Howard E. Hinnant
2016-06-26

---

julian

Contents

Introduction

This is a Julian calendar in the style of date.h. Everthing is the same here as for date.h, except that the calendrical arithmetic implements a proleptic Julian calendar.

The `julian` calendar can interoperate with date.h and tz.h just by paying attention to the namespace. For example to convert the *civil* date `2016_y/jun/26` to a julian date, just do:

```
#include "julian.h"
#include <iostream>

int
main()
{
    using namespace date::literals;
    std::cout << julian::year_month_day{2016_y/jun/26} << '\n';
}
```

This outputs:

```
2016-06-13
```

And here is the reverse conversion:

```
#include "julian.h"
#include <iostream>

int
main()
{
    using namespace julian::literals;
    std::cout << date::year_month_day{2016_y/jun/13} << '\n';
}
```

Which outputs:

```
2016-06-26
```

You can even convert directly to the ISO-week-based calendar:

```
#include "iso_week.h"
#include "julian.h"
#include <iostream>
```

```
int
main()
{
    using namespace julian::literals;
    std::cout << iso_week::year_weeknum_weekday{2016_y/jun/13} << '\n';
}
```

Which outputs:

```
2016-W25-Sun
```

You can find the current local julian date and time with:

```
#include "julian.h"
#include "tz.h"
#include <iostream>

int
main()
{
    auto zt = date::make_zoned(date::current_zone(), std::chrono::system_clock::now());
    auto ld = date::floor<date::days>(zt.get_local_time());
    julian::year_month_day ymd{ld};
    auto time = date::make_time(zt.get_local_time() - ld);
    std::cout << ymd << ' ' << time << '\n';
}
```

Example output:

```
2016-06-13 18:38:30.049598
```