This manual describes tools for analyzing and tuning the performance of parallel vector application programs. To help clarify the purposes, features, and requirements of the tools, this chapter discusses the following material:

- Performance measurement and accuracy
- Tools described in this manual
- Tools described in other manuals
- Tool interfaces
- Impact of tools
- Selection of tools
- Using the X Window System interface
- Component naming conventions

The following tools are described in this manual in alphabetical order:

- `atexpert(1)` - Gauges the effectiveness of Autotasking, by predicting the speedup in a dedicated environment
- `flowview(1)` - Displays times of subroutines and functions instrumented by the Flowtrace feature
- `hpm(1)` - Monitors machine performance during program execution
- `jumpview(1)` - Displays times of subroutines and functions generated by the Jumptrace feature
- `perfvie(1)` - Displays times of subroutines and functions instrumented by the Pertrace feature and the Hardware Performance Monitor device
- `procview(1)` - Displays program execution statistics on I/O, process, and memory generated by the `procstat(1)` command
- `profview(1)` - Displays the relative execution time used by individual parts of your program and generated by the Profiling feature

Table 1, page 3, through Table 5, page 8, in the following sections help define and differentiate the commands, subroutines, and features that are used in
performance tuning. Within the tables, the tools are listed alphabetically, irrespective of capitalization.

1.1 Performance Measurement and Accuracy

Tuning your code for better performance provides the following benefits:

- Reduces job turn-around time
- Makes efficient use of the computing resources available

Most of the tools in this manual measure or report information on the run-time performance of an application running under the UNICOS operating system. The performance data may reflect CPU execution time, multitasking efficiency, I/O use, or other types of data. When you use the results of these tools, it is important to remember that the numbers presented are not exact but are affected by the following factors:

- Instrumentation technique
- Granularity of information collection (for example, collecting from the program versus a routine or loop), especially compared to instrumentation overhead
- Other processes running on the UNICOS system
- Statistical variation

These factors produce the following inaccuracies:

- Numerical results from two runs of the same program with the same instrumentation technique will likely differ.
- Results from two runs of the same program with different instrumentation techniques, even if they purport to measure the same factors, will likely differ.

In most practical situations, these variations are not important. For example, although one tool indicates that a certain loop takes 25% of a program’s CPU time and a second tool indicates 27%, both results identify a loop that should be examined. Further, the fact that differences exist may provide you with additional information.

For example, the Profiling feature can show a heavily used routine, but the Flowtrace feature can help identify a single call in the source code that is accounting for most of the use of that routine. The fact that the numbers
displayed are different reflects different views of the code, not that one is more correct than the other.

1.2 Tools Described in This Manual

For every tool that gathers data from the execution of programs, a corresponding data reporting tool exists. Table 1, page 3 lists the tools that are described in this manual.

Table 1. Tools Described in This Manual

<table>
<thead>
<tr>
<th>Tool</th>
<th>Type</th>
<th>Description</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>atexpert(l)</td>
<td>Command</td>
<td>Predicts speedups of parallel processing in an autotasked program. The prediction is for a dedicated system and is made from data collected from one run on a system that is not dedicated.</td>
<td>Chapter 2, page 19</td>
</tr>
<tr>
<td>flowview(l)</td>
<td>Command</td>
<td>Displays information from the raw data generated by the Flowtrace library.</td>
<td>Chapter 3, page 67</td>
</tr>
<tr>
<td>Flowtrace</td>
<td>Feature</td>
<td>Times subroutines and functions during program execution. The flowview(l) command displays Flowtrace data.</td>
<td>Chapter 3, page 67</td>
</tr>
<tr>
<td>FLOWMARK</td>
<td>Subroutine</td>
<td>Times specified areas of programs (not necessarily subroutines or functions). FLOWMARK is a special feature of Flowtrace and Pertrace.</td>
<td>Chapter 3, page 67</td>
</tr>
<tr>
<td>hpm(l)</td>
<td>Command</td>
<td>Gathers Hardware Performance Monitor (HPM) statistics for an entire program. The perfview(l) command displays data from the hpm(l) command.</td>
<td>Chapter 4, page 97</td>
</tr>
<tr>
<td>jumpview(l)</td>
<td>Command</td>
<td>Displays information from the raw data generated by the jt(l) command.</td>
<td>Chapter 5, page 103</td>
</tr>
<tr>
<td>jt(l)</td>
<td>Command</td>
<td>Gathers timing statistics on code during execution. The jumpview(l) command displays data generated by the jt(l) command.</td>
<td>Chapter 5, page 103</td>
</tr>
<tr>
<td>Jumptrace</td>
<td>Feature</td>
<td>Provides exact timings of subroutines and functions during program execution. The jumpview(l) command displays Jumptrace data gathered by the jt(l) command.</td>
<td>Chapter 5, page 103</td>
</tr>
<tr>
<td>Tool</td>
<td>Type</td>
<td>Description</td>
<td>Section</td>
</tr>
<tr>
<td>-------------</td>
<td>--------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------</td>
</tr>
<tr>
<td>perfview(1)</td>
<td>Command</td>
<td>Displays information from raw data generated by the Pertrace library, the hpm(1) command, or the hpmall(8) administrator command.</td>
<td>Chapter 6, page 135</td>
</tr>
<tr>
<td>Pertrace</td>
<td>Feature</td>
<td>Times subroutines and functions based on statistics gathered from the Hardware Performance Monitor (HPM) device. The perfview(1) command displays Pertrace data.</td>
<td>Chapter 6, page 135</td>
</tr>
<tr>
<td>procview(1)</td>
<td>Command</td>
<td>Displays information from the raw data generated by the procstat(1) command.</td>
<td>Chapter 7, page 197</td>
</tr>
<tr>
<td>procstat(1)</td>
<td>Command</td>
<td>Gathers statistics about I/O, process, and memory as a program executes. The procview(1) command displays data generated by the procstat(1) command.</td>
<td>Chapter 7, page 197</td>
</tr>
<tr>
<td>profview(1)</td>
<td>Command</td>
<td>Displays information from the raw data generated by the prof(1) command.</td>
<td>Chapter 8, page 219</td>
</tr>
<tr>
<td>Profiling</td>
<td>Feature</td>
<td>Statistically gathers the relative execution time used by individual parts of a program. The profview(1) command displays Profiling data.</td>
<td>Chapter 8, page 219</td>
</tr>
</tbody>
</table>

### 1.3 Tools Described in Other Manuals

Table 2, page 5, lists tools that are related to performance analysis but are not described in this manual. See the associated man pages for information on these features.
Table 2. Other Performance Analysis Tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftnlist(1)</td>
<td>Command</td>
<td>Generates a Fortran listing.</td>
</tr>
<tr>
<td>hpmall(8)</td>
<td>Command</td>
<td>Gathers Hardware Performance Monitor (HPM) statistics for the entire system workload. The perfview(1) command displays data generated by the hpmall(8) command.</td>
</tr>
<tr>
<td>hpmflop(1)</td>
<td>Command</td>
<td>Reports hardware performance statistics for individual user jobs.</td>
</tr>
<tr>
<td>ja(1)</td>
<td>Command</td>
<td>Provides accounting data about many aspects of job processing.</td>
</tr>
<tr>
<td>mtdump(1)</td>
<td>Command</td>
<td>Produces various noninteractive reports from data written by the multitasking history trace feature.</td>
</tr>
<tr>
<td>SECOND(3)</td>
<td>Subroutine</td>
<td>Reports elapsed CPU time accumulated by all processes in a program.</td>
</tr>
<tr>
<td>time(1)</td>
<td>Command</td>
<td>Reports elapsed and execution times for a command.</td>
</tr>
<tr>
<td>xbrowse(1)</td>
<td>Command</td>
<td>Provides an interactive, graphic interface in which to view and edit Fortran.</td>
</tr>
</tbody>
</table>

1.4 Tool Interfaces

Table 3, page 6, identifies the user interfaces available for the performance tools that display output. Other tools, which gather data to send to a file, are not listed here.

The headings in Table 3, page 6, have the following meanings:

- In the Command line column, "Yes" means that command-line options can be used to produce reports in a file.
- In the Line-mode interactive column, "Yes" means that interactive commands can be used to control output to a terminal.
- In the X Window System column, "Yes" means that an X Window System interface can be used to control and display tool output.
### 1.5 Impact of Tools

Table 4, page 7, itemizes some considerations and requirements for using the performance tools that run with programs. The table does not include reporting tools.

In Table 4, page 7, the headings have the following meanings:

- In the CPU overhead column, "Low" indicates the tool incurs no significant overhead; "High" indicates the tool does incur significant overhead.

- In the Recompile, Reload, and Change source columns, "Yes" indicates the tool requires program recompiling or reloading, or source code changes; a dash (−) indicates it does not require program recompiling or reloading, or source code changes.

- In the Multitask column, "Yes" means the tool operates correctly with multitasked programs. The abbreviation "Cond" (conditional) indicates that

---

1 To report hpm and hpmall command output, use the perfview command.
the tools (Perftrace and procstat) can be run successfully with a multitasked program if the NCPUS environment variable is set to 1 before the user program begins execution.

Table 4. Performance Impact

<table>
<thead>
<tr>
<th>Tool</th>
<th>CPU overhead</th>
<th>Recompile</th>
<th>Reload</th>
<th>Change source</th>
<th>Multitask</th>
</tr>
</thead>
<tbody>
<tr>
<td>atexpert</td>
<td>Low $^2$</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>Flowtrace</td>
<td>High</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
<td>Yes 3 Cond</td>
</tr>
<tr>
<td>ja</td>
<td>Low</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>Jumptrace</td>
<td>High</td>
<td>Recommended</td>
<td>Yes</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>Perftrace</td>
<td>High</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
<td>Cond</td>
</tr>
<tr>
<td>FLOWMARK</td>
<td>High</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Cond</td>
</tr>
<tr>
<td>procstat</td>
<td>Low</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Cond</td>
</tr>
<tr>
<td>Profiling</td>
<td>Low</td>
<td>Recommended</td>
<td>Yes</td>
<td>-</td>
<td>Yes</td>
</tr>
</tbody>
</table>

1.6 Selection of Tools

Table 5, page 8, categorizes the tools according to the information they provide. The categories appear from general to specific; the tools named for each category are in the suggested order of use.

$^2$ Overhead is inversely proportional to the granularity and number of invocations of parallel regions.

$^3$ On UNICOS systems, the Flowtrace feature works with all types of Autotasking (automatically detected tasking and directive-based tasking), and with macrotasking that involves calls to the tskstart(3) routine, but it does not work with multitasking using the t_fork(3) routine.
Table 5. Information Provided by the Tools

<table>
<thead>
<tr>
<th>To identify . . .</th>
<th>Use these tools</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU time used by an entire program</strong></td>
<td>/bin/time (not C shell time)</td>
</tr>
<tr>
<td></td>
<td>ja</td>
</tr>
<tr>
<td></td>
<td>STOP Fortran statement ⁴</td>
</tr>
<tr>
<td><strong>Routine using the most CPU time</strong></td>
<td>Profiling (prof and profview)</td>
</tr>
<tr>
<td></td>
<td>Flowtrace (flowview)</td>
</tr>
<tr>
<td></td>
<td>Perfrace (perfview)</td>
</tr>
<tr>
<td></td>
<td>atexpert (used with Autotasking)</td>
</tr>
<tr>
<td></td>
<td>Jumptrace (jt and jumpview)</td>
</tr>
<tr>
<td><strong>CPU time used by one routine</strong></td>
<td>Flowtrace (flowview)</td>
</tr>
<tr>
<td></td>
<td>Perfrace (perfview)</td>
</tr>
<tr>
<td></td>
<td>FLOWMARK (Flowtrace, Perfrace)</td>
</tr>
<tr>
<td></td>
<td>Profiling (prof and profview)</td>
</tr>
<tr>
<td></td>
<td>Jumptrace (jt and jumpview)</td>
</tr>
<tr>
<td></td>
<td>atexpert (used with Autotasking)</td>
</tr>
<tr>
<td><strong>CPU time used in one loop or code section</strong></td>
<td>Profiling (prof and profview) FLOWMARK (Flowtrace, Perfrace)</td>
</tr>
<tr>
<td></td>
<td>Jumptrace (jt and jumpview)</td>
</tr>
<tr>
<td></td>
<td>atexpert (used with Autotasking)</td>
</tr>
<tr>
<td><strong>CPU time used in library routines</strong></td>
<td>Profiling (prof and profview)</td>
</tr>
<tr>
<td></td>
<td>Jumptrace (jt and jumpview)</td>
</tr>
<tr>
<td><strong>Routines that you should consider inlining</strong></td>
<td>Flowtrace (flowview)</td>
</tr>
<tr>
<td></td>
<td>Perfrace (perfview)</td>
</tr>
<tr>
<td></td>
<td>Jumptrace (jt and jumpview)</td>
</tr>
<tr>
<td><strong>Program’s call tree (dynamic)</strong></td>
<td>Flowtrace (flowview)</td>
</tr>
<tr>
<td><strong>Efficiency of system hardware use</strong></td>
<td>hpm (perfview)</td>
</tr>
<tr>
<td></td>
<td>Perfrace (perfview)</td>
</tr>
<tr>
<td></td>
<td>Jumptrace (jt and jumpview)</td>
</tr>
<tr>
<td><strong>Efficiency of parallel processing</strong></td>
<td>atexpert (used with Autotasking)</td>
</tr>
<tr>
<td></td>
<td>ja</td>
</tr>
<tr>
<td></td>
<td>hpm</td>
</tr>
<tr>
<td></td>
<td>mtdump (used with macrotasking)</td>
</tr>
</tbody>
</table>

⁴ See your CP90 language reference manual for more information on the STOP statement.
To identify . . .                Use these tools
Amount and speed of program I/O     procstat and procview
                                             ja
Amount of central memory used     procstat and procview
                                             STOP Fortran statement 4
Process creation/termination history  procstat and procview
                                             ja
Scope of variables for Autotasking   xbrowse
Use of Fortran common variables throughout program xbrowse

1.7 Using the X Window System Interface

This section assumes that you have the X Window System running on your
workstation and are familiar with its operation. The utilities referred to in this
chapter (and documented elsewhere in this manual) run on UNICOS systems
and communicate with the X Window System server software on your
workstation. These performance utilities have been run with workstations
manufactured by several vendors, but differences in implementation of the X
Window System may affect the behavior, appearance, or in extreme cases, the
usability of the utilities on a particular system. If you experience difficulty
running the utilities on your workstation, report the problem(s) to your system
support staff.

The X Window System was developed at the Massachusetts Institute of
Technology in 1984. It has been ported to most UNIX operating systems, and it
also runs under the VMS operating system and the MS-DOS operating system.
The UNICOS systems are based on the UNIX operating system, which also
supports the X Window System. For more information on the X Window
System, see any of the following O’Reilly and Associates, Inc., publications:

• XLIB Programming Manual
• X Window System User’s Guide
• X Toolkit Intrinsic Programming Manual

The following components of the X Window System interface will be discussed:
• Getting started
• Common options
Guide to Parallel Vector Applications

- Resources
- Troubleshooting
- Color
- Pointing device

1.7.1 Getting Started

To use the X Window System interface with the performance utilities, you must be using a workstation that is running the X Window System, version 11, release 3 or after. If your workstation does not support the X Window System, you cannot use this interface to the performance utilities, and you must use the line-mode version (see individual descriptions of the performance utilities). Contact your system administrator for information about the availability of the X Window System.

When your workstation is running the X Window System, use TCP/IP to establish a session on the UNICOS system. Be sure that the DISPLAY environment variable specifies the screen on which your performance utilities window should be presented. For example, if your workstation name is ursa, set the environment variable as follows:

Standard shell (sh) and Korn (ksh) shells:

```bash
$ DISPLAY=ursa:0.0; export DISPLAY
```

C shell (csh):

```csh
% setenv DISPLAY ursa:0.0
```

You must ensure that the system can make connections to the X Window System fileserver by executing the `xhost` client command on your workstation. Many users include such a command in their `.xinitrc` file. The following example adds host sn2022 to the access control list for the workstation:

```bash
% xhost + sn2022
```

The `xhost(1)` command is invoked on your workstation, not on the system.

When you enter any of the performance utility commands noted in this manual, the command detects whether to invoke the X Window System after checking the following options in the order listed:
1. If the \(-L\) option is specified on a performance utility command line, the utility assumes that you want the line-mode interface and does not attempt to run the X Window System mode.

2. If the \(-L\) option is not specified on the utility command line, the utility first checks for the presence of the \(\text{DISPLAY}\) environment variable and/or the \(-\text{display}\) command-line option. If either of these is specified, the tool will try to open contact with the specified workstation. If this is successful, the tool will run in the X Window System mode. If the contact is unsuccessful, the tool will run in interactive line-mode.

3. If you specify any of the following options on the command line, the utility assumes that you want the noninteractive command-line mode (\(\text{flowview, jumpview, procview, perfview, or profview}\)) or interactive line-mode (\(\text{ATEXpert}\)):

   \text{flowview: } -A, -a, -c, -H, -h, -j, -k, -l, -m, -T, -u, -v
   \text{jumpview: } -A, -c, -H
   \text{perfview: } -A, -a, -B, -c, -C, -H, -h, -l, -M, -m, -U, -u, -v
   \text{procview: } -a, -c, -E, -F, -H, -O, -P, -S, -z
   \text{profview: } -A, -a, -B, -b, -c, -D, -d, -H, -h, -i, -m, -y, -z
   \text{atexpert: } -r \text{ (can be used alone or with } -f\text{)}

4. If you specify none of the options listed in the previous step, the utility is usually invoked with the line-mode interface (if available).

5. If no options are specified on the command line, it defaults to an X Window System interface.

A few seconds after the performance utility is invoked with the X Window System interface, a cursor appears in the upper left corner of your screen. Using the mouse, move the cursor to the desired screen position and click the left mouse button. The performance utility window appears on your screen at the indicated position.

The behavior discussed in this section is for the \(\text{twm}\) window manager. Other window managers behave differently when new windows appear on the screen. For example, some place all windows in the upper left-hand corner of the screen, making you responsible for moving the window to the desired location. In some cases, you can specify the desired location by using the \(-\text{geometry}\) option on the command line. In this case, you should specify only the location.
Do not override the window size. The following example causes the window to appear at location X=100, Y=100 on the screen.

```
$ profview -geometry +100+100 raw.file
```

### 1.7.2 Common Options

Many X Window System client programs use a common set of names for their command-line arguments. The X Toolkit automatically handles the following arguments, which you can enter directly on the command line. These values override values set in your `.Xdefaults` file (if any). In many cases, you can provide abbreviations for these options (for example, `-g` or `-geo` rather than `-geometry`) but you should use the names and abbreviations as documented in the following list. Table 6, page 12, provides the common command-line arguments that the X Toolkit handles.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-bg color</code> or <code>-background color</code></td>
<td>Either option specifies the window background color, color.</td>
</tr>
<tr>
<td><code>-bd color</code> or <code>-bordercolor color</code></td>
<td>Either option specifies the window border color, color.</td>
</tr>
<tr>
<td><code>-bw number</code> or <code>-borderwidth number</code></td>
<td>Either option specifies the window border width, number, in pixels.</td>
</tr>
<tr>
<td><code>-display display</code></td>
<td>Specifies the name of the X Window System fileservr to use.</td>
</tr>
<tr>
<td><code>-fg color</code> or <code>-foreground color</code></td>
<td>Either option specifies the text or graphics color, color.</td>
</tr>
<tr>
<td><code>-fn font</code> or <code>-font font</code></td>
<td>Either option specifies the display text font, font.</td>
</tr>
<tr>
<td><code>-geometry geometry</code></td>
<td>Specifies the main window’s initial size and location. Use this option to change only the window location. Changing the window size with <code>-geometry</code> can interfere with the display of graphics.</td>
</tr>
<tr>
<td><code>-iconic</code></td>
<td>Specifies that your application should start out in an iconic state. The window manager that your system uses controls how this state is represented.</td>
</tr>
</tbody>
</table>
### 1.7.3 Resources

The man page for each tool describes the resources that may be changed to alter the visual appearance of an X Window system client. This is specific to each tool.

The man page for each tool describes the resources that may be changed, specific to each tool. You also can change the resources for common screen items used by a tool (for example, a text display window). These resources are specified in your `.Xdefaults` file. This file can reside in your home directory on the UNICOS system, or it can reside on your X Window System server (workstation). To make the file global, execute the `xrdp` command.

The contents of the `.Xdefaults` file is a series of ASCII records, usually generated using a text editor. A typical record contains an indicator of the application being changed, linked with the resource that is recognized by the application. This name is followed by a colon (:). The specified value for the resource follows the colon.

For example, the following line changes the width of the `Flowview` main text window to 900 pixels. The application's generic name is `Flowview`. The name of the resource checked by the tool is `mainTextWindowWidth`. These are

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-name name</code></td>
<td>Selects the name, <code>name</code>, under which resources for the application should be found. This option is useful when using shell aliases to distinguish between invocations of an application, and it eliminates the need to resort to creating links to alter the executable file name.</td>
</tr>
<tr>
<td><code>-rv</code> or <code>-reverse</code></td>
<td>Either option instructs the program to simulate reverse video, if possible, sometimes by swapping foreground and background colors. Usually, it is used only on monochromatic displays, and it may not be implemented correctly on all systems.</td>
</tr>
<tr>
<td><code>+rv</code></td>
<td>Instructs the program not to simulate reverse video. If the reverse video does not work correctly on your system, you can use this to override any defaults.</td>
</tr>
<tr>
<td><code>-title string</code></td>
<td>Specifies the window title as <code>string</code>. Depending on the window manager used, you may display the title information as a window header.</td>
</tr>
<tr>
<td><code>-xrm resourcestring</code></td>
<td>Specifies a name and value for the resource to override any defaults. You can also use it for setting resources that do not have explicit command-line arguments.</td>
</tr>
</tbody>
</table>
linked by an asterisk (*), which acts similarly to a wildcard used in shell commands.

Flowview*mainTextWindowWidth: 900

For example, the following line changes the pie chart radius in the profview command to 200 pixels. The application’s generic name is Profview. The name of the resource checked by the tool is drawPieChartRadius. These are linked by an asterisk (*), which acts similarly to a wildcard used in shell commands.

Profview*drawPieChartRadius: 200

X Window System resource designators can be considerably more complex than the previous example indicates. See the list of standard X Window System references described for each tool in the appropriate chapter.

1.7.4 Troubleshooting

If you have never worked with UNICOS systems with X Window System applications, you may have difficulty when first trying to bring up these performance tools. You first should try to run simple applications, such as /usr/bin/X11/xcalc. Be certain that the DISPLAY environment variable is set to your workstation, and that you have run the xhost command on your workstation (do not run xhost on UNICOS systems). It is also important that your UNICOS host knows the correct network pathway to your workstation. If you can use telnet to connect your workstation to the host, you could try using the numeric address of your workstation (followed by :0) as the DISPLAY environment variable, rather than its name. For example, if your workstation has address 127.162.83.10, you could try specifying a DISPLAY of 127.162.83.10:0. Try this method only if the UNICOS system does not recognize your workstation by name.

When you can run a simple application like xcalc, other X Window System tools should execute without difficulties.

1.7.5 Color

The X Window System tools described in this manual generally do not specify particular colors for their displays. However, if you are running the tools on a workstation that supports color, you can alter some of the colors in certain windows by specifying them as resources.
1.7.6 Pointing Device

The X Window System requires that your workstation have a pointing device, such as a mouse. Most of the X Window System tools described in this manual expect primary selections to be made by pressing the leftmost mouse button when the cursor is located on the desired menu item.

1.8 Component Naming Conventions

The terms buttons, menus, and keyboard accelerators are used to describe various components of the performance utilities. To avoid confusion, the following sections describe naming conventions.

1.8.1 Buttons

Buttons let you open menus and perform other functions. To activate a button, position the arrow cursor on the button and press the left mouse button. This is referred to as clicking on a button. The following types of buttons are used by the utilities:

- The menu bar at the top of the main window displays buttons that open menus. When you click on one of the buttons, the menu opens. (For other ways of opening menus, see Section 1.8.3, page 16.)

- The Cray Research logo also functions as a button. When activated, this button displays the product name, current version number, and applicable copyright information.

- Additional buttons appear on various pop-up displays. The Close button that appears on every pop-up display is an example of this type of button.

1.8.2 Menus

Menus are lists of available options. You can open a menu and make a selection in the following ways:

- Click on a menu button and a pop-up menu opens. Then click on the name of the menu option you want to use. The menu remains open until you move the cursor out of the menu boundaries and press the left mouse button. If you click on a second menu button, the first menu closes.

- Use the press and drag method to make a selection by positioning the cursor on a menu button. Then, press and hold the left mouse button while
dragging the mouse down. A pull-down menu opens. When the option you
want to use is highlighted, release the mouse button. The option is selected
and the menu closes automatically.

• After you have selected one menu, you can use the left and right arrow keys
to move from menu to menu. Using the arrow key to move to a new menu
button automatically closes the previous menu.

Menus display the following peel-off icon in the upper right-hand corner:

![Peel-off Icon]

This icon indicates that you can convert menus into separate X Window System
windows. After you have created a peel-off window, you can move, resize,
iconify, or otherwise manipulate the window in the same manner as any X
Window System window.

To create a peel-off window from a menu, position the cursor on the peel-off
icon and press any mouse button. To close a peel-off window, position the
cursor on the Close button and press any mouse button.

1.8.3 Keyboard Accelerators

Keyboard accelerators help speed experienced users through frequently used
menu options. To use a keyboard accelerator, make certain your cursor is inside
the tool’s main window. Accelerators work in the following ways:

• You can use the CONTROL key with another character key. For example, the
keyboard accelerator for closing a tool is CONTROL-Q, which performs the
same function as selecting Quit from the File menu. Keyboard accelerators
are not case-sensitive; therefore, CONTROL-Q and CONTROL-q perform the
same function. Menu notation uses the caret (^) symbol to signify the
CONTROL key. Where an accelerator is available for an option, it is included
in the option description.

• You can type the underlined letter on menu buttons. Typing the underlined
letter while pressing the CONTROL key when no menu is open, opens the
menu associated with the letter. Typing the underlined letter while pressing
the CONTROL key when a menu is open, opens the menu option associated
with the letter.
1.8.4 Cray Research Logo

Clicking on the Cray Research logo opens a display that shows current tool version and copyright information.

1.8.5 Help Menu

The Help menu options provide online information about how the tool works and displays version and copyright information. Figure 1, page 17 shows the Help menu.

![Help Menu](image)

Figure 1. Help Menu

1.8.5.1 Help Selection

Clicking on the Help option opens a separate window that contains help information. After the window is open, you can find information in several ways:

- Use the left mouse button to click on a topic in the Sections column of the window. You will move directly to that topic in the help text.

- Search for a string within the help text by entering it in the Search: field and then pressing the RETURN key.

- Click on one of the **bold, underlined** words in the text display on the right side of the window by positioning the cursor over the word and pressing the left mouse button. This links you automatically to more detailed information about that subject.

- Scroll through the text by pressing the SPACE key while the cursor is on the right side of the display.

- Use the cursor and mouse to scroll through the help text by using the scroll bar on the right side of the window.
To close the window, select Quit from the File menu.

1.8.5.2 Version Selection

Clicking on the Version selection opens a window that displays the current version of the tool and copyright information.

1.8.5.3 Release Information Selection

Clicking on the Release Information menu selection gives you information on what is new in the current release.