IBM C/C++ Tools: Execution Trace Analyzer Introduction

Document Number S61G-1398-00

IBM C/C++ Tools: Execution Trace Analyzer Introduction

S61G-1398-00

IBM C/C++ Tools: Execution Trace Analyzer Introduction

S61G-1398-00

First Edition (March 1993)

This edition applies to Version 1.0 of Execution Trace Analyzer (61G1398, 61G1176) and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Changes or additions to the text and illustrations are indicated by a vertical line to the left of the change or addition.

Requests for publications and for technical information about IBM products should be made to your IBM Authorized Dealer or your IBM Marketing Representative. Publications are not stocked at the address given below.

A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM Canada Ltd. Laboratory
Information Development
21/986/844/TOR
844 Don Mills Road
North York, Ontario, Canada. M3C 1V7

You can also send your comments by facsimile to (416) 448-6057 addressed to the attention of the RCF Coordinator. If you have access to Internet, you can send your comments electronically to **torrcf@vnet.ibm.com**; IBMLink, to **toribm(torrcf)**; IBM/PROFS, to **torolab4(torrcf)**; IBMMAIL, to **ibmmail(caibmwt9)**

If you choose to respond through Internet, please include either your entire Internet network address, or a postal address.

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.

Note!

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

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 licensed program in this publication is not intended to state or imply that only IBM's licensed program 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, is 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 the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

Trademarks and Service Marks

The following terms, denoted by an asterisk (*), used in this publication, are trademarks and service marks of IBM Corporation in the United States and other countries:

IBM	Presentation Manager
Operating System/2	OS/2

Conventions Used in this Book

The following highlighting conventions are used in this book:

Font	Convention
Bold	Names of windows, menus, menu choices, push buttons, and entry fields.
`Monospace`	Text displayed on the screen and text that you type.
Italics	Words that are used for emphasis and variables in command strings.

Notes About this Book

The words *program* and *application* are used synonymously in this book. Both words refer to the executable file that you are tracing.

Contents

Introduction

The IBM* Operating System/2* (OS/2*) Execution Trace Analyzer (hereafter referred to as EXTRA) is an OS/2-based application that helps you improve and understand the behavior of IBM C and C++ applications. EXTRA traces the execution of an application and creates a trace file. The trace file contains analysis data that can be displayed in various diagrams.

EXTRA can help you improve the performance of an application, examine what led to certain faults, and, in general, help you understand what happened when an application ran.

EXTRA does not replace static analyzers or debuggers, but it can compliment them by helping you understand aspects of the application that would otherwise be difficult or impossible to see.

Understanding EXTRA

You can use EXTRA to time and tune applications, to indicate where deadlocks occur, and to trace multi-threaded interactions.

Timing and tuning
EXTRA time stamps each trace event using a high resolution clock (about 838 nanoseconds per clock tick). As a result, the trace file contains a detailed record of when each traced function was called and when it returned. The information in a trace file can be used to compute the elapsed time for subintervals of the overall execution or to find hot spots. *Hot spots* are areas within an application where a disproportionate amount of time was spent. Plus, the information helps to determine which functions caused those hot spots.

Deadlocks
The execution trace provides a history of the events leading to the point where the application stopped.

Multi-threaded interactions
When multi-threaded applications are traced, the sequencing of functions across threads are seen in some of the trace diagrams. This can illuminate problems within critical areas of the application.

Two major features of EXTRA are as follows:

- The extra time needed to generate the trace analysis is removed. This makes the timings as accurate as possible without resorting to specialized timing hardware.
- In addition to tracing functions in the executable (EXE) file, EXTRA traces the entry points to system calls and some system provided dynamic link libraries (DLLs).

Using the Online Help

Help is available as *context-sensitive* help. Context-sensitive help lets you access help from any menu choice, window, or field to discover how to use that particular item.

To access a help screen, use one of the following methods:

- Select a choice from the **Help** menu.
- Press F1 or select the **Help** push button in any action window.
- Press F1 while highlighting any menu choice or menu bar choice.

Before You Begin

This chapter explains how to prepare your application for EXTRA and how to customize the operating environment.

Preparing an Application

To prepare an application, you must compile and link the application with the proper options. It is also possible to modify the source code of the application to generate events at desired locations. This is explained in "Defining User Events" on page 5.

Compiling an Application

You must compile your application again before using it with EXTRA. The following compiler options need to be specified:

/Gh Includes the *profile hooks* that allow EXTRA to monitor your executable file.

/Ti Includes debugging information in the compiled run file, which would be a file with the OBJ file extension. EXTRA needs this information the same as a debugger does.

Linking an Application

You must link the DDE4XTRA.OBJ file with your application. This enables EXTRA to hook your functions when gathering trace data.

The IBM LINK386 option is:

/DE This causes the linker to include debugging information in the run file (EXE or DLL). EXTRA needs this information the same as a debugger does.

Figure 1 is an example of a MAKE file.

```
pmlines.exe: pmlines.obj dde4xtra.obj pmlines.res
  link386 /DE /PMTYPE:PM pmlines profit,,pmlines,doscall /
  _pmwin _pmgpi;
  rc pmlines.res

pmlines.res: pmlines.rc pmlines.ico pmlines.h pmlines.dlg
  rc -r pmlines.rc

pmlines.obj: pmlines.c pmlines.h
  icc /c /Fa /S2 /Ti /Gh pmlines.c
```

Figure 1. Example MAKE File

To trace system calls, refer to “Tracing the System Calls” on page 5.

Customizing the Environment

The application and the data files must be available to EXTRA just as when your application is run by itself. EXTRA searches for the files in the current directory, and then in the current path.

Changing EXTRA Window Options

It is possible to generate a trace without the Presentation Manager* windows that EXTRA normally displays. To do this, set the following environment variable:

EXTRA=GO This option allows EXTRA to trace your application with no user interaction. The options last saved are used for this execution.

Tracing the System Calls

If you want to trace calls into the OS/2 Toolkit Application Interface (API), specify the following EXTRA libraries immediately before the OS/2 libraries in your link statement, as shown in Figure 1 on page 4.

The APIs and their corresponding libraries are as follows:

DosCalls	_DOSCALL.LIB
WinCalls	_PMWIN.LIB
GPICalls	_PMGPI.LIB

Notes:

1. Each library listed above also has an associated DLL.
2. The order is critical. If the replacement libraries are not immediately before the OS/2 libraries in the link statement, EXTRA may not interpret and trace the API calls.

Defining User Events

The DDE4XTRA.OBJ file contains an entry point called *EXTRA* that accepts calls from the application you are analyzing. After accepting the call, the entry point places the text strings, which are called *user events,* in the trace file. User events help you diagnose problems in the application. You can create a user event by adding the following statement to your application:

```
EXTRA(string);
```

where *string* is an ASCIIZ string. When such a call is made, the string is placed in the trace file. You can see the user events in the Call Nesting diagram and in the Statistical Summary.

You must declare a prototype for this function in the files that make the call to this function. The prototype is as follows:

```
VOID EXTRA (PSZ string);
```

Warning: The string must be unique static alphanumeric characters. Otherwise, unpredictable results can occur.

If you have created a MAKE file, no further steps are required.

Tracing File Accesses

EXTRA can track file accesses, such as DosOpen(), DosRead(), DosWrite(), DosClose(), and DosDupHandle(). To keep track of file accesses, select **File Access** from the **Options** menu in the **Trace Generation** window.

Note: You must link the _DOSCALL.LIB file with your application to make this choice available.

When you have selected the **File Access** choice, EXTRA monitors the five Dos calls listed above. When one of these calls references a file (as opposed to the console), EXTRA keeps track of the name of the file associated with it. Figure 2 shows an example of a file access in the Call Nesting diagram, where C:\MYFILE.OUT is the name of the file associated with DosOpen.

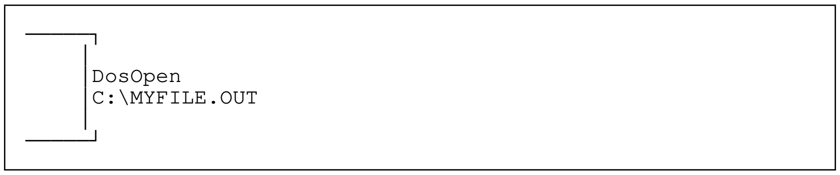

Figure 2. Example of a File Access in the Call Nesting Diagram

The Statistical Summary tracks how often the given file was accessed. Figure 3 on page 7 contains a sample Statistical Summary. In the example, there were a total of five Dos calls made by DosOpen: three calls to the MYFILE.OUT file and two calls to the MYFILE2.OUT file.

```
DosOpen            5
   C:\MYFILE.OUT     3
   C:\MYFILE2.OUT    2
```

Figure 3. Example of a File Access in the Statistical Summary

EXTRA considers any deviation in the file name as a distinct file. So, C:\MYFILE.OUT and MYFILE.OUT passed as ASCIIZ strings to DosOpen() are considered two different files during the file access tracing. EXTRA can track up to 100 open files at a time.

Note: EXTRA counts duplicate handles toward the 100 file total allowed.

EXTRA removes some, but not all, of the overhead associated with a file access. This is reflected in the display diagrams.

Getting Started

This chapter explains how to start EXTRA, and how to use the main EXTRA window called the **Trace Generation** window.

Starting EXTRA

To start EXTRA from the OS/2 command prompt, type the EXTRA command and the command you normally use to run your application, and then press Enter. For example, if you normally type the following to run your application:

```
            myprog /debug
```

where *myprof* is the name of your application and *debug* is a parameter, then you would type the following to run your application under the control of EXTRA:

```
            EXTRA myprog /debug
```

After pressing Enter, the IBM logo window is displayed. Select the **OK** push button to continue.

If you only type `EXTRA` and press Enter, the **Startup Information** action window is displayed. To continue, complete the entry fields as follows:

- In the **Program** entry field, type the name of the application you want to trace.
- In the **Parameters** entry field, type any parameters that you want to pass to your application.
- Select **OK** to accept the information entered, to close the action window, and to start EXTRA.

Warning: Do not start another EXTRA session while the first session is still in progress. Otherwise, unpredictable results will occur.

The **Trace Generation** window is displayed. From this window, you can select the various EXE and pre-loaded DLL files you want to analyze. You also can expand the traceable files and customize the trace analysis for your application.

Note: Online help is available for each entry field and push button in EXTRA. You can access the online help from the **Help** menu or by pressing F1.

Using the Trace Generation Window

The **Trace Generation** window is the main window of EXTRA. This window contains the file names of the executables for the application you want to analyze. An executable file is made up of OBJ files (or object files). An object file is made up of functions. An executable file or an object file that is displayed in black contains debugging information that allows EXTRA to trace your application. An executable file or an object file that does *not* contain debugging information is displayed in cyan.

By selecting the plus icon to the left of the executable file name, you can expand the executable to show the object files. Using the same technique, each object file in the executable can be expanded to show the functions.

Note: For more information about expanding executables and object files, refer to the online help.

Customizing the Trace

To customize your trace, select the **Option** menu from the **Trace Generation** window. The following are examples of what you can do:

- Log or not log time stamps.
- Select the size of the event log buffer to use. Select to write only the last set of events to the trace file by choosing the buffer wrap choice.
- Select the call nesting depth limit for each thread.
- Name the trace file.
- Set timeout values.
- Trace file accesses.
- Reset all options to their default value.

To save the setup changes for the execution trace, select the **Save Settings** check box at the bottom of the **Trace Generation** window. Any options you specified in this window can be saved, which prevents you from having to specify a complicated setup more than once. These settings are not saved until the window is closed or the application is traced. These settings are kept, for each application, until you reset them.

Viewing and Analyzing the Trace Files

This chapter describes the trace file, the trace analysis diagrams, and how to view these diagrams.

Understanding the Trace File

When EXTRA is run on your application, a trace file is produced. The default name of the trace file is *filename.TRC,* where *filename* is the name of the application's executable file. EXTRA places the trace file in the current directory unless you named a path when you started EXTRA. In that case, EXTRA places the trace file in the specified directory. You can change the path and name of the trace file by selecting the **Name Trace File** choice from the **Options** menu in the **Trace Generation** window.

Viewing the Trace Analysis Diagrams

After you have traced your application, the **Trace Analysis Selection** window is displayed. This window displays each trace analysis diagram as a push button. The choices from this window are:

- Statistics
- Call Nesting
- Time Line
- Exec Density
- Call Graph.

To analyze the trace file generated by EXTRA, you can use one or more of the trace analysis diagrams. These diagrams provide a meaningful presentation of the trace data. Several complementary interactive views of the trace are available and can be correlated. Because large trace files may be unmanageable when you are viewing them, EXTRA allows you to scale, filter, and summarize the trace data in several ways.

When you have viewed a diagram, you can exit the diagram or select another diagram to view.

Understanding the Statistical Summary

This diagram provides a textual report of execution time by function. Use this information to find hot spots in the overall program execution. It also provides summary information, such as:

- Number of executables or object files generating events
- Number of functions generating events
- Number of threads generating events
- Maximum nesting depth of each thread
- Number of user events
- Total number of events
- Number of buffer flushes
- Total trace time excluding overhead
- Trace overhead time.

The following statistics are also shown for each function:

- The total execution and active time as percentages of the total execution time
- The minimum, maximum, and average call time
- The total number of calls.

To display the Statistical Summary from the **Trace Analysis Selection** window, click on the **Statistics** push button and the diagram is displayed. To display this diagram from the command line, type the following command and press Enter:

```
istats filename.fileext
```

where *filename* is the name of the trace file generated by EXTRA and *fileext* is the extension.

Understanding the Call Nesting Diagram

This diagram displays the application execution as a vertical series of function calls and returns. It shows the sequence of nested function calls and returns performed by the application. A call is drawn as a step to the right and a return as a line back to the left. The calls are labeled with the name of the function being entered. Context switches between threads are displayed as dashed horizontal lines.

To display the Call Nesting diagram from the **Trace Analysis Selection** window, click on **Call Nesting** push button and the diagram is displayed. To display this diagram from the command line, type the following command and press Enter:

```
    icalnest filename.fileext
```

where *filename* is the name of the trace file generated by EXTRA and *fileext* is the extension.

Understanding the Time Line Diagram

This diagram displays the sequence of nested function calls and returns. The time stamp information is displayed in this diagram as well. EXTRA uses the time stamp information to determine the placement of each event along the time dimension on the vertical axis. This provides a direct and natural presentation of the chronological relationships of events.

To display the Time Line diagram from the **Trace Analysis Selection** window, click on the **Time Line** push button and the diagram is displayed. To display this diagram from the command line, type the following command and press Enter:

```
    itime filename.fileext
```

where *filename* is the name of the trace file generated by EXTRA and *fileext* is the extension.

Understanding the Execution Density Diagram

This diagram divides the entire execution time into a fixed number of horizontal time slices (or scan lines). The time slices are determined by the number of pages used for the entire diagram. Each function that logs events is assigned to a fixed column of the diagram. The slices are colored to show the percentage of time spent in each function.

To display the Execution Density diagram from the **Trace Analysis Selection** window, click on the **Exec Density** push button and the diagram is displayed. To display this diagram from the command line, type the following command and press Enter:

```
    iexcdens filename.fileext
```

where *filename* is the name of the trace file generated by EXTRA and *fileext* is the extension.

Understanding the Dynamic Call Graph Diagram

This diagram is a graphical view of the target application's execution. A node represents a function and an arc between a pair of nodes represents a call from one function to another. Only calls made during the application execution are displayed. The color and size of nodes and arcs depict the time spent in the node and the number of calls between nodes, respectively.

To display the Dynamic Call Graph diagram from the **Trace Analysis Selection** window, click on the **Call Graph** push button and the diagram is displayed. To display this diagram from the command line, type the following command and press Enter:

```
    idcgraph filename.fileext
```

where *filename* is the name of the trace file generated by EXTRA and *fileext* is the extension.

Using EXTRA Menu Choices

This chapter describes menu choices that allow you to correlate the diagrams and to reduce the amount of trace data displayed at one time.

Understanding Correlation

Three of the diagrams, Call Nesting, Time Line, and Execution Density, are called chronologically-scaled diagrams. **Correlation** is a menu choice available from these three diagrams. These diagrams can be correlated based on a particular event or a particular point in time. The following example shows how to use correlation on the Time Line and Call Nesting diagrams:

1. Open the Time Line diagram to get an overall view of the trace.
2. Select the **Open** cascading choice in the **File** menu to display a cascaded menu.
3. Select **Call Nesting** to open a new window.
4. Select a point of interest in the Time Line diagram by clicking and holding down mouse button two. This point represents some particular time or time interval.
5. Move the mouse pointer to the window showing the Call Nesting diagram. When you release mouse button two, the Call Nesting diagram scrolls the trace event or events whose time stamps most nearly correspond to the specified time or time interval in the Time Line diagram. EXTRA also highlights the corresponding parts of both diagrams to identify the correlation.

All three chronologically-scaled diagrams can be correlated with any other including another instance of the same diagram. For example, you can show the same diagram at a different scale and then correlate.

Understanding Data Reduction Techniques

Data reduction techniques let you manipulate the information that you want displayed in the analysis diagrams. These techniques include filtering, scaling, scrolling, using multiple views, summarizing execution time, using color, recognizing patterns, and interacting with the diagrams.

Using Filtering

You can select a subset of the traced events for presentation. The Call Nesting diagram and the Dynamic Call Graph let you show the threads that logged events. Then EXTRA displays only call sequences that involve one of the selected threads. The Time Line diagram offers a time threshold and does not show functions that completed execution in less time than the specified threshold.

Using Scaling

Scaling is a way to change how much detail is displayed. You can view the diagram as a whole or magnify areas to see the finer points. When details are hidden, you cannot spot patterns, anomalies, or features of the execution because there is too much information on the screen. All chronologically-scaled diagrams can be scaled along the time dimension and the Dynamic Call Graph can be scaled in both of its dimensions (height and width) by using the **Overview** menu choice.

Using Scrolling

The viewing window can be scrolled in all diagrams to focus on parts of interest. Correlation, described in “Understanding Correlation” on page 17, is also a valuable way to scroll the chronologically-scaled diagrams.

Using Multiple Views

The multiple presentations of the trace data described earlier let you view the traced application execution from different viewpoints simultaneously. So, any one diagram need not attempt to show all events contained in the trace or present all relationships and patterns among them.

Summarizing Execution Time

The Statistical Summary and the Dynamic Call diagram summarize execution time by function. The Execution Density diagram divides execution time into a fixed number of horizontal time slices or scan lines.

Using Color

EXTRA uses color to display information that would otherwise take up screen space. For example, the Execution Density diagram uses color to indicate the percentage of the total execution time that was used by a function during a time slice.

Recognizing Patterns

In the Call Nesting diagram, application loops cause the same sequence of calls and returns to be repeated in the trace. EXTRA lets you choose to display the diagram with the same sequences combined. By using the pattern recognition, you can reduce the amount of screen space the diagram uses.

Interacting with the Diagrams

Some information is shown on demand by making the diagrams interactive. EXTRA uses this to better manage the information in a diagram. Examples are as follows:

- Execution time rulers in the Time Line diagram and Execution Density diagram.
- Annotation in the Call Nesting diagram.
- Correlation in the Call Nesting diagram, Time Line diagram, and Execution Density diagram.
- Call stack list boxes in the Call Nesting diagram and Time Line diagram.
- Execution and active times in the Dynamic Call Graph.

Gathering Trace Data

EXTRA gathers trace data with minimal impact on the application being traced by capturing events, using time stamps, compensating for trace overhead, flushing the buffer, and reducing the trace volume.

Capturing Events

EXTRA uses compiler hooks to run monitor code on entry to each function. This monitor code traces an event and hooks the return address so a matching event can be logged during the function epilogue. So, the event recording functions is run in the address space of the application instead of in the trace monitor. This avoids the high overhead of an operating system context switch when events are recorded.

The monitor code does not affect application execution when EXTRA is not being used. This helps to minimize the difference between running the application with and without EXTRA. However, when the application is run without EXTRA, the hook causes a call and an immediate return.

Using Time Stamps

You can choose to either time stamp or not time stamp events. EXTRA uses an internal timer to get high resolution time stamps. Event logging adds a small amount to the normal runtime speed of the application. You may not be able to tell the difference between code that is being traced and code that is not, even for highly inter-active applications.

Compensating for Trace Overhead

EXTRA traces the execution of your application at a speed close to the application's real, unmodified speed. Each time EXTRA is run, it automatically calibrates itself to measure the overhead imposed by its trace generation code. The calibration numbers are written into the trace file. This allows the analysis diagrams to accurately adjust the measured execution times and compensate for the trace generation overhead. The size of the trace file is kept to a minimum.

The accuracy of the calibration is high. Ratios between the various timings are accurate.

Flushing the Buffer

Your application and EXTRA share memory with the trace buffer. This allows EXTRA to log events that are running in the address space of the application. When the trace buffer is full, EXTRA takes the following steps:

1. EXTRA time stamps the start of the buffer flush.
2. It writes the events in the buffer to the trace file on the disk. Then, it starts the application again.
3. EXTRA time stamps the ending of the buffer flush, which allows the analysis diagrams to remove the buffer-flushing overhead.

You can select the size of the trace buffer and, thereby, change the time spent flushing the buffers to the trace file on the disk.

Note: You can control the buffer size from the **Options** menu in the **Trace Generation** window. This choice causes EXTRA to keep only the most current events in the buffer and overwrite the older events. Since the buffer is flushed only when the application ends, less disk space is needed for the trace file.

Reducing the Trace Volume

In a highly interactive environment, large trace files take a lot of time to generate and analyze. Large trace files are created when you request detailed trace information. When you limit the trace to events of immediate interest, EXTRA collects and analyzes the data quicker. You can manipulate the EXTRA monitor by changing the parameters that control the size of the generated trace. These parameters fall into three areas, which are explained below:

- *Static controls* are set before running the application and require no runtime checks. EXTRA allows you to enable or disable tracing of the following:

Functions	Individual functions
Object files	All functions in an object file
Executables	All object files in an executable file
Threads	Selected threads

- *Dynamic controls* are also established prior to running the application and use runtime checks to control tracing. These controls include:

Call stack depth	Limits the number of traced functions on the call stack for each thread in the application. Functions do not log trace events beyond this limit.
Trigger functions	Designates functions as triggers that turn tracing on when called and turn tracing off when they return.
Buffer wrap	Writes to the trace file on the disk only when the target application ends. In this case, EXTRA wraps within its trace buffer and overwrites older events.
Timeout control	Sets a timeout period to end the application if no additional events are logged during the period.

- *Selective information* does not enable or disable tracing and augments the basic trace events with otherwise useful data.

Time stamps	If you do not need timing information, you may reduce the size of the trace file by having EXTRA *not* record time stamps.

®

Part Number: 61G1398
Program Number: 61G1176
61G1426

Printed in U.S.A.

61G1398

S61G-1398-00