NAME

gprof − display call graph profile data

SYNOPSIS

gprof [−[abcDhilLsTvwxyz]] [−[ACeEfFJnNOpPqQZ][*name*]]
[−I *dirs*] [−d[*num*]] [−k *from/to*]
[−m *min-count*] [−t *table-length*]
[−−[no−]annotated−source[=*name*]]
[−−[no−]exec−counts[=*name*]]
[−−[no−]flat−profile[=*name*]] [−−[no−]graph[=*name*]]
[−−[no−]time=*name*] [−−all−lines] [−−brief]
[−−debug[=*level*]] [−−function−ordering]
[−−file−ordering] [−−directory−path=*dirs*]
[−−display−unused−functions] [−−file−format=*name*]
[−−file−info] [−−help] [−−line] [−−min−count=*n*]
[−−no−static] [−−print−path] [−−separate−files]
[−−static−call−graph] [−−sum] [−−table−length=*len*]
[−−traditional] [−−version] [−−width=*n*]
[−−ignore−non−functions] [−−demangle[=*STYLE*]]
[−−no−demangle] [*image-file*] [*profile-file* ...]

DESCRIPTION

`gprof` produces an execution profile of C, Pascal, or Fortran77 programs. The effect of called routines is incorporated in the profile of each caller. The profile data is taken from the call graph profile file (*gmon.out* default) which is created by programs that are compiled with the **−pg** option of `cc`, `pc`, and `f77`. The **−pg** option also links in versions of the library routines that are compiled for profiling. `Gprof` reads the given object file (the default is `a.out`) and establishes the relation between its symbol table and the call graph profile from *gmon.out*. If more than one profile file is specified, the `gprof` output shows the sum of the profile information in the given profile files.

`Gprof` calculates the amount of time spent in each routine. Next, these times are propagated along the edges of the call graph. Cycles are discovered, and calls into a cycle are made to share the time of the cycle.

Several forms of output are available from the analysis.

The *flat profile* shows how much time your program spent in each function, and how many times that function was called. If you simply want to know which functions burn most of the cycles, it is stated concisely here.

The *call graph* shows, for each function, which functions called it, which other functions it called, and how many times. There is also an estimate of how much time was spent in the subroutines of each function. This can suggest places where you might try to eliminate function calls that use a lot of time.

The *annotated source* listing is a copy of the program's source code, labeled with the number of times each line of the program was executed.

OPTIONS

These options specify which of several output formats `gprof` should produce.

Many of these options take an optional *symspec* to specify functions to be included or excluded. These options can be specified multiple times, with different symspecs, to include or exclude sets of symbols.

Specifying any of these options overrides the default (**−p −q**), which prints a flat profile and call graph analysis for all functions.

`−A[`*symspec*`]`
`−−annotated−source[=`*symspec*`]`

The **−A** option causes `gprof` to print annotated source code. If *symspec* is specified, print output only for matching symbols.

`-b`
`--brief`
: If the **-b** option is given, `gprof` doesn't print the verbose blurbs that try to explain the meaning of all of the fields in the tables. This is useful if you intend to print out the output, or are tired of seeing the blurbs.

`-C[`*`symspec`*`]`
`--exec-counts[=`*`symspec`*`]`
: The **-C** option causes `gprof` to print a tally of functions and the number of times each was called. If *symspec* is specified, print tally only for matching symbols.

  If the profile data file contains basic-block count records, specifying the **-l** option, along with **-C**, will cause basic-block execution counts to be tallied and displayed.

`-i`
`--file-info`
: The **-i** option causes `gprof` to display summary information about the profile data file(s) and then exit. The number of histogram, call graph, and basic-block count records is displayed.

`-I` *`dirs`*
`--directory-path=`*`dirs`*
: The **-I** option specifies a list of search directories in which to find source files. Environment variable *GPROF_PATH* can also be used to convey this information. Used mostly for annotated source output.

`-J[`*`symspec`*`]`
`--no-annotated-source[=`*`symspec`*`]`
: The **-J** option causes `gprof` not to print annotated source code. If *symspec* is specified, `gprof` prints annotated source, but excludes matching symbols.

`-L`
`--print-path`
: Normally, source filenames are printed with the path component suppressed. The **-L** option causes `gprof` to print the full pathname of source filenames, which is determined from symbolic debugging information in the image file and is relative to the directory in which the compiler was invoked.

`-p[`*`symspec`*`]`
`--flat-profile[=`*`symspec`*`]`
: The **-p** option causes `gprof` to print a flat profile. If *symspec* is specified, print flat profile only for matching symbols.

`-P[`*`symspec`*`]`
`--no-flat-profile[=`*`symspec`*`]`
: The **-P** option causes `gprof` to suppress printing a flat profile. If *symspec* is specified, `gprof` prints a flat profile, but excludes matching symbols.

`-q[`*`symspec`*`]`
`--graph[=`*`symspec`*`]`
: The **-q** option causes `gprof` to print the call graph analysis. If *symspec* is specified, print call graph only for matching symbols and their children.

`-Q[`*`symspec`*`]`
`--no-graph[=`*`symspec`*`]`
: The **-Q** option causes `gprof` to suppress printing the call graph. If *symspec* is specified, `gprof` prints a call graph, but excludes matching symbols.

`-y`
`--separate-files`
: This option affects annotated source output only. Normally, `gprof` prints annotated source files to standard-output. If this option is specified, annotated source for a file named *path/filename* is generated in the file *filename-ann*. If the underlying filesystem would truncate *filename-ann* so that it overwrites the original *filename*, `gprof` generates annotated source in the file *filename.ann* instead (if

the original file name has an extension, that extension is *replaced* with *.ann*).

`-Z[`*`symspec`*`]`
`--no-exec-counts[=`*`symspec`*`]`
The **-Z** option causes `gprof` not to print a tally of functions and the number of times each was called. If *symspec* is specified, print tally, but exclude matching symbols.

`--function-ordering`
The **--function-ordering** option causes `gprof` to print a suggested function ordering for the program based on profiling data. This option suggests an ordering which may improve paging, tlb and cache behavior for the program on systems which support arbitrary ordering of functions in an executable.

The exact details of how to force the linker to place functions in a particular order is system dependent and out of the scope of this manual.

`--file-ordering` *`map_file`*
The **--file-ordering** option causes `gprof` to print a suggested .o link line ordering for the program based on profiling data. This option suggests an ordering which may improve paging, tlb and cache behavior for the program on systems which do not support arbitrary ordering of functions in an executable.

Use of the **-a** argument is highly recommended with this option.

The *map_file* argument is a pathname to a file which provides function name to object file mappings. The format of the file is similar to the output of the program `nm`.

```
c-parse.o:00000000 T yyparse
c-parse.o:00000004 C yyerrflag
c-lang.o:00000000 T maybe_objc_method_name
c-lang.o:00000000 T print_lang_statistics
c-lang.o:00000000 T recognize_objc_keyword
c-decl.o:00000000 T print_lang_identifier
c-decl.o:00000000 T print_lang_type
...
```

To create a *map_file* with GNU `nm`, type a command like `nm --extern-only --defined-only -v --print-file-name program-name`.

`-T`
`--traditional`
The **-T** option causes `gprof` to print its output in ''traditional'' BSD style.

`-w` *`width`*
`--width=`*`width`*
Sets width of output lines to *width*. Currently only used when printing the function index at the bottom of the call graph.

`-x`
`--all-lines`
This option affects annotated source output only. By default, only the lines at the beginning of a basic-block are annotated. If this option is specified, every line in a basic-block is annotated by repeating the annotation for the first line. This behavior is similar to `tcov`'s **-a**.

`--demangle[=`*`style`*`]`
`--no-demangle`
These options control whether C++ symbol names should be demangled when printing output. The default is to demangle symbols. The `--no-demangle` option may be used to turn off demangling. Different compilers have different mangling styles. The optional demangling style argument can be used to choose an appropriate demangling style for your compiler.

Analysis Options

`-a`
`--no-static`
: The **-a** option causes `gprof` to suppress the printing of statically declared (private) functions. (These are functions whose names are not listed as global, and which are not visible outside the file/function/block where they were defined.) Time spent in these functions, calls to/from them, etc, will all be attributed to the function that was loaded directly before it in the executable file. This option affects both the flat profile and the call graph.

`-c`
`--static-call-graph`
: The **-c** option causes the call graph of the program to be augmented by a heuristic which examines the text space of the object file and identifies function calls in the binary machine code. Since normal call graph records are only generated when functions are entered, this option identifies children that could have been called, but never were. Calls to functions that were not compiled with profiling enabled are also identified, but only if symbol table entries are present for them. Calls to dynamic library routines are typically *not* found by this option. Parents or children identified via this heuristic are indicated in the call graph with call counts of **0**.

`-D`
`--ignore-non-functions`
: The **-D** option causes `gprof` to ignore symbols which are not known to be functions. This option will give more accurate profile data on systems where it is supported (Solaris and HPUX for example).

`-k` *`from/to`*
: The **-k** option allows you to delete from the call graph any arcs from symbols matching symspec *from* to those matching symspec *to*.

`-l`
`--line`
: The **-l** option enables line-by-line profiling, which causes histogram hits to be charged to individual source code lines, instead of functions. If the program was compiled with basic-block counting enabled, this option will also identify how many times each line of code was executed. While line-by-line profiling can help isolate where in a large function a program is spending its time, it also significantly increases the running time of `gprof`, and magnifies statistical inaccuracies.

`-m` *`num`*
`--min-count=`*`num`*
: This option affects execution count output only. Symbols that are executed less than *num* times are suppressed.

`-n[`*`symspec`*`]`
`--time[=`*`symspec`*`]`
: The **-n** option causes `gprof`, in its call graph analysis, to only propagate times for symbols matching *symspec*.

`-N[`*`symspec`*`]`
`--no-time[=`*`symspec`*`]`
: The **-n** option causes `gprof`, in its call graph analysis, not to propagate times for symbols matching *symspec*.

`-z`
`--display-unused-functions`
: If you give the **-z** option, `gprof` will mention all functions in the flat profile, even those that were never called, and that had no time spent in them. This is useful in conjunction with the **-c** option for discovering which routines were never called.

Miscellaneous Options

`-d[`*num*`]`
`--debug[=`*num*`]`
The **-d** *num* option specifies debugging options. If *num* is not specified, enable all debugging.

`-O`*name*
`--file-format=`*name*
Selects the format of the profile data files. Recognized formats are **auto** (the default), **bsd**, **4.4bsd**, **magic**, and **prof** (not yet supported).

`-s`
`--sum`
The **-s** option causes `gprof` to summarize the information in the profile data files it read in, and write out a profile data file called *gmon.sum*, which contains all the information from the profile data files that `gprof` read in. The file *gmon.sum* may be one of the specified input files; the effect of this is to merge the data in the other input files into *gmon.sum*.

Eventually you can run `gprof` again without **-s** to analyze the cumulative data in the file *gmon.sum*.

`-v`
`--version`
The **-v** flag causes `gprof` to print the current version number, and then exit.

Deprecated Options

These options have been replaced with newer versions that use symspecs.

`-e` *function_name*
The **-e** *function* option tells `gprof` to not print information about the function *function_name* (and its children...) in the call graph. The function will still be listed as a child of any functions that call it, but its index number will be shown as **[not printed]**. More than one **-e** option may be given; only one *function_name* may be indicated with each **-e** option.

`-E` *function_name*
The `-E` *function* option works like the `-e` option, but time spent in the function (and children who were not called from anywhere else), will not be used to compute the percentages-of-time for the call graph. More than one **-E** option may be given; only one *function_name* may be indicated with each **-E** option.

`-f` *function_name*
The **-f** *function* option causes `gprof` to limit the call graph to the function *function_name* and its children (and their children...). More than one **-f** option may be given; only one *function_name* may be indicated with each **-f** option.

`-F` *function_name*
The **-F** *function* option works like the `-f` option, but only time spent in the function and its children (and their children...) will be used to determine total-time and percentages-of-time for the call graph. More than one **-F** option may be given; only one *function_name* may be indicated with each **-F** option. The **-F** option overrides the **-E** option.

FILES

a.out
the namelist and text space.

gmon.out
dynamic call graph and profile.

gmon.sum
summarized dynamic call graph and profile.

BUGS

The granularity of the sampling is shown, but remains statistical at best. We assume that the time for each execution of a function can be expressed by the total time for the function divided by the number of times the function is called. Thus the time propagated along the call graph arcs to the function's parents is directly proportional to the number of times that arc is traversed.

Parents that are not themselves profiled will have the time of their profiled children propagated to them, but they will appear to be spontaneously invoked in the call graph listing, and will not have their time propagated further. Similarly, signal catchers, even though profiled, will appear to be spontaneous (although for more obscure reasons). Any profiled children of signal catchers should have their times propagated properly, unless the signal catcher was invoked during the execution of the profiling routine, in which case all is lost.

The profiled program must call `exit`(2) or return normally for the profiling information to be saved in the *gmon.out* file.

SEE ALSO

monitor (3), *profil* (2), *cc* (1), *prof* (1), and the Info entry for *gprof*.

''An Execution Profiler for Modular Programs'', by S. Graham, P. Kessler, M. McKusick; Software − Practice and Experience, Vol. 13, pp. 671−685, 1983.

''gprof: A Call Graph Execution Profiler'', by S. Graham, P. Kessler, M. McKusick; Proceedings of the SIGPLAN '82 Symposium on Compiler Construction, SIGPLAN Notices, Vol. 17, No 6, pp. 120−126, June 1982.

COPYRIGHT