Profiling with gprof

- **Basic profiling tool under Linux: `gprof`**
- **Compiling for a profiling run**

  executable to be created

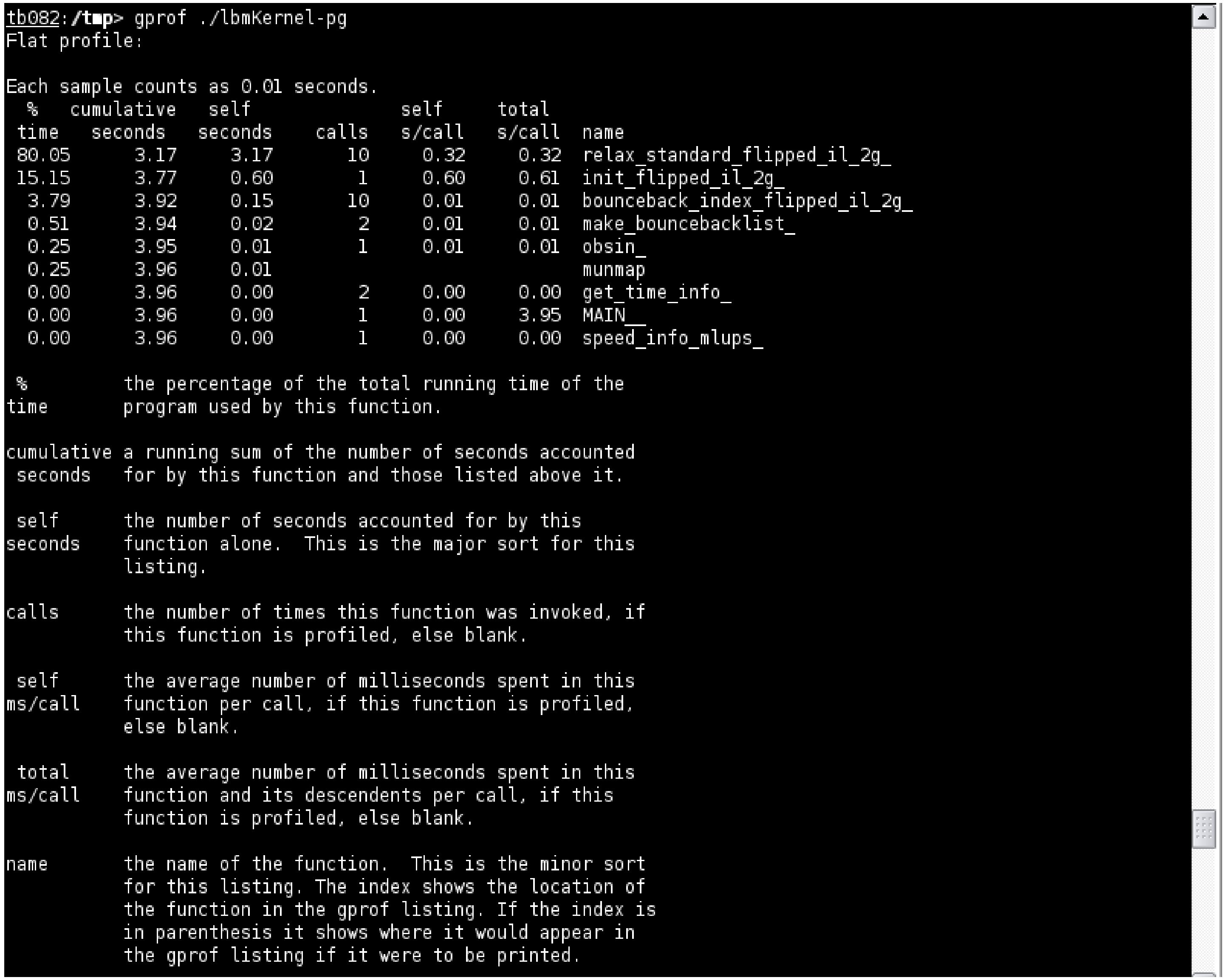
```
icc -pg …… -o a.out
./a.out
```

- **After running the binary, a file `gmon.out` is written to the current directory**
- **Human readable output:**

```
gprof a.out
```

- **Inlining should be disabled for profiling**
  - But then the executed code isn't what it should be…
- **Profiling may (substantially) reduce overall code performance**

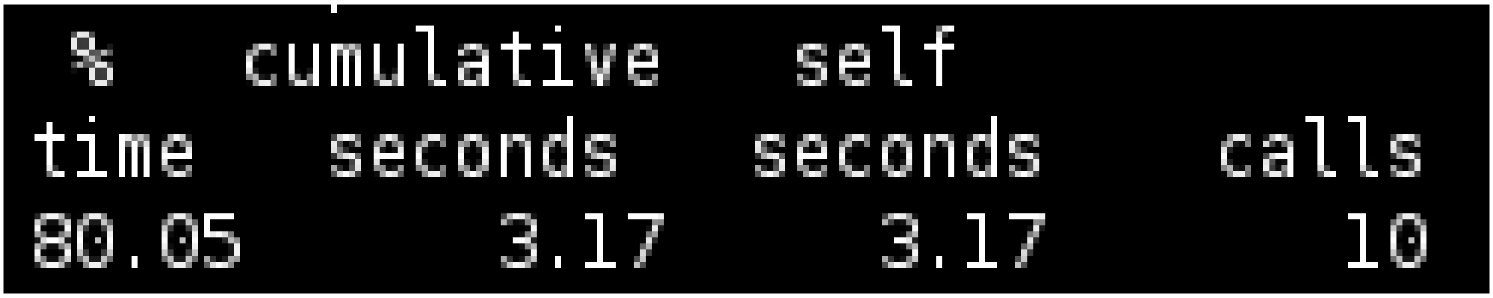
Profiling with gprof: Example (sample - output)



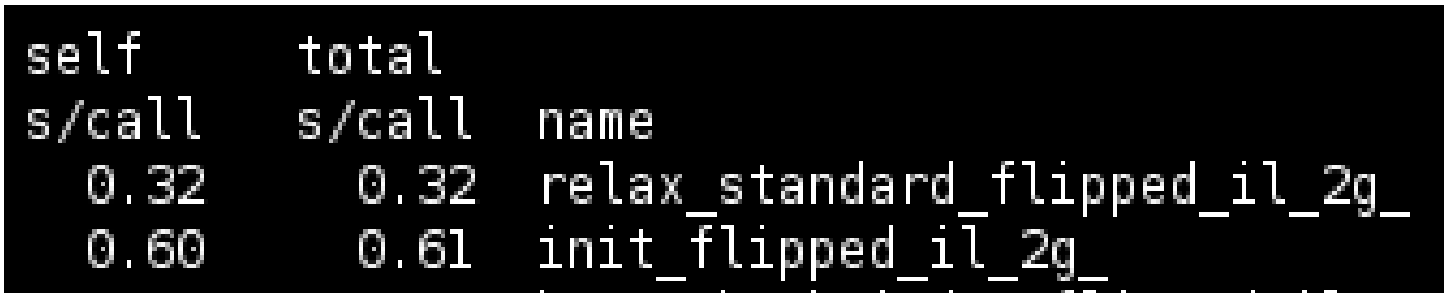
```
tb082:/tmp> gprof ./lbmKernel-pg
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls   s/call   s/call  name
 80.05      3.17     3.17       10     0.32     0.32  relax_standard_flipped_il_2g_
 15.15      3.77     0.60        1     0.60     0.61  init_flipped_il_2g_
  3.79      3.92     0.15       10     0.01     0.01  bounceback_index_flipped_il_2g_
  0.51      3.94     0.02        2     0.01     0.01  make_bouncebacklist_
  0.25      3.95     0.01        1     0.01     0.01  obsin_
  0.25      3.96     0.01                             munmap
  0.00      3.96     0.00        2     0.00     0.00  get_time_info_
  0.00      3.96     0.00        1     0.00     3.95  MAIN__
  0.00      3.96     0.00        1     0.00     0.00  speed_info_mlups_

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
           else blank.

 total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
           function is profiled, else blank.

name       the name of the function.  This is the minor sort
           for this listing. The index shows the location of
           the function in the gprof listing. If the index is
           in parenthesis it shows where it would appear in
           the gprof listing if it were to be printed.
```

Test of kernel routine:

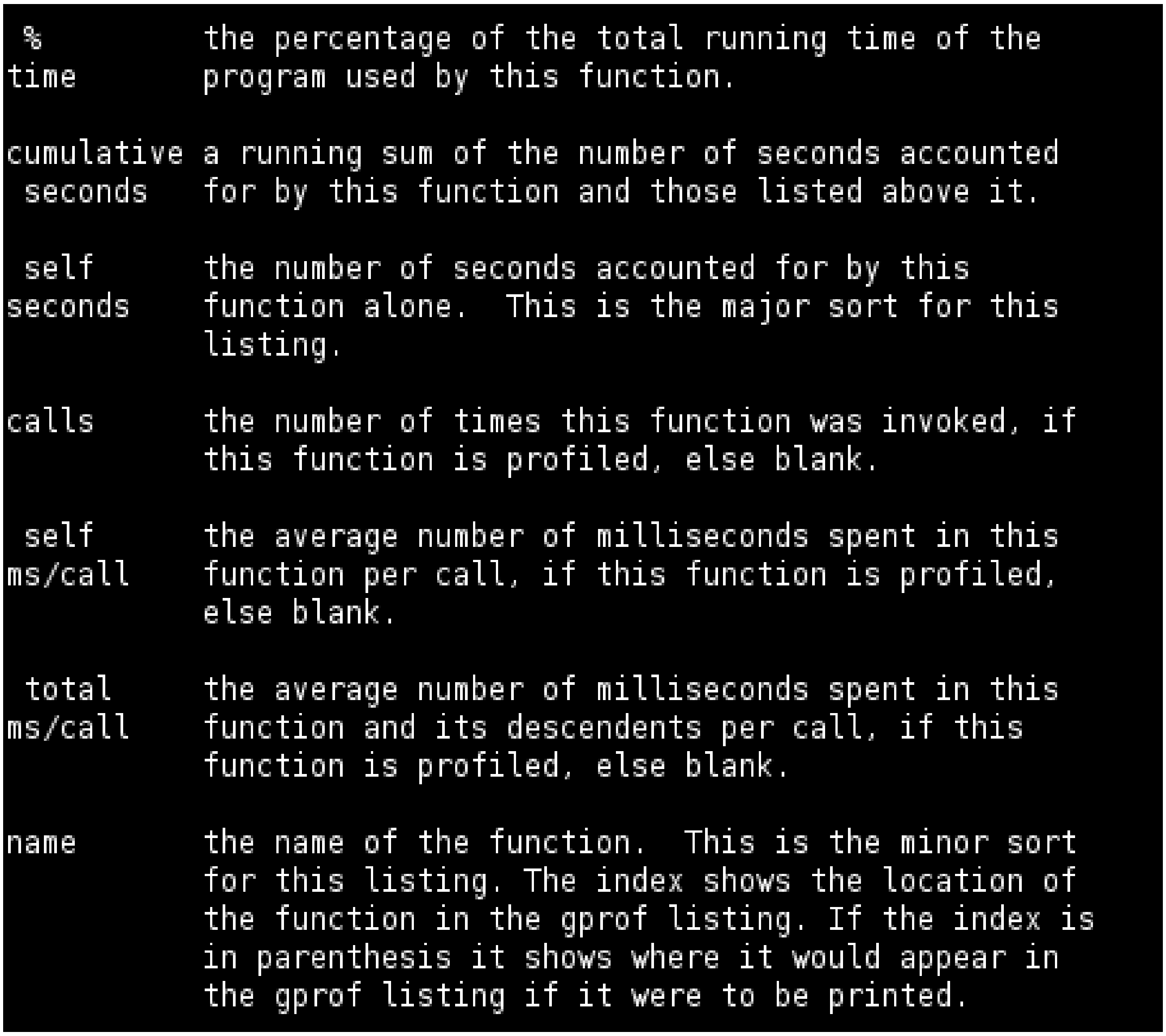
- Initialize
- Run the 2 computational kernels 10 times



```
 %   cumulative   self
time   seconds   seconds    calls
80.05     3.17     3.17       10
```

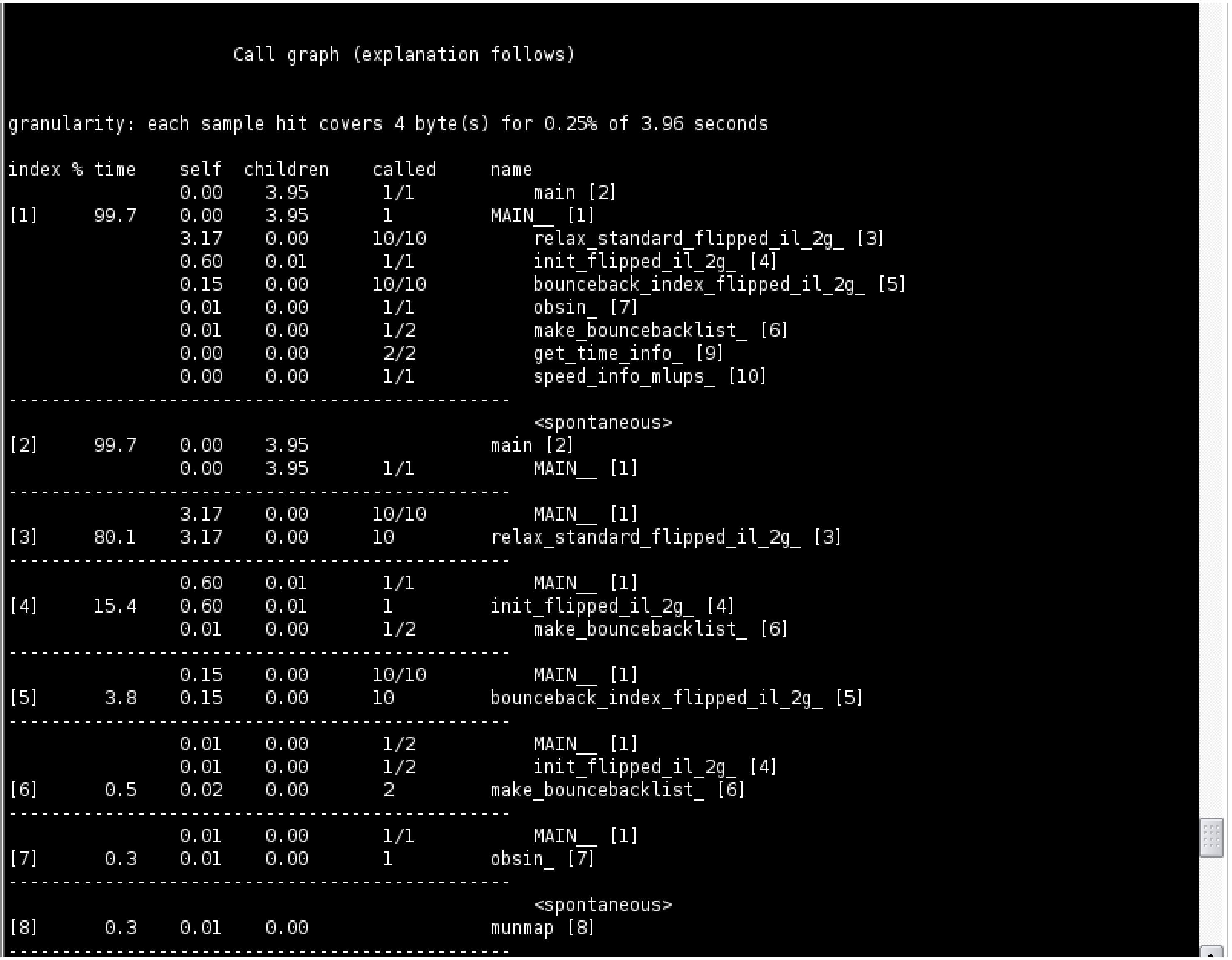


```
self     total
s/call   s/call  name
  0.32     0.32  relax_standard_flipped_il_2g_
  0.60     0.61  init_flipped_il_2g_
```



```
 %          the percentage of the total running time of the
time        program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self       the number of seconds accounted for by this
seconds     function alone.  This is the major sort for this
            listing.

calls       the number of times this function was invoked, if
            this function is profiled, else blank.

 self       the average number of milliseconds spent in this
ms/call     function per call, if this function is profiled,
            else blank.

 total      the average number of milliseconds spent in this
ms/call     function and its descendents per call, if this
            function is profiled, else blank.

name        the name of the function.  This is the minor sort
            for this listing. The index shows the location of
            the function in the gprof listing. If the index is
            in parenthesis it shows where it would appear in
            the gprof listing if it were to be printed.
```

Profiling with gprof: Example (sample - output)



```
                    Call graph (explanation follows)


granularity: each sample hit covers 4 byte(s) for 0.25% of 3.96 seconds

index % time    self  children    called     name
                0.00    3.95       1/1           main [2]
[1]     99.7    0.00    3.95       1         MAIN__ [1]
                3.17    0.00      10/10          relax_standard_flipped_il_2g_ [3]
                0.60    0.01       1/1           init_flipped_il_2g_ [4]
                0.15    0.00      10/10          bounceback_index_flipped_il_2g_ [5]
                0.01    0.00       1/1           obsin_ [7]
                0.01    0.00       1/2           make_bouncebacklist_ [6]
                0.00    0.00       2/2           get_time_info_ [9]
                0.00    0.00       1/1           speed_info_mlups_ [10]
-----------------------------------------------
                                                 <spontaneous>
[2]     99.7    0.00    3.95                 main [2]
                0.00    3.95       1/1           MAIN__ [1]
-----------------------------------------------
                3.17    0.00      10/10          MAIN__ [1]
[3]     80.1    3.17    0.00      10         relax_standard_flipped_il_2g_ [3]
-----------------------------------------------
                0.60    0.01       1/1           MAIN__ [1]
[4]     15.4    0.60    0.01       1         init_flipped_il_2g_ [4]
                0.01    0.00       1/2           make_bouncebacklist_ [6]
-----------------------------------------------
                0.15    0.00      10/10          MAIN__ [1]
[5]      3.8    0.15    0.00      10         bounceback_index_flipped_il_2g_ [5]
-----------------------------------------------
                0.01    0.00       1/2           MAIN__ [1]
                0.01    0.00       1/2           init_flipped_il_2g_ [4]
[6]      0.5    0.02    0.00       2         make_bouncebacklist_ [6]
-----------------------------------------------
                0.01    0.00       1/1           MAIN__ [1]
[7]      0.3    0.01    0.00       1         obsin_ [7]
-----------------------------------------------
                                                 <spontaneous>
[8]      0.3    0.01    0.00                 munmap [8]
-----------------------------------------------
```

Butterfly graph

Who calls whom and how often?

Profiling with gprof: Example (C++)

- **Example with wrapped `double` class:**

```
class D {
  double d;
public:
  D(double _d=0) : d(_d) {}
  D operator*(const D& o) {
    D r;
    r.d = d*o.d;
    return r;
  }
  operator double() {
    return d;
  }
};
```

Main program:

```
const int n=10000000;
D a[n],b[n];
D sum;

for(int i=0; i<n; ++i)
  a[i] = b[i] = 1.5;

double s = timestamp();
for(int k=0; k<10; ++k) {
  for(int i=0; i<n; ++i)
    sum = sum + a[i] * b[i];
}
```

Profiling with gprof: Example (C++) profiler output

- `icpc -O3 -pg perf.cc`

```
  %   cumulative   self              self     total
 time   seconds   seconds    calls  Ts/call  Ts/call  name
101.01      0.41     0.41                             main
```

- `icpc -O3 -fno-inline -pg perf.cc`

```
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ns/call  ns/call  name
 46.44      0.59     0.59 200000000     2.93     4.48  D::operator*(D const&)
 29.63      0.96     0.37 240000001     1.56     1.56  D::D(double)
 24.82      1.27     0.31                             main
```

- **But where did the time *actually* go?**
  - Butterfly (callgraph) profile also available
  - Real problem also with use of libraries (STL!)
  - Sometimes you have to roll your own little profiler