ENSC 351

Fall 2018

Lab 1

Retracing our steps

Deadline: September 24th, 2018
Last revision: September 10, 2018

Karol Swietlicki
Simon Fraser University

1 A trace library

I hope you like Google Chrome, because you will be using it a lot in this course.

Did you know that Chrome has a built-in trace reader? It does!

Open Chrome and go to chrome://tracing

See that "Record" button? Go on, hit that. Stop the recording within a second or two of starting it.

You can now see everything that happened in your browser over that second. At different levels of details you can see events start and stop, states, variable dumps...

But we aren't web developers and we don't care about the Record button. We care about the one labelled "Load". Yup, you can load your own, externally generated, traces into Chrome's trace viewer.

And the whole class will hinge on that.

2 The task at hand

Write a library that will emit a program trace to a file.

You need the following functions.

- void trace_start(char* filename)
- void trace_event_start(char* name, char* categories, char* arguments)
- void trace_event_end(char* arguments)
- void trace_instant_global(char* name)
- void trace_object_new(char* name, void* obj_pointer)
- void trace_object_gone(char* name, void* obj_pointer)
- void trace_counter(char* name, char* key, char* value)
- void trace_flush()
- void trace_end()

3 The trace format

The traces are text files. The trace starts with a [character and contains a bunch of events, each event enclosed in a pair of curly braces and events separated from each other by a comma. Whitespace is irrelevant.

The details of the format are in a second PDF document.

4 Design considerations

The job of the trace library is to capture the behavior of the program with minimum interference, including timing behavior. Ideally, your program would behave identically without the tracing enabled and with it on. A debugger is extremely heavy, so it won't do the job and it won't help much when dealing with threads. We need something else. We can approximate the desired properties with proper design.

It may be tempting to immediately write all trace data to a file as soon as a new data item arrives. This is a really bad idea, however. Tracing must be as non-invasive as possible for the image that you acquire to be clear of distortion.

Your trace system must only do file I/O when either ten thousand trace items have been accumulated, when the program crashes, when the trace is manually flushed to disk or when the trace is ended.

While the output that needs to land in a file is a string, your trace should not accumulate strings until the trace is about to be written out to disk. Store your trace as lightweight objects/structs and only render them to a string representation at the time of the trace flush. Make sure to issue only one I/O call to dump each chunk of the trace to the output file.

Your trace system must not allocate memory at any point in time, trace_start being the sole exception. Both your trace entry objects and your string buffer must be allocated there. Memory allocation can take a very long time, distorting the trace.

Your trace system must not acquire the thread ID or the process ID at any time, trace_start being the sole exception.

While this version of the trace library does not need to support more than one thread, the future versions will need such support. Plan accordingly.

Should your program crash, you need to output the trace up to the point of the crash. This is usually done through installing a SIGSEGV handler.

5 Grading

This lab will not be graded, though we will tell you if you have done your work correctly or not.

You will likely regret it if you don't do this lab, but we will let you skip it.

Future pieces of lab work rely very strongly on a functioning trace library. We won't be requiring you to use one, but it will be of great help to see the system's behavior. We won't be releasing an answer key, so you will either have to write your own eventually or try to debug your code blindly. Good luck to you in the latter case.