CPSC 427: Object-Oriented Programming

Michael J. Fischer

Lecture 14
March 8, 2016

Feedback on Midterm Exam

Demo: Stopwatch

Feedback on Midterm Exam

Exam Feedback

An exam is an assessment that is useful to student and instructor alike.

It allows the student to measure progress and to identify areas that remain unclear.

Likewise, it give the instructor feedback on topics that were not explained well or that need more attention.

Approximately an hour was spent going over each of the exam questions, both to cover what the answers were that I was looking for as well as to address common errors and misconceptions that were apparent in some of the incorrect answers.

Demo: Stopwatch

Realtime measurements

`StopWatch` is a class I wrote for measuring realtime performance of code.

It emulates a stopwatch with 3 buttons: `reset`, `start`, and `stop`.

At any time, the watch displays the cumulative time that the stopwatch has been running.

HirezTime class

HirezTime is a wrapper class for the system-specific functions to read the clock.

It hides the details of the underlying time representation and provides a simple interface for reading, computing, and printing times and time intervals.

HirezTime objects are intended to be copied rather than pointed at, and to behave like other numeric types.

Versions of `HirezTime`

There are two versions:

14-StopWatch (Linux/Unix/Darwin) Function `gettimeofday()` returns the clock in a `struct timeval`, which consists of two `long int`s representing seconds and microseconds. The resolution of the clock is system-dependent, typically 1 millisecond. (See demo `14-StopWatch`.)

14-StopWatch-hirez (Linux only) Function `clock_gettime()` returns the clock in a `struct timespec`, which consists of two `long int`s representing seconds and nanoseconds. The resolution of the clock is system-dependent and can be obtained with the `clock_getres()` function. (See demo `14-StopWatch-hirez`.)

HirezTime structure

- In C++, struct T and class T are very similar. In both cases, T becomes a new type name.
- struct members are public by default.
  class members are private by default.
- HirezTime is derived from struct timeval or struct timespec, depending on the version.
- It uses protected derivation to hide the underlying representation.
- It presents two interfaces to the world:
  1. The normal public interface treats HirezTime as an opaque object.
  2. A class derived from it can access the fields of the underlying timespec/timeval.

Printing a HirezTime number

Something seemingly simple like printing HirezTime values is not so simple. Naively, one might write:

```
cout << t.tv_sec << "." << t.tv_usec;
```

where tv_sec and tv_usec are the seconds and microseconds fields of a timeval structure.

If t represents 2 seconds and 27 microseconds, then what would print is 2.27, not the correct 2.000027.

The class contains a print function that fixes this problem.

StopWatch class

StopWatch contains five member variables to record

- Whether the watch is running or not.
- The cumulative run time to point when last stopped.
- The most recent start and stop times.

All functions are inline to minimize inaccuracies of measurement due to the overhead withing the stopwatch code itself.

Casting a StopWatch to a HirezTime

An operator extension defines a cast for reading the cumulative time from a stop watch:

```
    operator HirezTime() const { return cumSpan; }
```

Thus, if `sw` is a `StopWatch` instance,

```
    cout << sw;
```

will print `sw.cumSpan` using `sw.print()`.

Why it works

This works because `operator<<()` is not defined for righthand operands of type `StopWatch` but it is defined for `HirezTime`.

The compiler then **coerces** `sw` to something that is acceptable to the `<<` operator.

Because `operator HirezTime()` is defined for class `StopWatch`, the compiler will invoke it to obtain a `HirezTime` object, for which `<<` is defined.

Note that a similar coercion happens when one writes

```
    if(!in) {...}
```

to test if an `istream` object `in` is open for reading. Here, the `istream` object is coerced to a `bool` because `operator bool()` is defined inside the streams package.