TDD38 - Extra lecture

The chrono header

Eric Elfving

Department of Computer and Information Science
Linköping University

The `<chrono>` header has three basic types;

- `duration` represents a span of time (`count` number of ticks with a specific `period`)
- `time_point` to represent a specific position in time. Defined as a `duration` from a specific `epoch`
- Clocks - relate a timepoint to a physical time.

Clocks

There are at least three clocks available:

system_clock	Using the system-wide realtime clock - good to get actual time.
steady_clock	monotonic and stable. Member `now` will never return a value smaller than a previous call and each tick will be equal in time - to calculate difference in time.
high_resolution_clock	The clock with highest precision (smallest tick period). Usually the same as `steady_clock` or `system_clock`

Examples

```
duration<int,ratio<60*60*24> > one_day (1);

system_clock::time_point today = system_clock::now();
system_clock::time_point tomorrow = today + one_day;

std::time_t tt;

tt = system_clock::to_time_t ( today );
std::cout << "today is: " << ctime(&tt);

tt = system_clock::to_time_t ( tomorrow );
std::cout << "tomorrow will be: " << ctime(&tt);
```

`ctime` takes a `time_t *` and returns some string (`const char *`) representation...

Examples

```
auto now = system_clock::now();
time_t tt { system_clock::to_time_t(now) };
cout << "Todays date: " << put_time(localtime( &tt ), "%Y-%m-%d") << endl;
```

```
Todays date: 2015-12-01
```

Uses `put_time` from `<iomanip>`, see `http://www.cplusplus.com/reference/iomanip/put_time/` for all format specifiers.

Examples

```
steady_clock::time_point t1 = steady_clock::now();

// Do some complicated stuff...

steady_clock::time_point t2 = steady_clock::now();

duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

std::cout << "It took me " << time_span.count() << " seconds.";
std::cout << std::endl;
```

Examples

```
class Timer
{
    using sc = std::chrono::steady_clock;
public:
    Timer(): start{sc::now()}, stream{std::cout} {}
    Timer(std::ostream & os): start{sc::now()}, stream{os} {}
    ~Timer()
    {
        auto duration = sc::now() - start;
        auto diff = duration_cast<duration<long double>>(duration);
        stream << "lived for " << diff.count() << "seconds\n";
    }
private:
    sc::time_point start;
    std::ostream & stream;
};
```

User-defined literals

There are a few user-defined literals for duration values (in C++14)

```
auto lecture_length = 2h;
```

See `http://en.cppreference.com/w/cpp/chrono/duration` for all.

www.liu.se

LINKÖPING
UNIVERSITY