Performance Tools Notes

This document combines the ETO #20 Release Notes for MrPlus, MrProf, MrProfUtil, and PPCProf. The individual documents can be found conveniently by using the "Bookmarks" feature of Acrobat.

MrPlus v. 1.0d2c2 — "Prerelease"

Contents

Introduction

MrPlus is an MPW performance tool which operates on Apple PEF files. It is a multi-function tool which can do the following:

- Produce static information about the executable file, such as counts of important program structures, frequency of appearance of the various instructions, and frequency of appearance of references to the various registers.

- Produce a PEF file which is an enhanced version of the original file. It contains instrumentation code which, at the time of execution, outputs dynamic information about the program. Corresponding to the static information already listed will be dynamic data on the instruction mix. The distinction is that the dynamic information states the frequency of execution instead of the frequency of appearance. The dynamic information also includes a routine profile: the relative number of instructions executed in each routine.

- Produce a PEF file containing instrumentation that traps attempts to store into low memory (0–32K).
- Optimize the program by reordering portions of code in the PEF file to improve the utilization of the instruction cache and to reduce the number of page faults in a system with virtual memory enabled.

New Features

The following new options have been added in this version of MrPlus. (for details, see the “Options” section below.)

- `-report web`
- `-report unwind`
- `-arrout`
- `-progress`
- `-member`
- `-cntout`

Description

Static Analysis

The input to MrPlus is a PEF file and optionally an XCOFF file; if the latter is present, symbolic information will be included in reports and will be available for debugging. All reports produced by MrPlus go to Standard Error. Certain static reports are produced by default, i.e. even if no options are given. These include items such as code and data section sizes, and counts of the number of various program structures such as routines, switches implemented by jump tables, unconditional returns, conditional returns, etc. Other static report generation is controlled by the various `-report` options (see the section headed “Options” below).

Software Dynamic Analysis

In addition to the static reports, the options `-instrument calls` or `-instrument branches` control the dynamic analysis and optimization features of MrPlus (again, see the "Options" section). If either of these options is used, MrPlus will produce an instrumented PEF file (<program>.prof), a file that maps the instrumentation counters to program locations (<program>.pmap), and, if an XCOFF file had been originally input, an XCOFF file for the instrumented PEF file (<program>.prof.xcoff).

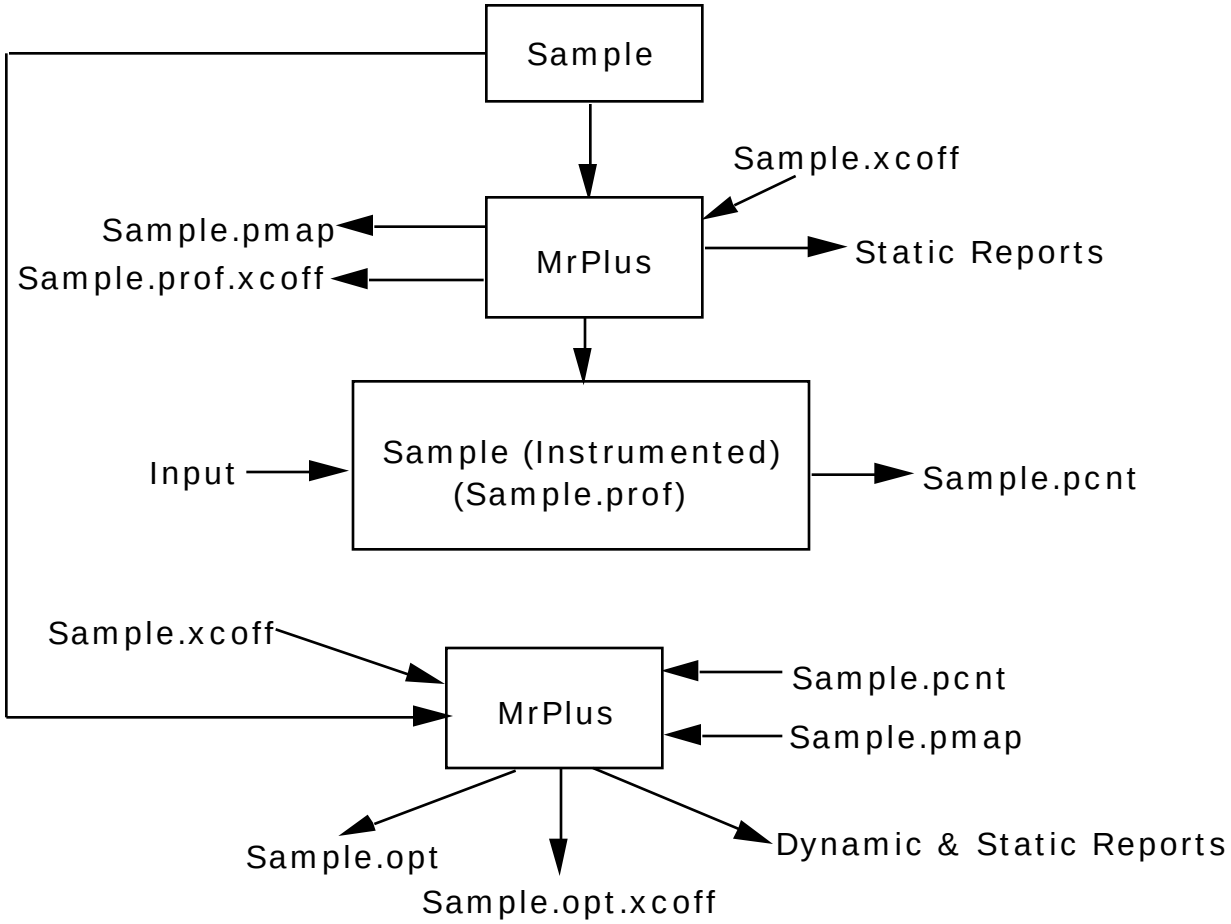
To explain the instrumentation, it is necessary to define the term basic block. A basic block, in any portion of a program, is the longest contiguous stretch of code which, except at its end points, contains no branches and no points to which branches are taken. The software counters record transitions between basic blocks. Such transitions are known in the compiler literature as "arcs." The .pmap file records, for each counter, the vital statistics of the source and destination basic blocks for which transitions are counted: their start addresses and their lengths. Execution of the instrumented program now produces a file containing the counter values. The user is prompted for a name for this file (but see the option `-cntout` in the "Options" section below). The name <program>.pcnt is assumed here.

Important: ProfileLib must be accessible to the instrumented program. However, linking with ProfileLib is not necessary because imports from it are added by MrPlus as part of the instrumentation step. Normally, the ProfileLib file is placed in the Extensions folder of the System Folder.

A second execution of MrPlus, using as inputs the original PEF file and the files <program>.pcnt, <program>.pmap, and optionally <program>.xcoff **[???]**, can produce several dynamic reports, a PEF file that has had certain optimizations applied (<program.opt>), and a corresponding XCOFF file (<program.opt.xcoff). (See the options `-arrange routines`, `-arrange blocks`, and `-opt dynamic` in the "Options" section below.)

The illustration below shows a flowchart for the processes just described.

Analysis of program "Sample"



Hardware Dynamic Analysis

If called with the `-monitor icache` option, MrPlus outputs a PEF file containing added instructions that cause hardware counters to gather information during execution. The information gathered is the execution time as reported by the hardware `time base` register. On a 604 processor, additional data gathered are the number of instructions completed and of instruction cache misses. The naming conventions for the output files are <program>.mon for the instrumented program and <program>.mon.xcoff for the corresponding XCOFF file.

Syntax, Parameters, and Options

Syntax

```
MrPlus
[-report imix | regs | glue | web | unwind ] |
[-instrument calls | branches | stores | none ] |
[-monitor icache] |
[-arrange routines | blocks | none ]
[-arrout arrfile]
[-opt static | dynamic | none ]
[-progress]
[-member name]
[-cntin PCNTfile] [-cntout PCNTfile]
[-mapin PMAPfile] [-mapout PMAPfile]
[-fragout PEFfile]
[-xcin XCOFFfile] [-xcout XCOFFfile]
PEFfile
```

Parameters

PEFfile Specifies the program to be analyzed and/or optimized.

Options

Note: In a number of the options below, the name “PEFfile” is used. It simply stands for the actual filename of the input program.

-report imix | regs | glue | web | unwind

imix

Report the static and dynamic occurrence count of classes of instructions. The dynamic count is reported only if the `-opt dynamic` or `-arrange` options are used.

reg

Report **static** register usages.

glue

Report occurrences of linker generated glue routines that enable cross TOC calls.

web

For each EXPORTED, INIT, TERM, and/or ENTRY function in PEFfile, list the pair “fragment name/function name” for all IMPORTED functions statically reachable from that function. Thus, `-report web` indicates for any code flow entering PEFfile which imports that code flow could possibly invoke.

unwind

Indicates how many functions save LR, CR, General Registers, and/or Floating Point Registers.

The above are not mutually exclusive. `-report` may be given repeatedly with different arguments.

-instrument calls | branches | stores

An instrumented version of the program is produced. The name of the file is PEFfile.prof unless changed by the option `-fragout`.

calls

The instrumentation counts only routine calls.

branches

The instrumentation counts all branches.

stores

The instrumentation detects attempts to store in low memory (0–32K). An alert box appears containing the low memory address, the name of the executing fragment, and the offset within the fragment of the trapped instruction. The alert box offers the choice of aborting or continuing.

-monitor icache

An instrumented version of the program is produced. The name of the file is PEFfile.mon unless changed by the option `-fragout`.

icache

The instrumentation enables hardware counters. The basic information is the execution time as reported by the hardware time base register. Additionally reported on a 604 are the number of completed instructions and the number of instruction cache misses. A calculated miss rate is also displayed.

The argument `icache` exists because the `-monitor` option may be extended with additional arguments in the future.

-arrange routines | blocks

This causes production of an optimized version of PEFfile. The filename is PEFfile.opt unless changed by the option `-fragout. -arrange` requires that dynamic information has been collected**.**

routines

Routines are rearranged relative to each other in order to improve utilization of the instruction cache and to reduce the number of page faults in a system with virtual memory enabled.

blocks

In addition to the actions for the argument `routines`, unexecuted blocks of routines are placed at the end of the code section.

In addition, for either `-arrange` option, a text file is written listing the preferred link order of routines. This file can be input to linkers which have been enhanced to use it in order to specify the desired routine ordering. The default name of this file is PEFfile.arr unless changed by the option `-arrout` below.

-arrout *arrfile*

This option specifies the name of the "link order" text file produced by the `-arrange` option above.

-opt static | dynamic

-opt static

Unneeded NOPs that follow some call instructions are removed, non-trivial epilog code for multiple return routines is shared, glue code called from 5 or fewer sites is inlined to call “pointer glue 12”, and load/store multiple instructions using 3 or fewer registers are replaced by the requisite number of simple load/stores.

-opt dynamic

In addition to the static optimizations just given, there is direct branching to dominant (of greater than 50% probability) switch targets and the setting of branch prediction hint bits. `-opt dynamic` requires that dynamic information has been collected. For `-opt dynamic`, any load/store multiple instructions executed are fully expanded with the requisite number of simple load/stores, while those not executed are not expanded.

-progress

MrPlus writes to Standard Error additional details about its progress in processing the PEFfile.

-member *name*

MrPlus can only process one code fragment member per invocation. *Name* is the name of the member within the PEFfile that MrPlus should process. This option is not required for the typical case where the PEFfile contains only one code fragment member.

When the PEFfile contains more than one member and the `-member` option is not supplied, MrPlus will simply list all the members. The user must run MrPlus with the `-member` option for each member to be processed, using as input for each step the PEFfile produced by the previous step, in order to recursively process each individual member.

-cntin *PCNTfile*

PCNTfile is the name of the file of counters produced when the software instrumented file is executed. At the end of that execution, the user was prompted for this filename. The option is used to communicate the name of this file to MrPlus.

-cntout *PCNTfile*

Normally, at the termination of a "PEFfile.prof" application which collected counters (that is, it was created with `-instrument calls` or `-instrument branches`) the user is prompted with a dialog box to supply the name of the file where the counters should be written.

For some applications, it may be inappropriate or cumbersome to interactively solicit the output PCNT filename. In this case, the `-cntout` option can be used to specify the *PCNTfile* name where counters are to be written when the application terminates, and no dialog box will appear.

Furthermore, sometimes the user wishes to run “PEFfile.prof” a number of times and automatically collect a different *PCNTfile* for each run. In this case, *PCNTfile* should contain a trailing underscore (example: “-cntout MyCounters_”) and ProfileLib will append the timestamp in hex to the filename in order to guarantee that each *PCNTfile* has a unique name (example: “MyCounters_AD6DE2BB", "MyCounters_AD6DF781”, etc.)

-mapin PMAPfile

PMAPfile is the name of a file produced by MrPlus when it produces a software instrumented PEF file. The option is used to communicate the name of this file to MrPlus for the subsequent analysis/optimization run.

-mapout *PMAPfile*

This option is used to override the default name for the output file that shows that mapping of the software counters. The default name is PEFfile.pmap.

-fragout *PEFfile*

This option is used to override the default names for the output PEF file. The default names are PEFfile.prof for a software instrumented file, PEFfile.mon for a hardware instrumented file, and PEFfile.opt for an optimized file.

-xcin *XCOFFfile*

This option is used to name an input XCOFF file. By default, PEFfile.xcoff will be read. No error is given if the .xcoff file cannot be opened.

-xcout *XCOFFfile*

This option is used to override the default names for the output XCOFF files. The default names are PEFfile.prof.xcoff for the case of instrumentation for software counters, PEFfile.mon.xcoff for the case of instrumentation for hardware counters, and PEFfile.opt.xcoff for the case of an optimized program file. If the `-fragout` option is used then the default name for the .xcoff file becomes instead <fragout>.xcoff, where <fragout> is the parameter for the `-fragout` option.

Note: The options `-arrange`, `-instrument`, `-monitor`, and `-opt` all may have the argument `none`. This is for convenience in the writing of scripts. Using the argument `none` has the same effect as omitting the option.

Cautions

Memory Requirements

Approximate memory requirements are given in the table below:

Option	Memory Requirement

`-instrument branches`	25 * (Code Size)
`-instrument calls`	15 * (Code Size)
`-instrument stores`	11 * (Code Size)
All other options	10 * (Code Size)

Known Outstanding Bugs

- Very simple libraries with no imports and no global variables have no TOC entries, and MrPlus `-instrument` options fail because it is not able to add TOC entries.
- MrPlus writes output information only to StdErr.
- MrPlus is unable to recognize and/or process some switch statements as implemented by some compilers. These problems can cause MrPlus to crash, to write a warning or error message, or to produce a PEFfile which will not run correctly.
- MrPlus is unable to recognize and/or process some forms of hand-coded assembly instructions or some unusual sequences of instructions as produced by some compilers. These problems can cause MrPlus to crash, to write a warning or error message, or to produce an incorrect PEFfile.
- MrPlus is sometimes confused by read-only data in the code section. Such data is often indicated by “Decode31 unknown” warning messages. In many cases, MrPlus will correctly process the PEFfile. But is other cases it may crash or produce an incorrect PEFfile.
- MrPlus does not update the “function size” fields of TraceBack entries when optimizing or instrumenting a PEF file.
- The `-cntout` option can be used in cases where the user does not want ProfileLib to prompt for a counter filename with a dialog box. However, if the filename already exists, ProfileLib will prompt the user for a new filename.

Bug Fixes in Rev1.0d2; (ETO #20)

- MrPlus did not handle PEF files containing multiple code fragments. these are now correctly handled using the `-member` option described in the “Options” section.
- Fragments with no init, term, or entry descriptors and no TVector exports would cause MrPlus to fail to locate the TOC base address.
- A variety of problems were fixed where MrPlus was unable to recognize and/or process some switch statements as implemented by some compilers. These problems caused MrPlus to crash, to write a warning or error message, or to produce an incorrect PEFfile.
- A variety of problems were fixed where MrPlus was unable to recognize and/or process read-only data (including TraceBacks) in the code section. These problems caused MrPlus to crash, to write a warning or error message, or to produce an incorrect PEFfile.
- PEFfiles with no code were not recognized and erroneous error messages were written.
- MrPlus did not recognize entry descriptors which pointed to routine descriptors (mixed mode UPP).
- `-instrument stores` would sometimes incorrectly trap indexed store instructions using R0 as storing to low memory when in fact they were correctly storing outside of low memory. This was because MrPlus was not using the contents of R0 in the address calculation.
- Some compilers do not have a SYMR as the first item in the reloation entries, and MrPlus would not produce instrumented or optimized PEFfiles in such cases.
- MrPlus would zero the low order 2 address bits or data section pointers to strings in the code section when producing instrumented or optimized PEFfiles.
- In the loader section, MrPlus did not update the addresses of exported symbols in the code section when producing instrumented or optimized PEFfiles.

MrProf v. 1.0d2c2 — “Prerelease”

Important: It is absolutely essential that you read these Release Notes. Although MrProf’s functionality has been implemented as a Macintosh application, it departs so radically from Macintosh Human Interface Guidelines that you will have great difficulty if you attempt to “wing it.” These notes, therefore, are written in the manner of a tutorial and are illustrated with a number of screen shots.

Bringing this application into conformity with the Guidelines will be a priority item in the immediate future.

Contents

General Description

MrProf is a Macintosh application that provides tabular and graphic presentations of data collected when executing programs that were instrumented by MrPlus. The input data to MrProf are a .pmap file, optionally a .xcoff file, and one or more .pcnt files. It is assumed that the reader is familiar with MrPlus, understands its uses of these files, and is familiar with the terms "basic block" and "arc".

The basic tabular output of MrProf shows, for each arc, the first and last addresses of the "from" basic block, the first and last addresses of the "to" basic block, the number of times the arc was traversed, and, for the "to" block, the source code line number and the routine name.

The graphical output is a call graph for all the routines in the analyzed program, and, for each routine, a diagram of the intra-routine branches.

Details and screen shots are in the tutorial section that follows.

Tutorial

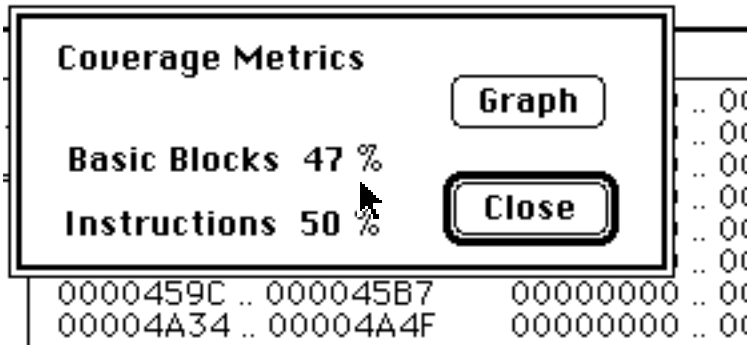
Loading the Data

Launch MrProf either directly or by “opening” a .pcnt or a .pmap file. This window has no close box, zoom box, or grow box. It also does not (yet) have scroll bars.

Now select “Open” in the “File” menu. A Standard-File dialog will appear, the only visible files being .pmap files. “Opening” a .pmap file will cause four columns of data and a vertical scroll bar to show in the window. Note: If the launch of MrProf was the result of “opening” a .pmap file, the data and scroll bar will appear in the window. This will not, however, occur if the launch was caused by “opening” a .pcnt file.

You will notice immediately that clicking in the scroll arrows moves the display one line per mouse click and clicking in the shaded area moves the display one page per click. Holding down the mouse button has no effect. The “page up” and “page down” keys on the extended keyboard will, when held down, cause continuous paging of the display.

Data from .pcnt and .xcoff files are loaded, respectively, by selecting “Load” in the “Counter” and “Symbol” menus. The preferred order is to load symbols first and then counter data. This is because a side effect of all selections from the “Counter” menu is the appearance of the dialog shown below:



In the event that you selected the two loads in the wrong order, just reselect “Load” from the “Counter” menu. This will harmlessly reload the count data and will again present the dialog. You will have noticed “Add” in the “Counter” menu. The purpose of this is add counter data from multiple executions of the subject program, using multiple .pcnt files.

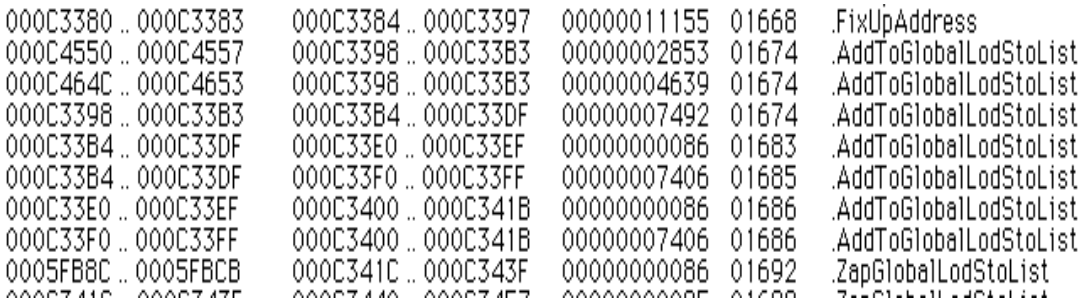
The data shown are the percentage of instructions actually executed and the percentage of basic block actually entered. The “Close” button dismisses the dialog.

Showing the Call Graph

The “Graph” button causes the display of a “call graph” for the entire program. The nodes represent routines. If a .xcoff file was loaded, the nodes will contain the routine names; otherwise, they will contain their starting addresses. On a black & white monitor, nodes actually entered are shown in black and those never entered are shown in white. On a color monitor, the nodes that were entered are green; the ones not entered are red. The graph is dismissed by clicking in its close box.

The Data Display

The appearance of the MrProf window is shown below in a small extract:



Each line represents an arc. The first two columns are the start and end addresses of the basic block at the “from” end of the arc. The next two are the corresponding addresses of the block at the “to” end of the arc. The next column is the count, the number of traversals of the arc. Next is the source line number of the start of the “to” block, followed by the name of the routine.

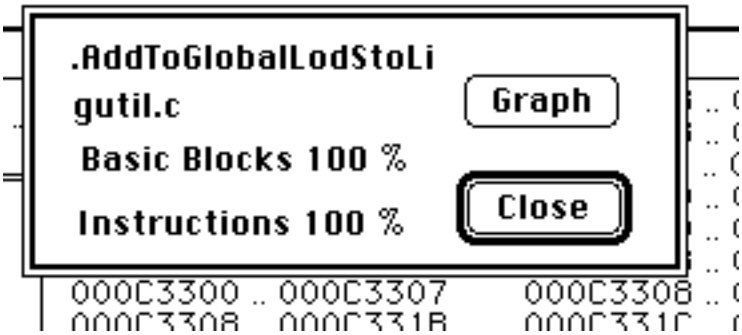
The “View” menu provides you with several ways of sorting the data. The choice “Address” sorts by “to” starting address as a primary key and by “from” starting address as a secondary key. Sorting by “Count” presents the data in descending order of count values. Sorting by “Name” is an alphabetic sort on the routine names. Finally, sorting by “Time” is in effect sorting by the number of executed instructions, because the sort key is the product of the count value and the size of the “to” block.

After sorting by name, count, or time, items are also sorted by the secondary key of “to” address.

The example shown above is an extract from the unsorted window.

Showing a Transition Graph

Clicking the mouse in the general area of a routine name, or where the name would be if the .xcoff file had not been loaded, produces the dialog shown below:

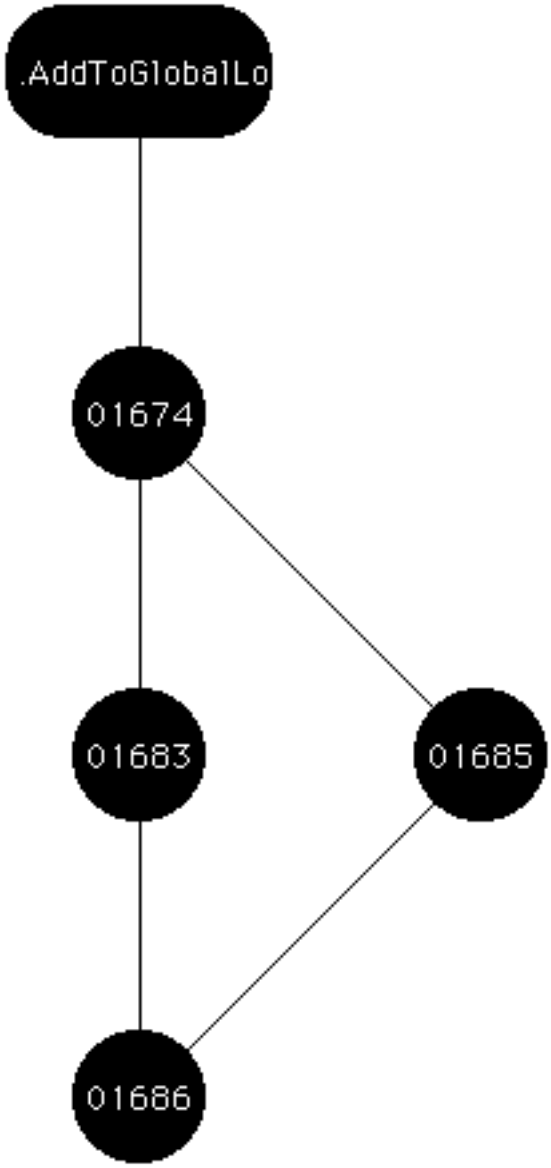


This shows the routine name, the source file name, the block and instruction coverage as in the first dialog, and again a choice between dismissing the dialog and showing a graph. If the .xcoff file had not been loaded, then the starting address of the routine would replace the symbolic information.

Choosing “Graph” presents the display reproduced below:

.AddToGlobalLodStoList

gutil.c

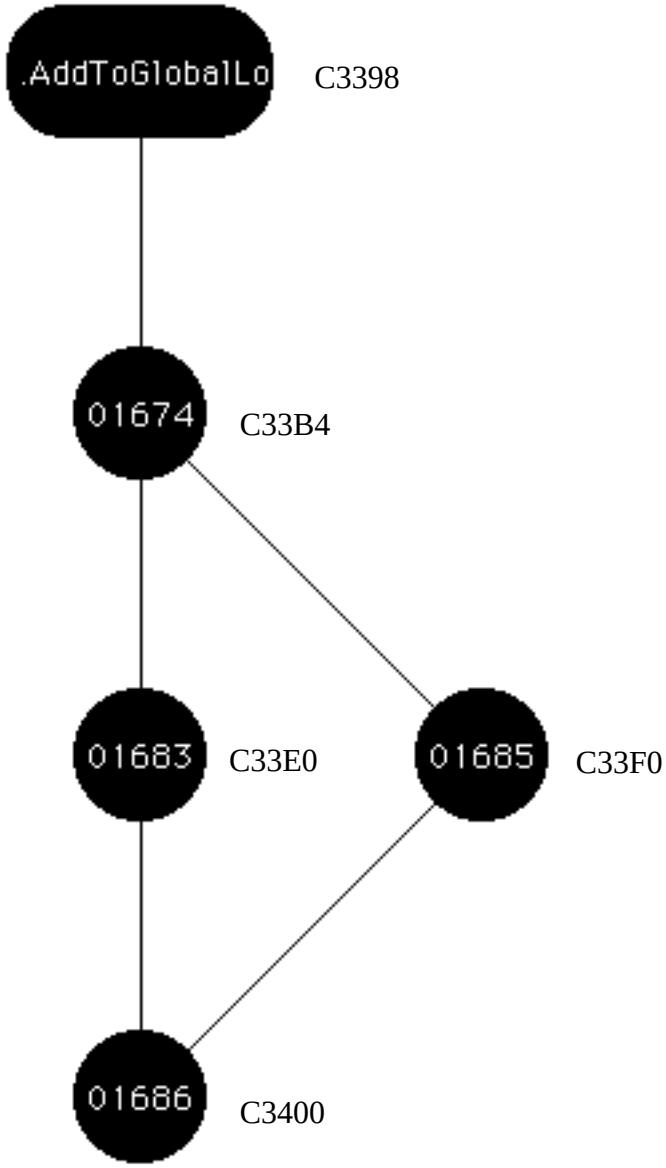


The nodes represent basic blocks. The numbers in the nodes are source line numbers. If no .xcoff file is loaded, the entry node will show the starting address of the routine; the other blocks will have a sequence number corresponding to their address order. The color representation is the same as for the call graph described earlier.

Adding addresses to this display will help you correlate it to the tabular information. So, writing in the address, one sees:

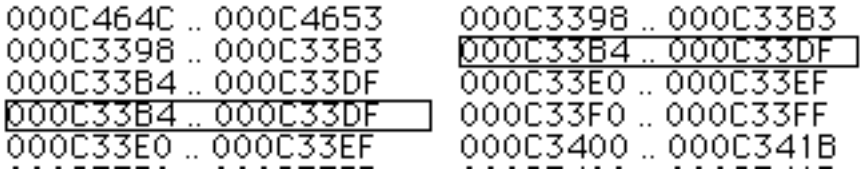
```
.AddToGlobalLodStoList

gutil.c
```



“Who branched to me”

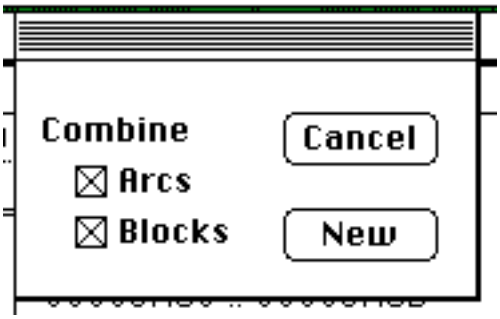
Putting the cursor on “from” block addresses in the table will enclose those addresses in a rectangle. Then selecting “Find” in the “Symbol” menu will cause a rectangle to be placed around that same pair of addresses in the “to” column. This is illustrated in the extract below:



Repetition of this process provides a way of “walking” up a call chain. It would be nice if all occurrences of the selected basic block in the “to” column were shown. Unfortunately, only one is shown: the one with the greatest count value.

Consolidation of Arcs and Blocks

Selecting “New” in the “File” menu will cause the following dialog to appear:



Pressing “New” will cause the appearance of a new window, again titled “MrProf”, but this time possessing a “close” box. There are four choices, “Arcs”, “Blocks”, both, or neither. The nature of the data in the window depends on the choice.

This can be done more than once, so that windows which, for example, respectively combine arcs and blocks can be open simultaneously. Be careful keeping track; the windows are definitely not labeled.

The table below is a repetition of the first table shown:

000C3380 .. 000C3383	000C3384 .. 000C3397	00000011155	01668	.FixUpAddress
000C4550 .. 000C4557	000C3398 .. 000C33B3	00000002853	01674	.AddToGlobalLodStoList
000C464C .. 000C4653	000C3398 .. 000C33B3	00000004639	01674	.AddToGlobalLodStoList
000C3398 .. 000C33B3	000C33B4 .. 000C33DF	00000007492	01674	.AddToGlobalLodStoList
000C33B4 .. 000C33DF	000C33E0 .. 000C33EF	00000000086	01683	.AddToGlobalLodStoList
000C33B4 .. 000C33DF	000C33F0 .. 000C33FF	00000007406	01685	.AddToGlobalLodStoList
000C33E0 .. 000C33EF	000C3400 .. 000C341B	00000000086	01686	.AddToGlobalLodStoList
000C33F0 .. 000C33FF	000C3400 .. 000C341B	00000007406	01686	.AddToGlobalLodStoList
0005FB8C .. 0005FBCB	000C341C .. 000C343F	00000000086	01692	.ZapGlobalLodStoList

The next table shows the result of combining blocks:

000592F4 .. 00059307	000C2F14 .. 000C3397	00000000000	01601	.FixUpAddress
000C464C .. 000C4653	000C3398 .. 000C341B	00000004639	01674	.AddToGlobalLodStoList
000C4550 .. 000C4557	000C3398 .. 000C341B	00000002853	01674	.AddToGlobalLodStoList
0005FB8C .. 0005FBCB	000C341C .. 000C3497	00000000086	01692	.ZapGlobalLodStoList

In the table above, all blocks within .AddToGlobalLodStoList are collapsed into one. The two lines represent calls to the routine from two different places. The table is obtained by putting a mark in the checkbox labeled blocks.

The next table shows the result of combining arcs:

000C3384 .. 000C3397	00000011155	01668	.FixUpAddress
000C3398 .. 000C33B3	00000007492	01674	.AddToGlobalLodStoList
000C33B4 .. 000C33DF	00000007492	01674	.AddToGlobalLodStoList
000C33E0 .. 000C33EF	00000000086	01683	.AddToGlobalLodStoList
000C33F0 .. 000C33FF	00000007406	01685	.AddToGlobalLodStoList
000C3400 .. 000C341B	00000007492	01686	.AddToGlobalLodStoList
000C341C .. 000C343F	00000000086	01692	.ZapGlobalLodStoList

In the table above, the “from” columns are not shown; they were empty. The data are simply the total count to each “to” block. What is shown, therefore, is the total of all arcs to a given block. The table is obtained by putting a mark in the checkbox labeled arcs.

Finally, one can see the result of combining both blocks and arcs:

```
000C2F14 .. 000C3397   00000011155  01601   .FixUpAddress
000C3398 .. 000C341B   00000007492  01674   .AddToGlobalLodStoList
000C341C .. 000C3497   00000000086  01692   .ZapGlobalLodStoList
```

Here we see only one line, the total count for the routine and no internal information. This table is obtained by marking both checkboxes..

Marking neither checkbox results in no combining; the new window will be a duplicate of the original one.

Known Outstanding Bugs

- The window opened by MrProf has neither a grow box nor a zoom box.
- The scroll bar in the MrProf window does not behave according to Mac standards regarding clicks on the arrow vs. the up/down region. Holding down the mouse button has no effect.
- MrProf’s facilities for opening the various files (pmap, pcnt, and xcoff) are incorrect according to Mac standards .
- Windows with text columns should have column headings, and heading should remain visible when information is scrolled.
- The Find menu item does not belong in the Symbol Menu.
- “Find” finds only one block. It should find all appropriate blocks. Repeated “finds” should move to subsequent blocks, scrolling as needed.
- Windows should be titled with the name of the code being analyzed.
- The method for moving nodes in a call graph is not up to Mac standards. It should be possible to “drag” nodes around and have “dotted outline” feedback.

- It should be possible to get coverage metrics and/or display a graph without having to load a file.

Bug Fixes in v. 1.0d2 (ETO #20)

- MrProf did not rotate the cursor when selecting different View options or loading files.
- When a new "from" was selected, the results of a previous “find” were not erased.
- When viewing by Name, Count, or Time, items were not sorted on a secondary key of “to” address.
- The font size was reduced for text windows, allowing them to fit on small screens.
- Horizontal graph layout was improved.

MrProfUtil v. 1.0d2c3 — “Prerelease”

Contents

Introduction

MrProfUtil is a simple MPW tool which operates on PCNTfiles produced with MrPlus. It provides two useful operations:

- MrProfUtil can add the corresponding counters of a set of multiple PCNTfiles together and create a single PCNTfile which represents the sum of all activity recorded in each of the files in the set.
- MrProfUtil can list a set of multiple PCNTfiles in the order of “most aggregate code coverage” so that users can determine which subset of tests provides the most coverage for the least amount of testing effort.

Syntax, Parameters, and Options

Syntax

```
MrProfUtil
-like PCNTfile
[-order] | [-sumout PCNTfile]
[-progress]
```

Options

-like *PCNTfile*

PCNTfile is the [directory:]filename of a PCNTfile. MrProfUtil will process all PCNTfiles in that directory which are “like” (have the same number of counters as) the specified PCNTfile. Thus, the `-like` option specifies the set of files to be processed by another option, such as `-order` or `-sumout`.

-order

Each file in the “like” set is analyzed for its individual code coverage (percent of counters which are non-zero) and for aggreagate coverage. The files are then listed in the following order. The file with the most coverage is listed first. Each successive file is the one which adds the most aggregate coverage to that provided by the files above it. In other words, for any integer *N*, the first *N* files listed provide the most aggreagte coverage of any set of *N* files.

For each row in the listing, the columns are (left to right): aggregate non-zero counters, aggregate percent coverage, filename, non-zero counters in this file, percent coverage for this file.

If each PCNTfile corresponds to a particular test case, `-order` can be used to determine which subset of all test cases provide the most coverage for the fewest number of tests run.

-sumout *PCNTfile*

As each file in the “like” set is processed, the corresponding individual counters are added together to produce a set of summary counters which are written to *PCNTfile*.

-progress

MrProfUtil writes to Standard Error the name of each PCNTfile it is examining.

PPCProff (v. 1.0a1) Tool

PPCProff prints an analysis of the `Profiler.pgh` data file generated when you run a PowerPC application or shared library that was linked using PPCLink's `-profile` option.

For more information, see the PPCLink release notes and Chapter 17 of Building and Managing Programs in MPW.

Syntax

```
PPCProff file1 [file2]… -xcoff fileName [option]…
```

Input

One or more `.pgh` data files generated during the execution of your program. PPCProff can accept more than one `.pgh` file on the command line. For example, you can specify several `.pgh` files generated during successive runs of your program and PPCProff will display the averages of the performance data collected in the files. PPCProff also requires as input a `.xcoff` symbolic information file which is specified using the `-xcoff` option. The `.xcoff` file is automatically generated by PPCLink if your program is linked using the `-profile` option. PPCProff does not accept standard input.

Output

The analysis of the `.pgh` data file(s) is sent to standard output unless you redirect it to a specified output file.

Status

PPCProff can return the following status codes:

0 no errors

1 fatal error

Parameters

***file1* [*file2*]...**

Specifies one or more `.pgh` data files generated during the execution of your program.

Options

PPCProff supports the following options:

-cutoff *n*

Ignores routines whose hierarchical times are less than *n*%. The default is to ignore routines whose hierarchical times are less than 0.005%.

-[no]mf

Suppresses or enables the use of Process Manager temporary memory.

-p

Writes progress and summary information to diagnostic output.

-procoffsets on | off

Controls whether call site offsets (or line numbers if available) should be displayed. Line numbers can only be displayed if the `.xcoff` file, specified by the `-xcoff` option, contains line number symbolics. If the line number information is not available, hex offsets are displayed instead.

on

Displays call site offsets (or line numbers if available).

off

Doesn't display call site offsets or line numbers.

-sortorder *keyword* [,*keyword*]...

Controls how the information is displayed by using the following keywords.

flat

The routines are sorted by their flat times. Flat time is the time spent within a routine, excluding the time spent in the routines it calls.

hier[archical]

The routines are sorted by their hierarchical times. Hierarchical time is the time spent within a routine including the time spent in the routines it calls.

[calls]from

The arcs are associated with the called routine. Use this to determine the callers of the profiled routines.

[calls]to

The arcs are associated with the calling routine. Use this to find out which routines are called by the profiled routines .

-unmangle on | off

Controls whether C++ symbol names will be mangled or unmangled.

on

Uses unmangled names.

off

Uses mangled names.

-v

Writes verbose progress to diagnostic output. Specifying this option also implies `-p`.

-w[arn]

Suppresses the display of warning messages.

-xcoff *fileName*

Specifies the name of the `.xcoff` file that was generated by PPCLink when your program was linked. This file contains symbolic information that is used to indicate routine names and source file names. If your source files were compiled with `-sym on`, then this file will also contain line number information. This option is required.

PPCProff (v. 1.0a1) Tool

PPCProff prints an analysis of the `Profiler.pgh` data file generated when you run a PowerPC application or shared library that was linked using PPCLink's `-profile` option.

For more information, see the PPCLink release notes and Chapter 17 of Building and Managing Programs in MPW.

Syntax

```
PPCProff file1 [file2]… -xcoff fileName [option]…
```

Input

One or more `.pgh` data files generated during the execution of your program. PPCProff can accept more than one `.pgh` file on the command line. For example, you can specify several `.pgh` files generated during successive runs of your program and PPCProff will display the averages of the performance data collected in the files. PPCProff also requires as input a `.xcoff` symbolic information file which is specified using the `-xcoff` option. The `.xcoff` file is automatically generated by PPCLink if your program is linked using the `-profile` option. PPCProff does not accept standard input.

Output

The analysis of the `.pgh` data file(s) is sent to standard output unless you redirect it to a specified output file.

Status

PPCProff can return the following status codes:

0 no errors

1 fatal error

Parameters

***file1* [*file2*]...**

Specifies one or more `.pgh` data files generated during the execution of your program.

Options

PPCProff supports the following options:

-cutoff *n*

Ignores routines whose hierarchical times are less than *n*%. The default is to ignore routines whose hierarchical times are less than 0.005%.

-[no]mf

Suppresses or enables the use of Process Manager temporary memory.

-p

Writes progress and summary information to diagnostic output.

-procoffsets on | off

Controls whether call site offsets (or line numbers if available) should be displayed. Line numbers can only be displayed if the `.xcoff` file, specified by the `-xcoff` option, contains line number symbolics. If the line number information is not available, hex offsets are displayed instead.

on

Displays call site offsets (or line numbers if available).

off

Doesn't display call site offsets or line numbers.

-sortorder *keyword* [,*keyword*]...

Controls how the information is displayed by using the following keywords.

flat

The routines are sorted by their flat times. Flat time is the time spent within a routine, excluding the time spent in the routines it calls.

hier[archical]

The routines are sorted by their hierarchical times. Hierarchical time is the time spent within a routine including the time spent in the routines it calls.

[calls]from

The arcs are associated with the called routine. Use this to determine the callers of the profiled routines.

[calls]to

The arcs are associated with the calling routine. Use this to find out which routines are called by the profiled routines .

-unmangle on | off

Controls whether C++ symbol names will be mangled or unmangled.

on

Uses unmangled names.

off

Uses mangled names.

-v

Writes verbose progress to diagnostic output. Specifying this option also implies `-p`.

-w[arn]

Suppresses the display of warning messages.

-xcoff *fileName*

Specifies the name of the `.xcoff` file that was generated by PPCLink when your program was linked. This file contains symbolic information that is used to indicate routine names and source file names. If your source files were compiled with `-sym on`, then this file will also contain line number information. This option is required.