CSCI251/CSCI851 Autumn-2020
Advanced Programming (**S3b**)

Getting organised II:
A little bit of Time

Outline

- OS time on capa.
- C-style: std::time
- C++11: std::chrono

Looking at the time …

```
$ time ./calling

real    0m0.046s
user    0m0.032s
sys     0m0.009s
```

- If we want to know how long our program takes overall, on capa we can use `time`.
- Real: Start to end.
- User: CPU time in user mode.
- Sys: CPU time in sys mode.
- Total CPU time: user+sys
- If you were using a multi-processor system you can have Real < user+sys.

C-style dates and times
std::time in header <ctime>

- See https://en.cppreference.com/w/cpp/chrono/c/time

```
#include <ctime>
#include <iostream>

int main()
{
      std::time_t result = std::time(nullptr);
      std::cout << std::asctime(std::localtime(&result))
            << result << " seconds since the Epoch\n";

      std::cout << CLOCKS_PER_SEC << std::endl;
}
```

Time in C++11

- http://en.cppreference.com/w/cpp/chrono
- Three main types are defined:
  - Clocks:
    - With an epoch and a tick rate.
      - The tick rate is effectively a number of steps within a second.
  - Time points:
    - Duration of time since epoch.
  - Durations:
    - Span of time associated with some number of ticks.

```
#include <iostream>
#include <chrono>

long fibonacci(unsigned n)
{
    if (n < 2) return n;
    return fibonacci(n-1) + fibonacci(n-2);
}

int main()
{
    auto start = std::chrono::system_clock::now();
    std::cout << "f(35) = " << fibonacci(35) << '\n';
    auto end = std::chrono::system_clock::now();

    std::chrono::duration<double> elapsed_seconds = end-start;
    std::time_t end_time = std::chrono::system_clock::to_time_t(end);

    std::cout << "finished computation at " << std::ctime(&end_time)
              << "elapsed time: " << elapsed_seconds.count() << "s\n";
}
```

But wait … Sleeping …

- https://en.cppreference.com/w/cpp/thread/sleep_for
- http://www.cplusplus.com/reference/chrono/
- With the headers `thread` and `chrono`, you can access functionality to have your program wait for some specified length of time.

```
std::this_thread::sleep_for(std::chrono::seconds(2));
std::this_thread::sleep_for(std::chrono::milliseconds(2));
```

- Time durations …

`Hours, minutes, seconds, milliseconds, microseconds, nanoseconds.`

- Why? → Monitoring across multiple threads.
- PLEASE: Don’t use sleep in the assignments.

C++20

- There is a lot of support introduced for managing calendars and time.
- There are some specific Clocks, associated with various times.
- And functionality for converting between clocks and for partitioning time.
- https://en.cppreference.com/w/cpp/chrono#Clocks