NAME

gprof - display call graph profile data

SYNOPSIS

gprof [**-abKlLsuz**] [**-C** *count*] [**-e** *name*] [**-E** *name*] [**-f** *name*] [**-F** *name*] [**-k** *fromname toname*]
[*a.out* [*a.out.gmon* ...]]

DESCRIPTION

The **gprof** utility 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 which is created by programs that are compiled with the **-pg** option of cc(1), pc(1), and f77(1). The **-pg** option also links in versions of the library routines that are compiled for profiling. By convention these libraries have their name suffixed with *_p*, i.e., the profiled version of *libc.a* is *libc_p.a* and if you specify libraries directly to the compiler or linker you can use **-l***c_p* instead of **-l***c*. Read the given object file (the default is *a.out*) and establishes the relation between its symbol table and the call graph profile. The default graph profile file name is the name of the executable with the suffix *.gmon* appended. If more than one profile file is specified, the **gprof** output shows the sum of the profile information in the given profile files.

The **gprof** utility 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. The first listing shows the functions sorted according to the time they represent including the time of their call graph descendants. Below each function entry is shown its (direct) call graph children, and how their times are propagated to this function. A similar display above the function shows how this function's time and the time of its descendants is propagated to its (direct) call graph parents.

Cycles are also shown, with an entry for the cycle as a whole and a listing of the members of the cycle and their contributions to the time and call counts of the cycle.

Second, a flat profile is given, similar to that provided by prof(1). This listing gives the total execution times, the call counts, the time that the call spent in the routine itself, and the time that the call spent in the routine itself including its descendants. The units for the per-call times are normally milliseconds, but they are nanoseconds if the profiling clock frequency is 10 million or larger, and if a function appears to be never called then its total self time is printed as a percentage in the self time per call column. The very high profiling clock frequencies needed to get sufficient accuracy in the per-call times for short-lived programs are only implemented for "high resolution" (non-statistical) kernel profiling.

Finally, an index of the function names is provided.

The following options are available:

-a Suppress the printing of statically declared functions. If this option is given, all relevant information about the static function (e.g., time samples, calls to other functions, calls from other functions) belongs to the function loaded just before the static function in the *a.out* file.

-b Suppress the printing of a description of each field in the profile.

-C *count*
Find a minimal set of arcs that can be broken to eliminate all cycles with *count* or more members. Caution: the algorithm used to break cycles is exponential, so using this option may cause **gprof** to run for a very long time.

-e *name*
Suppress the printing of the graph profile entry for routine *name* and all its descendants (unless they have other ancestors that are not suppressed). More than one **-e** option may be given. Only one *name* may be given with each **-e** option.

-E *name*
Suppress the printing of the graph profile entry for routine *name* (and its descendants) as **-e**, above, and also excludes the time spent in *name* (and its descendants) from the total and percentage time computations. (For example, **-E** *mcount* **-E** *mcleanup* is the default.)

-f *name*
Print the graph profile entry of only the specified routine *name* and its descendants. More than one **-f** option may be given. Only one *name* may be given with each **-f** option.

-F *name*
Print the graph profile entry of only the routine *name* and its descendants (as **-f**, above) and also uses only the times of the printed routines in total time and percentage computations. More than one **-F** option may be given. Only one *name* may be given with each **-F** option. The **-F** option overrides the **-E** option.

-k *fromname toname*
Will delete any arcs from routine *fromname* to routine *toname*. This can be used to break undesired cycles. More than one **-k** option may be given. Only one pair of routine names may be given with each **-k** option.

-K Gather information about symbols from the currently-running kernel using the sysctl(3) and kldsym(2) interfaces. This forces the *a.out* argument to be ignored, and allows for symbols in

kld(4) modules to be used.

-l Suppress the printing of the call-graph profile.

-L Suppress the printing of the flat profile.

-s A profile file *gmon.sum* is produced that represents the sum of the profile information in all the specified profile files. This summary profile file may be given to later executions of gprof (probably also with a **-s**) to accumulate profile data across several runs of an *a.out* file.

-u Suppress the printing of functions whose names are not visible to C programs. For the ELF object format, this means names that contain the '.' character. For the a.out object format, it means names that do not begin with a '_' character. All relevant information about such functions belongs to the (non-suppressed) function with the next lowest address. This is useful for eliminating "functions" that are just labels inside other functions.

-z Display routines that have zero usage (as shown by call counts and accumulated time).

FILES

a.out The namelist and text space.
a.out.gmon Dynamic call graph and profile.
gmon.sum Summarized dynamic call graph and profile.

SEE ALSO

cc(1), profil(2), clocks(7), pmcstat(8)

S. Graham, P. Kessler, and M. McKusick, "An Execution Profiler for Modular Programs", *Software - Practice and Experience*, 13, pp. 671-685, 1983.

S. Graham, P. Kessler, and M. McKusick, "gprof: A Call Graph Execution Profiler", *Proceedings of the SIGPLAN '82 Symposium on Compiler Construction, SIGPLAN Notices*, 6, 17, pp. 120-126, June 1982.

HISTORY

The **gprof** profiler appeared in 4.2BSD.

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(3) or return normally for the profiling information to be saved in the graph profile file.