Step 1. Enable profiling during compilation (use -pg option)

$ gcc -pg -o TestGprof TestGprof.c

Step 2. Execute the binary so that profiling data is generated

$ ./TestGprof

If the profiling is enabled then on executing the program, file gmon.out will be generated.

$ ls

gmon.out TestGprof TestGprof.c

Step 3. Run gprof on profiling data

$ gprof -b TestGprof gmon.out > analysis.out

This will give an human readable file. This file contains two tables:

flat profile: overview of the timing information of the functions

call graph: focuses on each function

-b option will suppress lot of verbose information which would be otherwise included in analysis file.

Below are the contents of analysis.out

Flat profile:

Each sample counts as 0.01 seconds.

% cumulative self self total

time seconds seconds calls s/call s/call name

100.55 20.38 20.38 100 0.20 0.20 StaticFunc

0.00 20.38 0.00 1 0.00 20.38 TestFunc

Call graph

granularity: each sample hit covers 2 byte(s) for 0.05% of 20.38 seconds

index % time self children called name

20.38 0.00 100/100 TestFunc [2]

[1] 100.0 20.38 0.00 100 StaticFunc [1]

-----------------------------------------------

0.00 20.38 1/1 main [3]

[2] 100.0 0.00 20.38 1 TestFunc [2]

20.38 0.00 100/100 StaticFunc [1]

-----------------------------------------------

[3] 100.0 0.00 20.38 main [3]

0.00 20.38 1/1 TestFunc [2]

-----------------------------------------------

Index by function name

[1] StaticFunc [2] TestFunc

Below are few more examples of gprof options

Suppress printing statically declared functions with -a option

$ gprof -b -a TestGprof gmon.out > analysis.out

Print only flat profile using -p option

$ gprof -b -p TestGprof

Print info related to specific function

$ gprof -b -pStaticFunc TestGprof

Suppress printing of flat profile using -P option

$ gprof -b -P TestGprof

Print only call graph using -q option

$ gprof -b -q TestGprof

Suppress printing of call graph using -Q option

$ gprof -b -Q TestGprof