perf: Flat Profile Usage

1. Specify the **`-g3`** and **`-Og`** options when compiling code. You can do this by compiling with `make debug`.

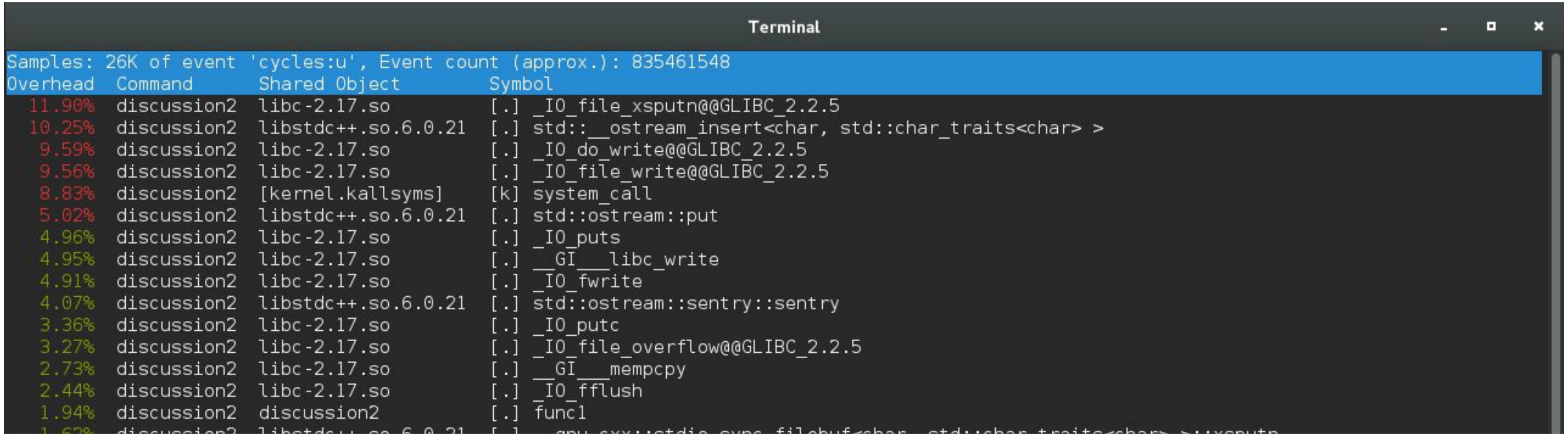
2. Run: `perf record -e cycles:u [normal command line]`

   a. This will create a file called perf.data in the current directory. This file may be very large, if you are using version control, it is a good idea to exclude it.

   b. Uses CPU cycle counters to measure proportional time spent in functions.

3. Run perf report to get a nice Textual User Interface (TUI) display of the recorded data.

perf: Flat Profile Report



```
Terminal

Samples: 26K of event 'cycles:u', Event count (approx.): 835461548
Overhead  Command       Shared Object         Symbol
  11.90%  discussion2   libc-2.17.so          [.] _IO_file_xsputn@@GLIBC_2.2.5
  10.25%  discussion2   libstdc++.so.6.0.21   [.] std::__ostream_insert<char, std::char_traits<char> >
   9.59%  discussion2   libc-2.17.so          [.] _IO_do_write@@GLIBC_2.2.5
   9.56%  discussion2   libc-2.17.so          [.] _IO_file_write@@GLIBC_2.2.5
   8.83%  discussion2   [kernel.kallsyms]     [k] system_call
   5.02%  discussion2   libstdc++.so.6.0.21   [.] std::ostream::put
   4.96%  discussion2   libc-2.17.so          [.] _IO_puts
   4.95%  discussion2   libc-2.17.so          [.] __GI___libc_write
   4.91%  discussion2   libc-2.17.so          [.] _IO_fwrite
   4.07%  discussion2   libstdc++.so.6.0.21   [.] std::ostream::sentry::sentry
   3.36%  discussion2   libc-2.17.so          [.] _IO_putc
   3.27%  discussion2   libc-2.17.so          [.] _IO_file_overflow@@GLIBC_2.2.5
   2.73%  discussion2   libc-2.17.so          [.] __GI___mempcpy
   2.44%  discussion2   libc-2.17.so          [.] _IO_fflush
   1.94%  discussion2   discussion2           [.] func1
   1.62%  discussion2   libstdc++.so.6.0.21   [.] __gnu_cxx::stdio_sync_filebuf<char, std::char_traits<char> >::xsputn
```

perf: Call Graph Usage

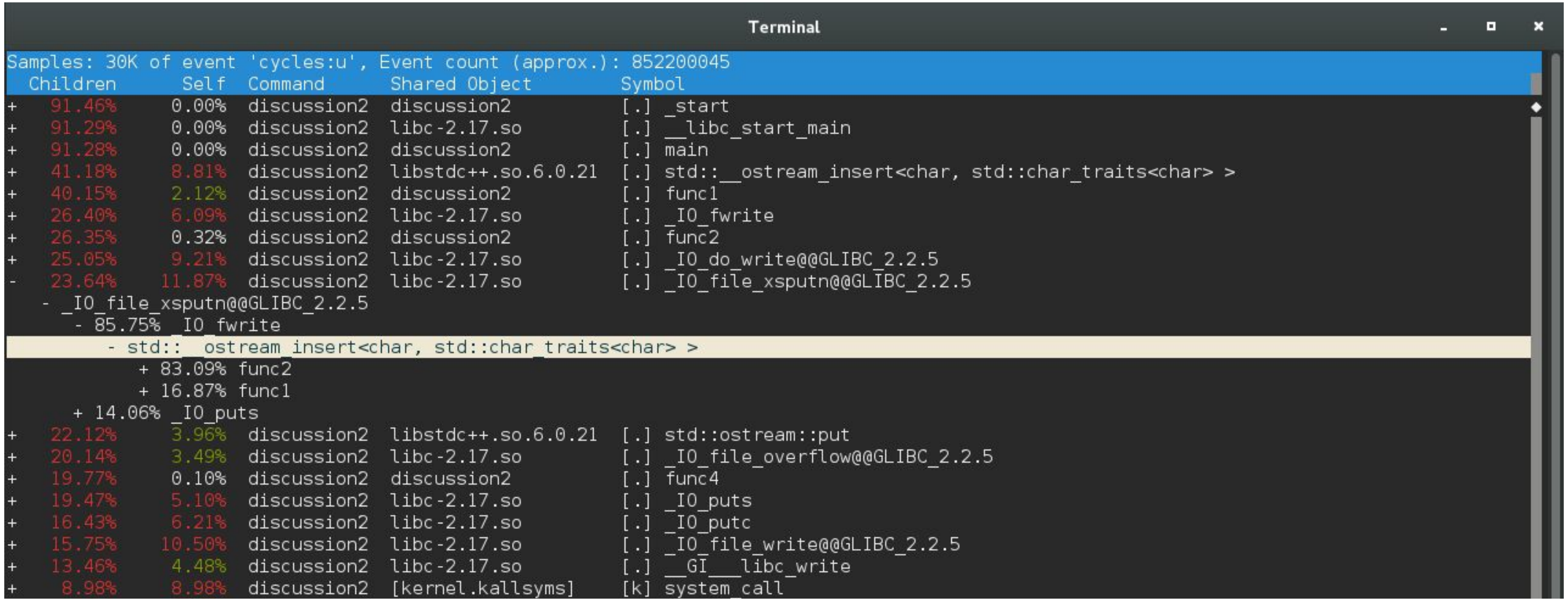
1. Specify the **`-g`** and **`-Og`** options when compiling code

2. Run: `perf record --call-graph dwarf -e cycles:u [normal command line]`

3. Run perf report to get a nice Textual User Interface (TUI) display of the recorded data.

4. When browsing the `perf report` view, use up/down arrow keys to move around, use <enter> to expand a graph, and use ‘q’ to exit.

perf: Call Graph Report



```
Samples: 30K of event 'cycles:u', Event count (approx.): 852200045
  Children      Self  Command       Shared Object          Symbol
+   91.46%     0.00%  discussion2   discussion2            [.] _start
+   91.29%     0.00%  discussion2   libc-2.17.so           [.] __libc_start_main
+   91.28%     0.00%  discussion2   discussion2            [.] main
+   41.18%     8.81%  discussion2   libstdc++.so.6.0.21    [.] std::__ostream_insert<char, std::char_traits<char> >
+   40.15%     2.12%  discussion2   discussion2            [.] func1
+   26.40%     6.09%  discussion2   libc-2.17.so           [.] _IO_fwrite
+   26.35%     0.32%  discussion2   discussion2            [.] func2
+   25.05%     9.21%  discussion2   libc-2.17.so           [.] _IO_do_write@@GLIBC_2.2.5
-   23.64%    11.87%  discussion2   libc-2.17.so           [.] _IO_file_xsputn@@GLIBC_2.2.5
   - _IO_file_xsputn@@GLIBC_2.2.5
      - 85.75% _IO_fwrite
         - std::__ostream_insert<char, std::char_traits<char> >
            + 83.09% func2
            + 16.87% func1
      + 14.06% _IO_puts
+   22.12%     3.96%  discussion2   libstdc++.so.6.0.21    [.] std::ostream::put
+   20.14%     3.49%  discussion2   libc-2.17.so           [.] _IO_file_overflow@@GLIBC_2.2.5
+   19.77%     0.10%  discussion2   discussion2            [.] func4
+   19.47%     5.10%  discussion2   libc-2.17.so           [.] _IO_puts
+   16.43%     6.21%  discussion2   libc-2.17.so           [.] _IO_putc
+   15.75%    10.50%  discussion2   libc-2.17.so           [.] _IO_file_write@@GLIBC_2.2.5
+   13.46%     4.48%  discussion2   libc-2.17.so           [.] __GI___libc_write
+    8.98%     8.98%  discussion2   [kernel.kallsyms]      [k] system_call
```

perf: Tips

- Prefer CAEN to your personal laptop, even if you run Linux
  - `ssh` is fine
  - **Do not** use a Linux VM on OS X/Windows for `perf`!
- Use large test cases for profiling! Smaller ones won't give you much info
- Use `-Og` or `-g3` optimization level
  - Using `-Og` shows just the time spent in your code
  - Using `-g3` shows the time spent in the STL *also*
- If you don't know what a function is in the flat profile, make a call graph and see where it's called from.
- Exclude `perf.data*` from version control

perf: Caveats

- Statistical accuracy
  - Do <u>not</u> interpret run times as absolute truth.

    There ***will*** be variance between running this on your local laptop and on a CAEN computer!
  - Run time will vary slightly from run to run
- In flat profile mode, times are summed across all calls to that function.
  - Function might be very slow on one kind of input, and very faster on every other kind
- Mutual Recursion A-->B-->A-->B
  - Makes the call graph more difficult to read by separating into cycles