Introducing the Cray TotalView Debugger

IN–2502 2.0

Cray Research, Inc.

Portions of this product may still be in development. The existence of those portions still in development is not a commitment of actual release or support by Cray Research, Inc. Cray Research, Inc. assumes no liability for any damages resulting from attempts to use any functionality or documentation not officially released and supported. If it is released, the final form and the time of official release and start of support is at the discretion of Cray Research, Inc.

The UNICOS operating system is derived from UNIX® System V. The UNICOS operating system is also based in part on the Fourth Berkeley Software Distribution (BSD) under license from The Regents of the University of California.

Cray Research Software Documentation Map

The information on the following pages highlights the major body of documentation available for Cray Research customers. The manuals are organized into categories by product designation:

Product	Description
Programming environments	Cray Research programming environments provide an integrated development environment by combining compilers with development tools and libraries.
UNICOS operating system	The UNICOS operating system provides basic system control and management functions. It provides services, including basic batch and interactive system access for users, operating system interfaces for programmers, networking facilities, and sophisticated resource management for administrators. UNICOS MAX, in conjunction with UNICOS on the host system, provides operating system functionality on CRAY T3D systems.
Networking products	Cray Research networking products provide media connections, network programming access, and batch and interactive access for users across a heterogeneous network of computers.
Operating systems supplementary products	The operating systems supplementary products include a variety of products that are used in conjunction with Cray Research operating systems but are not included on the same media.

To use the map, find the product designation closest to your specific needs or role as a Cray Research system user. Documents listed under other categories also may be of interest to you; documents are listed only once, underneath the product to which they most directly apply. The date on the bottom of the page tells you when the information was last revised.

For more information

In addition to the map, you can use the following publications to find documentation specific to your needs:

- *Software Documentation Ready Reference,* publication SQ–2122, serves as a general index to the Cray Research documentation set. The booklet lists documents and man pages according to topic.
- *Software Overview for Users*, publication SG–2052, provides an overview of Cray Research software, including operating systems, programming environments, libraries, tools, applications, and so on. It directs you to documentation that contains user-level information.
- *User Publications Catalog*, publication CP–0099, briefly describes all Cray Research manuals available to you, including some not shown on the map. Cray Research customers who subscribe to the Cray Inform (CRInform) program can access this information on CRInform.

Ordering

To obtain Cray Research publications, order them by publication number from the Distribution Center:

Cray Research, Inc.		
Distribution Center	Order desk	+1–612–683–5907
2360 Pilot Knob Road	Fax number	+1–612–452–0141
Mendota Heights, MN 55120	Email	`orderdsk@cray.com`
USA		

Programming Environments

Ada

Cray Ada Environment, Volume 1: Reference Manual (SR–3014)

Cray Ada Environment, Volume 2: Programming Guide (SR–3082)

Assemblers

Cray Assembler for MPP (CAM) Reference Manual (SR–2510)†

Cray Assembly Language (CAL) for Cray PVP Systems Ready Reference (SQ–3110)

Cray Assembly Language (CAL) for Cray PVP Systems Reference Manual (SR–3108)†

Fortran Compilers

CF90 Ready Reference (SQ–3900)

CF90 Commands and Directives Reference Manual (SR–3901)†

CF90 Fortran Language Reference Manual, Vol. 1 (SR–3902)†

CF90 Fortran Language Reference Manual, Vol. 2 (SR–3903)†

CF90 Fortran Language Reference Manual, Vol. 3 (SR–3905)†

CF90 SPARC Ready Reference (SQ–3904)

Introducing the CF90 SPARC Programming Environment (IN–2155)†

Cray MPP Fortran Reference Manual (SR–2504)†

CF77 Fortran Language Reference Manual (SR–3772)

Cray Standard C Compiler

Cray Standard C Reference Manual (SR–2074)†

Cray Standard C Ready Reference (SQ–2076)

Cray Standard C Reference Manual for MPP (SR–2506)†

C++ Compiler

Cray C/C++ Reference Manual (SR–2179)†

Cray C/C++ Ready Reference (SQ–2180)

CrayLibs

Application Programmer's I/O Guide (SG–2168)†

Application Programmer's Library Reference Manual (SR–2165)*

Introducing CrayLibs (IN–2167)†

Math Library Reference Manual (SR–2138)*

Parallel Virtual Machine (PVM) Reference Card (SQ–2512)

PVM and HeNCE Programmer's Manual (SR–2501)†

Scientific Libraries Reference Manual (SR–2081)*

CrayTools

Compiler Information File (CIF) Reference Manual (SR–2401)†

Cray TotalView Debugger Reference Card (SQ–2519)

Guide to Parallel Vector Applications (SG–2182)†

Introducing the Cray TotalView Debugger (IN–2502)†

Introducing the MPP Apprentice Tool (IN–2511)†

Introducing the Program Browser (IN–2140)†

UNICOS CDBX Symbolic Debugger Reference Card (SQ–2110)

UNICOS CDBX Symbolic Debugger Reference Manual (SR–2091)†

UNICOS CDBX Debugger User's Guide (SG–2094)†

General

Cray Research MPP Software Guide (SG–2508)†

CRAY T3D Emulator User's Guide (SG–2500)†

Installing Programming Environment Products (SG–5191)

Migrating to the CRAY T90 Series IEEE Floating Point (SN–2194)†

Optimizing Code on Cray PVP Systems (SG–2192)†

Programming Environment Installation Guide (SG–5217)

Message Passing Toolkit

MPT: PVM Programmer's Manual (SR–2196)†

MPT: MPI Programmer's Manual (SR–2197)†

† Available with CrayDoc
* Man pages available with the `man` command

UNICOS Operating System

General UNICOS

Software Documentation Ready Reference (SQ–2122)

Software Overview for Users (SG–2052)†

User's Guide to Online Information (SG–2143)†

UNICOS `ed` Reference Card (SQ–2055)

UNICOS Environment Variables Ready Reference (SQ–2117)

UNICOS File Formats and Special Files Reference Manual (SR–2014)*

UNICOS Standard Shell Reference Card (SQ–2115)

UNICOS User Commands Ready Reference (SQ–2056)*

UNICOS User Commands Reference Manual (SR–2011)*

UNICOS `vi` Reference Card (SQ–2054)

UNICOS Visual Interfaces User's Guide (SG–3094)†

Networking Feature Use and Administration

Front-end Protocol Internal Reference Manual (SM–0042)

FTA User and Administrator Manual (SG–2144)†

Kerberos Administrator's Guide (SG–2306)†

Kerberos User's Guide (SG–2409)†

Network Queuing System (NQS) User's Guide (SG–2105)†

ONC+ Technology for the UNICOS Operating System (SG–2169)†

Remote Procedure Call (RPC) Reference Manual (SR–2089)†

TCP/IP Network User's Guide (SG–2009)†

UNICOS Networking Facilities Administrator's Guide (SG–2304)†

UNICOS NQS and NQE Administrator's Guide (SG–2305)†

UNICOS Station Call Processor (USCP) Administrator's Guide (SG–2308)†

UNICOS MAX

Cray MPP Loader User's Guide (SG–2514)†

Cray MPP Simulator User's Guide (SG–2503)†

CRAY T3D Administrator's Guide (SG–2507)†*

UNICOS Basic Administration and Operation

CRAY EL Series IOS Commands Ready Reference (SQ–2162)*

CRAY EL Series IOS Commands Reference Manual (SR–2408)*

CRAY EL Series IOS Messages (SQ–2402)

CRAY IOS-V Commands Reference Manual (SR–2170)*

CRAY IOS-V Messages (SQ–2172)†

General UNICOS System Administration (SG–2301)†

`nmake` Reference Card (SQ–2146)

OLNET Online Diagnostic Network Communications Program Maintenance Manual for UNICOS (SD–1021)†

Online Maintenance Tools Guide for Cray PVP Systems (SD–1012)†

UNICOS Basic Administration Guide for CRAY J90 Series and CRAY EL Series (SG–2416)†

UNICOS Configuration Administrator's Guide (SG–2303)†

UNICOS Installation Guide (SG–2112)

UNICOS Installation Guide for the CRAY EL Series (SG–5201)

UNICOS Installation Guide for the CRAY J90 Series (SG–5271)

UNICOS Installation Menu System Reference Card (SQ–2411)

UNICOS Installation/Configuration Menu System User's Guide (SG–2412)†

UNICOS Resource Administration (SG–2302)†

UNICOS Shared File System (SFS) Administrator's Guide (SG–2114)†

UNICOS Feature Use and Administration

UNICOS Administrator Commands Reference Manual (SR–2022)*

UNICOS Administrator Commands Ready Reference (SQ–2413)*

UNICOS Message System Programmer's Guide (SG–2121)†

UNICOS Multilevel Security (MLS) Feature User's Guide (SG–2111)†

UNICOS System Security Overview for Administrators (SG–2141)†

UNICOS Tape Subsystem Administrator's Guide (SG–2307)†

UNICOS Tape Subsystem User's Guide (SG–2051)†

UNICOS under UNICOS Administrator's Guide (SG–2156)†

† Available with CrayDoc
* Man pages available with the `man` command

UNICOS Operating System (continued)

Programming Support

Segment Loader (SEGLDR) and `ld` Reference Manual (SR–0066)†

SEGLDR Reference Card (SQ–0303)

UNICOS System Libraries Ready Reference (SQ–2147)*

UNICOS System Libraries Reference Manual (SR–2080)*

Programming Support (continued)

UNICOS Macros and Opdefs Reference Manual (SR–2403)*

UNICOS System Calls Reference Manual (SR–2012)*

UNICOS X Window System Reference Manual (SR–2101)†

X Window System Resources Ready Reference (SQ–2123)

Networking Products

ATM

Asynchronous Transfer Mode (ATM) Administrator's Guide (SG–2193)†

fy Driver

fy Driver Administrator's Guide (SG–2132)

HIPPI

HIPPI Channel Administrator's Guide (SG–2159)

Network Queuing Environment (NQE)

NQE User's Guide (SG–2148)†

NQE Ready Reference (SG–2149)

NQE Administration (SG–2150)†

Introducing NQE (IN–2153)†

RQS MVS

RQS User's Guide for IBM MVS Systems (SG–2405)

RQS Administrator's Guide for IBM MVS Systems (SG–2406)

Fiber Distributed Data Interface (FDDI)

Fiber Distributed Data Interface (FDDI) Administrator's Guide (SG–3091)

RQS VMS

RQS/NQC User's Guide for DEC OpenVMS/VAX and AXP Systems (SV–3151)

RQS/NQC Administrator's Guide for DEC OpenVMS/VAX and AXP Systems (SV–3152)

SUPERLINK VM

SUPERLINK VM User's Guide: RQS Component (SI–0170)

SUPERLINK VM Administrator's Guide: RQS Component (SI–0171)

SUPERLINK VMS

SUPERLINK Administrator's Guide for DEC VAX/VMS Systems (SV–3154)

SUPERLINK User's Guide for DEC VAX/VMS Systems (SV–3153)

Operating Systems Supplementary Products

CCMT

Concurrent Maintenance Tools (CCMT) Guide (SD–1027)†

Cray/REELlibrarian (CRL)

Cray/REELlibrarian (CRL) Administrator's Guide (SG–2127)

Cray/REELlibrarian (CRL) User's Guide (SG–2126)

SMARTE

System Maintenance and Remote Testing Environment (SMARTE) Guide (SD–1017)†

Data Migration Facility (DMF)

Cray Data Migration Facility (DMF) Administrator's Guide (SG–2135)†*

Cray Data Migration (DMF) MSP Writer's Guide (SN–2098)†

OWS

Support System Reference Manual (SR–3077)*

Support System Operator's Guide (SG–3078)†

Support System Administrator's Guide (SG–3079)†

Support System Ready Reference (SQ–3080)

† Available with CrayDoc
* Man pages available with the `man` command

New Features

Introducing the Cray TotalView Debugger IN–2502 2.0

This manual contains all new window samples to illustrate the modified user interface, descriptions of new information displayed in the `process window`, an explanation of mouse button usage, expanded information on how to start a process, and a description of the `Release Information` option of the Help menu. Select the `Release Information` option to see a complete list of all feature changes for this release.

Record of Revision

The date of printing or software version number is indicated in the footer.

Version	***Description***
1.1	May 1994. Original printing. This manual introduces the Cray TotalView Debugger. Information from the *Cray TotalView Debugger Reference Manual*, publication SR–2502, now resides in online form, and can be accessed from the Cray TotalView debugger.
1.2	October 1994. Reprinted to support the Programming Environment MPP 1.2 release.
2.0	November 1995. Reprinted to support the Programming Environment 2.0 release.

Preface

This publication introduces the Cray TotalView debugger, an advanced, symbolic, source-level debugger designed for debugging the multiple processes of parallel Fortran, C, or C++ programs. The debugger is a part of the CrayTools package. CrayTools is a set of application development tools for debugging, performance analysis, and source code analysis. These tools help provide a powerful, user-friendly environment for the development and enhancement of new and existing code.

Related publications

The following documents contain additional information that may be helpful:

- *Cray MPP Fortran Reference Manual*, publication SR–2504
- *CF90 Fortran Language Reference Manual*, *Volume 1*, publication SR–3902
- *CF90 Fortran Language Reference Manual*, *Volume 2*, publication SR–3903
- *CF90 Fortran Language Reference Manual*, *Volume 3*, publication SR–3905
- *CF90 Commands and Directives Reference Manual*, publication SR–3901
- *Compiler Information File (CIF) Reference Manual*, publication SR–2401
- *Cray C/C++ Reference Manual*, publication SR–2179
- *Cray Standard C Reference Manual for MPP*, publication SR–2506
- *Cray Standard C Reference Manual*, publication SR–2074
- *Cray TotalView Debugger Reference Card*, publication SQ–2519
- *Introducing the Program Browser*, publication IN–2140
- *Introducing the MPP Apprentice Tool*, publication IN–2511

Ordering publications

The *User Publications Catalog*, publication CP–0099, describes the availability and content of all Cray Research hardware and software documents that are available to customers. Cray Research customers who subscribe to the Cray Inform (CRInform) program can access this information on the CRInform system.

To order a document, either call the Distribution Center in Mendota Heights, Minnesota, at +1–612–683–5907 or send a facsimile of your request to fax number +1–612–452–0141. Cray Research employees may send electronic mail to `orderdsk` (UNIX system users).

Customers who subscribe to the CRInform program can order software release packages electronically by using the `Order Cray Software` option.

Customers outside of the United States and Canada should contact their local service organization for ordering and documentation information.

If you are a CraySoft customer, you can order additional copies of this product, increase the number of licensed users, or order CraySoft publications, by contacting the CraySoft organization in one of the following ways:

- Call one of these telephone numbers:

  1–800–289–2729

  +1–612–683–3030

- Send electronic mail from any system connected to Internet to the following address:

  `craysoft@cray.com`

- Send a facsimile (fax) to the following telephone number:

  +1–612–683–5508

Conventions

The following conventions are used throughout this document:

Convention	Meaning
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
`manpage`(*x*)	Man page section identifiers appear in parentheses after man page names.

The following machine naming conventions may be used throughout this document:

Term	Definition
Cray PVP systems	All configurations of Cray parallel vector processing (PVP) systems, including the following: CRAY C90 series (CRAY C916, CRAY C92A, CRAY C94, CRAY C94A, and CRAY C98 systems) CRAY C90D series (CRAY C92AD, CRAY C94D, CRAY C98D systems) CRAY EL series (CRAY Y-MP EL, CRAY EL92, CRAY EL94, and CRAY EL98 systems) CRAY J90 series (CRAY J916 and CRAY J932 systems) CRAY T90 series (CRAY T94, CRAY T916, and CRAY T932 systems) CRAY Y-MP E series (CRAY Y-MP 2E, CRAY Y-MP 4E, CRAY Y-MP 8E, and CRAY Y-MP 8I systems) CRAY Y-MP M90 series (CRAY Y-MP M92, CRAY Y-MP M94, and CRAY Y-MP M98 systems)
Cray MPP systems	All configurations of Cray massively parallel processing systems, including the CRAY T3D series (CRAY T3D MC, CRAY T3D MCA, and CRAY T3D SC systems)

Term	Definition
All Cray Research systems	All configurations of Cray PVP and Cray MPP systems that support this release
SPARC systems	All SPARC platforms that run the Solaris operating system version 2.3 or later

The default shell in the UNICOS operating system, referred to in Cray Research documentation as the standard shell, is a version of the Korn shell that conforms to the following standards:

- Institute of Electrical and Electronics Engineers (IEEE) Portable Operating System Interface (POSIX) Standard 1003.2–1992
- X/Open Portability Guide, Issue 4 (XPG4)

The UNICOS operating system also supports the optional use of the C shell.

Cray UNICOS Version 9.0 is an X/Open Base 95 branded product.

The POSIX standard uses *utilities* to refer to executable programs that Cray Research documentation usually refers to as *commands*. Both terms may appear in this document.

Online information

The following types of online information products are available to Cray Research customers:

- CrayDoc online documentation reader, which lets you see the text and graphics of a document online. The CrayDoc reader is available on workstations. To start the CrayDoc reader at your workstation, use the `cdoc`(1) command.
- Man pages, which describe a particular element of the UNICOS operating system or a compatible product. To see a detailed description of a particular command or routine, use the `man`(1) command.
- UNICOS message system, which provides explanations of error messages. To see an explanation of a message, use the `explain`(1) command.

- Cray Research online glossary, which explains the terms used in a document. To get a definition, use the define(1) command.
- xhelp help facility. This online help system is available within tools such as the Program Browser (xbrowse) and the MPP Apprentice tool.

For detailed information on these topics, see the *User's Guide to Online Information*, publication SG–2143.

Reader comments

If you have comments about the technical accuracy, content, or organization of this document, please tell us. You can contact us in any of the following ways:

- Send us electronic mail from a UNICOS or UNIX system, using the following UUCP address:

  uunet!cray!publications

- Send us electronic mail from any system connected to the Internet, using the following Internet address:

  publications@timbuk.cray.com

- Call the CraySoft Customer Support Call Center using either of the following numbers:

  1–800–272–9349 (toll free from the United States and Canada)

  +1–770–631–2244

- Contact your Cray Research representative and ask that a Software Problem Report (SPR) be filed. Use PUBLICATIONS for the group name, PUBS for the command, and NO-LICENSE for the release name.

- Call our Software Publications Group in Eagan, Minnesota, through the Customer Service Call Center, using either of the following numbers:

  1–800–950–2729 (toll free from the United States and Canada)

  +1–612–683–5600

- Send a facsimile of your comments to the attention of “Software Publications Group” in Eagan, Minnesota, at fax number +1–612– 683–5599.

- Use the postage-paid Reader’s Comment Form at the back of the printed document.

We value your comments and will respond to them promptly.

Contents

	Page
Preface	iii
Related publications	iii
Ordering publications	iv
Conventions	v
Online information	vi
Reader comments	vii
Overview [1]	1
Totalview window	1
Process_window	3
Data_object_window	6
Breakpoint_window	7
Breakpoint_summary_window	9
Array_browser_window	10
Getting Started [2]	11
Before you start	11
Using the mouse buttons	12
Diving into items	12
Changing variables	14
Compiling C or Fortran programs	15
Setting the display	16
Starting the debugger	16
Setting search paths	18
Setting command arguments	19
Setting breakpoints	20
Starting a process	23
Getting Help [3]	25
Xhelp facility	26
Glossary	29
Index	37
Figure 1. Sample totalview window (no file specified)	2
Figure 2. Sample totalview window (file specified)	2
Figure 3. Sample process_window	5
Figure 4. Sample data_object_window	6
Figure 5. Sample conditional breakpoint coded in C	8
Figure 6. Sample breakpoint_summary_window	9
Figure 7. Sample array_browser_window	10
Figure 8. Selecting an item upon which to dive	13
Figure 9. Dive output	14

		Page
Figure 10.	prompt_for_string_window	15
Figure 11.	Searching directories for source files	19
Figure 12.	Two arguments entered	20
Figure 13.	Setting breakpoints	21
Figure 14.	After running to breakpoint	22
Figure 15.	xhelp window with help file loaded	27

Overview [1]

The Cray TotalView debugger is an advanced, symbolic, source-level debugger designed for debugging Fortran, C, or C++ programs. The principal advantage of the Cray TotalView debugger over conventional UNIX debuggers is that it can debug multiple processes of parallel programs. The debugger provides symbolic debugging of both single-tasked or multitasked applications through an X Window System or a line-mode user interface. If you set the `DISPLAY` environment variable before running TotalView, TotalView will bring up the X Window System. You can invoke the line-mode version of the TotalView debugger by using the following command line:

```
totalview –L
```

Online help is available within the line-mode interface itself.

> **Note:** This document describes the X Window System interface to the debugger . To obtain information about the line-mode interface, see the *Cray TotalView Debugger Reference Card*, publication SQ–2519, or the `totalview`(1) man page.

Before you can use the Cray TotalView debugger, you must have the `PrgEnv` module loaded. To load this module, enter the following command:

```
module load modules PrgEnv
```

Totalview window

1.1

The `totalview` window appears when you start the debugger. If you do not specify a file name on the `totalview` command line, the window appears with no processes listed. By using the `Load New Program` option of the File menu of this window you can load a file into TotalView thus bringing up a process window (see subsection 1.2 for a description of the process window). Figure 1 shows the `totalview` window as it appears if you do not specify a file name on the command line.

Figure 1. Sample `totalview` window (no file specified)

If you specify a file name on the `totalview` command line, the `totalview` window you see upon startup will look similar to the window in Figure 2.

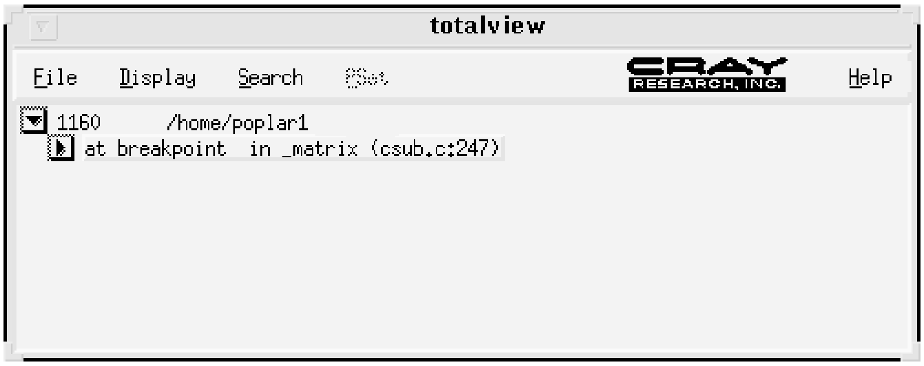

Figure 2. Sample `totalview` window (file specified)

When you specify the name of a parallel program on the command line the `totalview` window appears with only one entry visible. This entry represents your program and displays the program name, the process ID, and the current status. If you are running the TotalView debugger on a Massively Parallel Machine (MPP), the window also shows process set information. (See the glossary at the end of this document for a definition of process set.) If you use the left mouse button to click on one of the arrows, the entry will expand to show individual entries for each process element or task.

Process_window

1.2

The `process_window` contains four separate display areas (called *panes*) as well as a menu bar, status line, and control buttons. (A sample `process_window` is shown in Figure 3 on page 5.) The following list describes panes in the window.

Pane	Description
Call sequence	Upper left pane. This pane shows the calling sequence and the current routine of your process.
Current routine	Upper right pane. This pane shows the calling arguments and local variables for the currently displayed routine (selected from the call sequence). This pane may also display the current program counter (PC) or traceback information.
Source	Middle pane. This pane shows the source for the currently selected routine (selected from the call sequence).
Information	Bottom pane. This pane displays informational and status messages.

The menu bar, located below the window name, shows the menu names. To open a menu and see available options, click on the menu name by pressing the left mouse button.

The status line located below the menu bar, shows the following information:

`Status` <*state*> <*optional_error_code*> `PID` <*pid*> `<LWP, PE,` or `CPU>` <*pset*>

Item	Description
`Status`	Indicates the current status of the process being debugged. Status may be one of the following:

Status	Description
`At Breakpoint`	Process is stopped at a user -set breakpoint.
`Error` <*reason*>	Process encountered a serious error.
`Exited`	Process has finished and called `exit( )`.
`Never Created`	Process has not yet started.
`Running`	Process is running.
`Stopped` <*reason*>	Process is stopped but not at a breakpoint.

Item	Description
`PID` <*pid*>	Displays the process ID (PID) number for the current process.
`LWP`, `PE`, or `CPU`	Displays `LWP` to indicate the current Solaris lightweight process task number, `PE` to indicate a Massively Parallel Process (MPP) process element task number, or `CPU` to indicate a UNICOS multitasked program task number. Use this field to step through individual tasks by clicking on the arrows or by typing the task number in the input box following this field.
pset	Shows the current Pset (process set) if debugging a multiprocessed program.

The control buttons that appear in the middle of the `process_window` are explained in subsection 2.8, page 23.

In the sample `process_window` shown in Figure 3, the current process is stopped at a breakpoint. This is indicated by the status line at the top of the window and by the right arrow (—>) next to the line in the source pane where the breakpoint occurs.

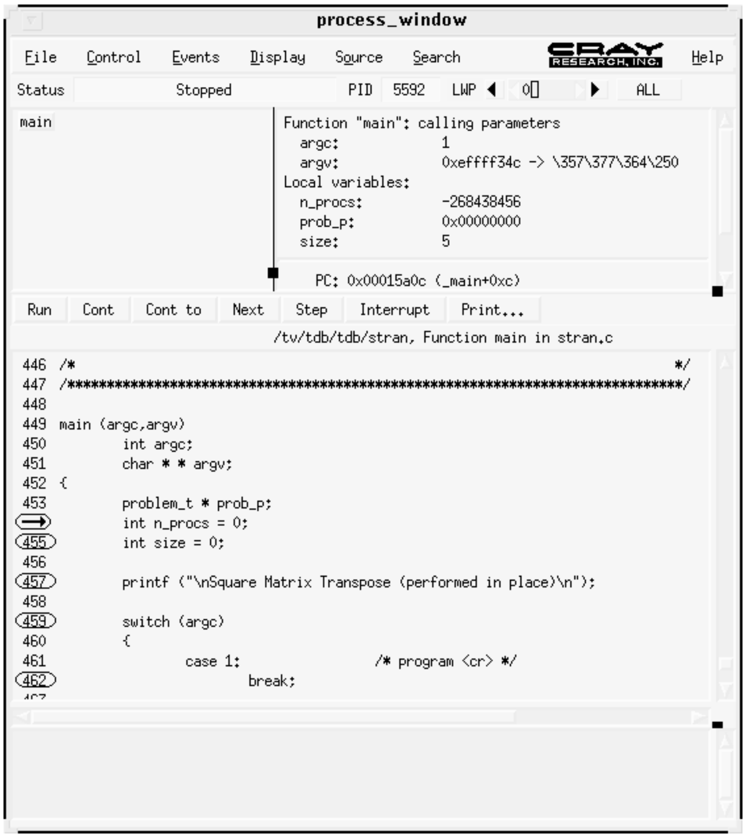

Figure 3. Sample `process_window`

Some of the functions you can perform from the `process_window` are as follows:

- Set or delete breakpoints
- Control the execution of a process
- Display information about variables

Data_object_window

1.3

The `data_object_window` displays information about variables, memory, and code segments. It shows the name of the variable, its address, type, and the data. From this display you can change the value, type, or address of the variable, or get more information about the variable. Figure 4 shows an example of a `data_object_window`.

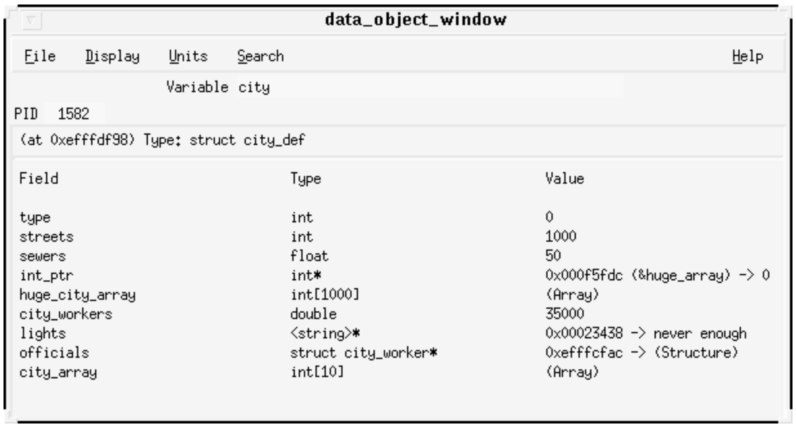

Figure 4. Sample `data_object_window`

Note: The input box and the slider bar appear only when debugging code running on MPP systems. The number shown in the input box reflects the current process element (PE) number; this number updates as you move the slider bar. The slider lets you access private data information across PEs. You can use either the arrows to move up and down the PE list, or you can use the slider to scan across PEs. You also can enter a specific PE number in the input box; valid values are current PE numbers.

`Breakpoint_window`

1.4

The `breakpoint_window` controls breakpoints. From this window you can set, enable, disable, or delete breakpoints. You can also determine the behavior of a breakpoint. For example, with multitasking applications, you can specify if a breakpoint should be shared among all of the related processes or if it should be active only for the current process. You can also control whether all of the processes should be stopped when the breakpoint is hit or if only the current process should be stopped.

To bring up the `breakpoint_window`, position the cursor on one of the oval icons that appear in the line number column at the far left of the source pane of the `process_window` and press the right mouse button.

Figure 5 is an example of the `breakpoint_window`. In this example, a conditional breakpoint is written in C. This breakpoint is shared among all processes; all of the processes will be stopped when any occurrence of the breakpoint is hit.

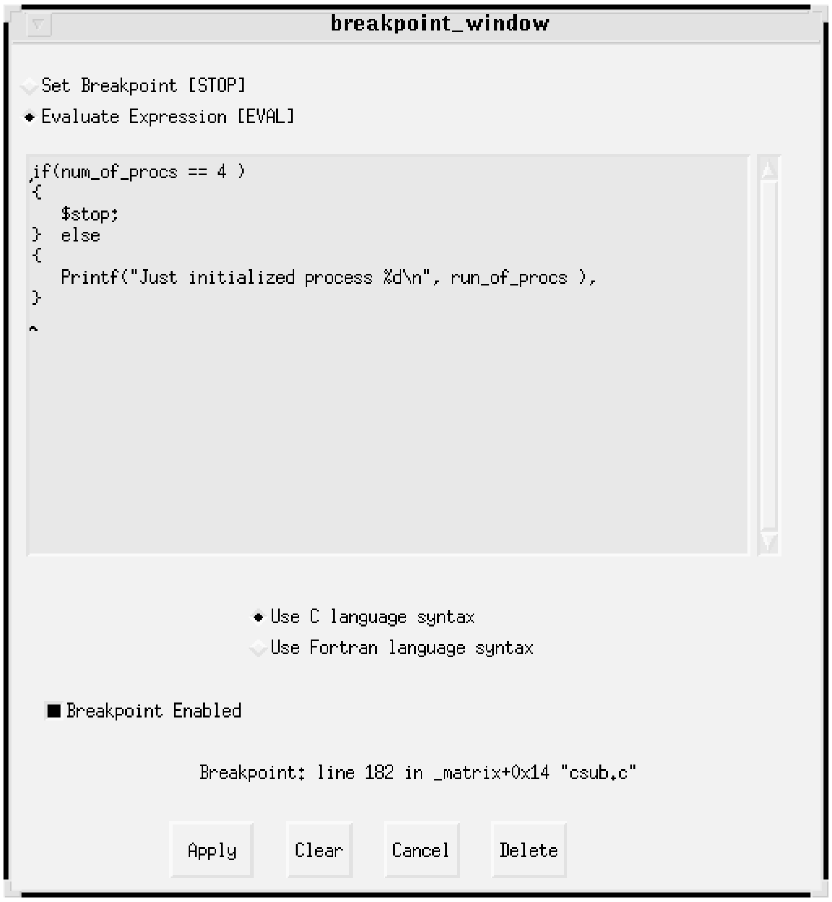

Figure 5. Sample conditional breakpoint coded in C

When setting a breakpoint, you can choose between a standard breakpoint or a conditional breakpoint. A standard breakpoint simply stops the process (or processes) when it is encountered. A conditional breakpoint allows you to attach a condition or an expression to its occurrence. The condition or expression can be written in the C or Fortran language.

Breakpoint_summary_window

1.5 The breakpoint_summary_window, shown in Figure 6, displays an overview of all of the breakpoints in your debugging session. From this window you can enable and disable individual breakpoints, clear all breakpoints, and select a breakpoint. Selecting a breakpoint from this window opens and positions a process_window at the breakpointed line.

The example in Figure 6 shows two standard breakpoints. Enabled breakpoints are indicated by a STOP sign. If a breakpoint is disabled, it is indicated by a dimmed STOP sign. A conditional breakpoint is distinguished from a standard breakpoint by an EVAL sign.

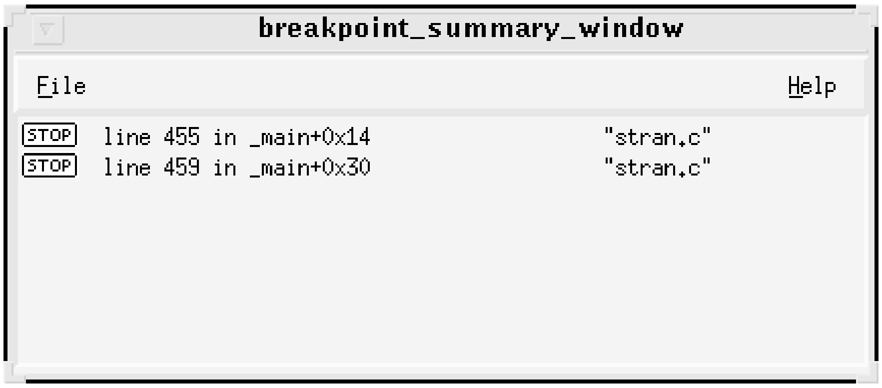

Figure 6. Sample breakpoint_summary_window

From the process_window you can temporarily enable or re-enable all breakpoints in your program, or save all of your breakpoints to a file for use in a later debugging session.

Array_browser_window

1.6

The array_browser_window provides information about array values. It has horizontal and vertical scroll bars that allow you to scroll the display. It also has several Dimension fields that allow you to specify which slice of an array you want displayed. From this display, you can go to a particular row and column dimension, and you can save the two-dimensional slice to a file.

Figure 7 shows a two-dimensional slice of a three-dimensional array in an array_browser_window. The sample two-dimensional slice shown in Figure 7 displays all rows and columns of 0 depth. The two asterisks (*,*) in the Dimension field indicate that a two-dimensional slice is being displayed.

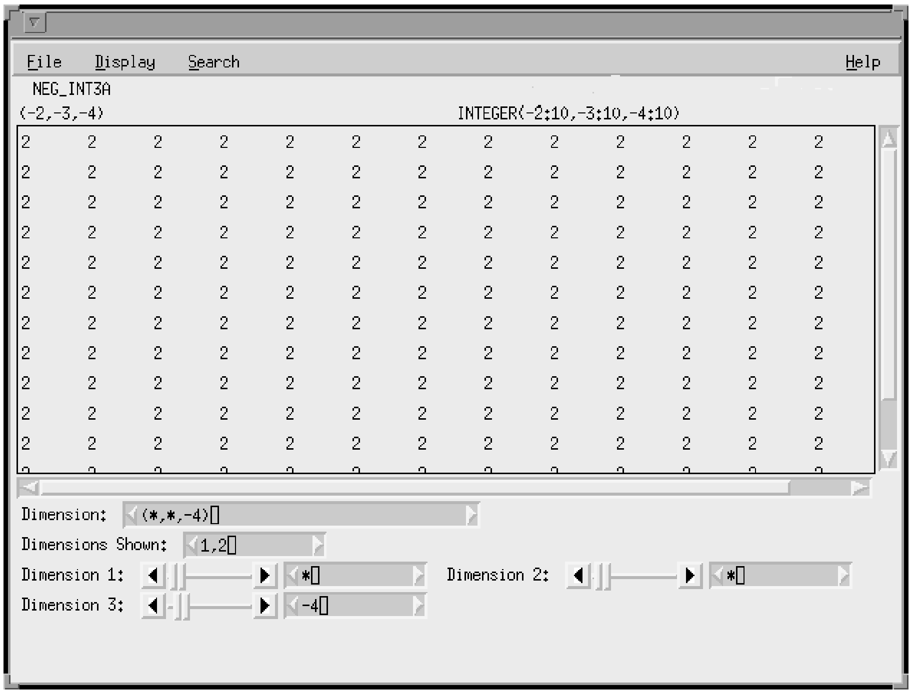

Figure 7. Sample array_browser_window

Getting Started [2]

Section 2

This section provides the following information you need to know before you start using the Cray TotalView debugger:

- Using mouse buttons
- Diving on items
- Changing variables

This section also describes how to start using the Cray TotalView debugger. It provides procedures for the following actions:

- Compiling a program
- Setting the display
- Starting the debugger
- Setting search paths
- Setting command arguments
- Setting breakpoints
- Starting a process

Before you start

2.1

Three of the most common actions you will perform while debugging your programs are using the mouse buttons, diving, and changing the value of variables.

Using the mouse buttons
2.1.1

When you start the Cray TotalView debugger, the totalview window appears on the screen by default. To display other windows and perform debugging operations, use the following mouse button functions:

Button	Description
Select item (left button)	Selects an item for further operations. For example, you can select a field and edit it, scroll windows, and resize panes.
Dive on an item (right button)	Displays additional information about an item. In many cases, a separate window appears, displaying the information. For example, diving into a variable displays a data object window, describing the address, data type, and value of that variable.

Diving into items
2.1.2

Diving is the action of displaying additional information about an item already displayed in a window. To dive into an item, place the cursor over the object and click the right mouse button. Figure 8 shows the variable `city` selected in the upper right pane of the `process_window`.

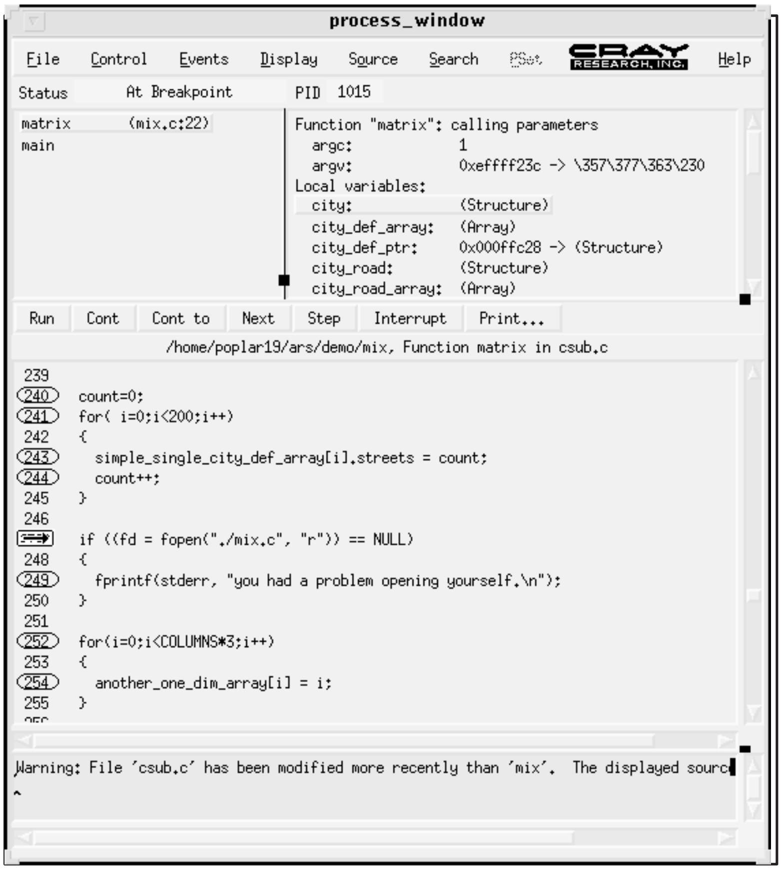

Figure 8. Selecting an item upon which to dive

Figure 9 shows the `data_object_window` displayed after diving on the `city` variable. Because `city` is a structure, all of its fields are shown automatically .

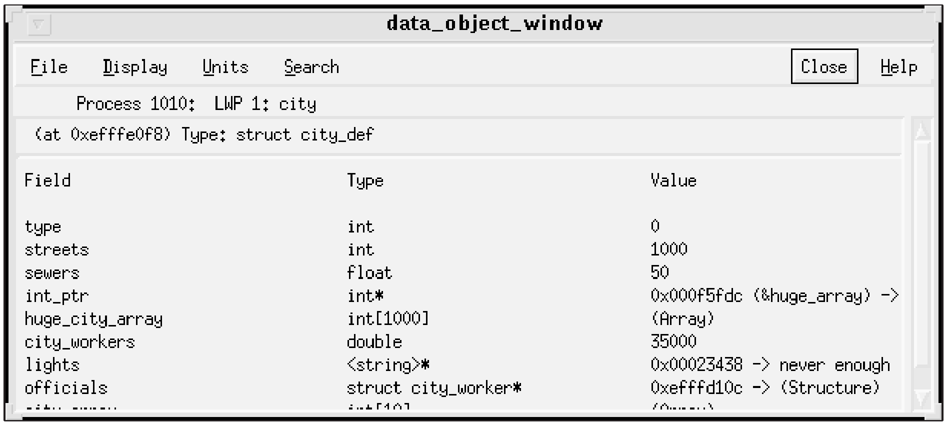

Figure 9. Dive output

Changing variables
2.1.3

To change the value of a variable, select the variable value in a `data_object_window` by positioning the cursor over the variable and clicking the left mouse button. A window appears in which you can change the value of the variable.

Figure 10 shows the `prompt_for_string_window` that appears after a dive on the value for variable `city.streets`. The current value is shown in the window. Click on the `Clear` button or use the backspace or delete key to erase the current value.

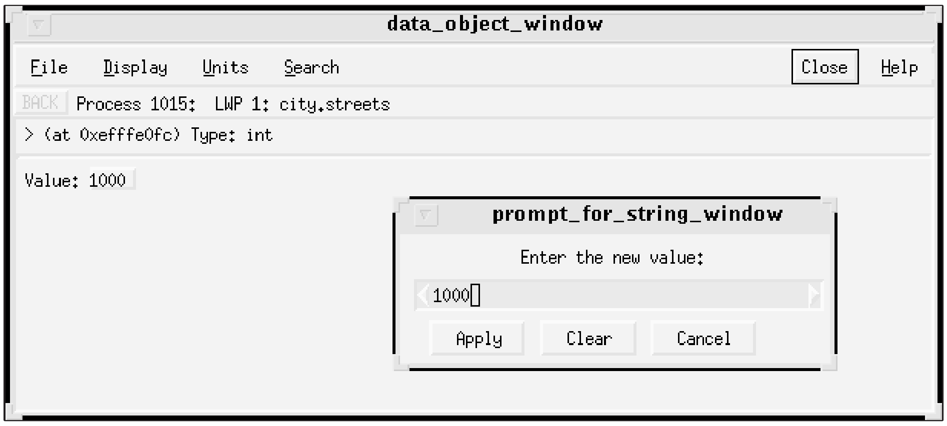

Figure 10. `prompt_for_string_window`

Compiling C or Fortran programs

2.2

To use the Cray TotalView debugger to debug a program, you must first compile the program, using the options appropriate to your situation.

To compile a C or Fortran program, use one of the following commands:

- C programs on Cray Research platforms:

  `cc -g` *progname* ... `-o` *executname* [*options*]

- C programs on SPARC platforms:

  `cc -g -xs` *progname* ... `-o` *executname* [*options*]

- C++ programs:

  `CC -g` *progname* `-o` ...

- FORTRAN 77 programs on MPP platforms:

  `cf77 -g -X`*n* `-C cray-t3d -o` *executname* [*options*]

Section 2

- FORTRAN 77 programs on SPARC platforms:

  `f77 -g -xs` *progname* ... `-o` *executname* [*options*]

- FORTRAN 77 programs on PVP platforms:

  `cf77 -g` *progname* ... `-o` *executname* [*options*]

- Fortran 90 programs:

  `f90 -g` *progname* ... `-o` *executname* [*options*]

The *progname* argument is the name of the source program, and *executname* is the name of the executable file. The `-g` option directs the compiler to generate symbol information for the Cray TotalView debugger.

Setting the display

2.3

Before you start the Cray TotalView debugger, you must set the X Window System `DISPLAY` environment variable, as follows:

If you use the `csh` shell command:

```
setenv DISPLAY workstation:0.0
```

If you use the `sh` or `ksh` shell command:

```
DISPLAY=workstation:0.0; export DISPLAY
```

Refer to the appropriate compiling system reference manual for more information on compiling programs for debugging.

Starting the debugger

2.4

The complete command-line syntax for TotalView is as follows:

`totalview [-L] [-c` *corefile* `] [-V] [-p pid] [-X` *npes* `] [`*Xoptions* `]` *filename*

Option	Description
`-L`	Requests that the line-mode interface to TotalView.
`-c` *corefile*	Specifies name of a core file to be debugged.
`-V`	Prints TotalView version, date, and time.
`-p` *pid*	Specifies process ID of a live process that, upon invocation of TotalView, is stopped and control of which is transferred to TotalView.
`-X` *npes*	Number of processor elements (PEs) in a partition.
Xoptions	Specifies standard X Window option values for TotalView to use.
filename	Specifies name of an executable file to be debugged. The name can be a quoted string containing a complete program invocation including arguments and standard I/O redirection.

Note: This document describes the X Window System interface for TotalView. Type `help` *commandname* at the prompt when in line mode to access online help for the line-mode interface.

The following examples show some common ways that you can start the debugger:

- `totalview`

  This command starts the debugger without loading a program or core file. After the debugger has started, you can load a program by using the `Load New Program` option of the File menu on the `totalview` window.

- `totalview` *filename*
  or
  `totalview` *filename* `-c` *corefile*
  or
  `totalview` *corefile* (MPP systems only)

These commands start the debugger in the default configuration, with *filename* and *corefile* specified as the name of the program and core file to be debugged.

- `totalview` *–otheroptions* "*filename arg1 arg2* "

  For example, a complete command line could look like the following example:

```
totalview –Xoptions "fred –a –b george pete < infile > outfile"
```

  This command starts the debugger in the default configuration and passes all subsequent arguments to the specified program. The quotes are required. You can put a complete command line, including standard I/O redirection, inside the quotes.

Setting search paths

2.5

If your source code resides in more than one directory, you can set search paths from the `process_window` by using the Control menu `Set Search Directory` option. The debugger searches directories in the following order for source code:

1. Current directory (`.`).
2. List of directories you specify with the `Set Search Directory` option. The debugger searches the directories in the order in which they appear in the dialog box.
3. The specified directory if you entered a full path name for the executable file.
4. Directories specified in your `PATH` environment variable.

These search paths apply to all processes that you are debugging.

After you change the list of directories, the debugger automatically searches again for the source file that is currently displayed.

Note: You can also permanently specify search directories with a parameter in the `.Xdefaults` or `TOtalview` file.

Figure 11 shows the `prompt_for_paragraph_window`, displayed when you select the `Set Search Directory` option.

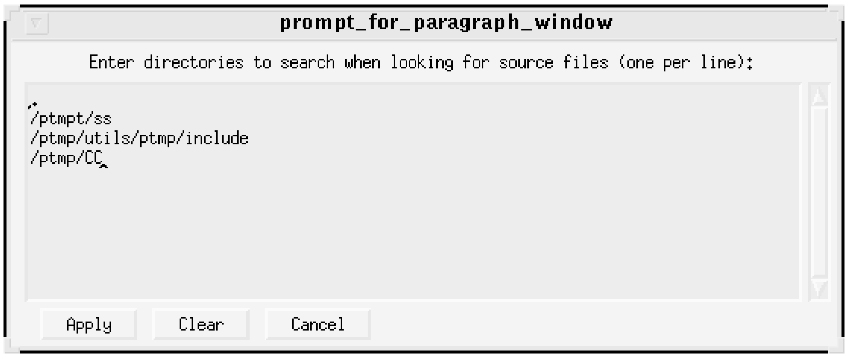

Figure 11. Searching directories for source files

Setting command arguments

2.6

When the debugger creates a process, it passes one argument to the program by default. This argument is the name of the file that contains the executable code for the process. To set additional command-line arguments, perform the following steps:

1. From the `process_window`, select the Control menu `Run (args)` option
2. Enter the arguments to be passed to the program. Use a space to separate arguments.
3. Select the `Apply` button to confirm your entries.
4. Repeat steps 1 through 3 for all programs you are debugging.

You can also set command-line arguments by using the `-a` option on the `totalview` command line.

Figure 12 shows the prompt for string window. This example shows that two arguments to your program have been entered: `file.data` and `file2.data`.

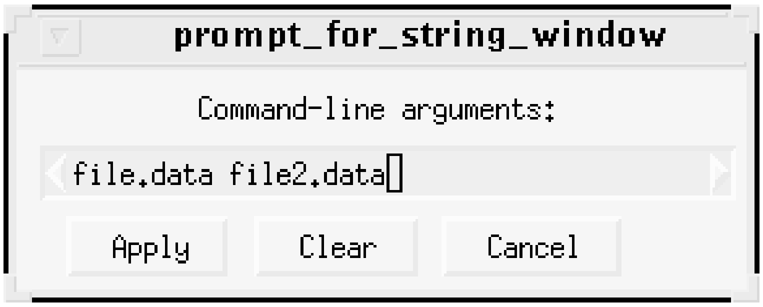

Figure 12. Two arguments entered

Setting breakpoints

2.7

A *breakpoint* is a point in a program at which you can suspend execution to examine or manipulate the data. Before starting your process, you might want to set breakpoints at locations that you know are going to be interesting. To set a breakpoint, perform the following steps:

1. Find the line in the source code at which you want to set the breakpoint.
2. Use the left mouse button to select the oval icon at the left of the line number .

To delete the breakpoint you have set, select it again.

Figure 13 shows a typical `process_window`. A breakpoint was set at line 187 where the `STOP` sign appears.

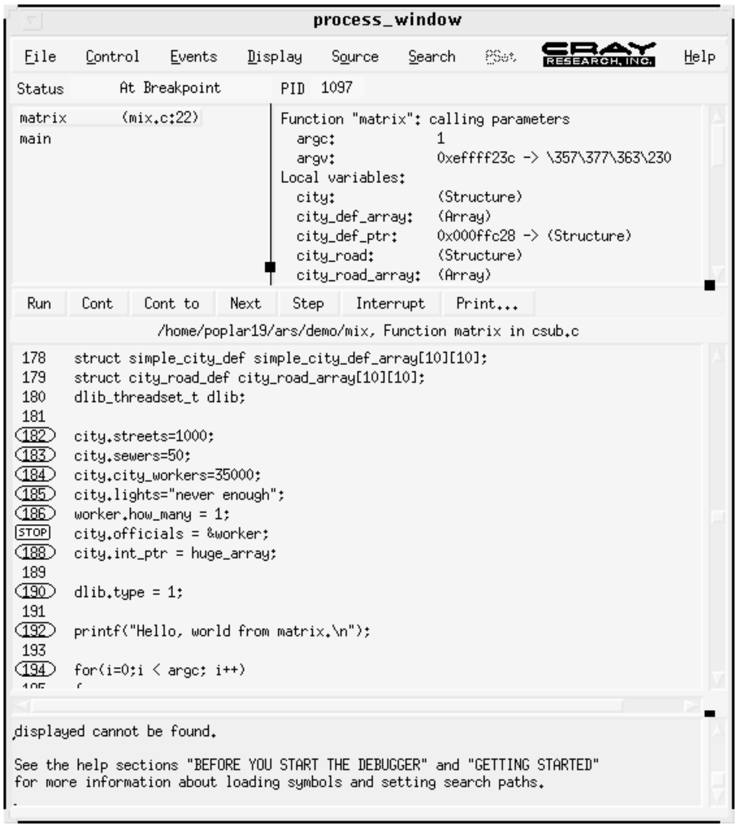

Figure 13. Setting breakpoints

Section 2

Figure 14 shows the debugging session from Figure 13 after running to the breakpoint set at line 187. The current line indicator now points to that line.

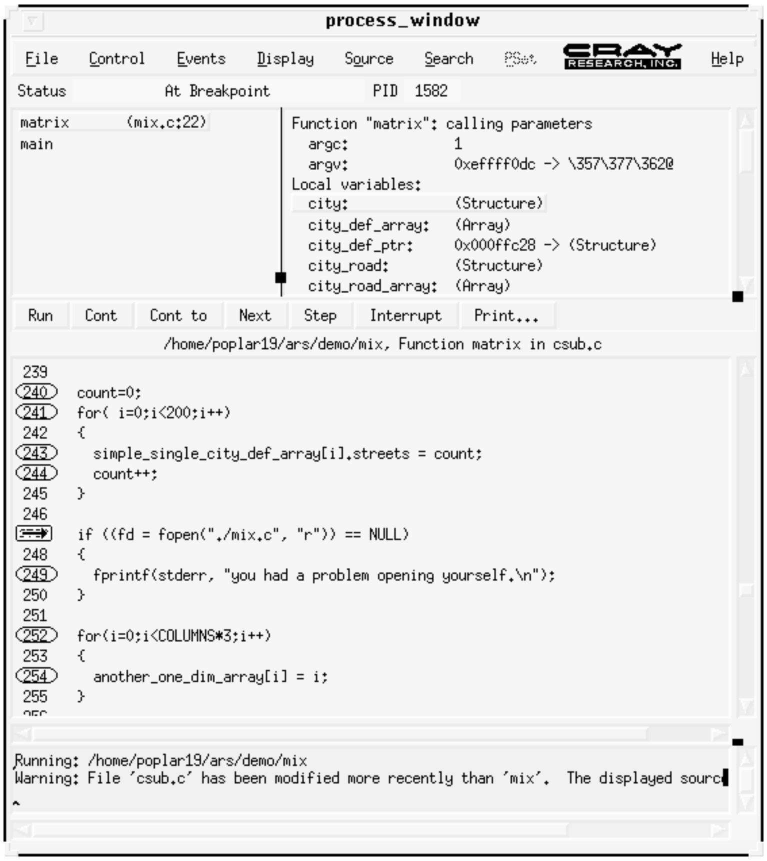

Figure 14. After running to breakpoint

Starting a process

2.8

From the `process_window`, you can start a new process or continue a process from a stopped state by using any of the following menu options, which are available through the Control menu, or by using the `Run`, `Cont`, `Cont to`, `Next`, and `Step` buttons located in the middle of the `process_window`:

Option	Description
`Run`	Starts your process from the beginning and runs to the first breakpoint (if any).
`Run (args)`	Opens a dialog box that lets you enter command-line arguments before running. This option always starts your process from the beginning and runs to the first breakpoint (if any).
`Cont`	Continues execution (or starts it if not already started) from the current location.
`Cont to`	First, highlight a source statement in the process window by selecting it with the left mouse button. Then use the `Cont to` command to start your program and execute to that statement.
`Step (source line)`	Executes the next source-level instruction while stepping through function calls.
`Stepi (instruction line)`	Executes the next machine-level instruction, while stepping you through any function calls, for the current process only and then stops. It enters subroutines where debug symbols are available. (No button exists for this option.)
`Next (source line)`	Executes the next source-level instruction without stepping through function calls.
`Nexti (instruction line)`	Executes the next machine-level instruction, while stepping you through any function calls, for the current process only and then stops. (No button exists for this option.)

Getting Help [3]

The online help system available within the Cray TotalView debugger puts information at your fingertips with the click of a button and ensures that help is always available when you need it. The Help menu gives you easy access to information about every aspect of the tool. By selecting among the following menu options, you can view documentation online without ever leaving your Cray TotalView debugger session:

- `READ ME FIRST`
- `Getting Started Help`
- `Window Help`
- `Version`
- `Release Information`

These options provide the following functionality:

Option	Description
`READ ME FIRST`	Invokes the Xhelp facility and opens the help file to the "Read Me First" section of the TotalView help file.
`Getting Started Help`	Invokes the Xhelp facility and opens the help file to the "Getting Started" section of the TotalView help file.
`Window Help`	Invokes the Xhelp facility and opens the help file to the section of the TotalView help file that contains information about options window from which you chose the `Window Help` option.
`Version`	Opens a window that displays version and copyright information about the tool.
`Release Information`	Opens a window that displays information about what is new in the current release and what software bugs are known to be problems.

Xhelp facility

3.1

The online help file available through Xhelp provides you with complete information about the Cray TotalView debugger. This file replaces paper documentation. The help file contains the following information:

- Introduces the various portions of the user interface and helps you understand how to navigate through the tool
- Provides task-oriented information about various tasks
- Describes all available windows and menu options
- Lists X Window System Toolkit options and environment variables available with TotalView

Output from the Xhelp facility is displayed in a separate window that is composed of two panes. The left pane lists items you can view, such as a list of help files available, the table of contents for a help file, or the index for a file. The right pane displays the help text. Figure 15, page 27 shows the help window with a portion of help text displayed.

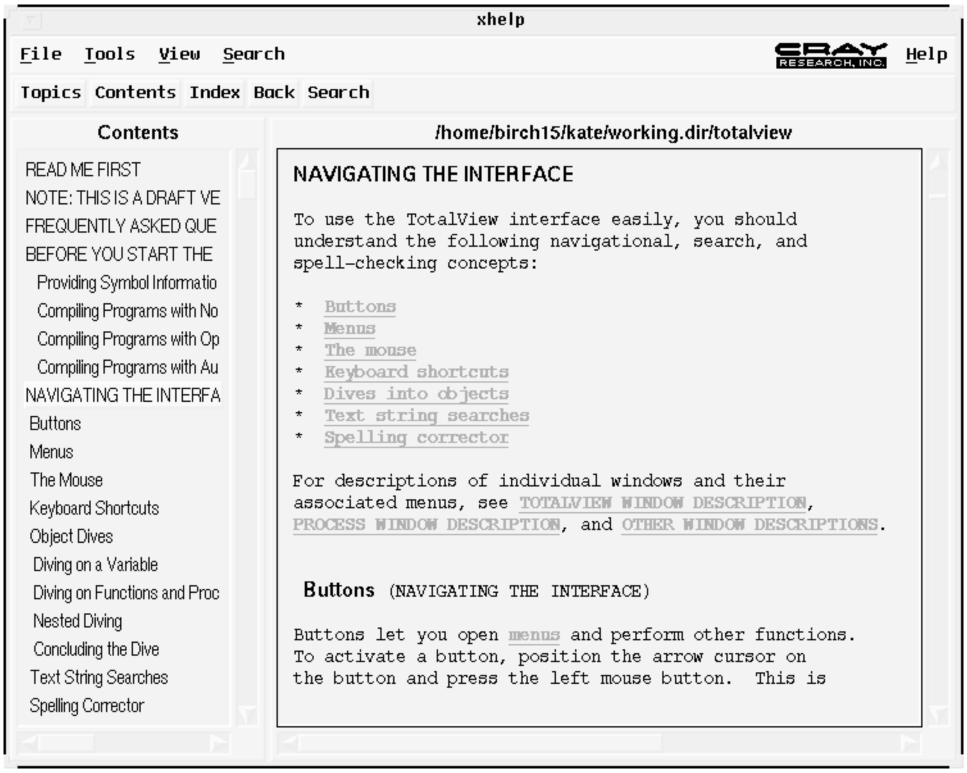

Figure 15. `xhelp` window with help file loaded

You can browse through help text in the following ways:

- Click on an entry in the Contents pane to move directly to a selection.
- Jump from area to area by using hypertext links. Within the text are bold, underlined words. This indicates that the words are active links to other information. Clicking on one of these words triggers the link, which then moves the text directly to the cross-referenced information.
- Use the search capability to look for an occurrence of a word or words. You can search forward and backward through text.
- Move gradually through the text by using the scroll bars.
- Locate topics by using the online index

Glossary

An expanded version of this glossary appears at the end of the TotalView online help file that is available through the Xhelp facility.

breakpoint

A point in a program at which execution can be conditionally suspended to permit examination and manipulation of data. Breakpoints that are shared by parent and child processes with the same executable file do not apply to child processes that have different executable files.

breakpoint icon

A blank oval character that the debugger displays to the left of an instruction that generates executable code, indicating that a breakpoint can be set at that location.

button

An icon that allows you to open menus, execute commands, and display pop-up windows. To activate, position the cursor on the button and press the left mouse button. (This is also called *clicking* on a button.)

call sequence pane

A pane that appears in the upper left portion of the process window. It contains a list of all of the functions or subroutines that are active in the current calling hierarchy of your program. The first routine that was executed appears at the bottom of the list and the routine most recently entered appears at the top of the list. When the program stops, the top routine in the call sequence pane is highlighted and the information about that routine is displayed in the current routine and source panes. You can dive on any of the routine names in the call sequence pane. That name will be highlighted and information about the routine will appear in the current routine and source panes.

click

To press a mouse button once and release it without moving the pointer.

core file

A file that contains the contents of memory. The operating system dumps (creates) a core file whenever a program exits because of a severe error (such as an attempt to store into an invalid address).

current routine pane

A pane that appears in the process window and describes the state of the routine that is highlighted in the call sequence pane. Listed in this pane are the routine name and all of the names and values of all calling parameters, local variables, and common blocks declared in the routine. You can dive on any variable in this pane to obtain additional information or to change the value. The current program counter (PC) value and traceback information also are displayed in this pane.

If the routine that appears in the current routine pane contains nested scopes with variables of the same name declared at different scope levels, the debugger will display all variables that are in scope at the current point of execution. To view the context of other variables, dive on the specific variable in the source pane.

`data_object_window`

A window that displays the name, address, data type, and value of a local variable, register, or global variable. It also displays the values and machine-level instructions stored in a block of memory.

debugger

A software tool for detecting, diagnosing, and eliminating errors in programs.

dialog box

A window that the system displays when it needs to gather information (such as a search string) before it can continue, or when it wants you to confirm your intention to take some action (such as quitting the debugger). You can also use a dialog box to modify variable information. When a dialog box appears, you must provide the requested information before continuing.

diving

The action of displaying more information about an object. For example, when you dive into a variable in the Cray TotalView debugger, a window appears with more information about the variable.

help facility

A facility that uses a common help tool called Xhelp. Many Cray Research, Inc. tools use Xhelp to present online help. Xhelp has help available about itself. Select `Help` from the menu bar of the Xhelp tool for information on the use of this tool.

menu bar

The area at the top of a window that contains menu selections.

object

A variable name, value, address, routine name, icon, type, source line, register value, process entry, or any other divable item.

pane

A partitioned area within a window that permits the window to display related pieces of information simultaneously. Each pane has a title bar and an optional scroll bar.

parallel processor

A computer that contains two or more central processing units (CPUs), working on one or more programs simultaneously, and often sharing memory , operating system kernels, and I/O management capabilities.

pop-up menu

A menu in a window that lists the commands available from that window.

process set

A set of program groups and share sets, associated with a multiprocess program. When you start a multiprocess program, the debugger adds each process to a process set as the process starts. The debugger groups processes based on the type of system call that created or changed the processes, as follows:

`fork()` system call

When a process calls `fork()`, the process creates child processes and thus becomes a parent process. The child processes inherit the same executable file as the parent. Thus, the parent and child share the same source code. Some programs may call `fork()` directly, while others call it indirectly.

`execve()` system call

In some programs, a child process makes a call to `execve()`. In this case, the child's executable file (which was identical to the parent's) is replaced with the executable file specified in the `execve()` call. Consequently, the parent and child no longer share the same source code. (If the parent calls `execve()`, the child processes' executable files remain the same.)

`process_window`

The main window for a process that consists of three panes: the call sequence pane, current routine pane, and the source code pane for the program.

process-related window

A class of window that displays information about a specific process. Thus, multiple instances of a given type of process-related window can exist, one per process (for example, an expression evaluation window). Compare with system window.

program counter

A counter (known as a PC) that you can set to a selected line of source code or to an absolute value. When you set the PC to a selected line, the PC points to the memory location at which the statement begins. When you set the PC to an absolute value, the PC points to the line that contains the absolute address.

program set

A set of processes that includes the parent process and all related processes. A program set includes child processes that were created by `fork()` (processes that share the same source code as the parent) and child processes that were created by `fork()` but with a subsequent call to `execve()` (processes that do not share the same source code as the parent). Compare with share set.

resource file

A file that customizes the Cray TotalView debugger's actions. The debugger currently uses two resource files, `.Xdefaults` and `TOtalview`.

`totalview` window

A window that displays information about each process you are debugging.

scroll bar

A box that appears on the right side and/or bottom of the window or pane and allows you to move up and down or left and right through the text in a window .

scrolling

Displaying additional text one page at a time, one line at a time, or an arbitrary number of lines at a time. Sometimes, the volume of information to be displayed exceeds the size of the window or pane, so you must scroll up/down or right/left to view the rest of the information.

set

The processes that belong to a multiprocess program.

share set

A set of processes that includes the parent process and any related processes that share the same source code as the parent. Thus, a share set includes only processes that were spawned with `fork()` (without calls to `execve()`). Compare with program set.

source pane

The pane in the process window that displays code for the routine that is highlighted in the call sequence pane. If symbol information is available for the routine and the debugger can locate the source files, it will display the program source code. Otherwise, the source pane will contain the machine level instructions. The boxes on the left edge of the pane indicate breakpoint locations. Selecting a box enables or disables a breakpoint at that location. Diving on a box brings up a breakpoint window to provide more detailed control of the breakpoint. You can dive on the names of any variable or function in the source pane. If you dive on a variable name, the debugger will display a data object window that describes that variable. If you dive on the function name, the source for that function will appear in the source pane, overlaying the original routine.

source-level debugger

A debugger that analyzes source programs written in high-level languages such as C and Fortran. On Cray Research systems, the Cray TotalView debugger and CDBX are examples of source-level debuggers.

symbol table

A table of symbolic names (for example, variables) used in a program and their memory locations. The symbol table is part of the executable object generated by the compiler and debuggers use it to analyze the program.

title bar

A box that appears at the top of the window or pane. It sometimes contains a shaded pattern and other descriptive information about the pane.

TotalView

A source-level debugger. The Cray TotalView debugger features a graphic interface (based on the X Window System) and operations for debugging multiple processes in parallel programs.

`TOtalview` file

A resource file in your home directory that contains options for customizing the behavior of the Cray TotalView debugger.

undiving

An action that causes a window to display its previous contents. To undive from a nested dive window, place the cursor over the undive icon (two upward-pointing angle brackets) on the left side of the title bar of the window and click either the left or the right button. The window displays the information that was displayed prior to the dive.

`.Xdefaults` file

A file in your home directory that contains command-line options for customizing the behavior of X Window System clients.

X Window System

A window system that was developed at the Massachusetts Institute of Technology (MIT) in 1984. The X Window System consists of display servers and clients. Display servers provide display capabilities and monitor user input; clients are applications programs that perform specific tasks.

Index

A

Array browser window, 10
Array value information, 10

B

Breakpoint summary window, 9
Breakpoint window, 7
Breakpoints
  conditional, 7
  enabling and disabling, 9
  setting, 7, 20
  stopping at, 5

C

Command arguments, setting, 19
Command line, examples, 17
Command–line syntax, 16
Common actions, 11
Compiling programs, 15
Copyright information, displaying, 25

D

Data object window, 6
Display setting, 16
`DISPLAY` variable, setting, 16
Diving, 12

E

Environment variables, 16

H

Help
  online, 25
  types available, 25

L

Line–mode interface, help for, 17
Loading `PrgEnv` module, 1

M

Modules, loading the `PrgEnv` module, 1
Mouse button functions, 12

N

New features, displaying, 25

O

Online help, 25
Online help file, 26

P

Panes
- call sequence, 3
- current routine, 3
- definition, 3
- information, 3
- source, 3

`Prg Env` module, 1
Process starting, 23
Process window, 3
Programs, compiling, 15

S

Search path setting, 18
Search paths, setting, 18
Setting command arguments, 19
Software problems, displaying, 25
Starting the debugger, 16

T

Totalview window, 1

V

Variables
- changing, 14
- displaying information, 6

Version information, displaying, 25

W

Windows
- array browser, 10
- breakpoint, 7
- breakpoint summary, 9
- data object, 6
- process, 3
- `totalview`, 1

Index

Reader's Comment Form

Introducing the Cray TotalView Debugger IN–2502 2.0

Your reactions to this manual will help us provide you with better documentation. Please take a moment to complete the following items, and use the blank space for additional comments.

List the operating systems and programming languages you have used and the years of experience with each.

Your experience with Cray Research computer systems: _____0-1 year _____1-5 year _____5+years

How did you use this manual: _____in a class _____as a tutorial or introduction _____as a procedural guide _____as a reference _____for troubleshooting _____other

Please rate this manual on the following criteria:

	Excellent			Poor
Accuracy	4	3	2	1
Appropriateness (correct technical level)	4	3	2	1
Accessibility (ease of finding information)	4	3	2	1
Physical qualities (binding, printing, illustrations)	4	3	2	1
Terminology (correct, consistent, and clear)	4	3	2	1
Number of examples	4	3	2	1
Quality of examples	4	3	2	1
Index	4	3	2	1

Please use the space below for your comments about this manual. Please include general comments about the usefulness of this manual. If you have discovered inaccuracies or omissions, please specify the number of the page on which the problem occurred.

Name ____________________
Title ____________________
Company ____________________
Telephone ____________________
Today's date ____________________

Address ____________________
City ____________________
State/Country ____________________
Zip code ____________________
Electronic mail address ____________________

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO 6184 ST. PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY RESEARCH, INC.

ATTN: Software Publications Group
655 LONE OAK DR BLDG F
EAGAN MN 55121-9957

Fold

Cut along this line

Reader's Comment Form

Introducing the Cray TotalView Debugger IN–2502 2.0

Your reactions to this manual will help us provide you with better documentation. Please take a moment to complete the following items, and use the blank space for additional comments.

List the operating systems and programming languages you have used and the years of experience with each.

Your experience with Cray Research computer systems: _____0-1 year _____1-5 year _____5+years

How did you use this manual: _____in a class _____as a tutorial or introduction _____as a procedural guide _____as a reference _____for troubleshooting _____other

Please rate this manual on the following criteria:

	Excellent			Poor
Accuracy	4	3	2	1
Appropriateness (correct technical level)	4	3	2	1
Accessibility (ease of finding information)	4	3	2	1
Physical qualities (binding, printing, illustrations)	4	3	2	1
Terminology (correct, consistent, and clear)	4	3	2	1
Number of examples	4	3	2	1
Quality of examples	4	3	2	1
Index	4	3	2	1

Please use the space below for your comments about this manual. Please include general comments about the usefulness of this manual. If you have discovered inaccuracies or omissions, please specify the number of the page on which the problem occurred.

Name ______________________

Title ______________________

Company ______________________

Telephone ______________________

Today's date ______________________

Address ______________________

City ______________________

State/Country ______________________

Zip code ______________________

Electronic mail address ______________________

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO 6184 ST. PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY RESEARCH, INC.

ATTN: Software Publications Group
655 LONE OAK DR BLDG F
EAGAN MN 55121-9957

Fold

Cut along this line