AProf

COLLABORATORS

	TITLE : AProf		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		December 17, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AProf

1.1 AProf.guide

Documentation for Amiga Profiler V3.2
© 1993 Michael G. Binz

1.2 Feedback

Contact me under

EMAIL: sun411@informatik.fh-augsburg.de

If it’s not possible to use EMAIL, write to:

Michael Binz
c/o Gisela Fahrner
Schießgrabenstr. 8a
8900 D-Augsburg

I'm very interested in tests of the profiler with many programming systems and Amiga configurations.

If you made tests with a compiler not listed in 'systems', please send me a small demo source file, the translated executable (with symbol hunk) and some information about creating symbol hunks with your compiler. If needed, send a description of problems and actions you performed to solve them, too.

1.3 Copyright

This software is freeware. You are allowed to copy, distribute and use it, as long as you don't change the software and distribute only the packed file (AProfilerX_XX.lzh).
You are not allowed to charge a fee higher than Fred Fish's for copying and distributing.

1.4 What is a profiler

A profiler is a development tool. It runs a specified program and collects information while it's executing.
This information can be used to find functions in your code, where optimizing will gain most effect.

1.5 Installation procedure

Copy all files included in Profilex_xx.lzh into a directory and add this directory to your PATH.

1.6 System requirements

The profiler needs at least Workbench/Kickstart 2.04 and a minimum of 500K available memory.

1.7 Compiler requirements

```
Your compiler must be able to create Amiga  symbol  hunks.  Symbols
included  in this hunks should only address the starting  addresses
of functions in your code (code symbols are automatically left  out
by the profiler).
If your compiler creates additional symbols, they must  be  removed
via the Misc/Mask Filter (Look in 'Systems', maybe your  systems is
listed there).
```

1.8 Using the profiler

```
If you use the compiler from CLI (which is recommended),  start  it
with

 1.DEV:c/app> AProf application

The profiler starts up, displays  its  user  interface,  loads  the
symbol tables from the file 'application' and waits for actions  to
be done by you. If the filename is missing, it will be requested.

@{ " Workbench usage " link ca-wb }
```

1.9 User interface

```
Menus:   @{ " Files " link m-files } @{ " Action " link m-action } @{ " Data "  ↩
    link m-data } @{ " Sort " link m-sort } @{ " Move " link m-move } @{ " Misc "  ↩
    link m-misc }
Gadgets: @{ " In/Exclusive timevalues " link gad-inex } @{ " Percentual/ ↩
    Millisecond timevalues " link gad-percmil } @{ " XTime " link gad-xtime }
         @{ " Button row " link gad-buttons }

Description of table entrys:

Function: This is the name of the symbol

HitCnt:   This is the number of times a function was called

Average:  This is the average execution time for that function

Over:     Overall execution time for that function

Min/Max:  Minimum/maximum execution time.

@{ " Bottom window frame                                           " link gad- ↩
    bottom }
```

1.10 Menu: Files

```
Open:    Read a new file
Save:    Save symbol table to file with same basename and suffix .pro
Save as: Save symbol table to selectable file
Reset:   Reset all timing values
Print:   not yet implemented
Exit:    Leave the program
About:   Display version, copyright
```

1.11 Menu: Action

```
Start: Starts the profile run. If you restart a profilee, then  the
       times are added.
```

1.12 Menu: Data

```
Exec details:
 In this requester, you can enter a command line (without  command
 name) and stack size for the profilee.

 Command line example:
   Loaded executable: 'list'
   Command line     : 'c:c#? all'
```

1.13 Menu: Sort

```
not yet implemented
```

1.14 Menu: Move

```
Find:          Enter a search string and start search
Find next:     Search for next occurence
Top:           Go to top of symbol table
Bottom:        Go to bottom of symbol table
Page up/down: One page up/down
```

1.15 Menu: Misc

```
Help:
  Start the @{ "AmigaGuide" link l-amigaguide }(TM) hypertext help system (if  ↩
      available)

Symbol filter:
  Enter a pattern for symbols you wish to exclude
```

```
 Example:
   .#?    Excludes all symbols starting with '.'
   #?%#?  Excludes all symbols including '%' in their name

  (For an explanation of Amiga pattern matching see your User Manual)

Refresh Window:
  Redisplay symbol table
```

1.16 Cycle: Inclusive/Exclusive subroutines

```
o Inclusive subroutines

 This adds the time values of all called subroutines to the callers
 time value.
 In C-programs the time value for 'main()' lies  near  the  overall
 execution time value, because normally  all  functions  are called
 from main().

o Exclusive subroutines

 Here the time values for a function exclude  the  time  needed  by
 called subroutines.
 If you add all values in the 'Over'-column of the symbol table the
 sum should be near to overall execution time of the profilee.
```

1.17 Cycle: Percentual/Millisecond timevalues

```
o Precentual timevalues

 All timevalues are in percent of the  overall  profilee  execution
 time

o Millisecond timevalues

 All timevalues are in milliseconds
```

1.18 XTime

```
Overall execution time of profilee in milliseconds
```

1.19 Button row

```
A quick way to reach some often needed menu entrys
```

1.20 Bottom window frame

```
In the bottom part of the window frame  the  compiler  will  display
user messages.
```

1.21 Configurability

```
All configurable items of AProf must be set as ToolTypes in Aprof.info.
The info file must be in the same directory as the profiler.

o WINDIM=left/top/width/height
   Use this to set profiler window size and position. (In version 3.2
   'width' must be given, but is not used)
```

1.22 Caveats

```
Here is a list of  known constructs which can result in problems if
you try to profile programs including them

@{ " Workbench based profiling " link ca-wb }
@{ " Non-standard startup modules " link ca-startup }
@{ " setjmp()/longjmp() " link ca-sljmp }
@{ " Use of CIA timers " link ca-timers }
@{ " Overlays " link ca-overlays }
@{ " Runtime limitation " link ca-runmax }
@{ " Static functions " link ca-static }
@{ " Traphandlers " link ca-traps }
@{ " Use of switch- and launch functions " link ca-swila }
```

1.23 Workbench based profiling

```
If started from workbench the profiler consumes the startup message
sent by workbench. Since profilees launched by the profiler inherit
the profilers process environment they start waiting for this work-
bench startup message, too (forever...).

As workaround use the simplest startup  module  possible.  In  most
cases this will prevent your program from waiting for this message.
On the other hand you also have to process commandline arguments in
your code.
```

1.24 Spezielle Startupmodule

```
The profilees must be able to run in the same  process  environment
as the profiler.  So it's not possible to use special startup codes
for detaching a program or making it resident.
```

1.25 setjmp()/longjmp()

```
If setjmp()/longjmp() combinations are used in the profilee, try to
execute an rts from the target function of longjmp.  This is needed
by the profilers trap logic, to know at what position  the  program
is executing.
```

1.26 CIA Timer

```
CIA timers are not available for profilees.
```

1.27 Overlays

```
Profiling of overlayed programs is not possible.
```

1.28 Runtime limitation

```
Profiler timers can measure maximum time spans of about 99 mins.
```

1.29 Static functions

```
Functions to be measured MUST be in the symbol table. This  is  not
the case for static functions in most programming environments.
```

1.30 Traphandlers

```
If your program uses a private trap handler, traps not handled must
be propagated to the previous handler.
Profiling is NOT possible if you are using trace traps (#9).
```

1.31 Switch-/Launchfunktionen

```
Profilees using members tc_Switch  and  tc_Launch  in  Exec's  Task
structure MUST propagate execution to previous defined handlers.
```

1.32 Utilities

```
 While developing the profiler, I wrote some utilities  which  some-
 times can be helpful:

@{ " StripB: " link util-stripb } Removes HUNK_SYMBOL and HUNK_DEBUG from  ↩
   executables
@{ " Sym:    "    link util-sym } Displays HUNK_SYMBOL
@{ " Seg:    "    link util-seg } Display of segment list
```

1.33 Utility: StripB

```
 StripB removes all HUNK_SYMBOL and HUNK_DEBUG from an Amiga executable

 Command line:

  StripB infile outfile
```

1.34 Utility: Sym

```
 Use Sym to display code symbols (HUNK_SYMBOL) in an Amiga executable.
                    ----

 Command line:

   Sym app ...

 Stucture of Sym output:

 Column:   1      2          3                4  5
 Data:    39   (  0 /      d90)             d3e  __Readfile__NoBuf

 Column 1: Counting number of symbol
        2: Counting number of hunk
      3: Size of hunk
      4: Address of symbol relative to current hunk
      5: Name of symbol

 Error messages:
  Symbol out of hunk bounds
      Symbol position beyond hunk limit (Used by some compilers)
```

1.35 Utility: Seg

```
 Seg displays segment list of an Amiga executable

 Command line:
  Seg prog
```

1.36 Getestete Systeme

```
 This is a list of systems I have tested the profiler with.  If  you
 find an error  or  if  you  have  tested a system not included here,
 send a message.

 @{ " Aztec C V3.2 - 5.2 " link manx-c}
 @{ " DICE V2.06.21 " link dice-c }
 @{ " GNU C/C++ Vx.x " link gnu-cpp}
 @{ " PCQ Pascal V1.2a " link pcq-pascal}
 @{ " Maxon C++ V1.2.1 " link maxon-cpp}
```

1.37 Manx Aztec C V3.2 - V5.2

```
Problems:
 None

Creation of symbol hunks:
 Use option -w for the linker

Other:
 You don't have to remove symbols named '_Hx_org'.
 Don't use detach.o with programs you want to profile.
```

1.38 SAS C Vx.x

```
 null
```

1.39 Maxon C++ 1.2.1

```
Problems:
 Sometimes mixing code and data symbols.  Check the symbol table and
 remove data symbols via menu Misc/Symbol Mask,

 Compiler is adding labels named L999 to symbol table.  This must be
 removed.

Creation of symbol hunks:
 For command line compiler use option -bs.
 In the integrated environment use menu compiler options.

Other:
 If you forget to remove L999 symbols guru follows.
 No unmangling of c++ symbols.
```

1.40 PCQ Pascal Compiler Version 1.2a + A68k V2.61 + BLink V6.7

```
Problems:
 Sometimes mixing code and data symbols.  Check the symbol table and
 remove data symbols via menu Misc/Symbol Mask,

 Compiler is adding labels named *%*. This must be removed.

Creation of symbol hunks:
 Use option -d for Assembler
```

1.41 DICE C V2.06.21 unregistered version

```
Problems:
 None.

Creation of symbol hunks:
 Use option -s for DCC.
```

1.42 GNU C/C++ Vx.x

```
Problems:
 It seems, there are no problems. I haven't tested this compiler, only
 tested some executables created with GNU C.

Creation of symbol hunks:
 ??
```

1.43 AmigaGuide (TM)

```
 AmigaGuide is a display program for text with hypertextfunctions.

 AmigaGuide is FreeWare for noncommercial users Freeware and is
 available via FTP.
```