

IBM Parallel Environment for AIX



Operation and Use, Volume 2, Part 2 Profiling

Version 2 Release 4

IBM Parallel Environment for AIX



Operation and Use, Volume 2, Part 2 Profiling

Version 2 Release 4

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page v.

Third Edition (October 1998)

This edition applies to Version 2, Release 4, Modification 0 of the IBM Parallel Environment for AIX (5765-543), and to all subsequent releases and modifications until otherwise indicated in new editions.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

IBM welcomes your comments. A form for your comments appears at the back of this publication. If the form has been removed, address your comments to:

IBM Corporation, Department 55JA, Mail Station P384

522 South Road

Poughkeepsie, NY 12601-5400

United States of America

FAX (United States and Canada: 1+914+432-9405

FAX (Other Countries)

Your International Access Code)+1+914+432-9405

IBMLink (United States customers only): IBMUSM10(MHVRCFS)

IBM Mail Exchange: USIB6TC9 at IBMMAIL

Internet e-mail: mhvrnf@vnet.ibm.com

World Wide Web: <http://www.rs6000.ibm.com> (select Parallel Computing)

If you would like a reply, be sure to include your name, address, telephone number, or FAX number.

Make sure to include the following in your comment or note:

- Title and order number of this book
- Page number or topic related to your comment

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Copyright International Business Machines Corporation 1998. All rights reserved. Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

© Copyright International Business Machines Corporation 1995, 1998. All rights reserved.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	v
Trademarks	vii
About This Book	ix
Who Should Use This Book	ix
How This Book is Organized	ix
Overview of Contents	ix
Typographic Conventions	x
Related Publications	x
IBM Parallel Environment for AIX Publications	x
Related IBM Publications	xi
Related Non-IBM Publications	xi
National Language Support	xii
Accessing Online Information	xii
Online Information Resources	xii
Getting the Books Online	xiii
Chapter 1. Profiling Parallel Programs with Xprofiler	1
Before You Begin	1
About Xprofiler	1
Requirements and Limitations	1
Xprofiler versus gprof	2
Compiling Applications to be Profiled	2
Starting Xprofiler	3
Xprofiler Command Line Options	3
Loading Files from the Xprofiler GUI	6
Setting the File Search Sequence	14
Understanding the Xprofiler Display	15
The Xprofiler Main Window	16
Using the Xprofiler Graphical User Interface	20
Using the Dialog Window Buttons	21
Using the Search Engine	21
Using the Save Dialog Windows	21
Using the Dialog Window Filters	22
Using the Radio/Toggle Buttons and Sliders	22
Manipulating the Function Call Tree	24
Zooming In on the Function Call Tree	24
Other Viewing Options	28
Filtering What You See	30
Clustering Libraries Together	36
Locating Specific Objects in the Function Call Tree	39
Getting Performance Data for Your Application	41
Getting Basic Data	41
Getting Detailed Data via Reports	46
Looking at Source Code	56
Saving Screen Images of Profiled Data	60
Appendix A. Parallel Environment Tools Commands	63
xprofiler	63

Appendix B. Customizing Tool Resources	67
Xprofiler Resource Variables	68
Controlling Fonts	68
Controlling the Appearance of the Xprofiler Main Window	68
Controlling Variables Related to the File Menu	69
Controlling Variables Related to the View Menu	72
Controlling Variables Related to the Filter Menu	73
Appendix C. Profiling Programs with the AIX prof and gprof Commands	75
Glossary of Terms and Abbreviations	77
Index	85

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing

IBM Corporation

500 Columbus Avenue

Thornwood, NY 10594

USA

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

Mail Station P300

522 South Road

Poughkeepsie, NY 12601-5400

USA

Attention: Information Request

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX
AIX/6000
IBM
LoadLeveler
Micro Channel
RISC System/6000
RS/6000
POWERparallel
SP

Microsoft, Windows, and the Windows logo are trademarks or registered trademarks of Microsoft Corporation.

PostScript is a trademark of Adobe Systems, Incorporated.

Motif is a trademark of Open Software Foundation.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names, which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

About This Book

This book describes the profiling facilities for the IBM Parallel Environment (PE) for AIX program product and how to use these profiling tools to analyze and tune your parallel programs.

This book concentrates on the actual commands, graphical user interfaces, and use of these tools as opposed to the writing of parallel programs. For this reason, you should use this book in conjunction with *IBM Parallel Environment for AIX: MPI Programming and Subroutine Reference*, (GC23-3894) and *IBM Parallel Environment for AIX: MPL Programming and Subroutine Reference* (GC23-3893).

This book assumes that AIX Version 4.3.2 or later, X-Windows**, and the PE software are already installed. It also assumes that you have been authorized to run the Parallel Operating Environment (POE). The PE software is designed to run on an IBM RS/6000 SP, an RS/6000 network cluster, or on a mixed system where additional RS/6000 processors supplement an SP system. For complete information on installing the PE software and setting up users, see *IBM Parallel Environment for AIX: Installation*, (GC23-3892). Also, see the appropriate AIX 4.3.2 or later documentation listed under "Related Publications" on page x. For information on POE and executing parallel programs, see *IBM Parallel Environment for AIX: Operation and Use, Volume 1, Using the Parallel Operating Environment* and *IBM Parallel Environment for AIX: Hitchhiker's Guide*

Who Should Use This Book

This book is designed primarily for end users and application developers. It is also intended for those who run parallel programs, and some of the information and tools covered should interest system administrators. Readers should have some experience with graphical user interface concepts such as windows, pull-down menus, and menu bars. They should also have knowledge of the AIX operating system and the X-Window system. Where necessary, this book provides some background information relating to these areas. More commonly, this book refers you to the appropriate documentation.

How This Book is Organized

Overview of Contents

This book contains the following information:

- Chapter 1, "Profiling Parallel Programs with Xprofiler" on page 1 describes how to profile your programs with the Parallel Environment's Xprofiler.
- Appendix A, "Parallel Environment Tools Commands" on page 63 contains the manual pages for the PE commands discussed throughout this book.
- Appendix B, "Customizing Tool Resources" on page 67 describes how to customize X-Windows resources for PE tools.
- Appendix C, "Profiling Programs with the AIX prof and gprof Commands" on page 75 describes how to use the AIX profilers **prof** and **gprof** to profile parallel programs.

Typographic Conventions

This book uses the following typographic conventions:

Type Style	Used For
bold	Bold words or characters represent system elements that you must use literally, such as command names, flag names, and path names. Bold words also indicate the first use of a term included in the glossary.
<i>italic</i>	<i>Italic</i> words or characters represent variable values that you must supply. <i>Italics</i> are also used for book titles and for general emphasis in text.
Constant width	Examples and information that the system displays appear in constant width typeface.

In addition to the highlighting conventions, this manual uses the following conventions when describing how to perform tasks. User actions appear in uppercase boldface type. For example, if the action is to enter the **xprofiler** command, this manual presents the instruction as:

ENTER **xprofiler**

The symbol “●” indicates the system response to an action. So the system's response to entering the **xprofiler** command would read:

- The **xprofiler** main window opens.

Related Publications

IBM Parallel Environment for AIX Publications

- *IBM Parallel Environment for AIX: General Information*, GC23-3906
- *IBM Parallel Environment for AIX: Hitchhiker's Guide*, GC23-3895
- *IBM Parallel Environment for AIX: Installation*, GC28-1981
- *IBM Parallel Environment for AIX: Operation and Use, Volume 1, Using the Parallel Operating Environment*, SC28-1979
- *IBM Parallel Environment for AIX: MPI Programming and Subroutine Reference*, GC23-3894
- *IBM Parallel Environment for AIX: MPL Programming and Subroutine Reference*, GC23-3893
- *IBM Parallel Environment for AIX: Messages*, GC28-1982
- *IBM Parallel Environment for AIX: Licensed Program Specification*, GC23-3896

As an alternative to ordering the individual books, you can use SBOF-8588 to order the entire IBM Parallel Environment for AIX library.

Related IBM Publications

- *IBM AIX Version 4 Getting Started*, SC23-2527
- *IBM AIX General Concepts and Procedures for RS/6000* GC23-2202
- *IBM AIX Version 4 Files Reference*, SC23-2512
- *IBM AIX Version 4 System Management Guide: Communications and Networks*, SC23-2526
- *IBM AIX Version 4.1 Installation Guide* SC23-2550
- *IBM AIX Version 4.2 Installation Guide* SC23-1924
- *IBM AIX Version 4 Commands Reference*, SBOF-1851 (all volumes)
- *IBM AIX Versions 3.2 and 4 Performance Tuning Guide* SC23-2365
- *IBM AIX Version 4 Messages Guide and Reference* SC23-2641
- *IBM AIX Version 4.1 Network Installation Management Guide and Reference*, SC23-2627
- *IBM AIX Version 4.2 Network Installation Management Guide and Reference*, SC23-1926
- *IBM AIX Version 4 System Management Guide: Operating System and Devices*, SC23-2525
- *IBM AIX Version 4 General Programming Concepts: Writing and Debugging Programs*, SC23-2533
- *IBM AIX Version 4 Communications Programming Concepts* SC23-2610
- *Diskless Workstation Management Guide*, SC23-2433
- *C++ for AIX/6000: Language Reference*, SC09-1606
- *C++ for AIX/6000: Standard Class Library Reference*, SC09-1604
- *C++ for AIX/6000: User's Guide*, SC09-1605
- *IBM Performance Toolbox 1.2 and 2 for AIX: Guide and Reference*, SC23-2625

Related Non-IBM Publications

- Almasi, G., Gottlieb, A., *Highly Parallel Computing* Benjamin-Cummings Publishing Company, Inc., 1989.
- Gropp, W., Lusk, E., Skjellum, A., *Using MPI*, The MIT Press, 1994.
- Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard* Version 1.1, University of Tennessee, Knoxville, Tennessee, June 6, 1995.
- Foster, I., *Designing and Building Parallel Programs* Addison-Wesley, 1995.
- Pfister, Gregory, F., *In Search of Clusters* Prentice Hall, 1995.

National Language Support

For National Language Support (NLS), all PE components and tools display messages located in externalized message catalogs. English versions of the message catalogs are shipped with the PE program product, but your site may be using its own translated message catalogs. The AIX environment variable **NLSPATH** is used by the various PE components to find the appropriate message catalog. **NLSPATH** specifies a list of directories to search for message catalogs. The directories are searched, in the order listed, to locate the message catalog. In resolving the path to the message catalog, **NLSPATH** is affected by the values of the environment variables **LC_MESSAGES** and **LANG**. If you get an error saying that a message catalog is not found, and want the default message catalog:

```
ENTER   export NLSPATH=/usr/lib/nls/msg/%L/%N
          export LANG=C
```

The PE message catalogs are in English, and are located in the following directories:

```
/usr/lib/nls/msg/C
/usr/lib/nls/msg/En_US
/usr/lib/nls/msg/en_US
```

If your site is using its own translations of the message catalogs, consult your system administrator for the appropriate value of **NLSPATH** or **LANG**. For additional information on NLS and message catalogs, see *IBM Parallel Environment for AIX: Messages* and *AIX for RS/6000: General Programming Concepts*.

Accessing Online Information

| In order to use the PE man pages or access the PE online (HTML) publications,
| the **ppe.pedocs** file set must first be installed. To view the PE online publications,
| you also need access to an HTML document browser such as Netscape. An index
| to the HTML files that are provided with the **ppe.pedocs** file set is installed in the
| **/usr/lpp/ppe.pedocs/html** directory.

Online Information Resources

| If you have a question about the SP, PSSP, or a related product, the following
| online information resources make it easy to find the information:

- Access the new SP Resource Center by issuing the command:
/usr/lpp/ssp/bin/resource_center. Note that the **ssp.resctr** fileset must be installed before you can do this.

| If you have the Resource Center on CD ROM, see the readme.txt file for
| information on how to run it.

- Access the RS/6000 Web Site at: <http://www.rs6000.ibm.com>.

Getting the Books Online

All of the PE books are available in Portable Document Format (PDF). They are included on the product media (tape or CD ROM), and are part of the **ppe.pedocs** file set. If you have a question about the location of the PE softcopy books, see your System Administrator.

To view the PE PDF publications, you need access to the Adobe Acrobat Reader 3.0.1. The Acrobat Reader is shipped with the AIX Version 4.3 Bonus Pack and is also freely available for downloading from the Adobe web site at URL **<http://www.adobe.com>**.

As stated above, you can also view or download the PE books from the IBM RS/6000 web site at **<http://www.rs6000.ibm.com>**. At the time this manual was published, the full path was **http://www.rs6000.ibm.com/resource/aix_resource/sp_books**. However, note that the structure of the RS/6000 web site can change over time.

Chapter 1. Profiling Parallel Programs with Xprofiler

This chapter describes how to profile your programs with the Xprofiler profiling tool of the IBM Parallel Environment for AIX. This chapter explains how to use the Xprofiler graphical user interface (GUI) to profile your application, so it is best to read it while you have the GUI up and running.

If you intend to use the AIX **gprof** command to profile your parallel application, see Appendix C, “Profiling Programs with the AIX prof and gprof Commands” on page 75 for information on how to do so. You may also find it helpful to consult the *IBM AIX Version 4 Commands Reference*

You do not need to be familiar with the AIX **gprof** command to use Xprofiler.

Xprofiler is a tool that helps you analyze your parallel application's performance quickly and easily. It uses data collected by the **-pg** compiling option to construct a graphical display of the functions within your application. Xprofiler provides quick access to the profiled data, which lets you identify the functions that are the most CPU-intensive. The graphical user interface also lets you manipulate the display in order to focus on the application's critical areas.

Before You Begin

About Xprofiler

Xprofiler lets you profile both serial and parallel applications. The difference is that when you run a serial application, a single profile data file is generated, while a parallel application produce multiple profile data files. Either way, you can use Xprofiler to analyze the resulting profiling information.

Xprofiler provides a set of resource variables that let you customize some of the features of the Xprofiler window and reports. For information about customizing resources for Xprofiler, see *IBM Parallel Environment for AIX: Operation and Use, Volume 2, Tools Reference*

The word *function* is used frequently throughout this chapter. Consider it to be synonymous with the terms *routine*, *subroutine*, and *procedure*.

Requirements and Limitations

To use Xprofiler, your application must be compiled with the **-pg** option. For more information about compiling, see “Compiling Applications to be Profiled” on page 2.

Like gprof, Xprofiler lets you analyze CPU (busy) usage only. It cannot give you other kinds of information such as CPU idle, I/O, or communication.

To run Xprofiler, AIX 4.2.1 (or later) must be installed on your machine.

If you compile your application on one machine, and then analyze it on another, you must first make sure that both machines have similar library configurations, at least for the system libraries used by the application. For instance, say you ran an HPF application on an SP, then tried to analyze the profiled data on a workstation. The levels of HPF runtime libraries must match, and must be placed in a location

that Xprofiler recognizes on the workstation. Otherwise, Xprofiler produces unpredictable results.

Since Xprofiler collects data by sampling, short-executing functions may show no CPU use.

Xprofiler does not give you information about the specific threads in a multi-threaded program. The data that Xprofiler presents is a summary of the activities of all the threads.

Xprofiler versus gprof

With Xprofiler, you can produce the same tabular reports that you may be accustomed to seeing with **gprof**. Just as with **gprof**, you can generate the Flat Profile, Call Graph Profile, and Function Index reports. Xprofiler is different from **gprof** in that it provides a graphical user interface (GUI) from which you can profile your application. It generates a graphical display of your application's performance, as opposed to just a text-based report. Unlike **gprof**, Xprofiler also lets you profile your application at the source statement level.

From the Xprofiler GUI, you can use all the same command line flags as **gprof**, plus a few more that are unique to Xprofiler.

Compiling Applications to be Profiled

In order to use Xprofiler, you must compile and link your application with the **-pg** option of the compiler command. This applies regardless of whether you are compiling a serial or parallel application. You can compile and link your application all at once, or perform the compile and link operations separately. Here's an example of how you would compile and link all at once:

```
cc -pg -o foo foo.c
```

And here's an example of how you would first compile your application and then link it. To compile:

```
cc -pg -c foo.c
```

To link:

```
cc -pg -o foo foo.o
```

Notice that when you compile and link separately, you must use the **-pg** option with *both* the compile and link commands.

The **-pg** option compiles and links the application so that when you run it, the CPU usage data gets written to one or more output files. For a serial application, this output consists of just one file called *gmon.out*, by default. For parallel applications, the output is written into multiple files, one for each task that is running in the application. To prevent each output file from overwriting the others, POE appends the task ID to each *gmon.out* file. For instance, *gmon.out.10*.

The **-pg** option is not a combination of the **-p** and the **-g** compiling options.

Note: You must set the LIBPATH environment variable to the profiled POE libraries in order to profile system libraries (like libc). For more information, see *IBM Parallel Environment for AIX: Operation and Use, Volume 2, Tools Reference*

In order to get a complete picture of your parallel application's performance, you must indicate all of its gmon.out files when you load the application into Xprofiler. When you specify more than one gmon.out file, Xprofiler shows you the sum of the profile information contained in each file.

The Xprofiler GUI provides the capability of viewing included functions. Note, however, that your application must also be compiled with the **-g** option in order for Xprofiler to display the included functions.

The **-g** option, in addition to the **-pg** option, is also required for source statement profiling.

Starting Xprofiler

You start Xprofiler from the AIX command line, using the **xprofiler** command. To use Xprofiler, you also need to specify the executable file (a.out), the profile data file (gmon.out), and any command line options.

Under some circumstances you may have multiple profile data files (gmon.out files). You will have more than one of these files if you are profiling a parallel application, because a gmon.out file is created for each task in the application when it is run. If you are running a serial application, there may be times when you want to summarize the profiling results from multiple runs of the application. In these cases, you will need to specify each of the profile data files you want to profile with Xprofiler.

You start Xprofiler by issuing the **xprofiler** command from the AIX command line. You also need to specify the executable, profile data file(s), and options, which you can do one of two ways. You can either specify them on the command line, with the **xprofiler** command, or you can issue the **xprofiler** command alone, then specify them from within the GUI. See "Loading Files from the Xprofiler GUI" on page 6.

To start Xprofiler and specify the executable, profile data file(s), and options:

ENTER **xprofiler** *a.out gmon.out...* [*options*]

where *a.out* is the binary executable file, *gmon.out* is the name of your profile data file(s), and *options* may be one or more of the options listed in "Xprofiler Command Line Options."

To print basic Xprofiler command syntax to the screen, use the **-h** or **-?** option with the **xprofiler** command from the command line. For example, **xprofiler -h**.

Xprofiler Command Line Options

You can specify the same command line options with the **xprofiler** command that you do with **gprof**, plus one additional option (**-disp_max**), which is specific to Xprofiler. The command line options let you control the way Xprofiler displays the profiled output.

When you enter an option, there must be a space between the option and its corresponding value. For example,

-e main

You can specify the following options from either the Xprofiler GUI or the command line. See “Specifying Command Line Options (from the GUI)” on page 11 for more information.

The Xprofiler command line options are:

<i>Table 1 (Page 1 of 3). Xprofiler Command Line Options</i>		
Use this flag:	To:	For example:
-b	Suppresses the printing of the field descriptions for the Flat Profile, Call Graph Profile, and Function Index reports when they are written to a file with the Save As option of the File menu.	To suppress printing of the field descriptions for the Flat Profile, Call Graph Profile, and Function Index reports in the saved file: file, ENTER <code>xprofiler -b a.out gmon.out</code>
-s	If multiple gmon.out files are specified when Xprofiler is started, produces the gmon.sum profile data file. The gmon.sum file represents the sum of the profile information in all the specified profile files. Note that if you specify a single gmon.out file, the gmon.sum file contains the same data as the gmon.out file.	To write the sum of the data from three profile data files, <i>gmon.out.1</i> , <i>gmon.out.2</i> , and <i>gmon.out.3</i> , into a file called <i>gmon.sum</i> : ENTER <code>xprofiler -s a.out gmon.out.1 gmon.out.2 gmon.out.3</code>
-z	Includes functions that have both zero CPU usage and no call counts in the Flat Profile, Call Graph Profile, and Function Index reports. A function will not have a call count if the file that contains its definition was not compiled with the -pg option, which is common with system library files.	To include all functions used by the application, in the Flat Profile, Call Graph Profile, and Function Index reports, that have zero CPU usage and no call counts: ENTER <code>xprofiler -z a.out gmon.out</code>
-a	Adds alternative paths to search for source code and library files, or changes the current path search order. When using this command line option, you can use the “at” symbol (@) to represent the default file path, in order to specify that other paths be searched before the default path.	To set the alternative file search path(s) so that Xprofiler searches <i>pathA</i> , the default path, then <i>pathB</i> : ENTER <code>xprofiler -a pathA:@.pathB</code>
-c	Loads the specified configuration file. If the -c option is used on the command line, the configuration file name specified with it will appear in the Configuration File (-c) : text field in the <i>Load Files Dialog</i> , and the Selection field of the <i>Load Configuration File Dialog</i> . When both the -c and -disp_max options are specified on the command line, the -disp_max option is ignored, but the value that was specified with it will appear in the Initial Display (-disp_max) : field in the <i>Load Files Dialog</i> , the next time it is opened.	To load the configuration file: ENTER <code>xprofiler a.out gmon.out -c config_file_name</code>
-disp_max	Sets the number of function boxes that Xprofiler initially displays in the function call tree. The value supplied with this flag can be any integer between 0 and 5,000. Xprofiler displays the function boxes for the most CPU-intensive functions through the number you specify. For instance, if you specify 50, Xprofiler displays the function boxes for the 50 functions in your program that consume the most CPU. After this, you can change the number of function boxes that are displayed via the Filter menu options. This flag has no effect on the content of any of the Xprofiler reports.	To display the function boxes for only 50 most CPU-intensive functions in the function call tree: ENTER <code>xprofiler -disp_max 50 a.out gmon.out</code>

Table 1 (Page 2 of 3). Xprofiler Command Line Options

Use this flag:	To:	For example:
-e	<p>De-emphasizes the general appearance of the function box(es) for the specified function(s) in the function call tree, and limits the number of entries for these function in the Call Graph Profile report. This also applies to the specified function's descendants, as long as they have not been called by non-specified functions.</p> <p>In the function call tree, the function box(es) for the specified function(s) appears greyed-out. Its size and the content of the label remain the same. This also applies to descendant functions, as long as they have not been called by non-specified functions.</p> <p>In the Call Graph Profile report, an entry for the specified function only appears where it is a child of another function, or as a parent of a function that also has at least one non-specified function as its parent. The information for this entry remains unchanged. Entries for descendants of the specified function do not appear unless they have been called by at least one non-specified function in the program.</p>	<p>To de-emphasize the appearance of the function boxes for <i>foo</i> and <i>bar</i>, as well as their qualifying descendants in the function call tree, and limit their entries in the Call Graph Profile report:</p> <p>ENTER <code>xprofiler -e foo -e bar a.out gmon.out</code></p>
-E	<p>Changes the general appearance and label information of the function box(es) for the specified function(s) in the function call tree. Also limits the number of entries for these functions in the Call Graph Profile report, and changes the CPU data associated with them. These results also apply to the specified function's descendants, as long as they have not been called by non-specified functions in the program.</p> <p>In the function call tree, the function box for the specified function appears greyed-out, and its size and shape also changes so that it appears as a square of the smallest allowable size. In addition, the CPU time shown in the function box label, appears as 0 (zero). The same applies to function boxes for descendant functions, as long as they have not been called by non-specified functions. This option also causes the CPU time spent by the specified function to be deducted from the left side CPU total in the label of the function box for each of the specified function's ancestors.</p> <p>In the Call Graph Profile report, an entry for the specified function only appears where it is a child of another function, or as a parent of a function that also has at least one non-specified function as its parent. When this is the case, the time in the <i>self</i> and <i>descendants</i> columns for this entry is set to 0 (zero). In addition, the amount of time that was in the <i>descendants</i> column for the specified function is subtracted from the time listed under the <i>descendants</i> column for the profiled function. As a result, be aware that the value listed in the <i>% time</i> column for most profiled functions in this report will change.</p>	<p>To change the display and label information for <i>foo</i> and <i>bar</i>, as well as their qualifying descendants in the function call tree, and limit their entries and data in the Call Graph Profile report:</p> <p>ENTER <code>xprofiler -E foo -E bar a.out gmon.out</code></p>
-f	<p>De-emphasizes the general appearance of all function boxes in the function call tree, <i>except</i> for that of the specified function(s) and its descendant(s). In addition, the number of entries in the Call Graph Profile report for the non-specified functions and non-descendant functions is limited. The -f flag overrides the -e flag.</p> <p>In the function call tree, all function boxes <i>except</i> for that of the specified function(s) and its descendant(s) appear greyed-out. The size of these boxes and the content of their labels remain the same. For the specified function(s), and its descendant(s), the appearance of the function boxes and labels remain the same.</p> <p>In the Call Graph Profile report, an entry for a non-specified or non-descendant function only appears where it is a parent or child of a specified function or one of its descendants. All information for this entry remains the same.</p>	<p>To de-emphasize the display of function boxes for all functions in the function call tree <i>except</i> for <i>foo</i>, <i>bar</i>, and their descendants, and limit their types if entries in the Call Graph Profile report:</p> <p>ENTER <code>xprofiler -f foo -f bar a.out gmon.out</code></p>

Table 1 (Page 3 of 3). Xprofiler Command Line Options

Use this flag:	To:	For example:
-F	<p>Changes the general appearance and label information of all function boxes in the function call tree <i>except</i> for that of the specified function(s) and its descendants. In addition, the number of entries in the Call Graph Profile report for the non-specified and non-descendant functions is limited, and the CPU data associated with them is changed. The -F flag overrides the -E flag.</p> <p>In the function call tree, all function boxes <i>except</i> for that of the specified function(s) and its descendant(s) appear greyed-out. The size and shape of these boxes also changes so that they appear as squares of the smallest allowable size. In addition, the CPU time shown in the function box label, appears as 0 (zero).</p> <p>In the Call Graph Profile report, an entry for a non-specified or non-descendant function only appears where it is a parent or child of a specified function or one of its descendants. When this is the case, the time in the <i>self</i> and <i>descendants</i> columns for this entry is set to 0 (zero). As a result, be aware that the value listed in the <i>% time</i> column for most profiled functions in this report will change.</p>	<p>To change the display and label information of the function boxes for all functions <i>except</i> the functions <i>foo</i> and <i>bar</i> and their descendants, and limit their types of entries and data in the Call Graph Profile:</p> <p>ENTER <code>xprofiler -F foo -F bar a.out gmon.out</code></p>
-L	<p>Sets the pathname for locating shared libraries. If you plan to specify multiple paths, use the <i>Set File Search Path</i> option of the File menu on the Xprofiler GUI. See "Setting the File Search Sequence" on page 14 for information.</p>	<p>To specify <code>/lib/profiled/libc.a:shr.o</code> as an alternate pathname for your shared libraries:</p> <p>ENTER <code>xprofiler -L /lib/profiled/libc.a:shr.o</code></p>

After you issue the **xprofiler** command, the Xprofiler main window appears, and displays your application's data.

Loading Files from the Xprofiler GUI

If you issue the **xprofiler** command without specifying an executable file, a profile data file, or options, you may do so from within the Xprofiler GUI. You use the *Load File* option of the File menu to do this.

When you issue the **xprofiler** command alone, the Xprofiler main window appears. Since you did not load an executable or specify a profile data file, the window will be empty.

If you issue the **xprofiler** command with the **-h** option only, Xprofiler displays the syntax for the command and then exits.



Figure 1. Xprofiler Main Window with No Executables Loaded

Select the File menu, and then the *Load File* option. The *Load Files Dialog* window appears.

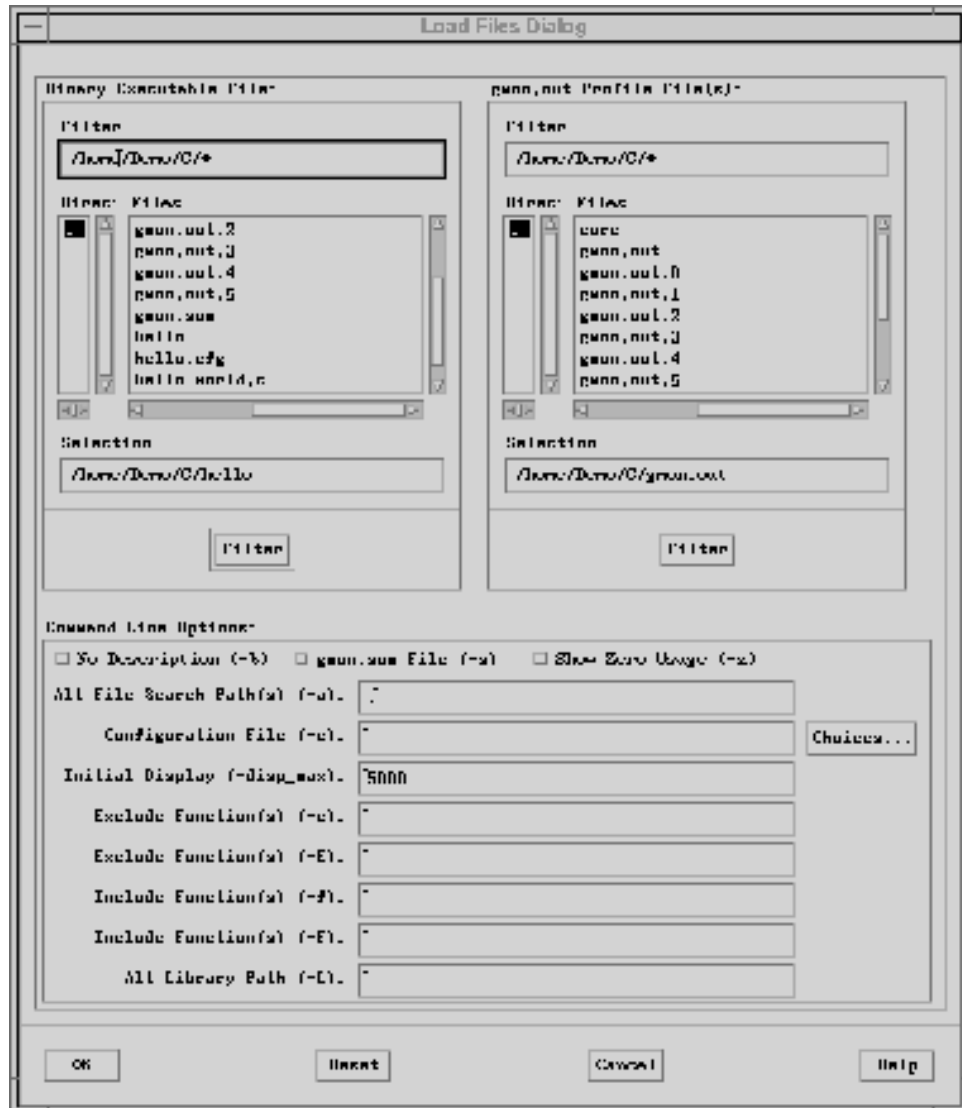


Figure 2. Load Files Dialog Window

The Load Files Dialog window lets you specify your application's executable, and its corresponding profile data (gmon.out) files. When you load a file, you can also specify the various command line options that let you control the way Xprofiler displays the profiled data.

To load the files for the application you wish to profile, you need to specify the:

- *Binary Executable* (required)
- *Profile Data File(s)* (required)
- *Command Line Options* (optional)

Specifying the Binary Executable

You specify the binary executable from the *Binary Executable File* of the *Load Files Dialog* window. The Binary Executable File area looks similar to this:

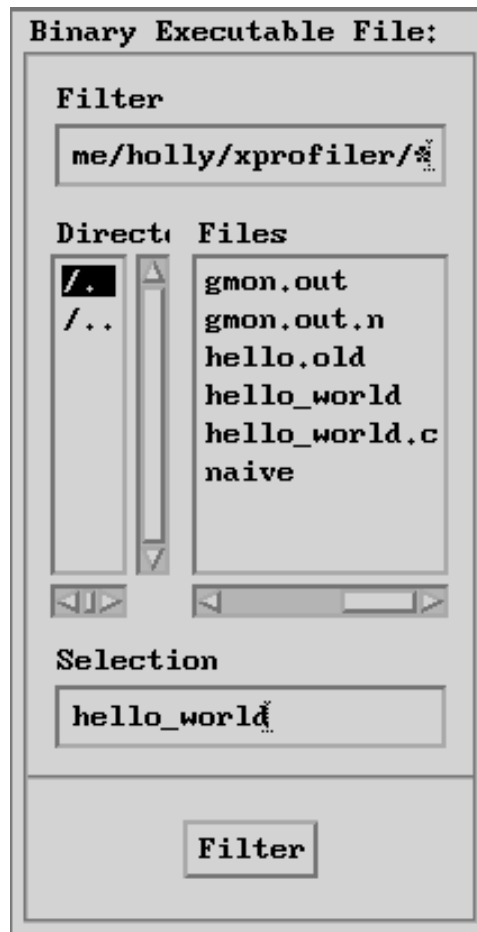


Figure 3. Binary Executable File Area

Use the scroll bars of the *Directories* and *Files* selection boxes to locate the executable file you wish to load. By default, all the files in the directory from which you invoked Xprofiler appear in the *Files* selection box. To select a file, click on it with the left mouse button.

To make locating your binary executable files easier, the Binary Executable File area includes a *Filter* button. Filtering lets you limit the files that are displayed in the *Files* selection box to those of a specific directory or of a specific type. For information on using the Filter, see “Using the Dialog Window Filters” on page 22.

Specifying the Profile Data File(s)

You specify the profile data file(s) from the *gmon.out Profile Data File(s)* area of the *Load Files Dialog* window. The *gmon.out Profile Data File(s)* area looks similar to this:

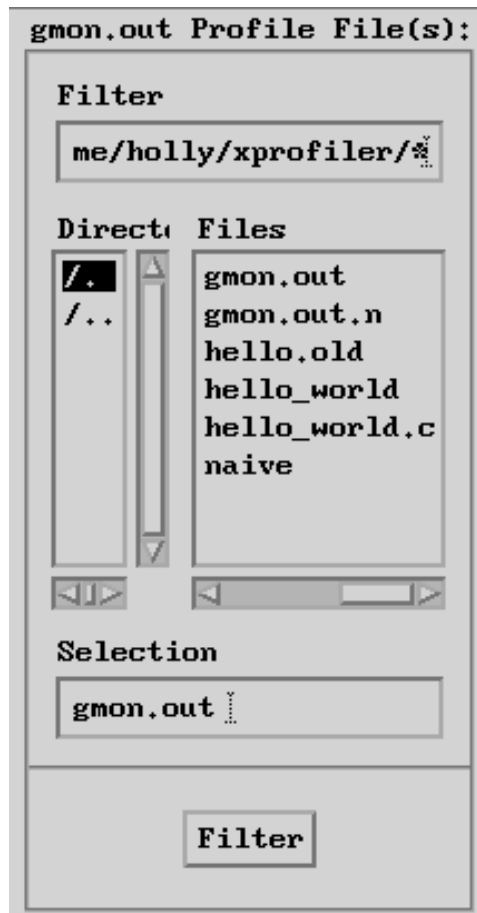


Figure 4. *gmon.out* Profile Data File(s) Area

When you started Xprofiler, with the **xprofiler** command, you were not required to indicate the name of the profile data file (which is probably why you are specifying it from the GUI). If you did not specify a profile data file, Xprofiler searches your directory for the presence of a file named *gmon.out* and, if found, places it in the *Selection* field of the *gmon.out* Profile File(s) area, as the default. Xprofiler then uses this file as input, even if it is not related to the binary executable file you specify. Since this will cause Xprofiler to display incorrect data, it is important that you enter the correct file into this field. So, if the profile data file you wish to use is named something other than what appears in the *Selection* field, you must replace it with the correct file name, as follows.

Use the scroll bars of the *Directories* and *Files* selection boxes to locate one or more of the profile data (*gmon.out*) files you wish to specify. The file you use does not have to be named *gmon.out*, and you may specify more than one profile data file. To select a file, click on it with the left mouse button. You can select multiple files by holding down the <Ctrl> key and clicking on each one with the left mouse button. To select multiple consecutive files, press and hold the left mouse button over the first file, and then drag the mouse over the other files. To de-select a file, press and hold the <Ctrl> key while clicking on the file.

To make locating your output files easier, the *gmon.out* Profile File(s) area includes a *Filter* button. Filtering lets you limit the files that get displayed in the *Files* selection box to those in a specific directory or of a specific type. For information on using the *Filter*, see "Using the Dialog Window Filters" on page 22.

Specifying Command Line Options (from the GUI)

You specify command line options from the *Command Line Options* area of the *Load Files Dialog* window. The Command Line Options area looks similar to this:



Figure 5. Command Line Options Area

You may specify one or more options as follows:

Use this flag:	To:	For example:
-b (button)	Suppresses the printing of the field descriptions for the Flat Profile, Call Graph Profile, and Function Index reports when they are written to a file with the Save As option of the File menu.	To suppress printing of the field descriptions for the Flat Profile, Call Graph Profile, and Function Index reports in the saved file, set the <i>-b</i> button to the pressed-in position.
-s (button)	If multiple <i>gmon.out</i> files are specified in the <i>gmon.out Profile File(s)</i> area, produces the <i>gmon.sum</i> profile data file. The <i>gmon.sum</i> file represents the sum of the profile information in all the specified profile files. Note that if you specify a single <i>gmon.out</i> file, the <i>gmon.sum</i> file contains the same data as the <i>gmon.out</i> file.	To write the sum of the data from three profile data files, <i>gmon.out.1</i> , <i>gmon.out.2</i> , and <i>gmon.out.3</i> , into a file called <i>gmon.sum</i> , set the <i>-s</i> button to the pressed-in position to activate this option.
-z (button)	Includes functions that have both zero CPU usage and no call counts in the Flat Profile, Call Graph Profile, and Function Index reports. A function will not have a call count if the file that contains its definition was not compiled with the -pg option, which is common with system library files.	To include all functions used by the application, in the Flat Profile, Call Graph Profile, and Function Index reports, that have zero CPU usage and no call counts, set the <i>-z</i> button to the pressed-in position to activate this option.
-a (field)	Adds alternative paths to search for source code and library files, or changes the current path search order. After clicking on the OK button, any modifications to this field are also made to the Enter Alt File Search Paths: field of the <i>Alt File Search Path Dialog</i> window. If both the <i>Load Files Dialog</i> window and the <i>Alt File Search Path Dialog</i> window are opened at the same time, when you make path changes in the <i>Alt File Search Path Dialog</i> and click the OK button, these changes are also made to the <i>Load Files Dialog</i> window. Also, when both of these windows are open concurrently, clicking on the OK or Cancel buttons in the <i>Load Files Dialog</i> causes both windows to close. If you wish to restore the Alt File Search Path(s) (-a): field to the same state as when the <i>Load Files Dialog</i> window was opened, click on the Reset button. You can use the “at” symbol (@) with this option to represent the default file path, in order to specify that other paths be searched before the default path.	To set the alternative file search path(s) so that Xprofiler searches <i>pathA</i> , the default path, then <i>pathB</i> , type <i>pathA:@:pathB</i> in the <i>Alt File Search Path(s) (-a)</i> field.

Table 2 (Page 2 of 4). Xprofiler GUI Command Line Options

Use this flag:	To:	For example:
-c (field)	<p>Loads the specified configuration file. If the -c option was used on the command line, or a configuration file had been previously loaded with the <i>Load Files Dialog</i> or <i>Load Configuration File Dialog</i> windows, the name of the most recently loaded file will appear in the Configuration File (-c): text field in the <i>Load Files Dialog</i>, as well as the Selection field of the <i>Load Configuration File Dialog</i>. If both the <i>Load Files Dialog</i> and <i>Load Configuration File Dialog</i> windows are open at the same time, when you specify a configuration file in the <i>Load Configuration File Dialog</i> and then click the OK button, the name of the specified file also appears in the <i>Load Files Dialog</i>. Also, when both of these windows are open concurrently, clicking on the OK or Cancel buttons in the <i>Load Files Dialog</i> causes both windows to close. When entries are made to both the Configuration File (-c): and Initial Display (-disp_max): fields in the <i>Load Files Dialog</i>, the value in the Initial Display (-disp_max): field is ignored, but is retained the next time this window is opened. If you wish to retrieve the file name that was in the Configuration File (-c): field when the <i>Load Files Dialog</i> window was opened, click on the Reset button.</p>	<p>To load the configuration file, type <i>gmon.out</i> in the <i>Configuration File (-c)</i> field.</p>
-disp_max (field)	<p>Sets the number of function boxes that Xprofiler initially displays in the function call tree. The value supplied with this flag can be any integer between 0 and 5,000. Xprofiler displays the function boxes for the most CPU-intensive functions through the number you specify. For instance, if you specify 50, Xprofiler displays the function boxes for the 50 functions in your program that consume the most CPU. After this, you can change the number of function boxes that are displayed via the <i>Filter</i> menu options. This flag has no effect on the content of any of the Xprofiler reports.</p>	<p>To display the function boxes for only 50 most CPU-intensive functions in the function call tree, type 50 in the <i>Init Display (-disp_max)</i> field</p>
-e (field)	<p>De-emphasizes the general appearance of the function box(es) for the specified function(s) in the function call tree, and limits the number of entries for these function in the <i>Call Graph Profile</i> report. This also applies to the specified function's descendants, as long as they have not been called by non-specified functions.</p> <p>In the function call tree, the function box(es) for the specified function(s) appears greyed-out. Its size and the content of the label remain the same. This also applies to descendant functions, as long as they have not been called by non-specified functions.</p> <p>In the <i>Call Graph Profile</i> report, an entry for the specified function only appears where it is a child of another function, or as a parent of a function that also has at least one non-specified function as its parent. The information for this entry remains unchanged. Entries for descendants of the specified function do not appear unless they have been called by at least one non-specified function in the program.</p>	<p>To de-emphasize the appearance of the function boxes for <i>foo</i> and <i>bar</i>, as well as their qualifying descendants in the function call tree, and limit their entries in the <i>Call Graph Profile</i> report, type <i>foo</i> and <i>bar</i> in the <i>Exclude Routines (-e)</i> field.</p> <p>You specify multiple functions by separating each one with a space.</p>

Table 2 (Page 3 of 4). Xprofiler GUI Command Line Options

Use this flag:	To:	For example:
-E (field)	<p>Changes the general appearance and label information of the function box(es) for the specified function(s) in the function call tree. Also limits the number of entries for these functions in the Call Graph Profile report, and changes the CPU data associated with them. These results also apply to the specified function's descendants, as long as they have not been called by non-specified functions in the program.</p> <p>In the function call tree, the function box for the specified function appears greyed-out, and its size and shape also changes so that it appears as a square of the smallest allowable size. In addition, the CPU time shown in the function box label, appears as 0 (zero). The same applies to function boxes for descendant functions, as long as they have not been called by non-specified functions. This option also causes the CPU time spent by the specified function to be deducted from the left side CPU total in the label of the function box for each of the specified function's ancestors.</p> <p>In the Call Graph Profile report, an entry for the specified function only appears where it is a child of another function, or as a parent of a function that also has at least one non-specified function as its parent. When this is the case, the time in the <i>self</i> and <i>descendants</i> columns for this entry is set to 0 (zero). In addition, the amount of time that was in the <i>descendants</i> column for the specified function is subtracted from the time listed under the <i>descendants</i> column for the profiled function. As a result, be aware that the value listed in the <i>% time</i> column for most profiled functions in this report will change.</p>	<p>To change the display and label information for <i>foo</i> and <i>bar</i>, as well as their qualifying descendants in the function call tree, and limit their entries and data in the Call Graph Profile report, type <i>foo</i> and <i>bar</i> in the <i>Exclude Routines (-E)</i> field.</p> <p>You specify multiple functions by separating each one with a space.</p>
-f (field)	<p>De-emphasizes the general appearance of all function boxes in the function call tree, <i>except</i> for that of the specified function(s) and its descendant(s). In addition, the number of entries in the Call Graph Profile report for the non-specified functions and non-descendant functions is limited. The -f flag overrides the -e flag.</p> <p>In the function call tree, all function boxes <i>except</i> for that of the specified function(s) and its descendant(s) appear greyed-out. The size of these boxes and the content of their labels remain the same. For the specified function(s), and its descendants, the appearance of the function boxes and labels remain the same.</p> <p>In the Call Graph Profile report, an entry for a non-specified or non-descendant function only appears where it is a parent or child of a specified function or one of its descendants. All information for this entry remains the same.</p>	<p>To de-emphasize the display of function boxes for all functions in the function call tree <i>except</i> for <i>foo</i>, <i>bar</i>, and their descendants, and limit their types if entries in the Call Graph Profile report, type <i>foo</i> and <i>bar</i> in the <i>Include Routines (-f)</i> field.</p> <p>You specify multiple functions by separating each one with a space.</p>

Table 2 (Page 4 of 4). Xprofiler GUI Command Line Options

Use this flag:	To:	For example:
-F (field)	<p>Changes the general appearance and label information of all function boxes in the function call tree <i>except</i> for that of the specified function(s) and its descendants. In addition, the number of entries in the Call Graph Profile report for the non-specified and non-descendant functions is limited, and the CPU data associated with them is changed. The -F flag overrides the -E flag.</p> <p>In the function call tree, all function boxes <i>except</i> for that of the specified function(s) and its descendant(s) appear greyed-out. The size and shape of these boxes changes so that they appear as squares of the smallest allowable size. In addition, the CPU time shown in the function box label, appears as 0 (zero).</p> <p>In the Call Graph Profile report, an entry for a non-specified or non-descendant function only appears where it is a parent or child of a specified function or one of its descendants. When this is the case, the time in the <i>self</i> and <i>descendants</i> columns for this entry is set to 0 (zero). As a result, be aware that the value listed in the <i>% time</i> column for most profiled functions in this report will change.</p>	<p>To change the display and label information of the function boxes for all functions <i>except</i> the functions <i>foo</i> and <i>bar</i> and their descendants, and limit their types of entries and data in the Call Graph Profile, type <i>foo</i> and <i>bar</i> in the <i>Include Routines (-F)</i> field.</p> <p>You specify multiple functions by separating each one with a space.</p>
-L (field)	Sets the alternate pathname for locating shared objects. If you plan to specify multiple paths, use the <i>Set File Search Path</i> option of the File menu on the Xprofiler GUI. See "Setting the File Search Sequence" on page 14 for information.	Type the alternate library pathname in this field.

Once you have specified the binary executable, the profile data file, and any command line options you wish to use, press the **OK** button to save the changes and close the dialog window. Xprofiler loads your application and displays its performance data.

Setting the File Search Sequence

You can specify where you want Xprofiler to look for your library files and source code files by using the *Set File Search Paths* option of the File menu. By default, Xprofiler searches the default paths first and then any alternative paths you specify.

Default Paths

For library files, Xprofiler uses the paths recorded in the specified gmon.out file(s). If you use the **-L** command line option, the path you specify with this option will be used instead of those in the gmon.out file.

Note: **-L** allows only one path to be specified and you can use this option only once.

For source code files, the paths recorded in the specified a.out file are used.

Alternative Paths

These are the paths you specify with the *Set File Search Paths* option of the File menu.

For library files, if everything else failed, the search will be extended to the path(s) specified in the LIBPATH environment variable.

To specify alternative path(s), do the following:

- Select the *File* menu, and then the *Set File Search Paths* option. The *Alt File Search Path Dialog* window appears.

- Enter the name of the path in the *Enter Alt File Search Path(s)* text field. You can specify more than one path by separating each with colon (:) or a space.

Notes:

1. You can use the “at” symbol (@) with this option to represent the default file path, in order to specify that other paths be searched before the default path. For example, to set the alternative file search path(s) so that Xprofiler searches *pathA*, the default path, then *pathB*, type *pathA:@:pathB* in the *Alt File Search Path(s) (-a)* field.
 2. If @ is used in the alternative search path, the two buttons in the Alt File Search Path Dialog will be greyed out, and have no effect on the search order.
- Click on the OK button. The paths you specified in the text field become the alternative paths.

Changing the Search Sequence: You can change the order of the search sequence for library files and source code files via the *Set File Search Paths* option of the File menu. To change the search sequence, do the following:

1. Select the *File* menu, and then the *Set File Search Paths* option. The *Alt File Search Path Dialog* window appears.
2. To indicate the file search should use alternative paths first, click on the *Check alternative path(s) first* button.
3. Click on the OK button. This changes the search sequence to the:
 - a. Alternative paths
 - b. Default paths
 - c. Path(s) specified in LIBPATH (library files only)

To return the search sequence back to its default order, repeat steps 1 through 3, but in step 2 above, click on the *Check default path(s) first* button. When the action is confirmed (by clicking on the OK button), the search sequence will start with the default paths again.

Keep in mind that if a file is found in one of the alternative paths or a path in LIBPATH, this path now becomes the default path for this file throughout the current Xprofiler session (until you exit this Xprofiler session or load a new set of data).

Understanding the Xprofiler Display

The primary difference between Xprofiler and the UNIX **gprof** command is that Xprofiler gives you a graphical picture of your application's CPU consumption in addition to textual data. This allows you to focus quickly on the areas of your application that consume a disproportionate amount of CPU.

Xprofiler displays your profiled program in a single main window. It uses several types of graphic images to represent the relevant parts of your program. Functions appear as solid green boxes (called *function boxes*), and the calls between them appear as blue arrows (called *call arcs*). The function boxes and call arcs that belong to each library within your application appear within a fenced-in area called

a *cluster box*. The way that functions, calls, and library clusters are depicted is discussed later.

The Xprofiler Main Window

The Xprofiler main window contains a graphical representation of the functions and calls within your application as well as their inter-relationships. It provides six menus, including one for online help.

The Xprofiler main window looks similar to this when an application has been loaded:

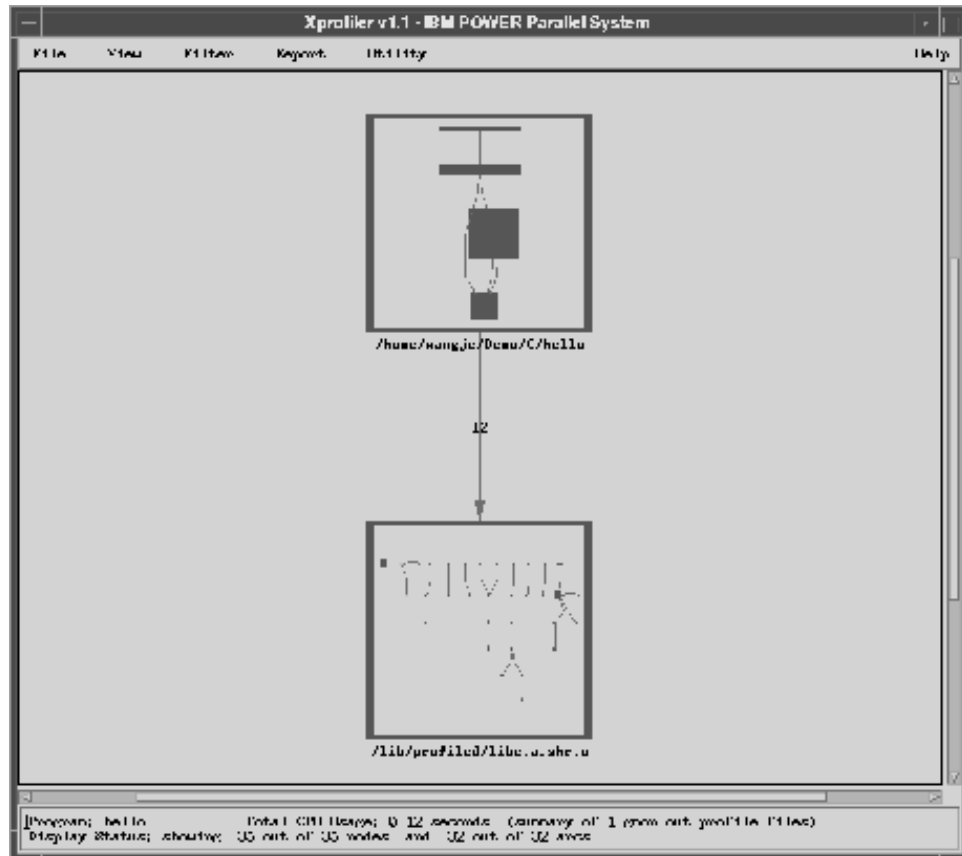


Figure 6. Sample Xprofiler Main Window

In the main window, Xprofiler displays the *function call tree*. The function call tree displays the function boxes, arcs, and cluster boxes that represent the functions within your application.

Note: When Xprofiler first opens, by default, the function boxes for your application will be *clustered* by library, as in the example above. This means that a cluster box appears around each library, and the function boxes and arcs within the cluster box are reduced in size. If you wish to see more detail, you need to uncluster the functions. To do this, select the *File* menu and then the *Uncluster Functions* option.

Xprofiler Main Menus

Along the upper portion of the main window is the *menu bar*. The left side of the menu bar contains the Xprofiler menus that let you work with your profiled data. On the right side of the menu bar, there is a Help menu for accessing online help.

The Xprofiler menus are described below:

File Menu: The File menu lets you specify the executable (a.out) files and profile data (gmon.out) files that Xprofiler will use. It also lets you control how your files are accessed and saved.

View Menu: The View menu lets you focus on specific portions of the function call tree in order to get a better view of the application's critical areas.

Filter Menu: The Filter menu lets you add, remove, and change specific parts of the function call tree. By controlling what Xprofiler displays, you can focus on the objects that are most important to you.

Report Menu: The Report menu provides several types of profiled data in a textual and tabular format. In addition to presenting the profiled data, the options of the Report menu let you:

- Display textual data
- Save it to a file
- View the corresponding source code
- Locate the corresponding function box or call arc in the function call tree.

Utility Menu: The Utility menu contains one option; *Locate Function By Name*, which lets you highlight a particular function in the function call tree.

Xprofiler Hidden Menus

Function Menu: The Function menu lets you perform a number of operations for any of the functions shown in the function call tree. You can access statistical data, look at source code, and control which functions get displayed.

The Function menu is not visible from the Xprofiler window. You access it by clicking on the function box of the function in which you are interested with your right mouse button. By doing this, you not only bring up the Function menu, but you select this function as well. Then, when you select actions from the Function menu, they are applied to this function.

Arc Menu: The Arc menu lets you locate the caller and callee functions for a particular *call arc*. A call arc is the representation of a call between two functions within the function call tree.

The Arc menu is not visible from the Xprofiler window. You access it by clicking on the call arc in which you are interested with your right mouse button. By doing this, you not only bring up the Arc menu, but you select that call arc as well. Then, when you perform actions with the Arc menu, they are applied to that call arc.

Cluster Node Menu: The Cluster Node menu lets you control the way your libraries are displayed by Xprofiler. In order to access the Cluster Node Menu, the function boxes, in the function call tree, must first be clustered by library. See "Clustering Libraries Together" on page 36 for information about clustering and

unclustering the function boxes of your application. When the function call tree is clustered, all the function boxes within each library appear within a *cluster box*.

The Cluster Node menu is not visible from the Xprofiler window. You access it by clicking on the edge of the cluster box in which you are interested with your right mouse button. By doing this, you not only bring up the Cluster Node menu, but you select that cluster as well. Then, when you perform actions with the Cluster Node menu, they are applied to the functions within that library cluster.

Display Status Field

At the bottom of the Xprofiler window is a single field that tells you:

- The name of your application
- The number of *gmon.out* files used in this session.
- The total amount of CPU used by the application.
- The number of functions and calls in your application, and how many of these are currently displayed

How Functions are Depicted

Functions are represented by green, solid-filled boxes in the function call tree. The size and shape of each function box indicates its CPU usage. The height of each function box represents the amount of CPU time it spent on executing itself. The width of each function box represents the amount of CPU time it spent executing itself, plus its descendant functions.

This type of representation is known as *summary mode*. In summary mode, the size and shape of each function box is determined by the total CPU time of multiple *gmon.out* files used on that function alone, and the total time used by the function and its descendant functions. A function box that is wide and flat represents a function that uses a relatively small amount of CPU on itself (it spends most of its time on its descendants). On the other hand, the function box for a function that spends most of its time executing only itself will be roughly square-shaped.

Functions can also be represented in *average mode*. In average mode, the size and shape of each function box is determined by the average CPU time used on that function alone, among all loaded *gmon.out* files, and the standard deviation of CPU time for that function among all loaded *gmon.out* files. The height of each function node represents the average CPU time, among all the input *gmon.out* files, used on the function itself. The width of each node represents the standard deviation of CPU time, among the *gmon.out* files, used on the function itself. The average mode representation is available only when more than one *gmon.out* file is entered. For more information on summary mode and average mode, see “Controlling the Representation of the Function Call Tree” on page 29.

Under each function box in the function call tree is a label that contains the name of the function and related CPU usage data. For information on the function box labels, see “Getting Basic Data” on page 41.

The example below shows the function boxes for two functions, *sub1* and *printf*, as they would appear in the Xprofiler display.

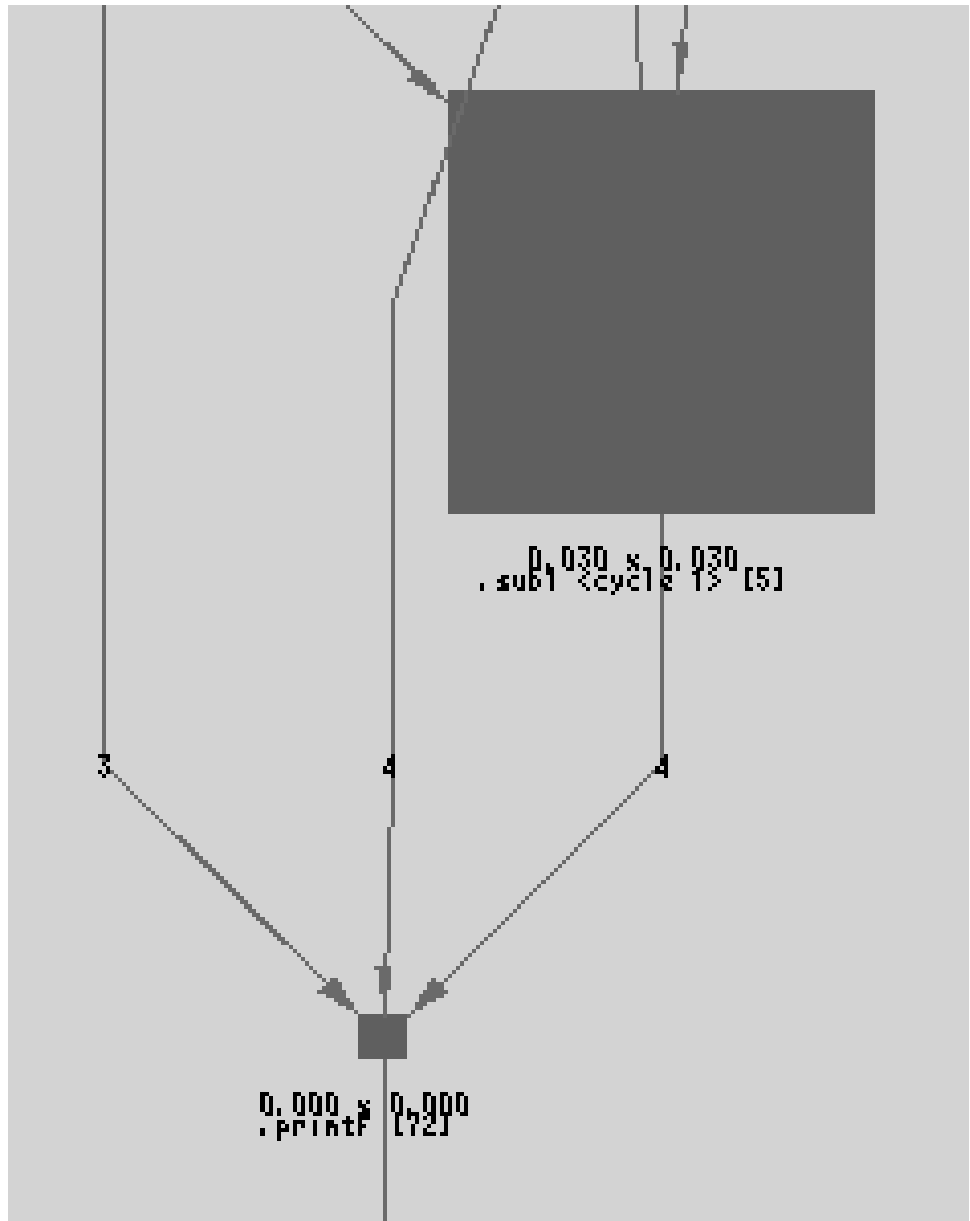


Figure 7. Example of Function Boxes and Arcs in Xprofiler Display

Each function box has its own menu. To access it, place your mouse cursor over the function box of the function in which you are interested, and press the right mouse button. Each function also has an information box that lets you get basic performance numbers quickly. To access the information box, place your mouse cursor over the function box of the function in which you are interested, and press the left mouse button.

How Calls Between Functions are Depicted

The calls made between each of the functions in the function call tree are represented by blue arrows extending between their corresponding function boxes. These lines are called *call arcs*. Each call arc appears as a solid blue line between two functions. The arrowhead indicates the direction of the call; the function represented by the function box it points to is the one that receives the call. The

function making the call is known as the *caller*, while the function receiving the call is known as the *callee*.

Each call arc includes a numerical label that tells you how many calls were exchanged between the two corresponding functions.

Figure 7 on page 19, above, shows several call arcs. For the call arc that connects *sub1* and *printf*, *sub1* is the caller and *printf* is the callee. The label tells you that *sub1* called *printf* four times.

Note that each call arc has its own menu that lets you locate the function boxes for its caller and callee functions. To access it, place your mouse cursor over the call arc for the call in which you are interested, and press the right mouse button. Each call arc also has an information box that shows you the number of times the caller function called the callee function. To access the information box, place your mouse cursor over the call arc for the call in which you are interested, and press the left mouse button.

How Library Clusters are Depicted

Xprofiler lets you collect the function boxes and call arcs that belong to each of your shared libraries into *cluster boxes*. Figure 6 on page 16 shows an example of an Xprofiler display in which the libraries are clustered.

Since there will be a box around each library, the individual function boxes and call arcs will be difficult to see. If you want to see more detail, you will need to uncluster the function boxes. To do this, select the Filter menu and then the *Uncluster Functions* option.

When viewing function boxes within a cluster box, note that the size of each function box is relative to those of the other functions within the same library cluster. On the other hand, when all the libraries are unclustered, the size of each function box is relative to all the functions in the application (as shown in the function call tree).

Each library cluster has its own menu that lets you manipulate the cluster box. To access it, place your mouse cursor over the edge of the cluster box you are interested in, and press the right mouse button. Each cluster also has an information box that shows you the name of the library and the total CPU usage (in seconds) consumed by the functions within it. To access the information box, place your mouse cursor over the edge of the cluster box you are interested in and press the left mouse button.

Using the Xprofiler Graphical User Interface

The Xprofiler graphical user interface (GUI) contains features and buttons that are common throughout the interface. This section explains how to use some of these common elements.

Using the Dialog Window Buttons

The buttons that appear on the Xprofiler dialog windows are explained below:

OK

Saves the changes, executes the action, and closes the dialog window.

Apply

Saves the changes, executes the action, but leaves the dialog window open.

Reset

Restores the fields of the dialog window to their original values (at the time you opened it), and keeps the dialog window open.

Cancel

Ignores changes and closes the dialog window.

Help

Brings up the Xprofiler online help.

Filter

Executes filtering criteria provided by you in the dialog window.

Using the Search Engine

Some of the Xprofiler windows that are accessible via the Report and Function menus provide a *Search Engine* field that lets you search for a specific string. In the Search Engine field, which is located at the bottom of these windows, you type the string in which you are interested. The first row that contains the string you specified is highlighted.

To use the Search Engine to search for a string:

1. Click on the *Search Engine* field with the left mouse button. The Search Engine field highlights to show that it is selected.
2. Type the string you are looking for in the Search Engine field.

Extended regular expressions are allowed. For more information, see the explanation of the **regcmp** and **regcomp** commands in *AIX Technical Reference, Volume 2: Base Operating System and Extensions (SC23-2615)*.

3. Press the <Enter> key. The first row, in the Report or Source Code window, that contains the string you specified is highlighted. Each time you press the <Enter> key, a subsequent occurrence of the string is highlighted. The Search wraps back to the first occurrence after all other occurrences have been highlighted.

Using the Save Dialog Windows

A *Save* dialog window appears when you choose the *Save As* option from any of the Xprofiler reports windows or from the File Menu. It allows you to save the data you see, in the report window that is currently open, to a file.

Note: If you choose the *Save As* option from one of the reports windows, the title of the dialog window included the name of the report (for example *Save Flat Profile*).

To save the current report data to a file using the Save dialog window:

1. Specify the file into which the data should be placed. You can specify either an existing file or a new one. If you specify an existing file, be aware that Xprofiler replaces the file altogether (instead of appending to the existing data). To replace an existing file, use the scroll bars of the *Directories* and the *Files* selection boxes to locate the file you want. To make locating your file easier, you can also use the *Filter* button (see “Using the Dialog Window Filters” for more information). To specify a new file, type its name in the *Selection* field.
2. Click on the **OK** button. A file containing the profiled window data appears in the directory you specified, under the name you gave it.

Using the Dialog Window Filters

Many of the Xprofiler dialog windows include a *Filter* button. The use of the Xprofiler Filter function follows the Motif standard. To use the Filter:

1. In the *Filter* field, specify the directory that contains the files that you wish to see in the *Files* selection box. You may specify an asterisk (*) as a wildcard.
2. Click on the *Filter* button with the left mouse button. The list of files in the Files selection box is updated to reflect your selection.

Using the Radio/Toggle Buttons and Sliders

Many of the dialog windows include buttons and sliders that allow you to select options and specify values.

Using the Buttons

Aside from push-buttons, the Xprofiler dialog windows also use radio buttons and toggle buttons. Radio buttons let you select one item from a set of items, while toggle buttons let you activate or de-activate a single item.

In the example below, the *Screen Dump Options Dialog* window uses both radio buttons and toggle buttons. For instance, under *Output To*, there are two radio buttons; **File** and **Printer**. You must select one or the other, but you cannot select both. Just above the *Default Directory* field, notice two toggle buttons; **Enable Landscape** and **Annotate Output**. By using the toggle buttons, you can activate either one or both of these options.

To select (or activate) an option with a radio or toggle button, set the button to the pressed-in position by clicking on it. When a button is pressed-in, it appears shaded. Under *Output To*, in the example below, **File** is selected and **Printer** is not.

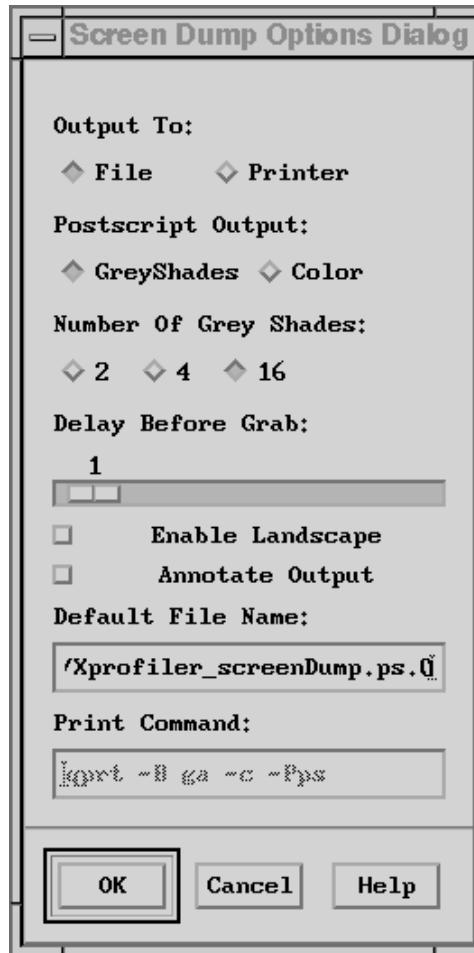


Figure 8. Example Showing Radio Buttons, Toggle Buttons, and Slider

Using the Sliders

Several of the Xprofiler dialog windows include *sliders* that let you specify a numerical value. In the example above, the *Delay Before Grab* slider lets you specify the number of seconds you want to pass before the screen image is actually captured.

Place your mouse cursor over the slider. Press and hold the left mouse button while moving the slider horizontally in either direction. The number above the slider changes as you move it, and indicates the number selected. Once the slider is at the setting you want, release the mouse button.

If the number of selectable values on the slider is high, you may want to have finer control over the placement of the slider. If so, click on the slider and then use the arrow keys on your keyboard to place it.

Manipulating the Function Call Tree

Xprofiler lets you look at your profiled data a number of ways, depending on what you want to see. It provides:

- Navigation that lets you move around the display and zoom in on specific areas
- Display options, based on your personal viewing preferences.
- Filtering capability, to let you include and exclude certain objects from the display

Zooming In on the Function Call Tree

Xprofiler lets you magnify specific areas of the window to get a better view of your profiled data. The View menu includes three options that let you do this:

- Overview
- Zoom In
- Zoom Out

To resize a specific area of the display, you can use either the *Overview* or *Zoom In* options of the View menu. To magnify an area with the *Overview* option:

1. Select the *View* menu, and then the *Overview* option. The Overview Window appears, as in the example below.

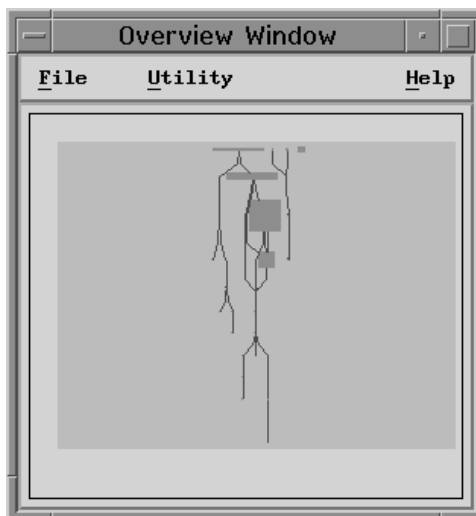


Figure 9. The Overview Window

The Overview Window contains a miniature view of the function call tree, just as it appears in the Xprofiler main display. When you open the Overview Window, the light blue highlight area represents the current view of the main window.

You control the size and placement of the highlight area with your mouse. Depending on where you place your mouse over the highlight area, your cursor changes to indicate the operation you can perform. Here is an explanation of the cursor images, and what they indicate to you:

When your cursor appears as two crossed arrows, it means that by pressing and holding your mouse button, you can control where the box is placed.

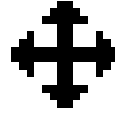


Figure 10. Cursor when movement of highlight box is under mouse control

When your cursor appears as a line with an arrow perpendicular to it, it means that your mouse button has grabbed the edge of the highlight area, and you now have the ability to resize it. By pressing and holding your mouse button, and dragging it in or out, you can increase or decrease the size of the box. Notice that as you move the edge in or out, the size of the entire highlight area changes.



Figure 11. Cursor when edge of highlight box is under mouse control

When your cursor appears as a right angle with an arrow pointing into it, it means that your mouse button has grabbed the corner of the highlight area and you now have the ability to resize it. By pressing and holding your mouse button, and dragging it diagonally up or down, you can increase or decrease the size of the box. Notice that as you move the corner up or down, the size of the entire highlight area changes.



Figure 12. Cursor when corner of highlight box is under mouse control

3. Place your mouse cursor within the light blue highlight area. Notice that the cursor changes to four crossed arrows. This indicates that your cursor has control over the placement of the box.

4. Move your cursor over one of the four corners of the highlight area. Notice that the cursor changes to a right angle with an arrow pointing into it. This indicates that you now have control over the corner of the highlight area.
5. Press and hold the left mouse button, and drag the corner of the box diagonally inward. The box shrinks as you move it. The example below shows the highlight area reduced in size, with only a few function boxes visible within it.

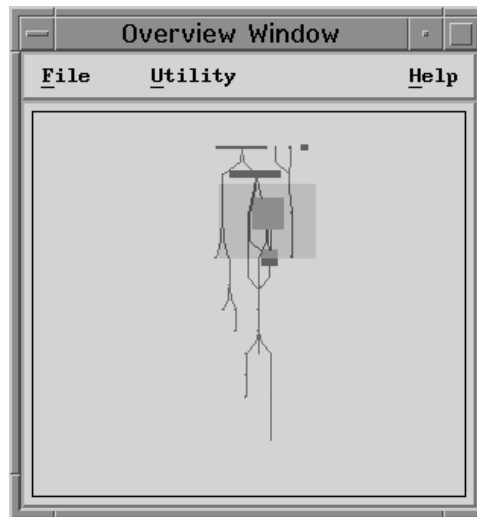


Figure 13. Highlight Area Reduced in Size

6. When the highlight area is as small as you would like it (or the smallest allowable size), release the mouse button. The Xprofiler main display redraws itself to contain only the functions within the highlight area, and in the same proportions. This has the effect of magnifying the items within the highlight area.
7. Place your mouse cursor over the highlight area. Your cursor again changes to four crossed arrows to indicate that you have control over the placement of the highlight area. Press and hold the left mouse button and drag the highlight area to the area of the Xprofiler display you want to magnify.
8. Release the mouse button. The Xprofiler main display now contains the items in which you are interested.

The example below shows the Xprofiler main display with the area, indicated by the highlight area in Figure 13, magnified. Note that the performance data, on the label of each function box, is now visible.

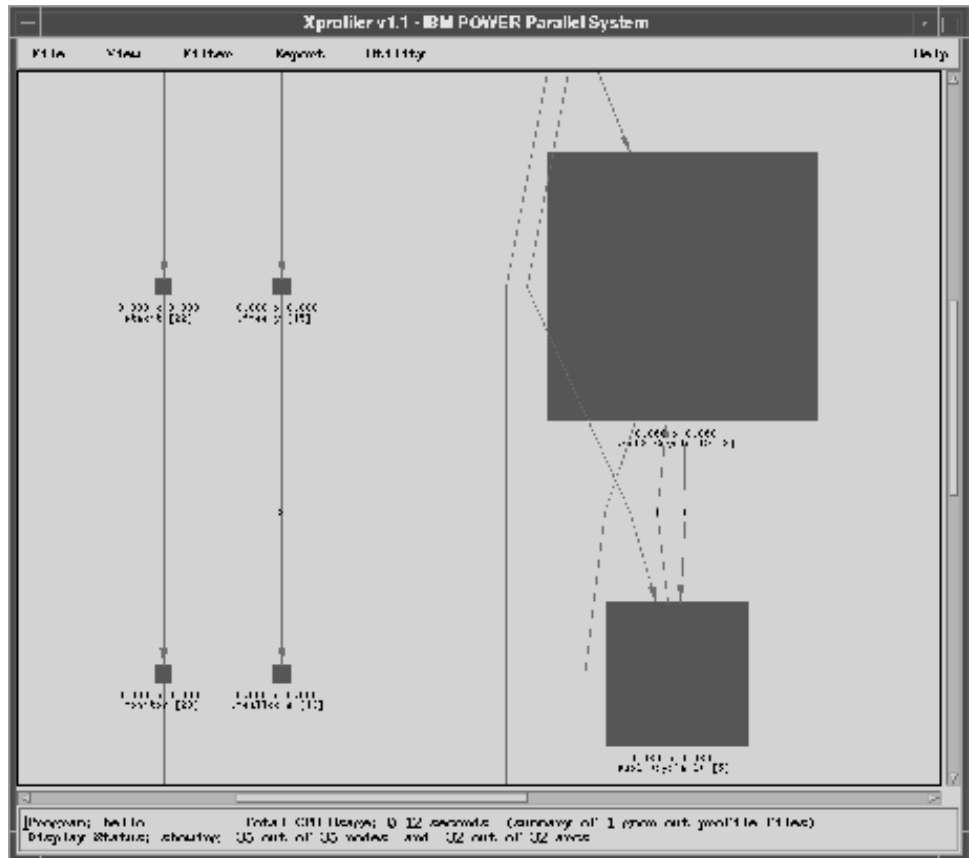


Figure 14. Magnified View of Xprofiler Display

To use the *Zoom In* option of the View menu to magnify a specific area of the function call tree:

1. Select the *View* menu, and then the *Zoom In* option. Once you select the *Zoom In* option, your cursor changes to a hand to indicate that your selection is active.
2. Place the mouse cursor in the upper left hand corner of the area you would like to view more closely. Press and hold the left mouse button while dragging it diagonally downward, until the rubber band box surrounds the area you want to view.
3. Release the mouse button. Xprofiler redraws the display so that the area of the function call tree you selected is centered and sized proportionately, according to the size of the rubber band box you drew.

To get an even closer view of the area you selected, choose the *Zoom In* option again and follow the steps above.

There may be times when you are looking at the function call tree too closely. The *Zoom Out* option lets you widen the view of the function call tree, as if you were taking a few steps back from a painting on a wall.

The *Zoom Out* option is most useful after using the *Zoom In* or *Overview* options to magnify an area of the function call tree. By default, the Xprofiler main window is completely zoomed out (it shows you the entire function call tree). The *Zoom Out* option helps you return the main window to this state.

To zoom out:

1. Select the *View* menu, and then the *Zoom Out* option. Once you select the *Zoom Out* option, your cursor changes to a hand to indicate that your selection is active.
2. Place the mouse cursor in the upper left hand corner of the area you want to view. Press and hold the left mouse button while dragging it diagonally downward, until the rubberband box surrounds the area you want to widen.
3. Release the mouse button. Xprofiler redraws the display so that the area of the function call tree you selected is centered and sized proportionately according to the size of the rubber band box that you drew.

To further step back from the area you selected, choose the *Zoom Out* option again, and follow the steps above.

Controlling How the Display is Updated

The Utility menu of the Overview window lets you choose the mode in which the display is updated. The default is the *Immediate Update* option, which causes the display to show you the items in the highlight area as you are moving it around. The *Delayed Update* option, on the other hand, causes the display to be updated only when you have moved the highlight area over the area in which you are interested, and released the mouse button. The *Immediate Update* option only applies to what you see when you move the highlight area; it has no effect on the resizing of items in highlight area, which is always delayed.

Other Viewing Options

Xprofiler lets you change the way it displays the function call tree, based on your own personal preferences.

Controlling the Graphic Style of the Function Call Tree

You can choose between 2-D and 3-D function boxes in the function call tree. The default style is 2-D, but you can change this to 3-D. To do this:

1. Select the *View* menu, and then the *3-D Image* option. The function boxes in the function call tree now appear in 3-D format.

Controlling the Orientation of the Function Call Tree

You can choose to have Xprofiler display the function call tree in either *Top-to-Bottom* or *Left-to-Right* format. The default is *Top-to-Bottom*. If you would rather see the function call tree displayed in *Left-to-Right* format:

1. Select the *View* menu, and then the *Layout: Left→Right* option. The function call tree appears in *Left-to-Right* format.

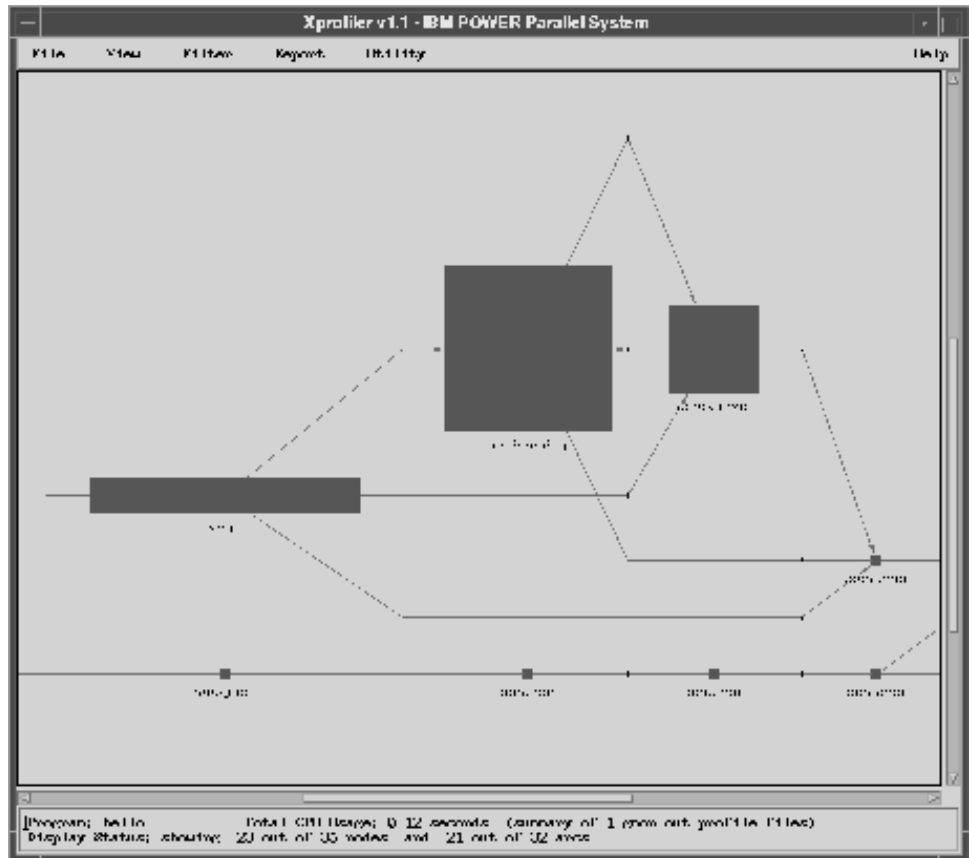


Figure 15. Left-to-Right Format

Controlling the Representation of the Function Call Tree

You can choose to have Xprofiler represent the function call tree in either *summary mode* or *average mode*.

When you select the *Summary Mode* option of the View menu, the size and shape of each function box is determined by the total CPU time of multiple *gmon.out* files used on that function alone, and the total time used by the function and its descendant functions. The height of each function node represents the total CPU time used on the function itself. The width of each node represents the total CPU time used on the function and its descendant functions. When the display is in summary mode, the *Summary Mode* option is greyed out and the *Average Mode* option is activated.

When you select the *Average Mode* option of the View menu, the size and shape of each function box is determined by the average CPU time used on that function alone, among all loaded *gmon.out* files, and the standard deviation of CPU time for that function among all loaded *gmon.out* files. The height of each function node represents the average CPU time, among all the input *gmon.out* files, used on the function itself. The width of each node represents the standard deviation of CPU time, among the *gmon.out* files, used on the function itself.

The purpose of average mode is to reveal workload balancing problems when an application is involved with multiple *gmon.out* files. In general, a function node with

large standard deviation has a wide width, and a node with small standard deviation has a slim width.

Note: Both summary mode and average mode only affect the appearance of the function call tree and the labels associated with it. All the performance data in Xprofiler reports and code displays are always summary data. If only one *gmon.out* file is given, both Summary Mode and Average Mode will be greyed out, and the display is always in Summary Mode.

Filtering What You See

When Xprofiler first opens, the entire function call tree appears in the main window. This includes the function boxes and call arcs that belong to your executable as well as the shared libraries that it utilizes. At times, you may want to simplify what you see in the main window, and there are a number of ways to do this.

Note: Filtering options of the Filter menu let you change the appearance of the function call tree only. The performance data contained in the reports (via the Reports menu) is not affected.

Restoring the Status of the Function Call Tree

Xprofiler allows you to undo operations that involve adding or removing nodes and arcs from the function call tree. When you undo an operation, you reverse the effect of any operation which adds or removes function boxes or call arcs to the function call tree. When you select the *Undo* option, the function call tree is returned to its appearance just prior to the performance of the add or remove operation. To undo an operation:

1. Select the *Filter* menu, and then the *Undo* option. The function call tree is returned to its appearance just prior to the performance of the add or remove operation.

Whenever you invoke the *Undo* option, the function call tree loses its zoom focus and zooms all the way out to reveal the entire function call tree in the main display. When you start Xprofiler, the *Undo* option is greyed out. It is activated only after an add or remove operation involving the function call tree takes place. After you undo an operation, the option greys out again until the next add or remove operation takes place.

The options that activate the *Undo* option include:

- In the main *File* menu:
 - Load Configuration
- In the main *Filter* menu:
 - Show Entire Call Tree
 - Hide All Library Calls
 - Add Library Calls
 - Filter by Function Names
 - Filter by CPU Time
 - Filter by Call Counts
- In the *Function* menu:
 - Immediate Parents

- All Paths To
- Immediate Children
- All Paths From
- All Functions on The Cycle
- Show This Function Only
- Hide This Function
- Hide Descendant Functions
- Hide This & Descendant Functions

If a dialog, like the Load Configuration Dialog or the Filter by CPU Time Dialog, is invoked and then cancelled immediately, the status of the *Undo* option is not affected. Once the option is available, it stays that way until you invoke it, or a new set of files is loaded into Xprofiler through the Load Files Dialog.

Displaying the Entire Function Call Tree

When you first open Xprofiler, by default, all the function boxes and call arcs of your executable and its shared libraries appear in the main window. After that, you may choose to filter out specific items from the window. However, there may be times when you want to see the entire function call tree again, without having to reload your application. To do this:

1. Select the *Filter* menu, and then the *Show Entire Call Tree* option. Xprofiler erases whatever is currently displayed in the main window and replaces it with the entire function call tree.

Excluding and Including Specific Objects

There are a number of ways that Xprofiler lets you control the items that get displayed in the main window. For the most part, you will want to include or exclude certain objects so that you can more easily focus on the things that are of most interest to you.

Filtering Shared Library Functions: In most cases, your application will call functions that are within shared libraries. By default, these shared libraries will appear in the Xprofiler window along with your executable. As a result, the window may get crowded and obscure the items that you really want to see. If this is the case, you may want to filter the shared libraries from the display. To do this:

1. Select the *Filter* menu, and then the *Remove All Library Calls* option.

The shared library function boxes disappear from the function call tree, leaving only the function boxes of your executable file visible.

If you removed the library calls from the display, you may want to add all or some of them back. To do this:

1. Select the *File* menu, and then the *Add Library Calls* option

The function boxes once again appear with the function call tree. Note, however, that all of the shared library calls that were in the initial function call tree may not be added back. This is because the *Add Library Calls* option only adds back in the function boxes for the library functions that were called by functions that are currently displayed in the Xprofiler window.

There may be times when you want to add only specific function boxes back into the display. To do this:

1. Select the *Filter* menu, and then the *Filter by Function Names* option. The *Filter By Function Names Dialog* window appears.
2. From the Filter By Function Names Dialog window, select the *add these functions to graph* button, and then type the name of the function you want to add in the *Enter function name* field. If you enter more than one function name, you must separate them by putting a blank space between each function name string.

If there are multiple functions in your program that include the string you enter in their names, the filter applies to each one. For example, say you specified *sub*, and *printf*, and your program also included functions named *sub1*, *psub1* and *printf*. The *sub*, *sub1*, *psub1*, *printf*, and *printf* functions would all be added to the graph.

3. Click on the **OK** button. The function box(es) appears in the Xprofiler display with the function call tree.

Filtering by Function Characteristics: The Filter menu of Xprofiler offers you three options that allow you to add or subtract function boxes from the main window, based on specific characteristics. The options are:

- Filter by Function Names
- Filter by CPU Time
- Filter by Call Counts

Each one of these options uses a different dialog window to let you specify the criteria by which you want to include or exclude function boxes from the window.

To filter by function names:

1. Select the *Filter* menu.
2. Select the *Filter by Function Names* option. The *Filter By Function Names Dialog* window appears.

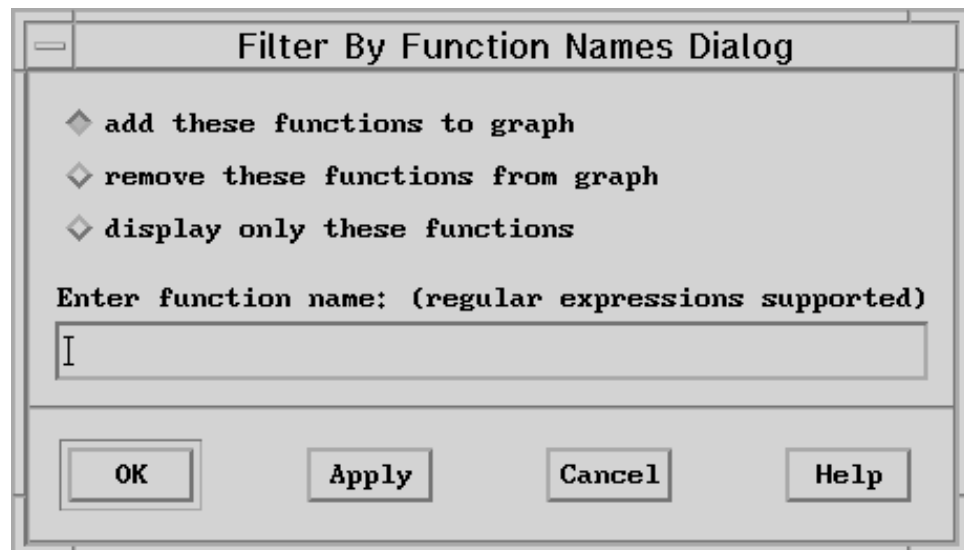


Figure 16. Filter By Function Names Dialog window

3. The Filter By Function Names Dialog window includes three options:

- *add these functions to graph*
- *remove these functions from the graph*
- *display only these functions*

From the Filter By Function Names Dialog window, select the option you want, and then type the name of the function(s) to which you want it applied in the *Enter function name* field. For example, say you wanted to remove function box for a function called *printf*, from the main window. You would click on the *remove this function from the graph* button and type *printf* in the *Enter function name* field.

You can enter more than one function name in this field. If there are multiple functions in your program that include the string you enter in their names, the filter will apply to each one. For example, say you specified *sub* and *print*, and your program also included functions named *sub1*, *psub1*, and *printf*. The option you chose would be applied to the *sub*, *sub1*, *psub1*, *print*, and *printf* functions.

4. Click on the **OK** button. The contents of the function call tree now reflect the filtering options you specified.

To filter by CPU time:

1. Select the *Filter* menu.
2. Select the *Filter by CPU Time* option. The *Filter By CPU Time Dialog* window appears.

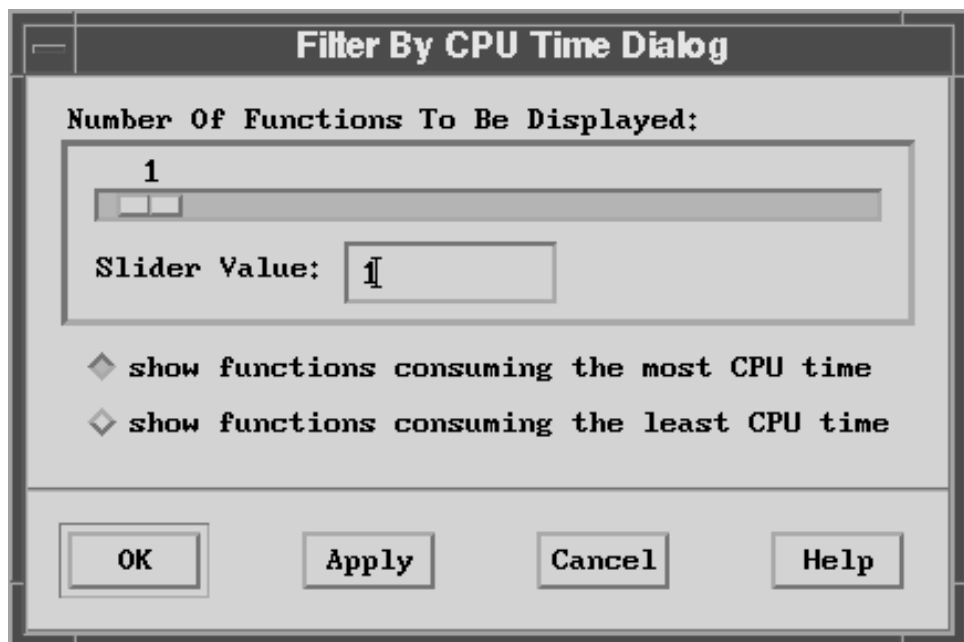


Figure 17. Filter By CPU Time Dialog window

3. The Filter By CPU Time Dialog window includes two options:

- *show functions consuming the most CPU time*
- *show functions consuming the least CPU time*

4. Click on the button for the option you want (*show functions consuming the most CPU time* is the default).
5. Select the number of functions to which you want it applied (1 is the default). You can move the slider in the *Functions* bar until the desired number appears, or you can enter the number in the *Slider Value* field. The slider and *Slider Value* field are synchronized so when the slider is updated, the text field value is updated also. If you enter a value in the text field, the slider is updated to that value when you click on the **Apply** button or the **OK** button.

For example, if you wanted to display the function boxes for the 10 functions in your application that consumed the most CPU, you would select the *show functions consuming the most CPU* button, and specify 10 with the slider or enter the value 10 in the text field.

6. Click on the **Apply** button to show the changes to the function call tree without closing the dialog. Click on the **OK** button to show the changes and close the dialog.

To filter by call counts:

1. Select the *Filter* menu.
2. Select the *Filter by Call Counts* option. The *Filter By Call Counts Dialog* window appears.

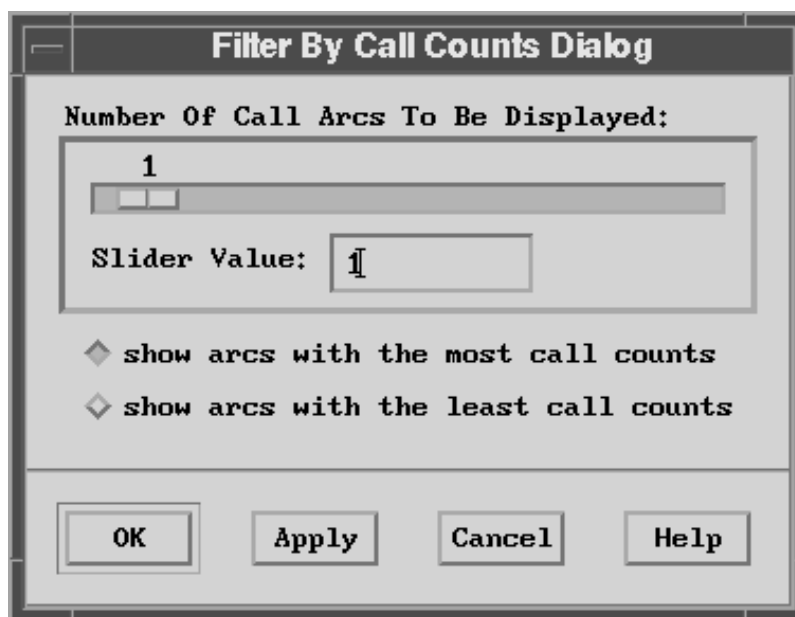


Figure 18. Filter By Call Counts Dialog window

3. The Filter By Call Counts Dialog window includes two options:
 - *show arcs with the most call counts*
 - *show arcs with the least call counts*
4. Click on the button for the option you want (*show arcs with the most call counts* is the default).
5. Select the number of call arcs to which you want it applied (1 is the default). You can move the slider in the *Call Arcs* bar until the desired number appears, or you can enter the number in the *Slider Value* field. The slider and *Slider Value* field are synchronized so when the slider is updated, the text field value

is updated also. If you enter a value in the text field, the slider is updated to that value when you click on the **Apply** button or the **OK** button.

For example, if you wanted to display the 10 call arcs in your application that represented the least number of calls, you would select the *show arcs with the least call counts* button, and specify 10 with the slider or enter the value 10 in the text field.

6. Click on the **Apply** button to show the changes to the function call tree without closing the dialog. Click on the **OK** button to show the changes and close the dialog.

Including and Excluding Parent and Child Functions: When tuning the performance of your application, you will want to know which functions consumed the most CPU time, and then you will need to ask several questions in order to understand their behavior:

- Where did each function spend most of the CPU time?
- What other functions called this function? Were the calls made directly or indirectly?
- What other functions did this function call? Were the calls made directly or indirectly?

Once you understand how these functions behave, and are able to improve their performance, you can move on to analyzing the functions that consume less CPU.

When your application is large, the function call tree will also be large. As a result, the functions that are the most CPU-intensive may be difficult to see in the function call tree. To get around this, use the *Filter by CPU* option of the Filter menu, which lets you display only the function boxes for the functions that consume the most CPU time. Once you've done this, the Function menu for each function lets you add the parent and descendant function boxes to the function call tree. By doing this, you create a smaller, simpler function call tree that displays the function boxes associated with most CPU-intensive area of the application.

A *child* function is one that is directly called by the function of interest. To see only the function boxes for the function of interest and its child functions:

1. Place your mouse cursor over the function box in which you are interested, and press the right mouse button. The Function menu appears.
2. From the Function menu, select the *Immediate Children* option, and then the *Show Child Functions Only* option.

Xprofiler erases the current display and replaces it with only the function boxes for the function you chose, plus its child functions.

A *parent* function is one that directly calls the function of interest. To see only the function box for the function of interest and its parent functions:

1. Place your mouse cursor over the function box in which you are interested, and press the right mouse button. The Function menu appears.
2. From the Function menu, select the *Immediate Parents* option, and then the *Show Parent Functions Only* option.

Xprofiler erases the current display and replaces it with only the function boxes for the function you chose, plus its parent functions.

There may be times when you may want to see the function boxes for both the parent and child functions of the function in which you are interested, without erasing the rest of the function call tree. This is especially true if you chose to display the function boxes for two or more of the most CPU-intensive functions with the *Filter by CPU* option of the Filter menu (you suspect that more than one function is consuming too much CPU). To do this:

1. Place your mouse cursor over the function box in which you are interested, and press the right mouse button. The Function menu appears.
2. From the Function menu, select the *Immediate Parents* option, and then the *Add Parent Functions to Tree* option.

Xprofiler leaves the current display as it is, but adds the parent function boxes.

3. Place your mouse cursor over the same function box and press the right mouse button. The Function menu appears.
4. From the Function menu, select the *Immediate Children* option, and then the *Add Child Functions to Tree* option.

Xprofiler leaves the current display as it is, but now adds the child function boxes in addition to the parents.

Clustering Libraries Together

When you first bring up the Xprofiler window, by default, the function boxes of your executable, and the libraries associated with it, are clustered. Since Xprofiler shrinks the call tree of each library when it places it in a cluster, you will need to uncluster the function boxes if you want to look closely at a specific function box label.

It is important to understand that you can see significantly more detail per function, when your display is in the unclustered or *expanded* state, than when it is in the clustered or *collapsed* state. So, depending on what you want to do, you will need to cluster or uncluster (collapse or expand) the display.

There may be times when the Xprofiler window is visually crowded. This is especially true if your application calls functions that are within shared libraries; function boxes representing your executable functions as well as the functions of the shared libraries get displayed. As a result, you may want to organize what you see in the Xprofiler window so you can focus on the areas that are most important to you. One way you can do this is to collect all the function boxes of each library into a single area, known as a library *cluster*.

When you choose to cluster your libraries, Xprofiler gathers all the functions for each one into a single area, and draws a green box around them. This is known as a *cluster box*. The name of the library also appears below the box.

The function boxes in the application shown in Figure 6 on page 16 have been clustered. Note that one cluster box holds the function boxes associated with the executable *hello_world*, while the other cluster box holds the function boxes of the shared library */lib/profiled/libc.a:shr.o*.

The example below shows the same application, *hello_world*, with its function boxes unclustered. Now that the function boxes of your executable and its shared libraries are displayed together, which means their relationships are more apparent.

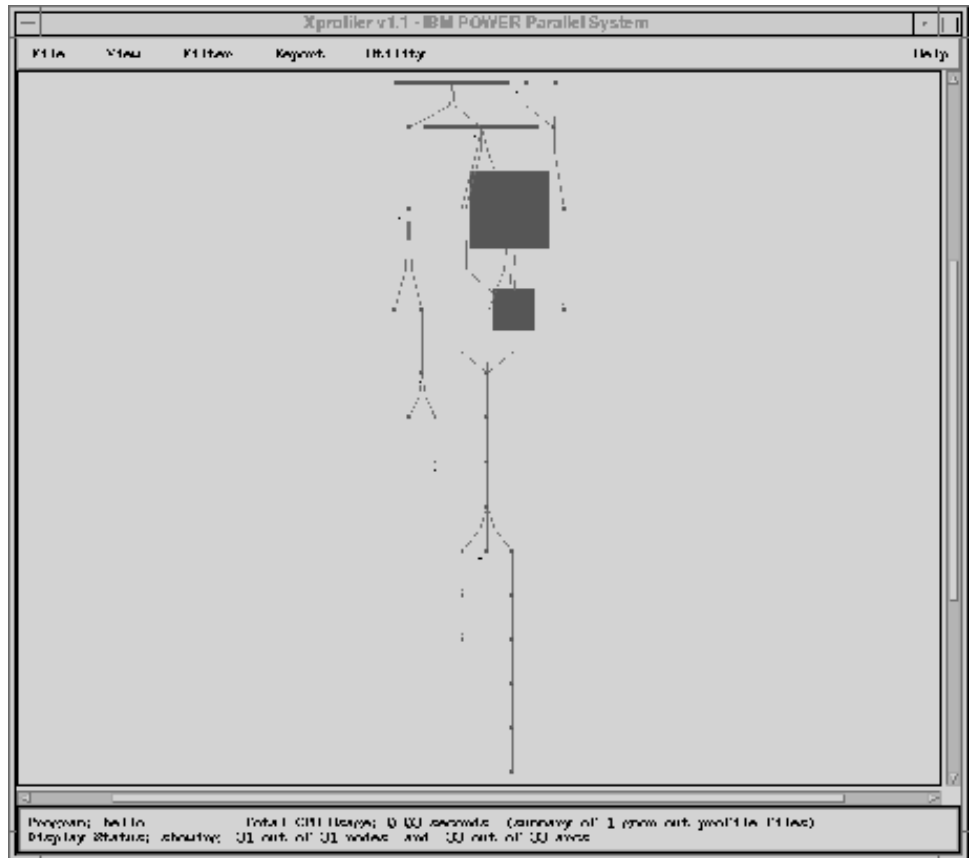


Figure 19. Xprofiler Window with Function Boxes Unclustered

Clustering Functions

If the functions within your application are unclustered, you can use an option of the Filter menu to cluster them.

1. Select the Filter menu, and then the *Cluster Functions by Library* option. The libraries within your application appear within their respective cluster boxes.

Once you cluster the functions in your application, as shown in Figure 6 on page 16, you can further reduce the size (also referred to as *collapse*) of each cluster box. To do this:

1. Place your mouse cursor over the edge of the cluster box and press the right mouse button. The Cluster Node Menu appears.
2. Select the *Collapse Cluster Node* option. The cluster box, and its contents, now appear as a small solid green box. In the example below, the library */lib/profiled/libc.a:shr.o* is collapsed.

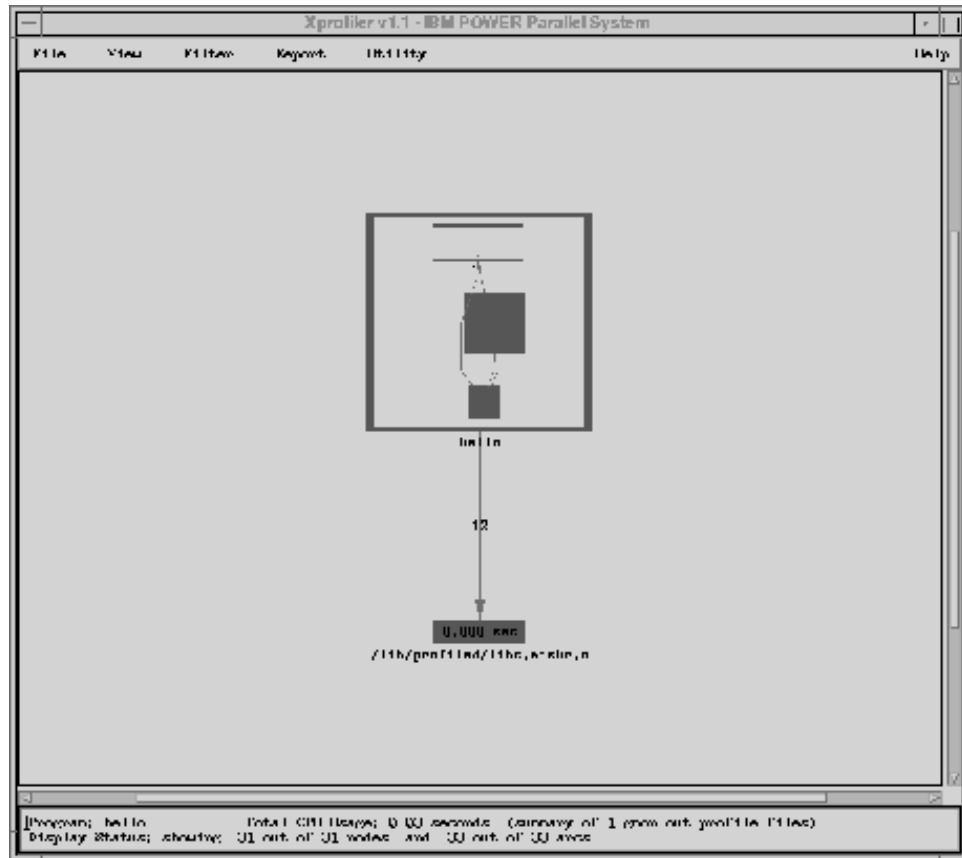


Figure 20. Xprofiler Window with One Library Cluster Box Collapsed

To return the cluster box to its original condition (*expand* it):

1. Place your mouse cursor over the collapsed cluster box and press the right mouse button. The Cluster Node Menu appears.
2. Select the *Expand Cluster Node* option. The cluster box, and its contents, appear once again.

Unclustering Functions

If the functions within your application are clustered, you can use an option of the Filter menu to uncluster them.

1. Select the Filter menu, and then the *Uncluster Functions* option. The cluster boxes disappear and the functions boxes of each library expand to fill the Xprofiler window.

If your functions have been clustered, you may want to remove one or more (but not all) cluster boxes. For example, say you wanted to uncluster only the functions of your executable, but keep its shared libraries within their cluster boxes. You would:

1. Place your mouse cursor over the edge of the cluster box that contains the executable and press the right mouse button. The Cluster Node Menu appears.
2. Select the *Remove Cluster Box* option. The cluster box disappears and the function boxes and call arcs, that represent the executable functions, now appear in full detail. The function boxes and call arcs of the shared libraries

remain within their cluster boxes, which now appear smaller to make room for the unclustered executable function boxes.

The example below shows the executable, *hello_world* with its cluster box removed. Its shared library, */lib/profiled/libc.a:shr.o*, remains within its cluster box.

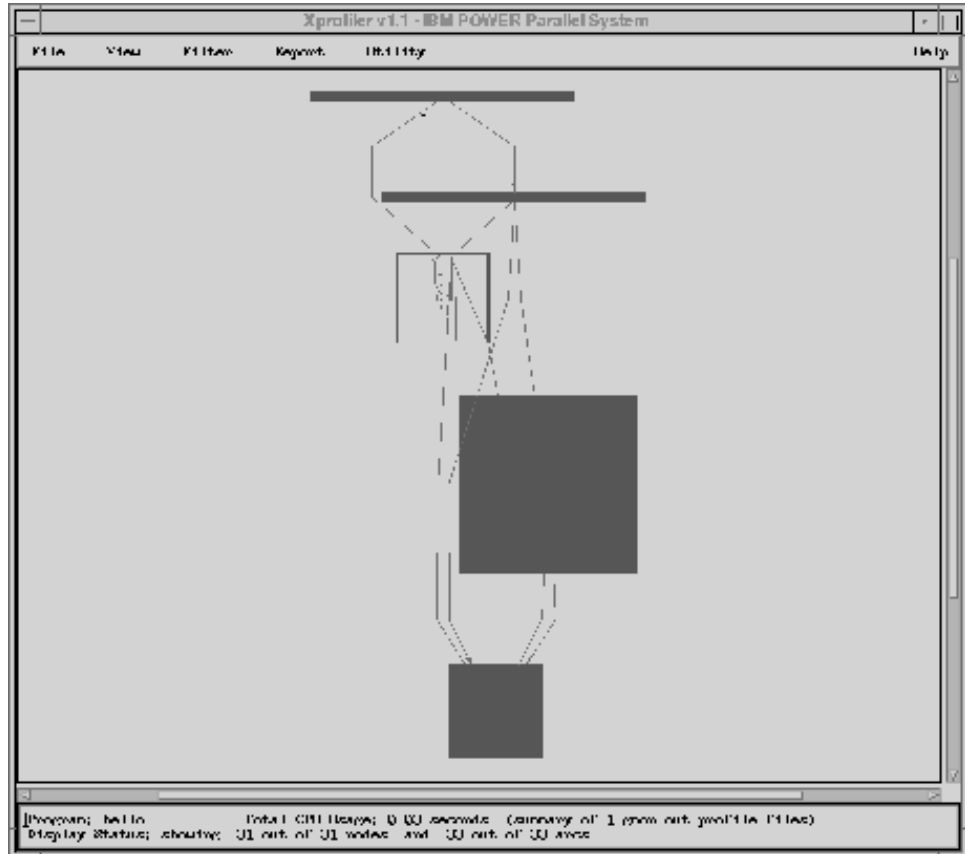


Figure 21. Xprofiler Window with One Library Cluster Box Removed

Locating Specific Objects in the Function Call Tree

If you are interested in one or more specific functions in a complex program, you may need help locating their corresponding function boxes in the function call tree.

If you would like to locate a single function, and you know its name, you can use the *Locate Function By Name* option of the Utility menu. To locate a function by name:

1. Select the Utility menu, and then the *Locate Function By Name* option. The *Search By Function Name Dialog* window appears.
2. Type the name of the function you wish to locate in the *Enter Function Name* field. The function name you type here must be a continuous string (it cannot include blanks).
3. Press the **OK** or **Apply** button. The corresponding function box is highlighted (its color changes to red) in the function call tree and Xprofiler zooms in on its location.

To display the function call tree in full detail again, go to the View menu and use the *Overview* option.

There may also be times when you want to see only the function boxes for the functions you are concerned with, plus other specific functions that are related to it. For instance, suppose you want to see all the functions that directly called the function in which you are interested. It might not be easy to pick out these function boxes when you view the entire call tree, so you would want to display them, plus the function of interest, alone.

Each function has its own menu, called a *Function menu*. Via the Function menu, you can choose to see the following for the function in which you are interested:

- Parent functions (functions that directly call the function of interest)
- Child functions (functions that are directly called by the function of interest)
- Ancestor functions (functions that can call, directly or indirectly, the function of interest)
- Descendant functions (functions that can be called, directly or indirectly, by the function of interest)
- Functions that belong to the same cycle

When you use these options, Xprofiler erases the current display and replaces it with only the function boxes for the function of interest and all the functions of the type you specified.

Locating and Displaying Parent Functions

A *parent* is any function that directly calls the function in which you are interested. To locate the parent function boxes of the function in which you are interested:

1. Click on the function box of interest with the right mouse button. The Function menu appears.
2. From the Function menu, select *Immediate Parents*→*Show Parent Functions Only*. Xprofiler redraws the display to show you only the function boxes for the function of interest and its parent functions.

Locating and Displaying Child Functions

A *child* is any function that is directly called by the function in which you are interested. To locate the child functions boxes for the function in which you are interested:

1. Click on the function box of interest with the right mouse button. The Function menu appears.
2. From the Function menu, select *Immediate Children*→*Show Child Functions Only*. Xprofiler redraws the display to show you only the function boxes for the function of interest and its child functions.

Locating and Displaying Ancestor Functions

An *ancestor* is any function that can call, directly or indirectly, the function in which you are interested. To locate the ancestor functions:

1. Click on the function box of interest with the right mouse button. The Function menu appears.

2. From the Function menu, select *All Paths To→Show Ancestor Functions Only*. Xprofiler redraws the display to show you only the function boxes for the function of interest and its ancestor functions.

Locating and Displaying Descendant Functions

A *descendant* is any function that can be called, directly or indirectly, by the function in which you are interested. To locate the descendant functions (all the functions that the function of interest can reach, directly or indirectly):

1. Click on the function box of interest with the right mouse button. The Function menu appears.
2. From the Function menu, select *All Paths From→Show Descendant Functions Only*. Xprofiler redraws the display to show you only the function boxes for the function of interest and its descendant functions.

Locating and Displaying Functions on a Cycle

To locate the functions that are on the same cycle as the function you are interested in:

1. Click on the function of interest with the right mouse button. The Function menu appears.
2. From the Function menu, select *All Functions on the Cycle→Show Cycle Functions Only*. Xprofiler redraws the display to show you only the function of interest and all the other functions on its cycle.

Getting Performance Data for Your Application

With Xprofiler, you can get performance data for your application on a number of levels, and in a number of ways. You can easily view data pertaining to a single function, or you can use the reports provided to get information on your application as a whole.

Getting Basic Data

Xprofiler makes it easy to get data on specific items in the function call tree. Once you've located the item you are interested in, you can get data a number of ways. If you are having trouble locating a function in the function call tree, see "Locating Specific Objects in the Function Call Tree" on page 39.

Basic Function Data

Below each function box in the function call tree is a label that contains basic performance data. The example below shows the function box for the function *main*, and its label.

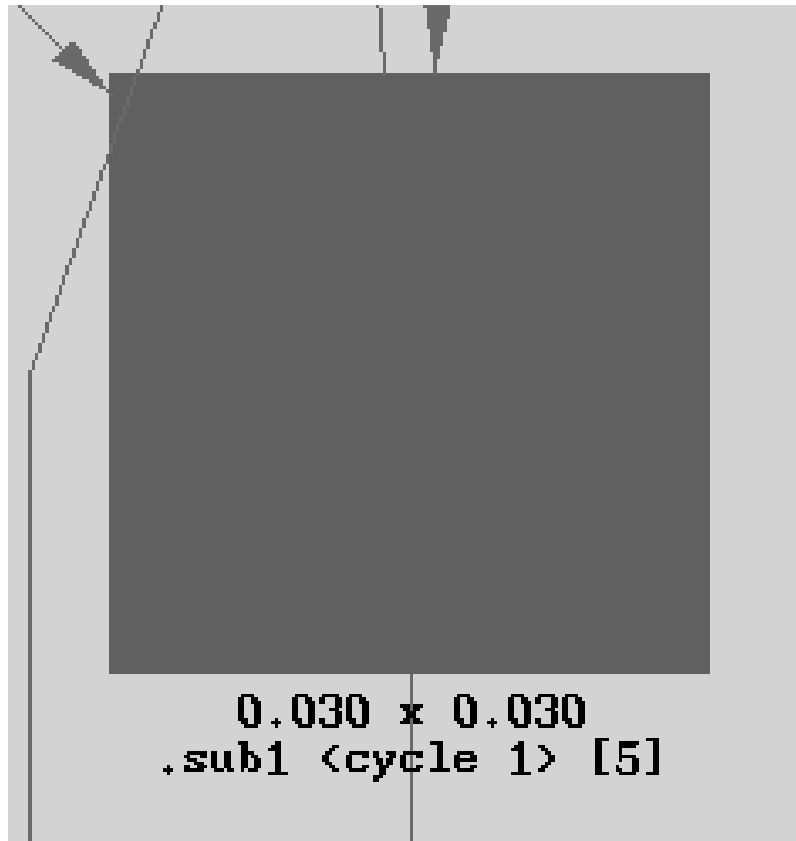


Figure 22. Example of a Function Box Label

The label contains the name of the function, its associated cycle, if any, and its index. In the example above, the name of the function is *sub1*. It is associated with cycle 1, and its index is 5. Also, depending on whether the function call tree is viewed in summary mode or average mode, the label will contain the information listed below. See “Controlling the Representation of the Function Call Tree” on page 29 for more about summary mode and average mode.

- In summary mode:
 - The total amount of CPU time (in seconds) this function spent on itself plus the amount of CPU time it spent on its descendants (the number on the left of the *x*). In the example above, the function *sub1* spent .030 seconds on itself, plus its descendants.
 - The amount of CPU time (in seconds) this function spent only on itself (the number on the right of the *x*). In the example above, the function *sub1* spent .030 seconds on itself.
- In average mode:
 - The average CPU time (in seconds), among all the input *gmon.out* files, used on the function itself.
 - The standard deviation of CPU time (in seconds), among all the input *gmon.out* files, used on the function itself.

Since labels are not always visible in the Xprofiler window when it is fully zoomed out, you may need to zoom in on it in order to see the labels. See “Zooming In on the Function Call Tree” on page 24 for information on how to do this.

Basic Call Data

Call arc labels appear over each call arc. The label shows you the number of calls that were made between the two functions (from caller to callee). For example:

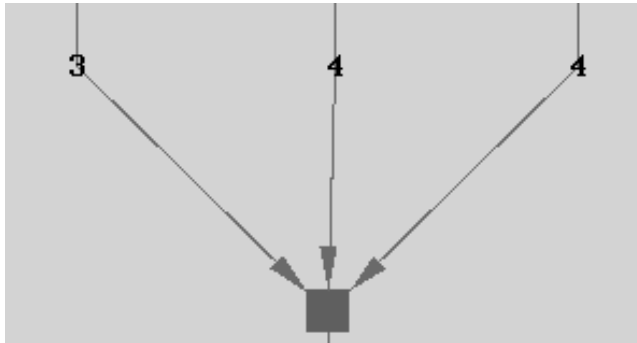


Figure 23. Example of a call arc label

In order to see a call arc label, you will probably need to zoom in on it. See “Zooming In on the Function Call Tree” on page 24 for information on how to do this.

Basic Cluster Data

Cluster box labels tell you the name of the library that is represented by that cluster. If it is a shared library, the label shows its full pathname.

Information Boxes

For each function box, call arc, and cluster box, there is a corresponding information box that you can access with your mouse. It gives you the same basic data that appears on the label. This is useful when the Xprofiler display is fully zoomed out and the labels are not visible. To access the information box, click on the function box, call arc, or cluster box (place it over the edge of the box) with the left mouse button. The information box appears.

For a function, the information box contains:

- The name of the function, its associated cycle, if any, and its index.
- The amount of CPU used by this function. There are two values supplied in this field. The first is the amount of CPU time spent on this function plus the time spent on its descendants. The second value represents the amount of CPU time this function spent only on itself.
- The number of times this function was called (by itself or any other function in the application).

For a call, the information box contains:

- The caller and callee functions (their names) and their corresponding indexes.
- The number of times the caller function called the callee.

For cluster, the information box contains:

- The name of the library
- The total CPU usage (in seconds) consumed by the functions within it.

Function Menu Statistics Report Option

You can get performance statistics for a single function via the *Statistics Report* option of the Function menu. It lets you see data on the CPU usage and call counts of the selected function. If you are using more than one gmon.out file, this option breaks down the statistics per each gmon.out file you use.

When you select the *Statistics Report* menu option, the *Function Level Statistics Report* window appears, as in the example below.

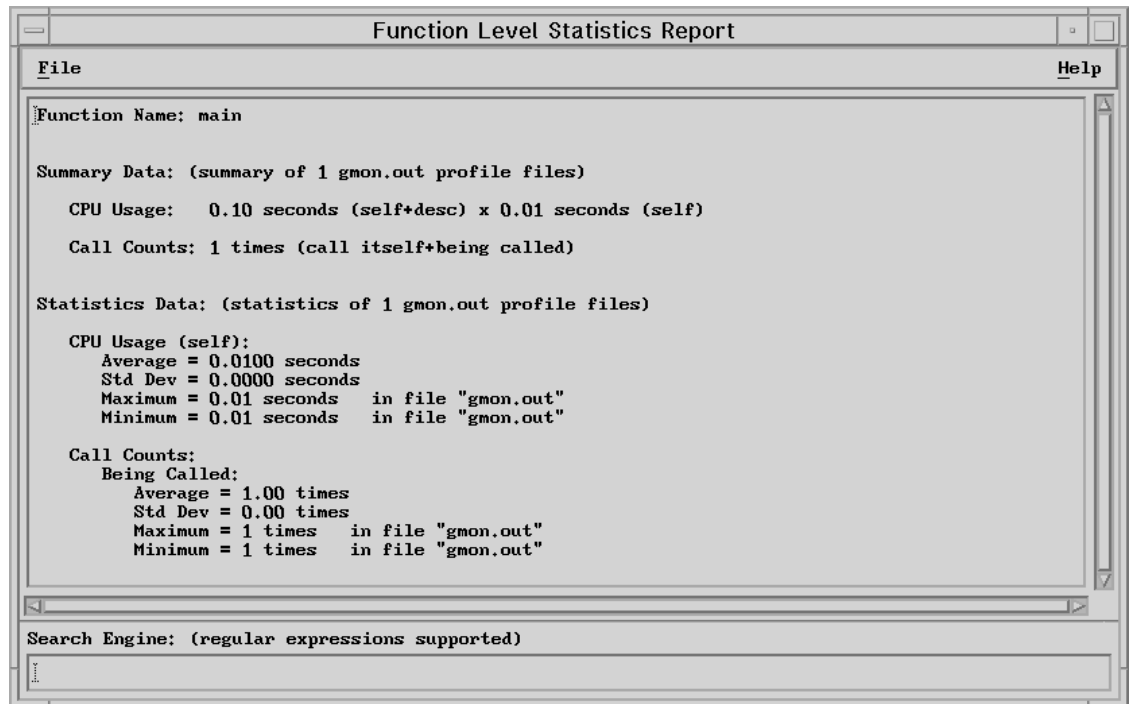


Figure 24. Function Level Statistics Report window

The Function Level Statistics Report window provides the following information:

Function Name

The name of the function you selected. In Figure 24, the function name is *main*.

Summary Data

The total amount of CPU used by this function. If you used multiple gmon.out files, the value shown here represents their sum.

The *CPU Usage* field shows you:

- The amount of CPU used by this function. There are two values supplied in this field. The first is the amount of CPU time spent on this function plus the time spent on its descendants. The second value represents the amount of CPU time this function spent only on itself.

In Figure 24, CPU usage is listed as *0.10 seconds (self+desc) x 0.10 seconds (self)*

The *Call Counts* field shows you:

- The number of times this function called itself, plus the number of times it was called by other functions.

In Figure 24, the value in the Call Counts field is *1 times*.

Statistics Data

The CPU usage and calls made to or by this function, broken down by gmon.out file.

The *CPU Usage* field shows you:

- Average

The average CPU time used by the data in each gmon.out file. In Figure 24 on page 44, the Average is listed as *0.0100 seconds*.

- Std Dev

Standard deviation. A value that represents the difference in CPU usage samplings, per function, from one gmon.out file to another. The smaller the standard deviation, the more balanced the workload. In Figure 24 on page 44, the Std Dev is listed as *0.0000 seconds*.

- Maximum

Of all the gmon.out files, the maximum amount of CPU time used. The corresponding gmon.out file appears to the right. In Figure 24 on page 44, the Maximum is listed as *0.01 seconds*.

- Minimum

Of all the gmon.out files, the minimum amount of CPU time used. The corresponding gmon.out file appears to the right. In Figure 24 on page 44, the Minimum is listed as *0.01 seconds*.

The *Call Counts* field shows you:

- Average

The average number of calls made to this function or by this function, per gmon.out file. In Figure 24 on page 44, the Average is *1.00 times*.

- Std Dev

Standard deviation. A value that represents the difference in call count sampling, per function, from one gmon.out file to another. A small standard deviation value in this field means that the function was almost always called the same number of times in each gmon.out file. In Figure 24 on page 44, the Std Dev is *0.00 times*.

- Maximum

The maximum number of calls made to this function or by this function in a single gmon.out file. The corresponding gmon.out file appears to the right. In Figure 24 on page 44, the Maximum is *1 times*.

- Minimum

The minimum number of calls made to this function or by this function in a single gmon.out file. The corresponding gmon.out file appears to the right. In Figure 24 on page 44, the Minimum is *1 times*.

Getting Detailed Data via Reports

Xprofiler provides performance data in textual and tabular format. This data is provided in various tables called *reports*. If you are a **gprof** user, you are familiar with the *Flat Profile*, *Call Graph Profile*, and *Function Index* reports. Xprofiler generates these same reports, in the same format, plus two others.

You can access the Xprofiler reports from the Report menu. The Report menu lets you see the following reports:

- Flat Profile
- Call Graph Profile
- Function Index
- Function Call Summary
- Library Statistics

Each report window includes a File menu. Under the File menu is the *Save As* option which allows you to save the report to a file. For information on using the *Save File Dialog* window to save a report to a file, see “Using the Save Dialog Windows” on page 21.

Each report window also includes a *Search Engine* field, which is located at the bottom of the window. The Search Engine lets you search for a specific string in the report. For information on using the Search Engine field, see “Using the Search Engine” on page 21.

Note: If you select the *Save As* option from the Flat Profile, Function Index, or Function Call Summary report windows, you must either complete the save operation or cancel it before you can select any other option from the menus of these reports. You can, however, use the other menus of Xprofiler before completing the save operation or canceling it, with the exception of the Load Files option, of the File menu, which remains greyed out.

Each of the Xprofiler reports are explained below.

Flat Profile Report

When you select the *Flat Profile* menu option, the *Flat Profile* window appears. The Flat Profile report shows you the total execution times and call counts for each function (including shared library calls) within your application. The entries for the functions that use the greatest percentage of the total CPU usage appear at the top of the list, while the remaining functions appear in descending order, based on the amount of time used.

Unless you specified the **-z** command line option, the Flat Profile report does not include functions whose CPU usage is 0 (zero) and have no call counts.

Note that the data presented in the Flat Profile window is the same data that is generated with the UNIX **gprof** command.

The Flat Profile report looks similar to this:

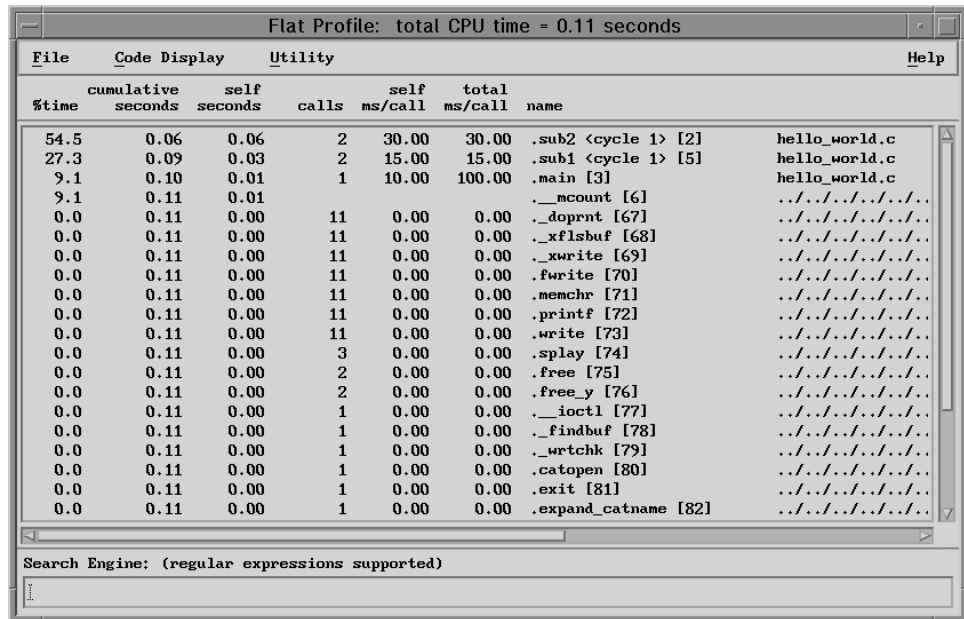


Figure 25. Flat Profile Report

Flat Profile Window Fields: The Flat Profile window fields are explained below.

- %time
The percentage of the program's total CPU usage that is consumed by this function.
- cumulative seconds
A running sum of the number of seconds used by this function and those listed above it.
- self seconds
The number of seconds used by this function alone. The *self seconds* values are what Xprofiler uses to sort the functions of the Flat Profile report.
- calls
The number of times this function was called (if this function is profiled). Otherwise, it is blank.
- self ms/call
The average number of milliseconds spent in this function per call (if this function is profiled). Otherwise, it is blank.
- total ms/call
The average number of milliseconds spent in this function and its descendants per call (if this function is profiled). Otherwise, it is blank.
- name
The name of the function. The *index* appears in brackets [] to the right of the function name. The index serves as the function's identifier within Xprofiler. It also appears below the corresponding function in the function call tree.

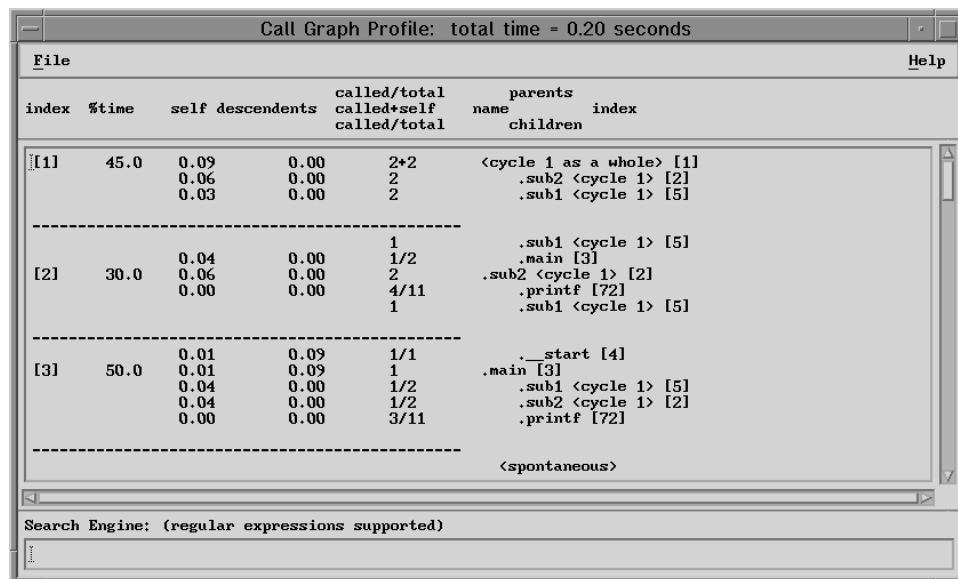
Call Graph Profile Report

The *Call Graph Profile* menu option lets you view the functions of your application, sorted by the percentage of total CPU usage that each function, and its descendants, consumed. When you select this option, the *Call Graph Profile* window appears.

Unless you specified the **-z** command line option, the Call Graph Profile report does not include functions whose CPU usage is 0 (zero) and have no call counts.

Note that the data presented in the Call Graph Profile window is the same data that is generated with the UNIX **gprof** command.

The Call Graph Profile report looks similar to this:



The screenshot shows a window titled "Call Graph Profile: total time = 0.20 seconds". The window contains a table with the following columns: index, %time, self, descendants, called/total, called+self, called/total, parents, name, index, children. The data is as follows:

index	%time	self	descendants	called/total	called+self	called/total	parents	name	index	children
[1]	45.0	0.09	0.00	2+2	2	2	<cycle 1 as a whole>		[1]	.sub2 <cycle 1> [2] .sub1 <cycle 1> [5]
[2]	30.0	0.04	0.00	1/2	2	2	.sub1 <cycle 1>	[5]	[5]	.main [3] .sub2 <cycle 1> [2] .printf [72] .sub1 <cycle 1> [5]
[3]	50.0	0.01	0.09	1/1	1	1	__start	[4]	[4]	.main [3] .sub1 <cycle 1> [5] .sub2 <cycle 1> [2] .printf [72]
		0.04	0.00	1/2	1	1				
		0.00	0.00	3/11	1	1				
										<spontaneous>

Figure 26. Call Graph Profile Report

Call Graph Profile Window Fields: The fields of the Call Graph Profile are explained below.

- index

The index of the function in the Call Graph Profile. Each function in the Call Graph Profile has an associated index number which serves as the function's identifier. The same index also appears with each function box label in the function call tree, as well as other Xprofiler reports.

- %time

The percentage of the program's total CPU usage that was consumed by this function and its descendants.

- self

The number of seconds this function spends within itself.

- descendants

The number of seconds spent in the descendants of this function, on behalf of this function.

- called/total, called+self, called/total

The heading of this column refers to the three different kinds of calls that take place within your program. The values in this field correspond to the functions listed in the *name*, *index*, *parents*, *children* field to its right. Depending on whether the function is a parent, child, or the function of interest (the function who's index is listed in the *index* field of this row), this value can stand for one of the following:

- Number of times a parent called the function of interest
- Number of times the function of interest called itself, recursively
- Number of times the function of interest called a child

In the example below, *sub2* is the function of interest, *sub1* and *main* are its parents, and *printf* and *sub1* are its children.

called/total called+self called/total	parents name children	index

1	.sub1 <cycle 1>	[5]
1/2	.main	[3]
2	.sub2 <cycle 1>	[2]
4/11	.printf	[72]
1	.sub1 <cycle 1>	[5]

Figure 27. *called/total*, *call/self*, *called/total* field

- *called/total*

For a parent function, this refers to the number of calls made to the function of interest, as well as the total number of calls it made to all functions. In the example above, one of the parent functions, *main*, made two calls; one to the function of interest, *sub2*, and one to another function.

- *called+self*

This refers to the number of times the function of interest called itself, recursively. In the example above, the function of interest, *sub2*, called itself two times. For a child function, this refers to the number of times the function of interest called this child. In the example above, one of the child functions, *printf*, was called eleven times; four times by the function of interest, *sub2*, and seven times by other functions.

- *name*, *index*, *parents*, *children*

The layout of the heading of this column is indicative of the information that is provided. To the left is the name of the function, and to its right is the function's index number. Appearing above the function are its parents, and below are its children.

parents	
name	index
children	
.sub1 <cycle 1>	[5]
.main	[3]
.sub2 <cycle 1>	[2]
.printf	[72]
.sub1 <cycle 1>	[5]

Figure 28. name/index/parents/children field

- name

The name of the function, with an indication of its membership in a cycle, if any. Note that the function of interest appears to the left, while its parent and child functions are indented above and below it. In the example above, the name of the function is *sub2*.
- index

The index of the function in the Call Graph Profile. This number corresponds to the index that appears in the *index* column of the Call Graph Profile and the on the function box labels in the function call tree. In the example above, the index of *sub2* is [2].
- parents

The parents of the function. A *parent* is any function that directly calls the function in which you are interested. In the example above, the parents are *sub1* and *main*.

If any portion of your application was not compiled with the **-pg** option, Xprofiler will not be able to identify the parents for the functions within those portions. As a result, these parents will be listed as *spontaneous* in the Call Graph Profile report.
- children

The children of the function. A *child* is any function that is directly called by the function in which you are interested. In the example above, the children are *printf* and *sub1*.

Function Index Report

The *Function Index* menu option lets you view a list of the function names included in the function call tree. When you select this option, the *Function Index* window appears, and displays the function names in alphabetical order. To the left of each function name is its *index*, enclosed in brackets []. The index is the function's identifier, which is assigned by Xprofiler. An index also appears on the label of each corresponding function box in the function call tree as well as other reports.

Unless you specified the **-z** command line option, the Function Index report does not include functions whose CPU usage is 0 (zero) and have no call counts.

The *Function Index* menu option includes a *Code Display* menu, like the *Flat Profile* menu option, allowing you to view source code or disassembler code. For more information on viewing code, see “Viewing Source Code” on page 56 and “Viewing Disassembler Code” on page 58.

The Function Index report looks similar to this:

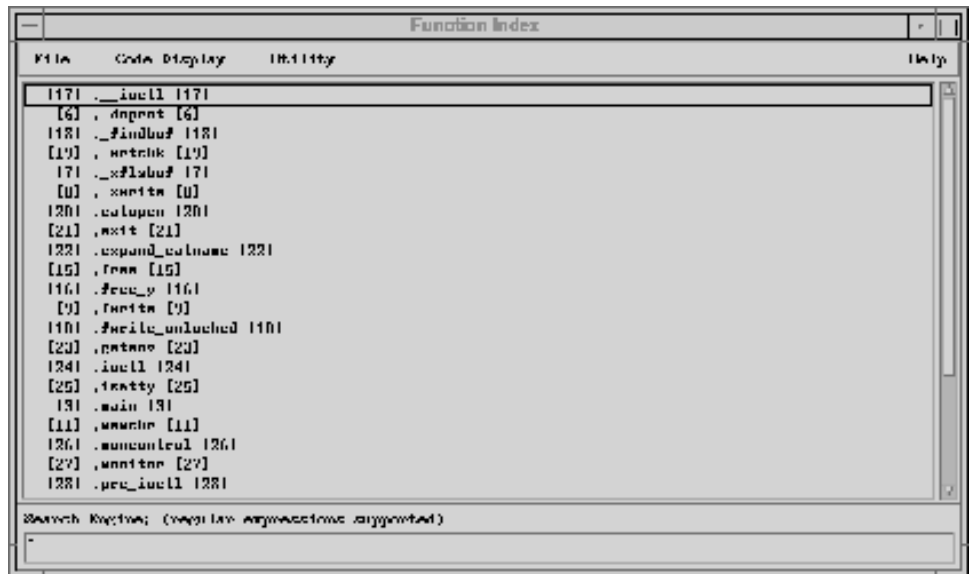


Figure 29. Sample Function Index Report

Function Call Summary Report

The *Function Call Summary* menu option lets you display all the functions in your application that call other functions. They appear as caller-callee pairs (call arcs, in the function call tree), and are sorted by the number of calls in descending order. When you select this option, the *Function Call Summary* window appears.

The Function Call Summary report looks similar to this:

%total	calls	function
10.78%	11	calls from .printf [72] to ._doprint [67]
10.78%	11	calls from ._doprint [67] to .fwrite [70]
10.78%	11	calls from ._xflsbuf [68] to ._xwrite [69]
10.78%	11	calls from ._xwrite [69] to .write [73]
10.78%	11	calls from .fwrite [70] to .memchr [71]
10.78%	11	calls from .fwrite [70] to ._xflsbuf [68]
3.92%	4	calls from .sub2 <cycle 1> [2] to .printf [72]
3.92%	4	calls from .sub1 <cycle 1> [5] to .printf [72]
2.94%	3	calls from .free_y [76] to .splay [74]
2.94%	3	calls from .main [3] to .printf [72]
1.96%	2	calls from .free [75] to .free_y [76]
0.98%	1	calls from .ioctl [84] to ._ioctl [77]
0.98%	1	calls from .setlocale [89] to .saved_category_name [88]
0.98%	1	calls from .monitor [87] to .catopen [80]
0.98%	1	calls from .monstn [1465] to .free [75]
0.98%	1	calls from .monstartup [1463] to .free [75]
0.98%	1	calls from .monitor [87] to .moncontrol [86]
0.98%	1	calls from .expand_catname [82] to .getenv [83]
0.98%	1	calls from .expand_catname [82] to .setlocale [89]
0.98%	1	calls from .isatty [85] to .ioctl [84]
0.98%	1	calls from .catopen [80] to .expand_catname [82]

Search Engine: (regular expressions supported)

Figure 30. Sample Function Call Summary Report

Function Call Summary Window Fields: The fields of the Function Call Summary window are explained below.

- %total
The percentage of the total number of calls generated by this caller-callee pair.
- calls
The number of calls attributed to this caller-callee pair.
- function
The name of the caller function and callee function.

Library Statistics Report

The *Library Statistics* menu option lets you display the CPU time consumed and call counts of each library within your application. When you select this option, the *Library Statistics* window appears.

The Library Statistics report looks similar to this:

The screenshot shows a window titled "Library Statistics" with a menu bar containing "File" and "Help". Below the menu bar is a table with the following data:

total seconds	%total time	total calls	%total calls	%calls out of	%calls into	%calls within	load unit
0.10	90.91	5	4.90	11.76	0.00	4.90	hello_world
0.01	9.09	97	95.10	0.00	11.76	83.33	/lib/profiled/libc.a : shr.o
0.00	0.00	NA	--	0.00	--	--	/lib/profiled/libc.a : meth.o

Below the table is a search engine field with the text "Search Engine: (regular expressions supported)" and an empty input box.

Figure 31. Sample Library Statistics Report

Library Statistics Window Fields: The fields of the Library Statistics window are explained below.

- total seconds
The total CPU usage of the library, in seconds.
- %total time
The percentage of the total CPU usage that was consumed by this library.
- total calls
Total number of calls generated by this library.
- %total calls
The percentage of the total calls that were generated by this library.
- %calls out of
The percentage of the total number of calls made from this library to other libraries.
- %calls into
The percentage of the total number of calls made from other libraries into this library.
- %calls within
The percentage of the total number of calls made between the functions within this library.
- load unit
The library's full path name.

Saving Reports to a File

Xprofiler lets you save any of the reports you generate with the Report menu to a file. You can do this via the File and Report menus of the Xprofiler GUI.

Saving a Single Report: To save a single report, go to the Report menu, on the Xprofiler main window, and select the report you would like to save. Each report window includes a File menu. Select the File menu and then the *Save As* option to save the report. A *Save* dialog window appears, which is named according to the report from which you selected the *Save As* option. For instance, if you chose *Save As* from the Flat Profile window, the dialog window is named *Save Flat Profile Dialog*.

Saving the Call Graph Profile, Function Index, and Flat Profile Reports to File: You can save the Call Graph Profile, Function Index, and Flat Profile reports to a single file with the the File menu of the Xprofiler main window. The information you generate here is identical to the output of the UNIX **gprof** command. From the File menu, select the *Save As* option. The *Save File Dialog* window appears.

To save the report(s):

1. Specify the file into which the profiled data should be placed. You can specify either an existing file or a new one. To specify an existing file, use the scroll bars of the *Directories* and the *Files* selection boxes to locate the file you want. To make locating your files easier, you can also use the *Filter* button (see “Using the Dialog Window Filters” on page 22 for more information). To specify a new file, type its name in the *Selection* field.
2. Click on the **OK** button. A file containing the profiled data appears in the directory you specified, under the name you gave it.

Note: Once you select the *Save As* option from the File menu, and the *Save Profile Reports* window opens, you must either complete the save operation or cancel it before you can select any other option from the menus of its parent window. For example, if you select the *Save As* option from the Flat Profile report, and the *Save File Dialog* window appears, you cannot use any other option of the Flat Profile report window.

The *File Selection* field of the *Save File Dialog* window follows Motif standards.

Saving Summarized Data from Multiple Profile Data Files: If you are profiling a parallel program, you could specify more than one profile data (*gmon.out*) file when you started Xprofiler. The *Save gmon.sum As* option of the File menu lets you save a summary of the data in each of these files to a single file.

The Xprofiler *Save gmon.sum As* option produces the same result as the Xprofiler and **gprof -s** command line option. If you run Xprofiler later, you can use the file you create here as input with the **-s** option. In this way, you can accumulate summary data over several runs of your application.

To create a summary file:

1. Select the File menu, and then the *Save gmon.sum As* option. The *Save gmon.sum Dialog* window appears.
2. Specify the file into which the summarized, profiled data should be placed. By default, Xprofiler puts the data into a file called *gmon.sum*, but you can designate a different file. You can either specify a new file or an existing one.

To specify a new file, type its name in the selection field. To specify an existing file, use the scroll bars of the *Directories* and *Files* selection boxes to locate the file you want. To make locating your files easier, you can also use the Filter button (see “Using the Dialog Window Filters” on page 22 for information).

3. Click on the OK button. A file, containing the summary data, appears in the directory you specified, under the name you gave it.

Saving a Configuration File: The *Save Configuration* menu option lets you save the names of the functions that are displayed currently to a file. Later, in the same Xprofiler session or a different session, you can read in this configuration file using the *Load Configuration* option. See the following section, “Loading a Configuration File,” for more information.

To save a configuration file:

1. Select the File menu, and then the *Save Configuration* option. The *Save Configuration File Dialog* window opens with the *program.cfg* file as the default value in the *Selection* field. “Program” is the name of the input *a.out* file.

You can use the default file name, enter a file name in the *Selection* field, or select a file from the dialog's files list.

2. Specify a file name in the *Selection* field and click on the **OK** button. A configuration file is created containing the name of the program and the names of the functions that are displayed currently.
3. Specify an existing file name in the *Selection* field and click on the **OK** button. An *Overwrite File Dialog* window appears so you can check the file before overwriting it.

If you select the *Forced File Overwriting* option in the *Runtime Options Dialog* window, the *Overwrite File Dialog* does not open and the specified file is overwritten without warning.

Loading a Configuration File: The *Load Configuration* menu option lets you read in a configuration file that you saved. See the previous section, “Saving a Configuration File,” for more information. The *Load Configuration* option automatically reconstructs the function call tree according to the function names recorded in the configuration file.

To load a configuration file:

1. Select the File menu, and then the *Load Configuration* option. The *Load Configuration File Dialog* window opens. If a configuration files were loaded previously during the current Xprofiler session, the name of the file that was most recently loaded will appear in the **Selection** field of this dialog.

You can also load the file with the **-c** command line option. See “Specifying Command Line Options (from the GUI)” on page 11 for more information.

2. Select a configuration file from the dialog's **Files** list or specify a file name in the **Selection** field, and click on the **OK** button. The function call tree is redrawn to show only those function boxes for functions that are listed in the configuration file and are called within the program that is currently represented in the display. All corresponding call arcs are also drawn.

| If the *a.out* name, that is, the program name in the configuration file, is different
| from the *a.out* name in the current display, a confirmation dialog appears to
| allow you to decide whether or not you still wish to load the file.

- | 3. If after loading a configuration file, you wish to return the function call tree back
| to its previous state, select the *Filter* menu, and then the *Undo* option.

Looking at Source Code

Xprofiler provides several ways for you to view your source code. You can view the source or disassembler code for your application on a per-function basis. This also applies to any included function code your application may use.

When you view source or included function code, you use the *Source Code* window. When you view disassembler code, you use the *Disassembler Code* window. You can access these windows through the Report menu of the Xprofiler GUI or the Function menu of the function in which you are interested.

Viewing Source Code

Both the Function menu and Report menu provide the means for you to access the *Source Code* window, from which you will view your code.

To access the *Source Code* window via the Function menu:

1. Click on the function box you are interested in with the right mouse button. The Function menu appears.
2. From the Function menu, select the *Show Source Code* option. The *Source Code* window appears.

To access the *Source Code* window via the Report menu:

1. Select the Report menu, and then the *Flat Profile* option. The *Flat Profile* window appears.
2. From the *Flat Profile* window, select the function you would like to view by clicking on its entry in the window. The entry highlights to show that it is selected.
3. Select the *Code Display* menu, and then the *Show Source Code* option. The *Source Code* window appears, containing the source code for the function you selected.

Using the Source Code Window: The Source Code window shows you only the source code file for the function you specified from the Flat Profile window or the Function menu. The Source Code Window looks similar to this:

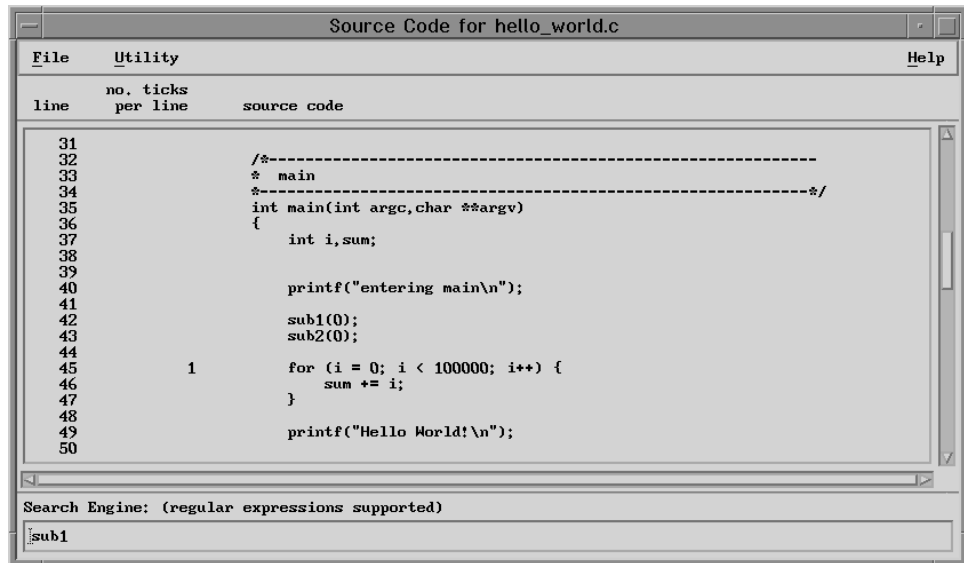


Figure 32. Sample Source Code Window

The Source Code Window contains information in the following fields:

- line

The source code line number.

- no. ticks per line

Each tick represents .01 seconds of CPU time used. The number that appears in this field represents the number of ticks used by the corresponding line of code. For instance, if the number 3 appeared in this field, for a source statement, this source statement would have used .03 seconds of CPU time. Note that the CPU usage data only appears in this field if you used the **-g** option when you compiled your application. Otherwise, this field is blank.

- source code

The application's source code.

The *Search Engine* field, at the bottom of the Source Code window, lets you search for a specific string in your source code. For information on using the Search Engine field, see “Using the Search Engine” on page 21

The Source Code window contains the following menus:

- File

The *Save As* option lets you save the annotated source code to a file. When you select this option, the Save File Dialog window appears. For more information on using the Save File Dialog window, see “Using the Save Dialog Windows” on page 21

Select *Close* if you wish to close the Source Code window.

- Utility

The *Utility* menu contains only one option; *Show Included Functions*.

For C++ users, the *Show Included Functions* option lets you view the source code of included function files that are included by the application's source code.

|
| If a selected function does not have an included function file associated with it or
| does not have the function file information available because the **-g** option was not
| used for compiling, the *Utility* menu will be greyed out. The availability of the *Utility*
| menu serves as an indication of whether or not there is any included function file
| information associated with the selected function.

When you select the *Show Included Functions* option, the *Included Functions Dialog* window appears, which lists all of the included function files. Specify a file by either clicking on one of the entries in the list with the left mouse button, or by typing the file name in the **Selection** field. Then click on the **OK** or **Apply** button. After selecting a file from the *Included Functions Dialog* window, the *Included Function File* window appears, displaying the source code for the file that you specified.

Viewing Disassembler Code

Both the Function menu and Report menu provide the means for you to access the *Disassembler Code* window, from which you can view your code.

To access the *Disassembler Code* window via the Function menu:

1. Click on the function you are interested in with the right mouse button. The Function menu appears.
2. From the Function menu, select the *Show Disassembler Code* option. The *Disassembler Code* window appears.

To access the *Disassembler Code* window via the Report menu:

1. Select the Report menu, and then the *Flat Profile* option. The *Flat Profile* window appears.
2. From the *Flat Profile* window, select the function you would like to view by clicking on its entry in the window. The entry highlights to show that it is selected.
3. Select the *Code Display* menu, and then the *Show Disassembler Code* option. The *Disassembler Code* window appears, and contains the disassembler code for the function you selected.

Using the Disassembler Code Window: The *Disassembler Code* window shows you only the disassembler code for the function you specified from the Flat Profile window. The Disassembler Code Window looks similar to this:

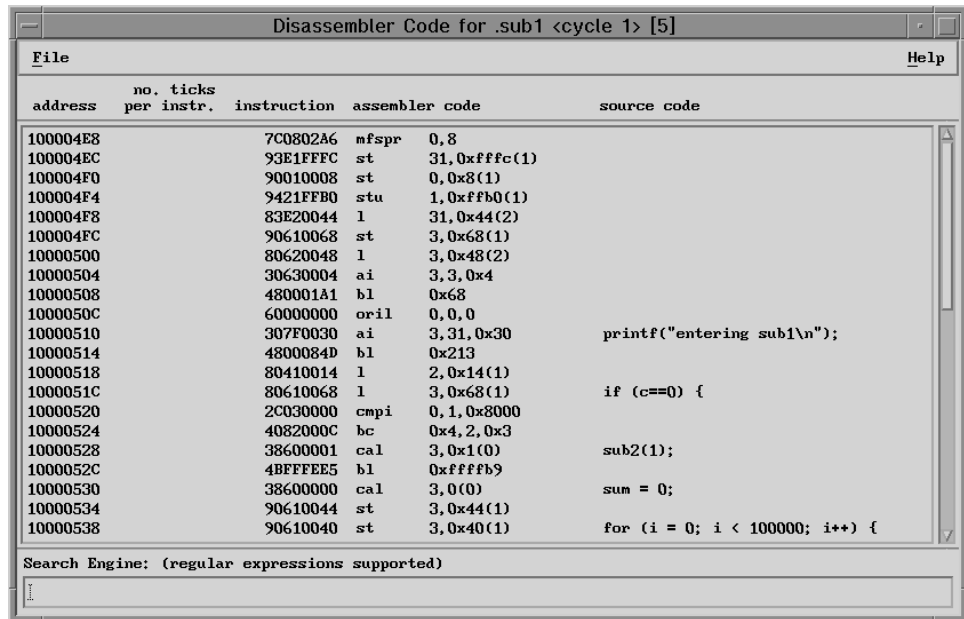


Figure 33. Sample Disassembler Code Window

The Disassembler Code window contains information in the following fields:

- address

The address of each instruction in the function you selected (from either the Flat Profile window or the function call tree).

- no. ticks per instr.

Each tick represents .01 seconds of CPU time used. The number that appears in this field represents the number of ticks used by the corresponding instruction. For instance, if the number 3 appeared in this field, this instruction would have used .03 seconds of CPU time.

- instruction

The execution instruction.

- assembler code

The execution instruction's corresponding assembler code.

- source code

The line in your application's source code that corresponds to the execution instruction and assembler code. In order for information to appear in this field, you must have compiled your application with the `-g` compile option.

The *Search Engine* field, at the bottom of the Disassembler Code window, lets you search for a specific string in your disassembler code. For information on using the Search Engine field, see "Using the Search Engine" on page 21.

The Disassembler Code window contains only one menu:

- File

Select *Save As* to save the annotated disassembler code to a file. When you select this option, the Save File Dialog window appears. For information on

using the Save File Dialog window, see “Using the Save Dialog Windows” on page 21.

Select *Close* if you wish to close the Disassembler window.

Saving Screen Images of Profiled Data

The File menu of the Xprofiler GUI includes an option called *Screen Dump* that lets you capture an image of the Xprofiler main window. This option is useful if you want to save a copy of the graphical display to refer to later. You can either save the image as a file in PostScript format, or send it directly to a printer.

To capture a window image:

1. Select the *File*→*Screen Dump* options. The Screen Dump menu opens.
2. From the Screen Dump menu, select the *Set Option* option. The *Screen Dump Options Dialog* window appears.

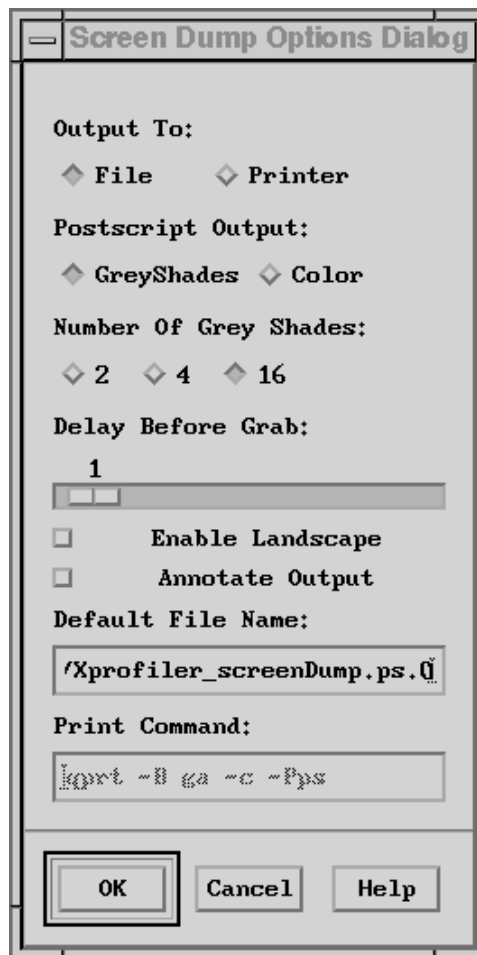


Figure 34. Screen Dump Options Dialog Window

3. Make the appropriate selections in the fields of the *Screen Dump Dialog Window* as follows:

- Output To:

This option lets you specify whether you want to save the captured image as a PostScript file or send it directly to a printer.

If you would like to save the image to a file, select the *File* button. This file, by default, is named *Xprofiler.screenDump.ps.0*, and is displayed in the *Default File Name* field of this dialog window. When you select the *File* button, the text in the *Print Command* field greys out.

If you would like to send the image directly to a printer, select the *Printer* button. The image is sent to the printer you specify in the *Print Command* field of this dialog window. Note that when you specify the *Print* option, a file of the image is not saved. Also, selecting this option causes the text in the *Default File Name* field to grey out.

- **PostScript Output:**

This option lets you specify whether you want to capture the image in shades of grey or in color.

If you want to capture the image in shades of grey, select the *GreyShades* button. You must also select the number of shades you want the image to include with the *Number of Grey Shades* option, as discussed below.

If you want to capture the image in color, select the *Color* button. *GreyShades*.

- **Number of Grey Shades**

This option lets you specify the number of grey shades that the captured image will include. Select either the 2, 4, or 16 buttons, depending on the number of shades you want to use. Typically, the more shades you use, the longer it will take to print the image. 16.

- **Delay Before Grab**

This option lets you specify how long of a delay will occur between activating the capturing mechanism and when the image is actually captured. By default, the delay is set to one second, but you may need time to arrange the window the way you want it. Setting the delay to a longer interval gives you some extra time to do this. You set the delay with the slider bar of this field. The number above the slider indicates the time interval in seconds. You can set the delay to a maximum of thirty seconds.

To set the delay, place the mouse cursor over the slider. Next, press and hold the left mouse button while moving the slider to the right. When the slider is at the desired number, release the mouse button.

- **Enable Landscape (button)**

This option lets you specify that you want the output to be in landscape format (the default is portrait). To select landscape format, select the *Enable Landscape* button.

- **Annotate Output (button)**

This option lets you specify that you would like information about how the file was created to be included in the PostScript image file. By default, this information is not included. To do this, select the *Annotate Output* button.

- **Default File Name**

If you chose to put your output in a file, this field lets you specify the file name. The default file name is *Xprofiler.screenDump.ps.0*. If you want to

| change to a different file name, type it over the one that appears in this
| field.

| If you specify the output file name with an integer suffix (that is, the file
| name ends with *xxx.nn*, where *nn* is a non-negative integer), the suffix
| automatically increases by one every time a new output file is written in the
| same Xprofiler session.

- Print Command

If you chose to send the captured image directly to a printer, this field lets you specify the print command. The default print command is *qprt -B ga -c -Pps*. If you would like to use a different command, type the new command over the one that appears in this field.

Press the **OK** button. The *Screen Dump Options Dialog* window closes.

Once you have set your screen dump options, you need to select the window, or portion of a window, you wish to capture. From the Screen Dump menu, select the *Select Target Window* option. A cursor in the image of a hand appears after the number of seconds you specified. At any time you wish to cancel the capture, you may do so by clicking on the right mouse button. The hand-shaped cursor will change back to normal and the operation will be terminated.

To capture the entire Xprofiler window, place the cursor in the window and then click the left mouse button.

To capture a portion of the Xprofiler window:

1. Place the cursor in the upper left corner of the area you wish to capture.
2. Press and hold the middle mouse button and drag the cursor diagonally downward, until the area you wish to capture is within the rubberband box.
3. Release the middle mouse button to set the location of the rubberband box.
4. Press the left mouse button to capture the image.

If you chose to save the image as a file, the file is stored in the directory you specified. If you chose to print the image, the image is sent to the printer you specified.

Appendix A. Parallel Environment Tools Commands

This appendix contains the manual pages for the PE tools commands discussed throughout this book. Each manual page is organized into the sections listed below. The sections always appear in the same order, but some appear in all manual pages while others are optional.

NAME	Provides the name of the command described in the manual page, and a brief description of its purpose.
SYNOPSIS	Includes a diagram that summarizes the command syntax, and provides a brief synopsis of its use and function. If you are unfamiliar with the typographic conventions used in the syntax diagrams, see “Typographic Conventions” on page x.
FLAGS	Lists and describes any required and optional flags for the command.
DESCRIPTION	Describes the command more fully than the NAME and SYNOPSIS sections.
ENVIRONMENT VARIABLES	Lists and describes any applicable environment variables.
EXAMPLES	Provides examples of ways in which the command is typically used.
FILES	Lists and describes any files related to the command.
RELATED INFORMATION	Lists commands, functions, file formats, and special files that are employed by the command, that have a purpose related to the command, or that are otherwise of interest within the context of the command.

xprofiler

NAME

xprofiler – Invokes the Xprofiler, a GUI-based performance profiling tool.

SYNOPSIS

```
xprofiler [program] [-b] [-h]
[-s] [-z] [-a] [-c]
[-L pathname]
[[-e name]...]
[[-E name]...]
[[-f name]...]
[[-F name]...]
[-disp_max number_of_functions]
[[gmon.out]...]
```

The **xprofiler** command invokes the Xprofiler, a GUI-based performance profiling tool.

FLAGS

- b** Suppresses the printing of the field descriptions for the Flat Profile, Call Graph Profile, and Function Index reports when they are written to a file with the **Save As** option of the File menu.
- s** Produces the *gmon.sum* profile data file, if multiple *gmon.out* files are specified when Xprofiler is started. The *gmon.sum* file represents the sum of the profile information in all the specified profile files. Note that if you specify a single *gmon.out* file, the *gmon.sum* file contains the same data as the *gmon.out* file.
- z** Includes functions that have both zero CPU usage and no call counts in the Flat Profile, Call Graph profile, and Function Index reports. A function will not have a call count if the file that contains its definition was not compiled with the **-pg** option, which is common with system library files.
- a** Adds alternative paths to search for source code and library files, or changes the current path search order. When using this command line option, you can use the "at" symbol (@) to represent the default file path, in order to specify that other paths be searched before the default path.
- c** Loads the specified configuration file. If the **-c** option is used on the command line, the configuration file name specified with it will appear in the **Configuration File (-c)**: text field in the *Load Files Dialog*, and the **Selection** field of the *Load Configuration File Dialog*. When both the **-c** and **-disp_max** options are specified on the command line, the **-disp_max** option is ignored, but the value that was specified with it will appear in the **Initial Display (-disp_max)**: field in the Load Files Dialog, the next time it is opened.
- disp_max** Sets the number of function boxes that Xprofiler initially displays in the function call tree. The value supplied with this flag can be any integer between 0 and 5,000. Xprofiler displays the function boxes for the most CPU-intensive functions through the number you specify. For instance, if you specify 50, Xprofiler displays the function boxes for the 50 functions in your program that consume the most CPU. After this, you can change the number of function boxes that are displayed via the Filter menu options. This flag has no effect on the content of any of the Xprofiler reports.
- e** De-emphasizes the general appearance of the function box(es) for the specified function(s) in the function call tree, and limits the number of entries for these function in the Call Graph Profile report. This also applies to the specified function's descendants, as long as they have not been called by non-specified functions.

In the function call tree, the function box(es) for the specified function(s) appears greyed-out. Its size and the content of the label remain the same. This also applies to descendant functions, as long as they have not been called by non-specified functions.

In the Call Graph Profile report, an entry for the specified function only appears where it is a child of another function, or as a parent of a function that also has at least one non-specified function as its

parent. The information for this entry remains unchanged. Entries for descendants of the specified function do not appear unless they have been called by at least one non-specified function in the program.

-E

Changes the general appearance and label information of the function box(es) for the specified function(s) in the function call tree. Also limits the number of entries for these functions in the Call Graph Profile report, and changes the CPU data associated with them. These results also apply to the specified function's descendants, as long as they have not been called by non-specified functions in the program.

In the function call tree, the function box for the specified function appears greyed-out, and its size and shape also changes so that it appears as a square of the smallest allowable size. In addition, the CPU time shown in the function box label, appears as 0 (zero). The same applies to function boxes for descendant functions, as long as they have not been called by non-specified functions. This option also causes the CPU time spent by the specified function to be deducted from the left side CPU total in the label of the function box for each of the specified function's ancestors.

In the Call Graph Profile report, an entry for the specified function only appears where it is a child of another function, or as a parent of a function that also has at least one non-specified function as its parent. When this is the case, the time in the *self* and *descendants* columns for this entry is set to 0 (zero). In addition, the amount of time that was in the *descendants* column for the specified function is subtracted from the time listed under the *descendants* column for the profiled function. As a result, be aware that the value listed in the *% time* column for most profiled functions in this report will change.

-f

De-emphasizes the general appearance of all function boxes in the function call tree, *except* for that of the specified function(s) and its descendant(s). In addition, the number of entries in the Call Graph Profile report for the non-specified functions and non-descendant functions is limited. The **-f** flag overrides the **-e** flag.

In the function call tree, all function boxes *except* for that of the specified function(s) and its descendant(s) appear greyed-out. The size of these boxes and the content of their labels remain the same. For the specified function(s), and its descendants, the appearance of the function boxes and labels remain the same.

In the Call Graph Profile report, an entry for a non-specified or non-descendant function only appears where it is a parent or child of a specified function or one of its descendants. All information for this entry remains the same.

-F

Changes the general appearance and label information of all function boxes in the function call tree *except* for that of the specified function(s) and its descendants. In addition, the number of entries in the Call Graph Profile report for the non-specified and non-descendant functions is limited, and the CPU data associated with them is changed. The **-F** flag overrides the **-E** flag.

xprofiler(1)

In the function call tree, the function box for the specified function appears greyed-out, and its size and shape also changes so that it appears as a square of the smallest allowable size. In addition, the CPU time shown in the function box label, appears as 0 (zero).

In the Call Graph Profile report, an entry for a non-specified or non-descendant function only appears where it is a parent or child of a specified function or one of its descendants. The time in the *self* and *descendants* columns for this entry is set to 0 (zero). When this is the case, the time in the *self* and *descendants* columns for this entry is set to 0 (zero). As a result, be aware that the value listed in the *% time* column for most profiled functions in this report will change.

- L Uses an alternate path name for locating shared libraries. If you plan to specify multiple paths, use the *Set File Search Path* option of the File menu on the Xprofiler GUI.
- h Prints basic Xprofiler command syntax to the screen.

DESCRIPTION

Xprofiler is a GUI-based performance profiling tool, which can be used to analyze the performance of sequential as well as parallel programs. Xprofiler provides graphical function call tree display and textual profile reports to help you understand your program's CPU usage and function call counts information.

EXAMPLES

To use **xprofiler**, you first compile your program (for example, `foo.c`) with **-pg**:

```
xlc -pg -o foo foo.c
```

When the program `foo` is executed, one *gmon.out* file will be generated for each processor involved in the execution. To invoke **xprofiler**, enter:

```
xprofiler foo [[gmon.out]...]
```

FILES

`/usr/lib/X11/app-defaults/Xprofiler`

RELATED INFORMATION

Commands: **gprof(1)**, **xlc(1)**, **xlf(1)**

Appendix B. Customizing Tool Resources

You can customize certain features of an X-Window. For example, you can customize its colors, fonts, orientation, and so on. This section lists each of the resource variables you can set for Xprofiler.

You may customize resources by assigning a value to a resource name in a standard X-Windows format. Several resource files are searched according to the following X-Windows convention:

```
/usr/lib/X11/$LANG/app-defaults/file_name
```

```
/usr/lib/X11/app-defaults/file_name
```

```
$XAPPLRESDIR/file_name
```

```
$HOME/.Xdefaults
```

Where *file_name* is *Xprofiler*. Options in the *.Xdefaults* file take precedence over entries in the preceding files. This allows you to have certain specifications apply to all users in the *app-defaults* file as well as user specific preferences set for each user in their *\$HOME/.Xdefaults* file.

You customize a resource by setting a value to a *resource variable* associated with that feature. You store these *resource settings* in a file called *.Xdefaults* in your home directory. You can create this file on a server, and so customize a resource for all users. Individual users may also want to customize resources. The resource settings are essentially your own personal preferences as to how the X-Windows should look.

For example, consider the following resource variables for a hypothetical X-Windows tool:

```
TOOL*MainWindow.foreground:
```

```
TOOL*MainWindow.background:
```

In this example, say the resource variable *TOOL*MainWindow.foreground* controls the color of text on the tool's main window. The resource variable *TOOL*MainWindow.background* controls the background color of this same window. If you wanted the tool's main window to have red lettering on a white background, you would insert the following lines into the *.Xdefaults* file.

```
TOOL*MainWindow.foreground:    red
```

```
TOOL*MainWindow.background:    white
```

Customizable resources and instructions for their use for Xprofiler are defined in ***/usr/lib/X11/app-defaults/Xprofiler***, as well as ***/usr/lpp/ppe.xprofiler/defaults/Xprofiler.ad***. In this file is a set of X resources for defining graphical user interfaces based on the following criteria:

- Window geometry
- Window title

- Push button and label text
- Color maps
- text font (in both textual reports and the graphical display).

Xprofiler Resource Variables

You can use the resource variables listed below to control the appearance and behavior of Xprofiler. Note that the values supplied here are the defaults, but you may change them to suit your own preferences.

Controlling Fonts

To specify the font for the labels that appear with function boxes, call arcs, and cluster boxes:

Use this resource variable:	Specify this default, or a value of your own choice:
*narc*font	-ibm-block-medium-r-normal-*-0-0-*-*-*-ibm-*

To specify the font used in textual reports:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*fontList	rom10

Controlling the Appearance of the Xprofiler Main Window

To specify the size of the main window:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*mainW.height	700
Xprofiler*mainW.width	900

To specify the foreground and background colors of the main window:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*foreground	black
Xprofiler*background	light gray

To specify the number of function boxes that are displayed when you first open the Xprofiler main window:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*InitialDisplayGraph	5000

You can use the *-disp_max* command line option to override this value.

To specify the colors of the function boxes and call arcs of the function call tree:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*defaultNodeColor	forest green
Xprofiler*defaultArcColor	royal blue

To specify the color in which a specified function box or call arc is highlighted:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*HighlightNode	red
Xprofiler*HighlightArc	red

To specify the color in which de-emphasized function boxes appear:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*SuppressNode	gray

Function boxes are de-emphasized with the *-e*, *-E*, *-f*, and *-F* options.

Controlling Variables Related to the File Menu

To specify the size of the Load Files Dialog box:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*loadFile.height	785
Xprofiler*loadFile.width	725

The Load Files Dialog box is invoked via *Load Files* option of the File menu.

To specify whether a confirmation dialog box should appear whenever a file will be overwritten:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*OverwriteOK	False

The value *True* would be equivalent to selecting the *Set Options* option from the File menu, and then selecting the *Forced File Overwriting* option from the Runtime Options Dialog box.

To specify the alternative search paths for locating source or library files:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*fileSearchPath	<i>search path</i>

The value you specify for *search path* is equivalent to the search path you would designate from the Alt File Search Path Dialog box. To get to this dialog box, you would choose the *Set File Search Paths* option from the File menu.

To specify the file search sequence (whether the default or alternative path is searched first):

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*fileSearchDefault	True

The value *True* is equivalent to selecting the *Set File Search Paths* from the File menu, and then the *Check default path(s) first* option from the Alt File Search Path Dialog box.

Controlling Variables Related to the Screen Dump Option

To specify whether a screen dump will be sent to a printer or placed in a file:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*PrintToFile	True

The value *True* is equivalent to selecting the *File* button in the *Output To* field of the Screen Dump Options Dialog box. You access the Screen Dump Options Dialog box by selecting the *Screen Dump*→*Set Option* options from the File menu.

To specify whether the PostScript screen dump will be created in grey shades or color:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*ColorPscript	False

The value *False* is equivalent to selecting the *GreyShades* button in the *PostScript Output* area of the Screen Dump Options Dialog box. You access the Screen Dump Options Dialog box by selecting the *Screen Dump*→*Set Option* options from the File menu.

To specify the number of grey shades that the PostScript screen dump will include (if you selected *GreyShades* in the *PostScript Output* field):

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*GreyShades	16

The value *16* is equivalent to selecting the *16* button in the *Number of Grey Shades* field of the Screen Dump Options Dialog box. You access the Screen Dump Options Dialog box by selecting the *Screen Dump*→*Set Option* options from the File menu.

To specify the number of seconds that Xprofiler waits before capturing a screen image:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*GrabDelay	1

The value *1* is the default for the *Delay Before Grab* option of the Screen Dump Options Dialog box, but you may specify a longer interval by entering a value here. You access the Screen Dump Options Dialog box by selecting the *Screen Dump*→*Set Option* options from the File menu.

To specify the maximum number of seconds that may be specified with the slider of the *Delay Before Grab* option:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*grabDelayScale.maximum	30

The value *30* is the default for the *Delay Before Grab* option of the Screen Dump Options Dialog box. This means that users cannot set the slider scale to a value greater than 30. You access the Screen Dump Options Dialog box by selecting the *Screen Dump*→*Set Option* options from the File menu.

To specify whether the screen dump is created in Landscape or Portrait format:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*Landscape	True

The value *True* is the default for the *Enable Landscape* option of the Screen Dump Options Dialog box. You access the Screen Dump Options Dialog box by selecting the *Screen Dump*→*Set Option* options from the File menu.

To specify whether or not you would like information about how the image was created to be added to the PostScript screen dump:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*Annotate	False

The value *False* is the default for the *Annotate Output* option of the Screen Dump Options Dialog box. You access the Screen Dump Options Dialog box by selecting the *Screen Dump*→*Set Option* options from the File menu.

To specify the directory that will store the screen dump file (if you selected *File* in the *Output To* field):

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*PrintDirectory	<i>directory</i>

The value you specify for *directory* is equivalent to the directory you would designate in the *Default Directory* field of the Screen Dump Dialog box. You access the Screen Dump Options Dialog box by selecting the *Screen Dump*→*Set Option* options from the File menu.

To specify the printer destination of the screen dump (if you selected *Printer* in the *Output To* field):

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*PrintCommand	qprt -B ga -c -Pps

The value *qprt -B ga -c -Pps* is the default print command, but you may supply a different one here.

Controlling Variables Related to the View Menu

To specify the size of the Overview window:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*overviewMain.height	300
Xprofiler*overviewMain.width	300

To specify the color of the highlight area of the Overview window:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*overviewGraph*defaultHighlightColor	sky blue

To specify whether the function call tree is updated as the highlight area is moved (Immediate) or only when it is stopped and the mouse button released (Delayed):

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*TrackImmed	True

The value *True* is equivalent to selecting the *Immediate Update* option from the Utility menu of the Overview window. You access the Overview window by selecting the *Overview* option from the View menu.

To specify whether the function boxes in the function call tree appear in 2-D or 3-D format:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*Shape2D	True

The value *True* is equivalent to selecting the *2-D Image* option from the *View* menu.

To specify whether the function call tree appears in Top-to-Bottom or Left-to-Right format:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*LayoutTopDown	True

The value *True* is equivalent to selecting the *Layout: Top→Bottom* option from the *View* menu.

Controlling Variables Related to the Filter Menu

To specify whether the function boxes of the function call tree are clustered or unclustered when the Xprofiler main window is first opened:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*ClusterNode	True

The value *True* is equivalent to selecting the *Cluster Functions by Library* option from the *Filter* menu.

To specify whether the call arcs of the function call tree are collapsed or expanded when the Xprofiler main window is first opened:

Use this resource variable:	Specify this default, or a value of your own choice:
Xprofiler*ClusterArc	True

The value *True* is equivalent to selecting the *Collapse Library Arcs* option from the *Filter* menu.

Appendix C. Profiling Programs with the AIX prof and gprof Commands

The difference between profiling serial and parallel applications with the AIX profilers is that serial applications can be run to generate a single profile data file, while a parallel application can be run to produce many.

You request parallel profiling by setting the compile flag to **-p** or **-pg** as you would with serial compilation. The parallel profiling capability of PE creates a monitor output file for each task. The files are created in the current directory, and are identified by the name *mon.out.taskid* or *gmon.out.taskid*, where *taskid* is a number between 0 and one less than the number of tasks.

Following the traditional method of profiling using the AIX operating system, you compile a serial application and run it to produce a single profile data file that you can then process using either the **prof** or **gprof** commands. With a parallel application, you compile and run it to produce a profile data file for each parallel task. You can then process one, some, or all the data files produced using either the **prof** or **gprof** commands. The following table describes how to profile parallel programs. For comparison, the steps involved in profiling a serial program are shown in the left-hand column of the table.

To Profile a Serial Program:	To Profile a Parallel Program:
<p>Step 1: Compile the application source code using the cc command with either the -p or -pg flag.</p>	<p>Step 1: Compile the application source code using the command mpcc (for C programs), mpCC (for C++ programs), or mpxlf (for Fortran programs) as described in <i>IBM Parallel Environment for AIX: Operation and Use, Volume 1, Using the Parallel Operating Environment</i>. You should use one of the standard profiling compiler options – either -p or -pg – on the compiler command. For more information on the compiler options -p and -pg, refer to their use on the cc command as described in <i>IBM AIX Version 4 Commands Reference</i> and <i>IBM AIX Version 4 General Programming Concepts: Writing and Debugging Programs</i>.</p>
<p>Step 2: Run the executable program to produce a profile data file. If you have compiled the source code with the -p option, the data file produced is named <i>mon.out</i>. If you have compiled the source code with the -pg option, the data file produced is named <i>gmon.out</i>.</p>	<p>Step 2: Before you run the parallel program, set the environment variable MP_EUILIBPATH=/usr/lpp/ppe.poe/lib/profiled:/usr/lib/profiled:/lib/profiled:/usr/lpp/ppe.poe/lib. If your message passing library is not in /usr/lpp/ppe.poe/lib, substitute your message passing library path. Run the parallel program. When the program ends, it generates a profile data file for each parallel task. The system gives unique names to the data files by appending each task's identifying number to <i>mon.out</i> or <i>gmon.out</i>. If you have compiled the source code with the -p option, the data files produced take the form:</p> <p style="text-align: center;">mon.out.taskid</p> <p>If the source code has been compiled with the -pg option, the data files produced take the form:</p> <p style="text-align: center;">gmon.out.taskid</p> <p>Note: The current directory must be writable from all remote nodes. Otherwise, the profile data files will have to be manually moved to the home node for analysis with prof and gprof. You can also use the mcpqath command to move the files. See <i>IBM Parallel Environment for AIX: Operation and Use, Volume 1, Using the Parallel Operating Environment</i> for more about mcpqath.</p>

To Profile a Serial Program:	To Profile a Parallel Program:
<p>Step 3: Use either the prof or the gprof command to process the profile data file. You use the prof command to process the <i>mon.out</i> data file, and the gprof command to process the <i>gmon.out</i> data file.</p>	<p>Step 3: Use either the prof or gprof command to process the profile data files. The prof command processes the <i>mon.out</i> data files, and gprof processes the <i>gmon.out</i> data files. You can process one, some, or all of the data files created during the run. You must specify the name(s) of the profile data file(s) to read, however, because the prof and gprof commands read <i>mon.out</i> or <i>gmon.out</i> by default. On the prof command, use the -m flag to specify the name(s) of the profile data file(s) it should read. For example, to specify the profile data file for task 0 with the prof command:</p> <pre>ENTER prof -m mon.out.0</pre> <p>You can also specify that the prof command should take profile data from some or all of the profile data files produced. For example, to specify three different profile data files – the ones associated with tasks 0, 1, and 2 – on the prof command:</p> <pre>ENTER prof -m mon.out.0 mon.out.1 mon.out.2</pre> <p>On the gprof command, you simply specify the name(s) of the profile data file(s) it should read on the command line. You must also specify the name of the program on the gprof command, but no option flag is needed. For example, to specify the profile data file for task 0 with the gprof command:</p> <pre>ENTER gprof program gmon.out.0</pre> <p>As with the prof command, you can also specify that the gprof command should take profile data from some or all of the profile data files produced. For example, to specify three different profile data files – the ones associated with tasks 0, 1, and 2 – on the gprof command:</p> <pre>ENTER gprof program gmon.out.0 gmon.out.1 gmon.out.2</pre>

The parallel utility, **mp_profile()**, may also be used to selectively profile portions of a program. To start profiling, call **mp_profile(1)**. To suspend profiling, call **mp_profile(0)**. The final profile data set will contain counts and CPU times for the program lines that are delimited by the start and stop calls. In C, the calls are **mpc_profile(1)**, and **mpc_profile(0)**. By default, profiling is active at the start of the user's executable.

Note: Like the sequential version of **prof/gprof**, if more than one profile file is specified, the parallel version of the **prof/gprof** command output shows the sum of the profile information in the given profile files. There is no statistical analysis contacted across the multiple profile files.

Glossary of Terms and Abbreviations

This glossary includes terms and definitions from:

- The *Dictionary of Computing*, New York: McGraw-Hill, 1994.
- The *American National Standard Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute (ANSI). Copies can be purchased from the American National Standards Institute, 1430 Broadway, New York, New York 10018. Definitions are identified by the symbol (A) after the definition.
- The *ANSI/EIA Standard - 440A: Fiber Optic Terminology*, copyright 1989 by the Electronics Industries Association (EIA). Copies can be purchased from the Electronic Industries Association, 2001 Pennsylvania Avenue N.W., Washington, D.C. 20006. Definitions are identified by the symbol (E) after the definition.
- The *Information Technology Vocabulary* developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

This section contains some of the terms that are commonly used in the Parallel Environment books and in this book in particular.

IBM is grateful to the American National Standards Institute (ANSI) for permission to reprint its definitions from the American National Standard *Vocabulary for Information Processing* (Copyright 1970 by American National Standards Institute, Incorporated), which was prepared by Subcommittee X3K5 on Terminology and Glossary of the American National Standards Committee X3. ANSI definitions are preceded by an asterisk (*).

Other definitions in this glossary are taken from *IBM Vocabulary for Data Processing, Telecommunications, and Office Systems* (GC20-1699).

A

address. A value, possibly a character or group of characters that identifies a register, a device, a particular part of storage, or some other data source or destination.

AIX. Abbreviation for Advanced Interactive Executive, IBM's licensed version of the UNIX operating system. AIX is particularly suited to support technical computing applications, including high function graphics and floating point computations.

AIXwindows Environment/6000. A graphical user interface (GUI) for the RS/6000. It has the following components:

- A graphical user interface and toolkit based on OSF/Motif
- Enhanced X-Windows, an enhanced version of the MIT X Window System
- Graphics Library (GL), a graphical interface library for the applications programmer which is compatible with Silicon Graphics' GL interface.

API. Application Programming Interface.

application. The use to which a data processing system is put; for example, topayroll application, an airline reservation application.

argument. A parameter passed between a calling program and a called program or subprogram.

attribute. A named property of an entity.

B

bandwidth. The total available bit rate of a digital channel.

blocking operation. An operation which does not complete until the operation either succeeds or fails. For example, a blocking receive will not return until a message is received or until the channel is closed and no further messages can be received.

breakpoint. A place in a program, specified by a command or a condition, where the system halts execution and gives control to the workstation user or to a specified program.

broadcast operation. A communication operation in which one processor sends (or broadcasts) a message to all other processors.

buffer. A portion of storage used to hold input or output data temporarily.

C

C. A general purpose programming language. It was formalized by ANSI standards committee for the C language in 1984 and by Uniform in 1983.

C++. A general purpose programming language, based on C, which includes extensions that support an object-oriented programming paradigm. Extensions include:

- strong typing
- data abstraction and encapsulation
- polymorphism through function overloading and templates
- class inheritance.

call arc. The representation of a call between two functions within the Xprofiler function call tree. It appears as a solid line between the two functions. The arrowhead indicates the direction of the call; the function it points to is the one that receives the call. The function making the call is known as the *caller*, while the function receiving the call is known as the *callee*.

chaotic relaxation. An iterative relaxation method which uses a combination of the Gauss-Seidel and Jacobi-Seidel methods. The array of discrete values is divided into sub-regions which can be operated on in parallel. The sub-region boundaries are calculated using Jacobi-Seidel, whereas the sub-region interiors are calculated using Gauss-Seidel. See also *Gauss-Seidel*.

client. A function that requests services from a server, and makes them available to the user.

cluster. A group of processors interconnected through a high speed network that can be used for high performance computing. It typically provides excellent price/performance.

collective communication. A communication operation which involves more than two processes or tasks. Broadcasts, reductions, and the MPI_Allreduce subroutine are all examples of collective communication operations. All tasks in a communicator must participate.

command alias. When using the PE command line debugger, pdbx, you can create abbreviations for existing commands using the **pdbx alias** command. These abbreviations are known as *command aliases*.

Communication Subsystem (CSS). A component of the IBM Parallel System Support Programs for AIX that provides software support for the High Performance Switch. It provides two protocols; IP (Internet Protocol)

for LAN based communication and US (user space) as a message passing interface that is optimized for performance over the switch. See also *Internet Protocol* and *User Space*.

communicator. An MPI object that describes the communication context and an associated group of processes.

compile. To translate a source program into an executable program.

condition. One of a set of specified values that a data item can assume.

control workstation. A workstation attached to the RS/6000 SP that serves as a single point of control allowing the administrator or operator to monitor and manage the system using IBM Parallel System Support Programs for AIX.

core dump. A process by which the current state of a program is preserved in a file. Core dumps are usually associated with programs that have encountered an unexpected, system-detected fault, such as a Segmentation Fault, or severe user error. The current program state is needed for the programmer to diagnose and correct the problem.

core file. A file which preserves the state of a program, usually just before a program is terminated for an unexpected error. See also *core dump*.

current context. When using either of the PE parallel debuggers, control of the parallel program and the display of its data can be limited to a subset of the tasks that belong to that program. This subset of tasks is called the *current context*. You can set the current context to be a single task, multiple tasks, or all the tasks in the program.

D

data decomposition. A method of breaking up (or decomposing) a program into smaller parts to exploit parallelism. One divides the program by dividing the data (usually arrays) into smaller parts and operating on each part independently.

data parallelism. Refers to situations where parallel tasks perform the same computation on different sets of data.

dbx. A symbolic command line debugger that is often provided with UNIX systems. The PE command line debugger, **pdbx**, is based on the **dbx** debugger.

debugger. A debugger provides an environment in which you can manually control the execution of a

program. It also provides the ability to display the program's data and operation.

distributed shell (dsh). An IBM Parallel System Support Programs for AIX command that lets you issue commands to a group of hosts in parallel. See the *IBM RISC System/6000 Scalable POWERparallel Systems: Command and Technical Reference* (GC23-3900-00) for details.

domain name. The hierarchical identification of a host system (in a network), consisting of human-readable labels, separated by decimals.

E

environment variable. 1. A variable that describes the operating environment of the process. Common environment variables describe the home directory, command search path, and the current time zone. 2. A variable that is included in the current software environment and is therefore available to any called program that requests it.

event. An occurrence of significance to a task; for example, the completion of an asynchronous operation such as an input/output operation.

Ethernet. Ethernet is the standard hardware for TCP/IP LANs in the UNIX marketplace. It is a 10 megabit per second baseband type network that uses the contention based CSMA/CD (collision detect) media access method.

executable. A program that has been link-edited and therefore can be run in a processor.

execution. To perform the actions specified by a program or a portion of a program.

expression. In programming languages, a language construct for computing a value from one or more operands.

F

fairness. A policy in which tasks, threads, or processes must be allowed eventual access to a resource for which they are competing. For example, if multiple threads are simultaneously seeking a lock, then no set of circumstances can cause any thread to wait indefinitely for access to the lock.

FDDI. Fiber distributed data interface (100 Mbit/s fiber optic LAN).

file system. In the AIX operating system, the collection of files and file management structures on a

physical or logical mass storage device, such as a diskette or minidisk.

fileset. 1) An individually installable option or update. Options provide specific function while updates correct an error in, or enhance, a previously installed product. 2) One or more separately installable, logically grouped units in an installation package. See also *Licensed Program Product* and *package*.

foreign host. See *remote host*.

Fortran. One of the oldest of the modern programming languages, and the most popular language for scientific and engineering computations. It's name is a contraction of *FORmula TRANslation*. The two most common Fortran versions are Fortran 77, originally standardized in 1978, and Fortran 90. Fortran 77 is a proper subset of Fortran 90.

function call tree. A graphical representation of all the functions and calls within an application, which appears in the Xprofiler main window. The functions are represented by green, solid-filled rectangles called function boxes. The size and shape of each function box indicates its CPU usage. Calls between functions are represented by blue arrows, called call arcs, drawn between the function boxes. See also *call arcs*.

function cycle. A chain of calls in which the first caller is also the last to be called. A function that calls itself recursively is not considered a function cycle.

functional decomposition. A method of dividing the work in a program to exploit parallelism. One divides the program into independent pieces of functionality which are distributed to independent processors. This is in contrast to data decomposition which distributes the same work over different data to independent processors.

functional parallelism. Refers to situations where parallel tasks specialize in particular work.

G

Gauss-Seidel. An iterative relaxation method for solving Laplace's equation. It calculates the general solution by finding particular solutions to a set of discrete points distributed throughout the area in question. The values of the individual points are obtained by averaging the values of nearby points. Gauss-Seidel differs from Jacobi-Seidel in that for the $i+1$ st iteration Jacobi-Seidel uses only values calculated in the i th iteration. Gauss-Seidel uses a mixture of values calculated in the i th and $i+1$ st iterations.

global max. The maximum value across all processors for a given variable. It is global in the sense that it is global to the available processors.

global variable. A variable defined in one portion of a computer program and used in at least one other portion of the computer program.

gprof. A UNIX command that produces an execution profile of C, Pascal, Fortran, or COBOL programs. The execution profile is in a textual and tabular format. It is useful for identifying which routines use the most CPU time. See the man page on **gprof**.

GUI (Graphical User Interface). A type of computer interface consisting of a visual metaphor of a real-world scene, often of a desktop. Within that scene are icons, representing actual objects, that the user can access and manipulate with a pointing device.

H

High Performance Switch. The high-performance message passing network, of the RS/6000 SP(SP) machine, that connects all processor nodes.

HIPPI. High performance parallel interface.

hook. **hook** is a **pdbx** command that allows you to re-establish control over all task(s) in the current context that were previously unhooked with this command.

home node. The node from which an application developer compiles and runs his program. The home node can be any workstation on the LAN.

host. A computer connected to a network, and providing an access method to that network. A host provides end-user services.

host list file. A file that contains a list of host names, and possibly other information, that was defined by the application which reads it.

host name. The name used to uniquely identify any computer on a network.

hot spot. A memory location or synchronization resource for which multiple processors compete excessively. This competition can cause a disproportionately large performance degradation when one processor that seeks the resource blocks, preventing many other processors from having it, thereby forcing them to become idle.

I

IBM Parallel Environment for AIX. A program product that provides an execution and development environment for parallel Fortran, C, or C++ programs. It also includes tools for debugging, profiling, and tuning parallel programs.

installation image. A file or collection of files that are required in order to install a software product on a RS/6000 workstation or on SP system nodes. These files are in a form that allows them to be installed or removed with the AIX **installp** command. See also *fileset*, *Licensed Program Product*, and *package*.

Internet. The collection of worldwide networks and gateways which function as a single, cooperative virtual network.

Internet Protocol (IP). 1) The TCP/IP protocol that provides packet delivery between the hardware and user processes. 2) The High Performance Switch library, provided with the IBM Parallel System Support Programs for AIX, that follows the IP protocol of TCP/IP.

IP. See *Internet Protocol*.

J

Jacobi-Seidel. See *Gauss-Seidel*.

job management system.

| The software you use to manage the jobs across your
| system, based on the availability and state of system
| resources.

K

Kerberos. A publicly available security and authentication product that works with the IBM Parallel System Support Programs for AIX software to authenticate the execution of remote commands.

kernel. The core portion of the UNIX operating system which controls the resources of the CPU and allocates them to the users. The kernel is memory-resident, is said to run in *kernel mode* (in other words, at higher execution priority level than *user mode*) and is protected from user tampering by the hardware.

L

Laplace's equation. A homogeneous partial differential equation used to describe heat transfer, electric fields, and many other applications.

latency. The time interval between the instant at which an instruction control unit initiates a call for data transmission, and the instant at which the actual transfer of data (or receipt of data at the remote end) begins. Latency is related to the hardware characteristics of the system and to the different layers of software that are involved in initiating the task of packing and transmitting the data.

Licensed Program Product (LPP). A collection of software packages, sold as a product, that customers pay for to license. It can consist of packages and filesets a customer would install. These packages and filesets bear a copyright and are offered under the terms and conditions of a licensing agreement. See also *fileset* and *package*.

LoadLeveler. A job management system that works with POE to allow users to run jobs and match processing needs with system resources, in order to better utilize the system.

local variable. A variable that is defined and used only in one specified portion of a computer program.

loop unrolling. A program transformation which makes multiple copies of the body of a loop, placing the copies also within the body of the loop. The loop trip count and index are adjusted appropriately so the new loop computes the same values as the original. This transformation makes it possible for a compiler to take additional advantage of instruction pipelining, data cache effects, and software pipelining.

See also *optimization*.

M

menu. A list of options displayed to the user by a data processing system, from which the user can select an action to be initiated.

message catalog. A file created using the AIX Message Facility from a message source file that contains application error and other messages, which can later be translated into other languages without having to recompile the application source code.

message passing. Refers to the process by which parallel tasks explicitly exchange program data.

MIMD (Multiple Instruction Multiple Data). A parallel programming model in which different processors perform different instructions on different sets of data.

MPMD (Multiple Program Multiple Data). A parallel programming model in which different, but related, programs are run on different sets of data.

MPI. Message Passing Interface; a standardized API for implementing the message passing model.

N

network. An interconnected group of nodes, lines, and terminals. A network provides the ability to transmit data to and receive data from other systems and users.

node. (1) In a network, the point where one or more functional units interconnect transmission lines. A computer location defined in a network. (2) In terms of the RS/6000 SP, a single location or workstation in a network. An SP node is a physical entity (a processor).

node ID. A string of unique characters that identifies the node on a network.

nonblocking operation. An operation, such as sending or receiving a message, which returns immediately whether or not the operation was completed. For example, a nonblocking receive will not wait until a message is sent, but a blocking receive will wait. A nonblocking receive will return a status value that indicates whether or not a message was received.

O

object code. The result of translating a computer program to a relocatable, low-level form. Object code contains machine instructions, but symbol names (such as array, scalar, and procedure names), are not yet given a location in memory.

optimization. A not strictly accurate but widely used term for program performance improvement, especially for performance improvement done by a compiler or other program translation software. An optimizing compiler is one that performs extensive code transformations in order to obtain an executable that runs faster but gives the same answer as the original. Such code transformations, however, can make code debugging and performance analysis very difficult because complex code transformations obscure the correspondence between compiled and original source code.

option flag. Arguments or any other additional information that a user specifies with a program name. Also referred to as *parameters* or *command line options*.

P

package. A number of filesets that have been collected into a single installable image of program products, or LPPs. Multiple filesets can be bundled together for installing groups of software together. See also *fileset* and *Licensed Program Product*.

parallelism. The degree to which parts of a program may be concurrently executed.

parallelize. To convert a serial program for parallel execution.

Parallel Operating Environment (POE). An execution environment that smooths the differences between serial and parallel execution. It lets you submit and manage parallel jobs. It is abbreviated and commonly known as POE.

parameter. * (1) In Fortran, a symbol that is given a constant value for a specified application. (2) An item in a menu for which the operator specifies a value or for which the system provides a value when the menu is interpreted. (3) A name in a procedure that is used to refer to an argument that is passed to the procedure. (4) A particular piece of information that a system or application program needs to process a request.

partition. (1) A fixed-size division of storage. (2) In terms of the RS/6000 SP, a logical definition of nodes to be viewed as one system or domain. System partitioning is a method of organizing the SP into groups of nodes for testing or running different levels of software of product environments.

Partition Manager. The component of the Parallel Operating Environment (POE) that allocates nodes, sets up the execution environment for remote tasks, and manages distribution or collection of standard input (STDIN), standard output (STDOUT), and standard error (STDERR).

pdbx. **pdbx** is the parallel, symbolic command line debugging facility of PE. **pdbx** is based on the **dbx** debugger and has a similar interface.

PE. The IBM Parallel Environment for AIX program product.

performance monitor. A utility which displays how effectively a system is being used by programs.

POE. See Parallel Operating Environment.

pool. Groups of nodes on an SP that are known to the Resource Manager, and are identified by a number.

point-to-point communication. A communication operation which involves exactly two processes or tasks. One process initiates the communication through a *send* operation. The partner process issues a *receive* operation to accept the data being sent.

procedure. (1) In a programming language, a block, with or without formal parameters, whose execution is invoked by means of a procedure call. (2) A set of

related control statements that cause one or more programs to be performed.

process. A program or command that is actually running the computer. It consists of a loaded version of the executable file, its data, its stack, and its kernel data structures that represent the process's state within a multitasking environment. The executable file contains the machine instructions (and any calls to shared objects) that will be executed by the hardware. A process can contain multiple threads of execution.

The process is created via a **fork()** system call and ends using an **exit()** system call. Between **fork** and **exit**, the process is known to the system by a unique process identifier (pid).

Each process has its own virtual memory space and cannot access another process's memory directly. Communication methods across processes include pipes, sockets, shared memory, and message passing.

prof. A utility which produces an execution profile of an application or program. It is useful to identifying which routines use the most CPU time. See the man page for **prof**.

profiling. The act of determining how much CPU time is used by each function or subroutine in a program. The histogram or table produced is called the execution profile.

Program Marker Array. An X-Windows run time monitor tool provided with Parallel Operating Environment, used to provide immediate visual feedback on a program's execution.

pthread. A thread that conforms to the POSIX Threads Programming Model.

R

reduction operation. An operation, usually mathematical, which reduces a collection of data by one or more dimensions. For example, the arithmetic SUM operation is a reduction operation which reduces an array to a scalar value. Other reduction operations include MAXVAL and MINVAL.

remote host. Any host on a network except the one at which a particular operator is working.

remote shell (rsh). A command supplied with both AIX and the IBM Parallel System Support Programs for AIX that lets you issue commands on a remote host.

Report. In Xprofiler, a tabular listing of performance data that is derived from the gmon.out files of an application. There are five types of reports that are generated by Xprofiler, and each one presents different statistical information for an application.

Resource Manager. A server that runs on one of the nodes of an RS/6000 SP (SP) machine. It prevents parallel jobs from interfering with each other, and reports job-related node information.

RISC. Reduced Instruction Set Computing (RISC), the technology for today's high performance personal computers and workstations, was invented in 1975.

S

shell script. A sequence of commands that are to be executed by a shell interpreter such as C shell, Korn shell, or Bourne shell. Script commands are stored in a file in the same form as if they were typed at a terminal.

segmentation fault. A system-detected error, usually caused by referencing an invalid memory address.

server. A functional unit that provides shared services to workstations over a network; for example, a file server, a print server, a mail server.

signal handling. A type of communication that is used by message passing libraries. Signal handling involves using AIX signals as an asynchronous way to move data in and out of message buffers.

source line. A line of source code.

source code. The input to a compiler or assembler, written in a source language. Contrast with object code.

SP. RS/6000 SP; a scalable system from two to 128 processor nodes, arranged in various physical configurations, that provides a high powered computing environment.

SPMD (Single Program Multiple Data). A parallel programming model in which different processors execute the same program on different sets of data.

standard input (STDIN). In the AIX operating system, the primary source of data entered into a command. Standard input comes from the keyboard unless redirection or piping is used, in which case standard input can be from a file or the output from another command.

standard output (STDOUT). In the AIX operating system, the primary destination of data produced by a command. Standard output goes to the display unless redirection or piping is used, in which case standard output can go to a file or to another command.

stencil. A pattern of memory references used for averaging. A 4-point stencil in two dimensions for a given array cell, $x(i,j)$, uses the four adjacent cells, $x(i-1,j)$, $x(i+1,j)$, $x(i,j-1)$, and $x(i,j+1)$.

subroutine. (1) A sequence of instructions whose execution is invoked by a call. (2) A sequenced set of instructions or statements that may be used in one or more computer programs and at one or more points in a computer program. (3) A group of instructions that can be part of another routine or can be called by another program or routine.

synchronization. The action of forcing certain points in the execution sequences of two or more asynchronous procedures to coincide in time.

system administrator. (1) The person at a computer installation who designs, controls, and manages the use of the computer system. (2) The person who is responsible for setting up, modifying, and maintaining the Parallel Environment.

System Data Repository. A component of the IBM Parallel System Support Programs for AIX software that provides configuration management for the SP system. It manages the storage and retrieval of system data across the control workstation, file servers, and nodes.

System Status Array. An X-Windows run time monitor tool, provided with the Parallel Operating Environment, that lets you quickly survey the utilization of processor nodes.

T

task. A unit of computation analogous to an AIX process.

thread. A single, separately dispatchable, unit of execution. There may be one or more threads in a process, and each thread is executed by the operating system concurrently.

tracing. In PE, the collection of data for the Visualization Tool (VT). The program is *traced* by collecting information about the execution of the program in trace records. These records are then accumulated into a trace file which a user visualizes with VT.

tracepoint. Tracepoints are places in the program that, when reached during execution, cause the debugger to print information about the state of the program.

trace record. In PE, a collection of information about a specific event that occurred during the execution of your program. For example, a trace record is created for each send and receive operation that occurs in your program (this is optional and may not be appropriate). These records are then accumulated into a trace file which allows the Visualization Tool to visually display the communications patterns from the program.

U

unrolling loops. See *loop unrolling*.

US. See *user space*.

user. (1) A person who requires the services of a computing system. (2) Any person or any thing that may issue or receive commands and message to or from the information processing system.

user space (US). A version of the message passing library that is optimized for direct access to the SP High Performance Switch, that maximizes the performance capabilities of the SP hardware.

utility program. A computer program in general support of computer processes; for example, a diagnostic program, a trace program, a sort program.

utility routine. A routine in general support of the processes of a computer; for example, an input routine.

V

variable. (1) In programming languages, a named object that may take different values, one at a time. The values of a variable are usually restricted to one data type. (2) A quantity that can assume any of a given set of values. (3) A name used to represent a data item whose value can be changed while the program is running. (4) A name used to represent data whose

value can be changed, while the program is running, by referring to the name of the variable.

view. (1) In an information resource directory, the combination of a variation name and revision number that is used as a component of an access name or of a descriptive name.

Visualization Tool. The PE Visualization Tool. This tool uses information that is captured as your parallel program executes, and presents a graphical display of the program execution. For more information, see *IBM Parallel Environment for AIX: Operation and Use, Volume 2, Tools Reference*

VT. See *Visualization Tool*.

X

X Window System. The UNIX industry's graphics windowing standard that provides simultaneous views of several executing programs or processes on high resolution graphics displays.

xpdbx. This is the former name of the PE graphical interface debugging facility, which is now called **pedb**.

Xprofiler. An AIX tool that is used to analyze the performance of both serial and parallel applications, via a graphical user interface. Xprofiler provides quick access to the profiled data, so that the functions that are the most CPU-intensive can be easily identified.

Index

A

AIX profilers
 gprof 1
 prof 1

C

commands, PE 63
conventions x
customizing resources 67
customizing tool resources 67

G

gprof 1

I

IBM Parallel Environment for AIX ix

M

mixed system ix

P

Parallel Operating Environment (POE) ix
parallel profiling capability 75
parallel programs
 profiling 1
PE commands 63
 xprofiler 63
POE environment variables
 MP_EUILIBPATH 75
prof 1
profilers, AIX 1
profiling parallel programs 1
publications, related x

R

resource settings 67
resources, customizing 67

S

serial program 75

T

trademarks v

X

Xprofiler 1
xresources, customizing 67

Communicating Your Comments to IBM

IBM Parallel Environment for AIX
Operation and Use, Volume 2, Part 2
Profiling
Version 2 Release 4
Publication No. SC28-1980-02

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you are mailing a reader's comment form (RCF) from a country other than the United States, you can give the RCF to the local IBM branch office or IBM representative for postage-paid mailing.

- If you prefer to send comments by mail, use the RCF at the back of this book.
- If you prefer to send comments by FAX, use this number:
 - FAX: (International Access Code)+1+914+432-9405
- If you prefer to send comments electronically, use this network ID:
 - IBM Mail Exchange: USIB6TC9 at IBMMAIL
 - Internet e-mail: mhvrcfs@us.ibm.com
 - World Wide Web: <http://www.s390.ibm.com/os390>

Make sure to include the following in your note:

- Title and publication number of this book
- Page number or topic to which your comment applies

Optionally, if you include your telephone number, we will be able to respond to your comments by phone.

Reader's Comments — We'd Like to Hear from You

**IBM Parallel Environment for AIX
Operation and Use, Volume 2, Part 2
Profiling
Version 2 Release 4
Publication No. SC28-1980-02**

You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Today's date: _____

What is your occupation?

Newsletter number of latest Technical Newsletter (if any) concerning this publication:

How did you use this publication?

- | | | | |
|--------------------------|-------------------------------|--------------------------|------------------------|
| <input type="checkbox"/> | As an introduction | <input type="checkbox"/> | As a text (student) |
| <input type="checkbox"/> | As a reference manual | <input type="checkbox"/> | As a text (instructor) |
| <input type="checkbox"/> | For another purpose (explain) | | |

Is there anything you especially like or dislike about the organization, presentation, or writing in this manual? Helpful comments include general usefulness of the book; possible additions, deletions, and clarifications; specific errors and omissions.

Page Number:

Comment:

Name

Address

Company or Organization

Phone No.

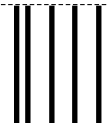


Cut or Fold
Along Line

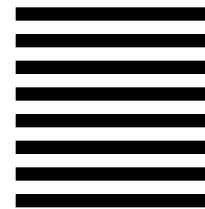
Fold and Tape

Please do not staple

Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM Corporation
Department 55JA, Mail Station P384
522 South Road
Poughkeepsie NY 12601-5400



Fold and Tape

Please do not staple

Fold and Tape

Cut or Fold
Along Line



Program Number: 5765-543



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC28-1980-02

