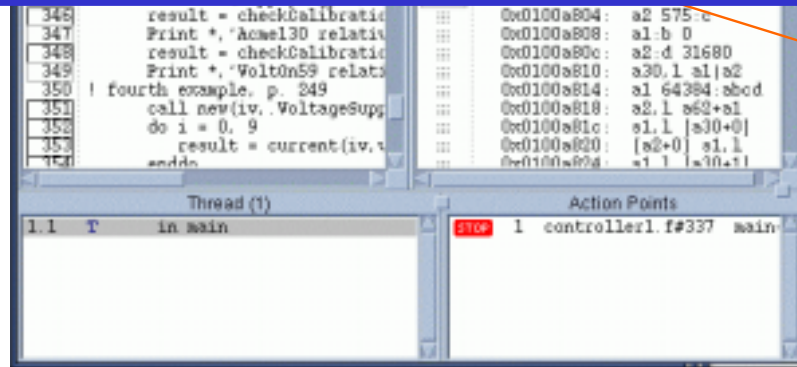
```

...
... 0x0100a7ec: a59:b 63212
...
... 0x0100a7f0: j, a60 a61, sr
...
→ 0x0100a7f4: [a62-143] s1, w
...
... 0x0100a7f8: s1, w [a62-143]
...
... 0x0100a7fc: [a62-135] s1, w
...
343 0x0100a800: a1 0:a
...
... 0x0100a804: a2 575:c
...

```

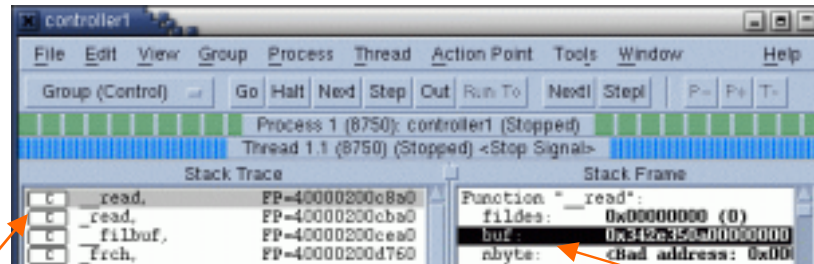
Still in call?

Just after jsr, but not yet at line 343





Halting the Run



Mixed
lan

Diving on
data

Diving
on

Diving on
tion points

Stack Trace

c	_read,	FP=40000200c8a0
c	_read,	FP=40000200cba0
c	_filbuf,	FP=40000200cea0
c	_frch,	FP=40000200d760
c	_sr_endrec,	FP=40000200db60
c	_FRF,	FP=40000200e100
f90	RECEIVE,	FP=40000200e520
f90	READ_VM,	FP=40000200e7e0
f90	CHECKCALIBRATION,	FP=40000200eaa0
f90	main,	FP=40000200f180
	__start,	FP=40000200f200



Diving on Data



Stack Trace:

Function	FP	Stack Frame
c __read	FP=40000200c8a0	Function "__read":
c __read	FP=40000200cba0	files: 0x00000000 (0)
c filbuf	FP=40000200cea0	buf: 0x342e350a00000000
c frch	FP=40000200d760	nbyte: cBad address: 0x00
c ar_endrc	FP=40000200db60	Local variables:

read.c#__read#buf

```

(at 0x024e4de0) Type: void *
Value: 0x342e350a00000000 -> <Bad address: 0x342e350a00000000>
    
```

Result of diving on *buf*

```

read.c#__read#buf - 1.1
(at 0x024e4de0) Type: void *
Value: 0x342e350a00000000 -> <Bad address:
    
```



Diving on Data



The screenshot shows the Etnus TotalView interface. The top window displays a stack trace for process 1 (8750) controller1, showing a call to `read`. The bottom window shows the variable `buf` at address `0x342e350a00000000` with a value of `0x342e350a00000000`. A third window shows the variable type being edited from `void *` to `void [*]`.

Result of diving on *buf*

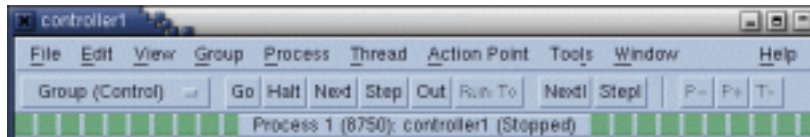
Editing type of *buf*

```

read.c#__read#buf - 1.1
(at 0x024e4de0) Type: void [*
Value: 0x342e350a00000000 -> <Bad address:
    
```



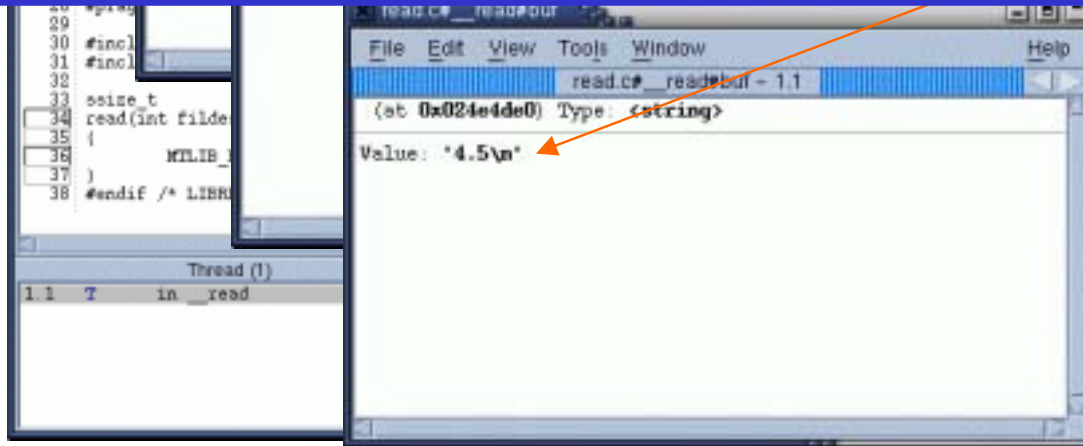
Diving on Data



```

read.c#__read#buf - 1.1
(at 0x024e4de0) Type: <string>

Value: "4.5\n"
    
```





More Data

```

fch.cr_fch#stat
File Edit View Tools Window Help
fch.cr_fch#stat - 1.1
(at 0x40000200cf40) Type: struct ffsw
Field Type Value
sw_flag unsigned int:1 0x00 (0)
sw_error unsigned int:31 0x00000000 (0)
(padding) <char>[4] (Array)
sw_count long 0x0000000000000000 (0)
sw_stat unsigned int:16 0x0000 (0)
(padding) <char>[2] (Array)
sw_user int:32 0x00000001 (1)
sw_iptr void * 0x02484020 -> 0x0000000000000000 (0)
sw_sptr void * 0x00000000
sw_rsvl int 0xfffffaec (-1300)
(padding) <char>[4] (Array)
aiochp struct aiochp (Struct)
aio_fildes int 0x00000000 (0)
(padding) <char>[4] (Array)
aio_buf void * 0x013a9668 (& p_SVPT_cwp0 and swap)
aio_nbytes unsigned long int 0x000040000200d260 (70368777785952)
aio_offset long 0xffffeba800000000 (-22368189677568)
aio_reqprio int 0x00000000 (0)
(padding) <char>[4] (Array)
aio_sigevent struct sigevent (Struct)
sigev_notify int 0x00000000 (0)
(padding) <char>[4] (Array)
sigev_notifyinfo union notifyinfo (Union)
sigev_value union sigval (Union)
sigev_notify_function void (*) (void) 0x013aee5c : pthread_mutex_unlock+0x
sigev_notify_attributes struct @?e_type_20 + 0x013a92a8 (& __p__SVPT_cwp_and_swap)
sigev_reserved unsigned long int[11] (Array)
[0] 0x000040000200d2a0 (70368777786016)
[1] 0x0000000000000000 (0)
[2] 0x000000000000002b (43)
[3] 0x00000000024e6e00 (38694400)
[4] 0x00000000010743f4 (17253364)
[5] 0x000000000000002b (43)
[6] 0x0000000000000000 (0)
[7] 0x0000000000000000 (0)
[8] 0x000000000112a7d0 (17999824)
[9] 0xfffffa9000000000 (1844673809511507)
[10] 0x0000000000000001 (1)
sigev_pad unsigned long int[6] (Array)
[0] 0x00000000010f4d50 (17780064)
[1] 0x0000000001055af4 (17128180)
[2] 0x0000000000000000 (0)
[3] 0x0000000000000000 (0)

```

The 45th CUG Conference
CRAY USER GROUP
 FLIGHT TO INSIGHT
 May 13-16, 2003 - Columbus, Ohio



CLI



- Stand-alone or with GUI
- Command line ASCII input
- Tcl based interface
 - ‘d’ prefix on TotalView commands
 - Watch out for special characters!!
 - \$, [,], {, }, #, ;, “, space
- Programmable



CLI Startup



```
sn702> totalviewcli controller1
```

```
d1.<> dbreak 337
```

```
1
```

```
d1.<> dgo
```

```
Created process 1 (71894), named "controller1"
```

```
Thread 1.1 has appeared
```

```
Thread 1.1 hit breakpoint 1 at line 337 in  
"main"
```

```
d1.<> dcont
```



Application Input



(Acme130 now at address 12)

(GPIB instrument # 12 sends value 3.5999999)

(VoltyMetrics now at address 14)

(Acme130 now at address 12)

(GPIB instrument # 12 sends value 1.)

(Please enter number for GPIB instrument # 14)

4.5

Acme130 relative error at 1 volt is: 3.5999999

(GPIB instrument # 13 sends value 1.)

(Please enter number for GPIB instrument # 14)

Thread 1.1 received a signal (Interrupt)



dwwhere



d1.<> dwwhere

```
>0 __read      PC=0x0110e810, FP=0x40000200c7e0 [read.c#23]
 1 _read       PC=0x0110ee0c, FP=0x40000200cae0 [read.c#36]
 2 __filbuf    PC=0x0110c614, FP=0x40000200cde0 [_filbuf.c#65]
 3 _frch       PC=0x010e51b4, FP=0x40000200d6a0 [frch.c#224]
 4 _sr_endrec  PC=0x0108ece8, FP=0x40000200daa0 [rf.c#876]
 5 _FRF        PC=0x01041d00, FP=0x40000200e040 [rf90.c#333]
 6 RECEIVE     PC=0x01003c3c, FP=0x40000200e460 [cont.f#50]
 7 READ_VM    PC=0x01005af0, FP=0x40000200e720 [cont.f#142]
 8 CHECKCAL   PC=0x01009928, FP=0x40000200e9e0 [cont.f#363]
 9 main       PC=0x0100abac, FP=0x40000200f0c0 [cont.f#348]
10 __start    PC=0x010019d4, FP=0x40000200f140 [cont]
```



dlist



```
d1.<> dlist
16  #include <sys.s>
17  #include "sys/sv2/syscall.h"
18  #include <unistd.h>
19
20  ssize_t
21  __read(int fildes, void *buf, size_t nbyte)
22  {
23 >    USUAL_SYSCALL(ssize_t, SYS_read, ARG_LIST3(fildes,
24                                                    buf, nbyte));
25  }
26
27  #ifndef LIBRESTART_LIBC
28  /* Wrapper for pthread callback */
29  #pragma weak read = _read
```



dprint



```
d1.<> dprint *buf
```

```
*buf = <Bad address: 0x342e350a00000000>
```

```
d1.<> dwhat buf
```

```
In thread 1.1:
```

```
Name: buf; Type: void *; Size: 8 bytes;Addr: 0x024e4de0
```

```
Scope: ##controller1#read.c#__read(Scope class: Any)
```

```
Address class: reference_param(Reference parameter)
```

```
d1.<> dprint *(<string>*)&buf
```

```
*(<string>*)&buf = "4.5\n"
```



TCL “for”



```
d1.<> for {set i 57} {$i < 64} {incr i} {dprint \a$i}
$a57 = 0x00002ffffffffffe0 (52776558133216)
$a58 = 0x0000300000000000 (52776558133248)
$a59 = 0xfffffb8c00000000 (-4896262717440)
$a60 = 0x000000000110ee0c (17886732)
$a61 = 0x000000000110e7c8 (17885128)
$a62 = 0x000040000200c7e0 (70368777783264)
$a63 = 0x000040000200c500 (70368777782528)
```



Summary



- Etnus TotalView is a strong base for our debugging platform.
- There is more to be done.
- Management has increased our resources.