Tick Tock:
Maintaining Time

Jacob Pratt
2026–09–10
Montréal

RustConf
2026

Background

- `time` used to be part of Rust's standard library

Background

- `time` used to be part of Rust's standard library
- 0.1 series was unmaintained since 2016

Background

- `time` used to be part of Rust's standard library
- 0.1 series was unmaintained since 2016
- 0.2 series was a greenfield rewrite
  - released December 2019

Background

- `time` used to be part of Rust's standard library
- 0.1 series was unmaintained since 2016
- 0.2 series was a greenfield rewrite
  - released December 2019
  - Great learning experience!

Background

- `time` used to be part of Rust's standard library
- 0.1 series was unmaintained since 2016
- 0.2 series was a greenfield rewrite
  - released December 2019
  - Great learning experience!
  - API inspiration from other libraries in Rust, Java, C++, Python, JavaScript, etc.

Background

- `time` used to be part of Rust's standard library
- 0.1 series was unmaintained since 2016
- 0.2 series was a greenfield rewrite
  - released December 2019
  - Great learning experience!
  - API inspiration from other libraries in Rust, Java, C++, Python, JavaScript, etc.
  - cumulative deprecations, mistakes, and quirks required a breaking change

Background

- `time` used to be part of Rust's standard library
- 0.1 series was unmaintained since 2016
- 0.2 series was a greenfield rewrite
  - released December 2019
  - Great learning experience!
  - API inspiration from other libraries in Rust, Java, C++, Python, JavaScript, etc.
  - cumulative deprecations, mistakes, and quirks required a breaking change
  - parallel API surfaces: panicking and fallible

The 0.3 Series

- applied lessons from 0.2 in one breaking release

The 0.3 Series

- applied lessons from 0.2 in one breaking release
- over 50 releases and counting

The 0.3 Series

- applied lessons from 0.2 in one breaking release
- over 50 releases and counting
- stay up-to-date with the language

The 0.3 Series

- applied lessons from 0.2 in one breaking release
- over 50 releases and counting
- stay up-to-date with the language
- security is essential

The 0.3 Series

- applied lessons from 0.2 in one breaking release
- over 50 releases and counting
- stay up-to-date with the language
- security is essential
- ensure forward compatibility whenever possible

The 0.3 Series

- applied lessons from 0.2 in one breaking release
- over 50 releases and counting
- stay up-to-date with the language
- security is essential
- ensure forward compatibility whenever possible
- maintain a high bar for quality and correctness

The 0.3 Series

- applied lessons from 0.2 in one breaking release
- over 50 releases and counting
- stay up-to-date with the language
- security is essential
- ensure forward compatibility whenever possible
- maintain a high bar for quality and correctness
- API evolves, but your build never breaks

Types Overview

- Core

Types Overview

- Core
  - `Month`
  - `Weekday`

Types Overview

- Core
  - `Month`
  - `Weekday`
  - `SignedDuration`

Types Overview

- Core
  - Month
  - Timestamp
  - Weekday
  - SignedDuration

Types Overview

- Core
  - `Month`
  - `Timestamp`
  - `Weekday`
  - `Date`
  - `SignedDuration`
  - `Time`

Types Overview

- Core
  - `Month`
  - `Timestamp`
  - `PlainDateTime`
  - `Weekday`
  - `Date`
  - `SignedDuration`
  - `Time`

Types Overview

- Core
  - `Month`
  - `Timestamp`
  - `PlainDateTime`
  - `Weekday`
  - `Date`
  - `UtcOffset`
  - `SignedDuration`
  - `Time`

Types Overview

- Core
  - `Month`
  - `Timestamp`
  - `PlainDateTime`
  - `Weekday`
  - `Date`
  - `UtcOffset`
  - `SignedDuration`
  - `Time`
  - `OffsetDateTime`

Types Overview

- Core
  - `Month`
  - `Timestamp`
  - `PlainDateTime`
  - `UtcDateTime`
  - `Weekday`
  - `Date`
  - `UtcOffset`
  - `SignedDuration`
  - `Time`
  - `OffsetDateTime`

Types Overview

- Core
  - `Month`
  - `Timestamp`
  - `PlainDateTime`
  - `UtcDateTime`
  - `Weekday`
  - `Date`
  - `UtcOffset`
  - `SignedDuration`
  - `Time`
  - `OffsetDateTime`
- Iterator
  - `DateIter`
  - `MonthIter`
  - `WeekdayIter`

Types Overview

- Core
  - `Month`
  - `Weekday`
  - `SignedDuration`
  - `Timestamp`
  - `Date`
  - `Time`
  - `PlainDateTime`
  - `UtcOffset`
  - `OffsetDateTime`
  - `UtcDateTime`
- Iterator
  - `DateIter`
  - `MonthIter`
  - `WeekdayIter`
- Format description

Types Overview

- Core
  - `Month`
  - `Timestamp`
  - `PlainDateTime`
  - `UtcDateTime`
  - `Weekday`
  - `Date`
  - `UtcOffset`
  - `SignedDuration`
  - `Time`
  - `OffsetDateTime`
- Iterator
  - `DateIter`
  - `MonthIter`
  - `WeekdayIter`
- Format description
- Low-level types for formatting & parsing

Helper Macros

Helper Macros

```
Date                    date!(2026-09-10)
```

Helper Macros

Date	`date!(2026-09-10)`
Time	`time!(2:47:00 PM)`

Helper Macros

```
Date              date!(2026-09-10)
Time              time!(2:47:00 PM)
PlainDateTime     datetime!(2026-253 2:47 PM)
```

Helper Macros

```
Date              date!(2026-09-10)
Time              time!(2:47:00 PM)
PlainDateTime     datetime!(2026-253 2:47 PM)
UtcOffset         offset!(-04:00)
```

Helper Macros

```
Date              date!(2026-09-10)
Time              time!(2:47:00 PM)
PlainDateTime     datetime!(2026-253 2:47 PM)
UtcOffset         offset!(-04:00)
OffsetDateTime    datetime!(2026-W37-4 14:47 -04)
```

Helper Macros

```
Date              date!(2026-09-10)
Time              time!(2:47:00 PM)
PlainDateTime     datetime!(2026-253 2:47 PM)
UtcOffset         offset!(-04:00)
OffsetDateTime    datetime!(2026-W37-4 14:47 -04)
UtcDateTime       utc_datetime!(2026-09-10 18:47)
```

Helper Macros

```
Date              date!(2026-09-10)
Time              time!(2:47:00 PM)
PlainDateTime     datetime!(2026-253 2:47 PM)
UtcOffset         offset!(-04:00)
OffsetDateTime    datetime!(2026-W37-4 14:47 -04)
UtcDateTime       utc_datetime!(2026-09-10 18:47)
Timestamp         timestamp!(1789066020)
```

Helper Macros

Date	`date!(2026-09-10)`
Time	`time!(2:47:00 PM)`
PlainDateTime	`datetime!(2026-253 2:47 PM)`
UtcOffset	`offset!(-04:00)`
OffsetDateTime	`datetime!(2026-W37-4 14:47 -04)`
UtcDateTime	`utc_datetime!(2026-09-10 18:47)`
Timestamp	`timestamp!(1789066020)`
	`timestamp!(2026-09-10 18:47:00)`

Spin-Off Crates

Not everything belongs in `time`!

Spin-Off Crates

Not everything belongs in `time`!

`standback` — `std` polyfills for older `rustc` versions

Spin-Off Crates

Not everything belongs in `time`!

`standback`	`std` polyfills for older `rustc` versions
`num_threads`	number of threads in a running process

Spin-Off Crates

Not everything belongs in `time`!

`standback`	`std` polyfills for older `rustc` versions
`num_threads`	number of threads in a running process
`powerfmt`	helpers for implementing `Display`

Spin-Off Crates

Not everything belongs in `time`!

`standback`	`std` polyfills for older `rustc` versions
`num_threads`	number of threads in a running process
`powerfmt`	helpers for implementing `Display`
`num-conv`	integer conversions without `as` casts

Spin-Off Crates

Not everything belongs in `time`!

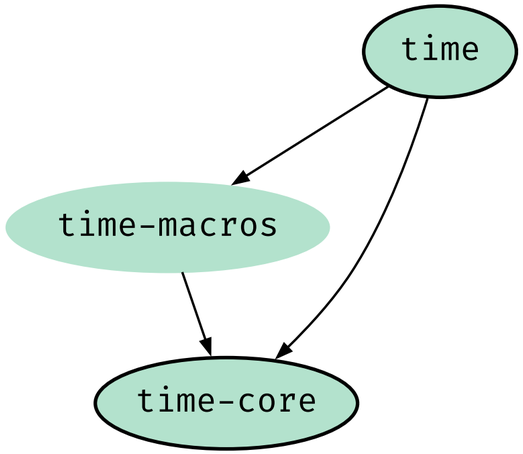
`standback`	`std` polyfills for older `rustc` versions
`num_threads`	number of threads in a running process
`powerfmt`	helpers for implementing `Display`
`num-conv`	integer conversions without `as` casts
`deranged`	ranged integers

Spin-Off Crates

Not everything belongs in `time`!

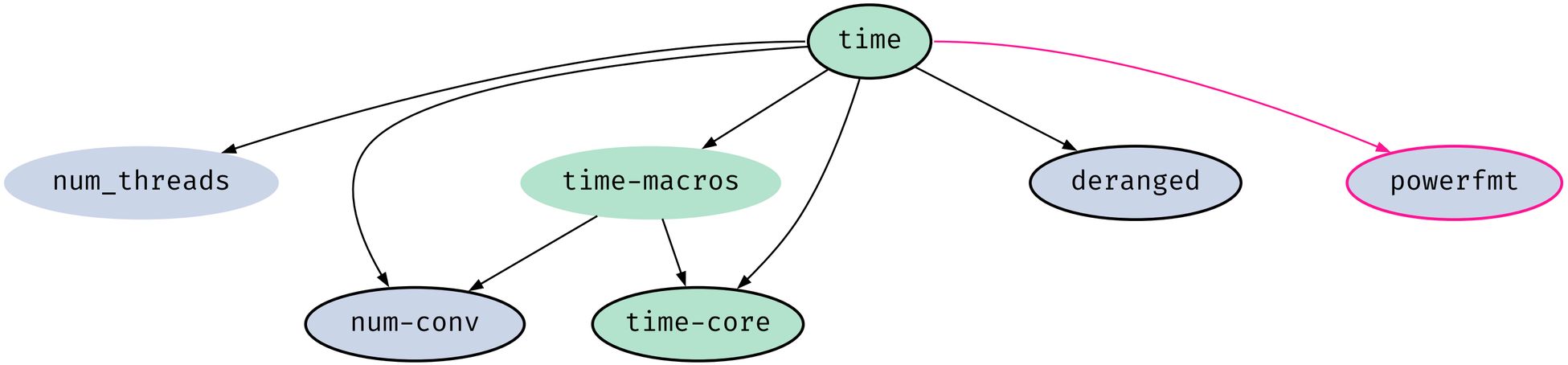
`standback`	`std` polyfills for older `rustc` versions
`num_threads`	number of threads in a running process
`powerfmt`	helpers for implementing `Display`
`num-conv`	integer conversions without `as` casts
`deranged`	ranged integers
`zoneinfo`	TZif file reading & writing
`zoneinfo-static`	precompiled time zone database

Dependency Tree



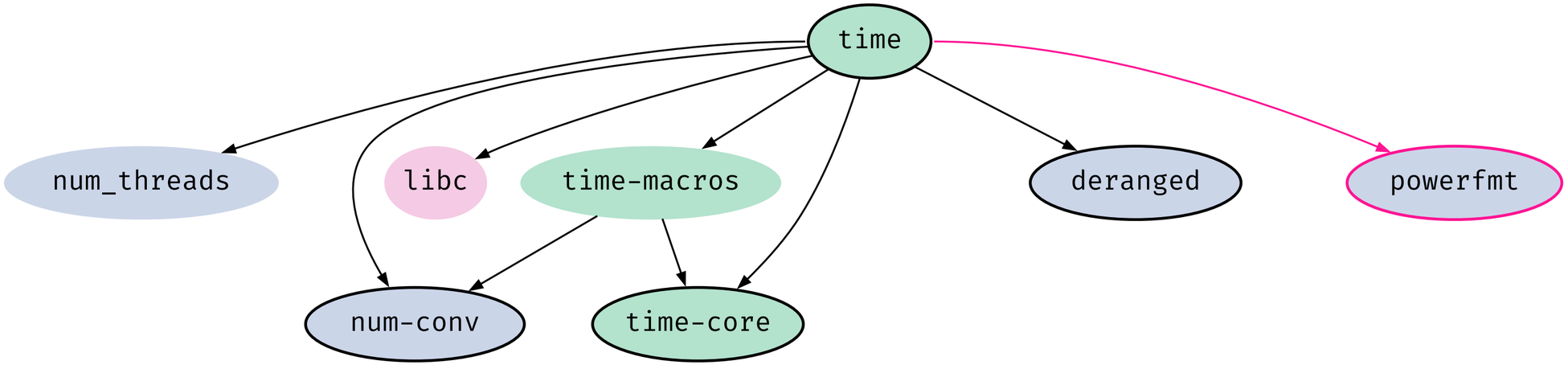
☐ mandatory ■ `time` workspace

Dependency Tree



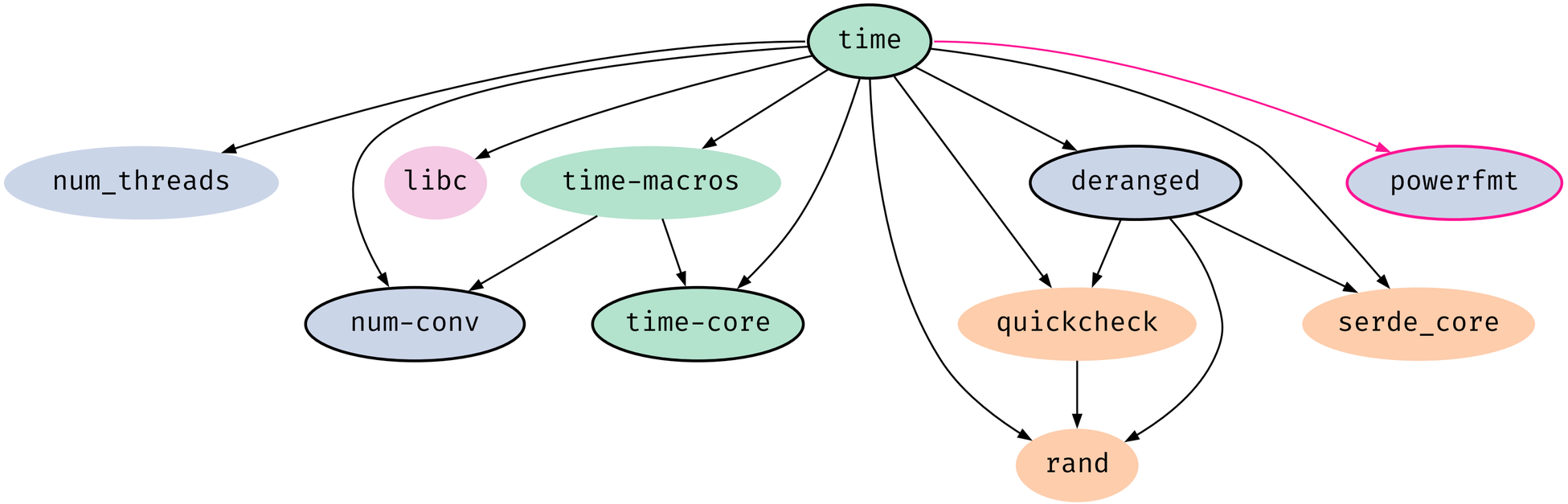
mandatory
deprecated
time workspace
spinoff crate

Dependency Tree



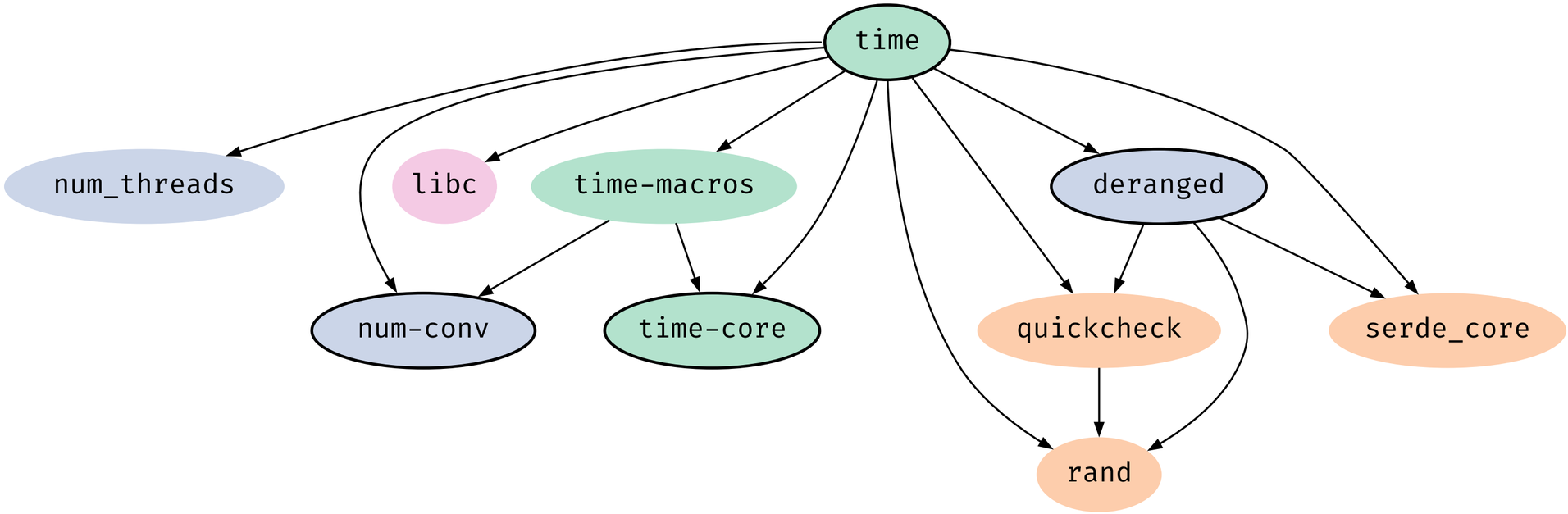
mandatory
deprecated
time workspace
local offset
spinoff crate

Dependency Tree



mandatory
deprecated
time workspace
local offset
spinoff crate
third party interop

Dependency Tree
(future possibility)



☐ mandatory

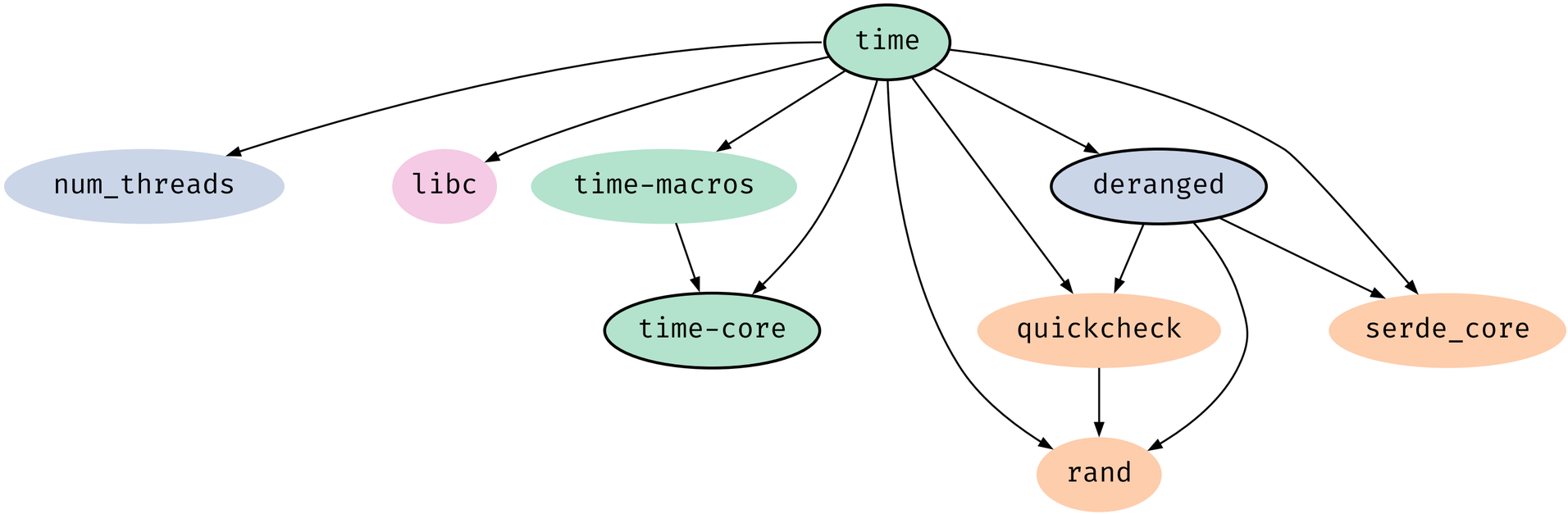
time workspace

local offset

spinoff crate

third party interop

Dependency Tree
(future possibility)



☐ mandatory

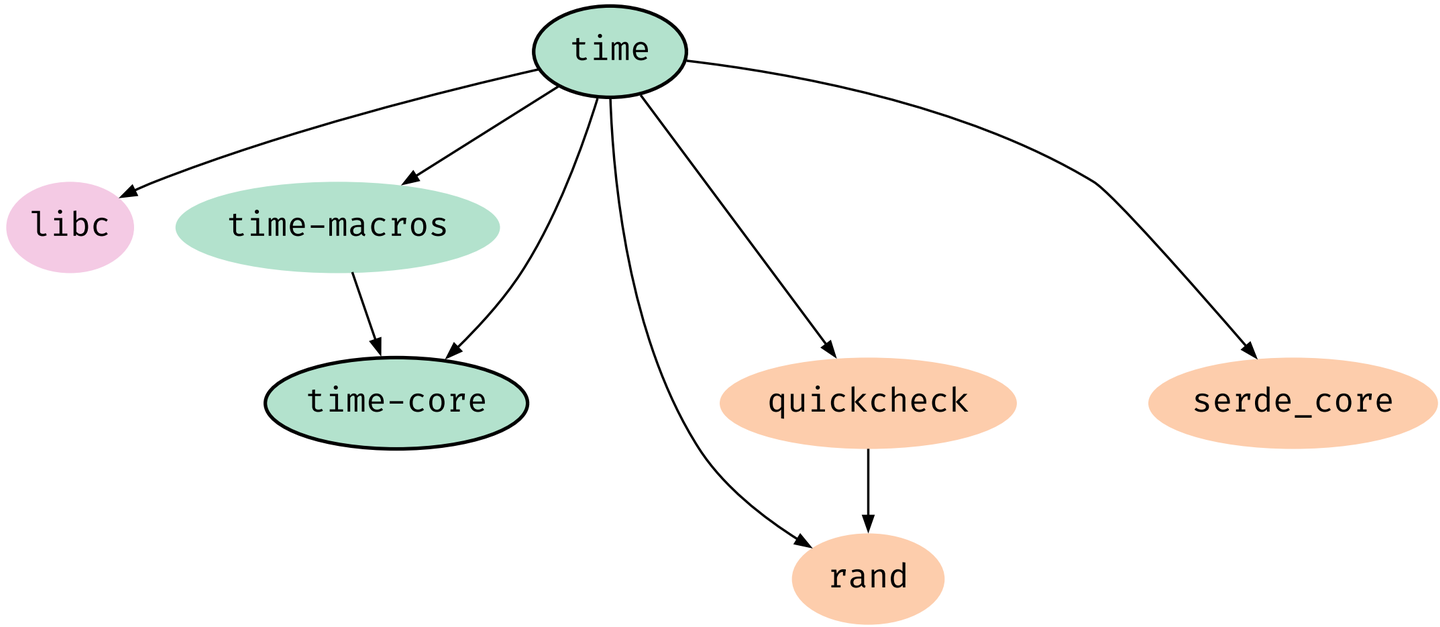
time workspace

local offset

spinoff crate

third party interop

Dependency Tree
(future possibility)



☐ mandatory

time workspace

local offset

third party interop

Formatting and Parsing

Format descriptions

Formatting and Parsing

Format descriptions

C	`%a %h %d`

Formatting and Parsing

Format descriptions

```
C               %a %h %d
time            [weekday repr:short] [month repr:short] [day]
```

Formatting and Parsing

Format descriptions

`C`	`%a %h %d`
`time`	`[weekday repr:short] [month repr:short] [day]`
`well-known`	ISO 8601 (international standard) RFC 2822 (email) RFC 3339 (similar to ISO 8601)

Formatting and Parsing

Format descriptions

`C`	`%a %h %d`
`time`	`[weekday repr:short] [month repr:short] [day]`
`well-known`	ISO 8601 (international standard) RFC 2822 (email) RFC 3339 (similar to ISO 8601)

Example of *completely valid* RFC 2822:

HT Sat,CR
(HTfoo012FOO!)(SOH VT SO DEL)(\NUL)(\SOH\HT\(\)\\DEL)\
(\
\VT)02
2021 03:04:05 GMT

Formatting and Parsing

Formatting and Parsing

It's a compiler?

Formatting and Parsing

It's a compiler?

```
let fmt_rt: Vec<_> = parse_owned::<2>(
    "[year]-[month]-[day]"
)?;
```

Formatting and Parsing

It's a compiler?

```
let fmt_rt: Vec<_> = parse_owned::<2>(
    "[year]-[month]-[day]"
)?;

const FMT_V2: &[BorrowedFormatItem<'_>] =
    format_description!(
        version = 2,
        "[year]-[month]-[day]"
    );
```

Formatting and Parsing

It's a compiler?

```rust
let fmt_rt: Vec<_> = parse_owned::<2>(
    "[year]-[month]-[day]"
)?;

const FMT_V2: &[BorrowedFormatItem<'_>] =
    format_description!(
        version = 2,
        "[year]-[month]-[day]"
    );

const FMT_V3: FormatDescriptionV3<'_> =
    format_description!(
        version = 3,
        "[year]-[month]-[day]"
    );
```

Formatting and Parsing

It's a compiler?

```
const {
    use ::time::format_description::__private::*;
    use ::time::format_description::modifier::*;
    FormatDescriptionV3Inner::BorrowedCompound(const {
        &[
            FormatDescriptionV3Inner::CalendarYearFullStandardRange(
                CalendarYearFullStandardRange::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::MonthNumerical(
                MonthNumerical::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::Day(Day::default())
        ]
    })
    .into_opaque()
}
```

Formatting and Parsing

It's a compiler?

```
const {
    use ::time::format_description::__private::*;
    use ::time::format_description::modifier::*;
    FormatDescriptionV3Inner::BorrowedCompound(const {
        &[
            FormatDescriptionV3Inner::CalendarYearFullStandardRange(
                CalendarYearFullStandardRange::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::MonthNumerical(
                MonthNumerical::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::Day(Day::default())
        ]
    })
    .into_opaque()
}
```

Formatting and Parsing

It's a compiler?

```
const {
    use ::time::format_description::__private::*;
    use ::time::format_description::modifier::*;
    FormatDescriptionV3Inner::BorrowedCompound(const {
        &[
            FormatDescriptionV3Inner::CalendarYearFullStandardRange(
                CalendarYearFullStandardRange::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::MonthNumerical(
                MonthNumerical::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::Day(Day::default())
        ]
    })
    .into_opaque()
}
```

Formatting and Parsing

It's a compiler?

```
const {
    use ::time::format_description::__private::*;
    use ::time::format_description::modifier::*;
    FormatDescriptionV3Inner::BorrowedCompound(const {
        &[
            FormatDescriptionV3Inner::CalendarYearFullStandardRange(
                CalendarYearFullStandardRange::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::MonthNumerical(
                MonthNumerical::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::Day(Day::default())
        ]
    })
    .into_opaque()
}
```

Formatting and Parsing

It's a compiler?

```
const {
    use ::time::format_description::__private::*;
    use ::time::format_description::modifier::*;
    FormatDescriptionV3Inner::BorrowedCompound(const {
        &[
            FormatDescriptionV3Inner::CalendarYearFullStandardRange(
                CalendarYearFullStandardRange::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::MonthNumerical(
                MonthNumerical::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::Day(Day::default())
        ]
    })
    .into_opaque()
}
```

Formatting and Parsing

It's a compiler?

```
const {
    use ::time::format_description::__private::*;
    use ::time::format_description::modifier::*;
    FormatDescriptionV3Inner::BorrowedCompound(const {
        &[
            FormatDescriptionV3Inner::CalendarYearFullStandardRange(
                CalendarYearFullStandardRange::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::MonthNumerical(
                MonthNumerical::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::Day(Day::default())
        ]
    })
    .into_opaque()
}
```

Formatting and Parsing

It's a compiler?

```
const {
    use ::time::format_description::__private::*;
    use ::time::format_description::modifier::*;
    FormatDescriptionV3Inner::BorrowedCompound(const {
        &[
            FormatDescriptionV3Inner::CalendarYearFullStandardRange(
                CalendarYearFullStandardRange::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::MonthNumerical(
                MonthNumerical::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::Day(Day::default())
        ]
    })
    .into_opaque()
}
```

Formatting and Parsing

It's a compiler?

```
const {
    use ::time::format_description::__private::*;
    use ::time::format_description::modifier::*;
    FormatDescriptionV3Inner::BorrowedCompound(const {
        &[
            FormatDescriptionV3Inner::CalendarYearFullStandardRange(
                CalendarYearFullStandardRange::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::MonthNumerical(
                MonthNumerical::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::Day(Day::default())
        ]
    })
    .into_opaque()
}
```

Formatting and Parsing

It's a compiler?

```
const {
    use ::time::format_description::__private::*;
    use ::time::format_description::modifier::*;
    FormatDescriptionV3Inner::BorrowedCompound(const {
        &[
            FormatDescriptionV3Inner::CalendarYearFullStandardRange(
                CalendarYearFullStandardRange::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::MonthNumerical(
                MonthNumerical::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::Day(Day::default())
        ]
    })
    .into_opaque()
}
```

Formatting and Parsing

It's a compiler?

```
const {
    use ::time::format_description::__private::*;
    use ::time::format_description::modifier::*;
    FormatDescriptionV3Inner::BorrowedCompound(const {
        &[
            FormatDescriptionV3Inner::CalendarYearFullStandardRange(
                CalendarYearFullStandardRange::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::MonthNumerical(
                MonthNumerical::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::Day(Day::default())
        ]
    })
    .into_opaque()
}
```

Formatting and Parsing

It's a compiler?

```
const {
    use ::time::format_description::__private::*;
    use ::time::format_description::modifier::*;
    FormatDescriptionV3Inner::BorrowedCompound(const {
        &[
            FormatDescriptionV3Inner::CalendarYearFullStandardRange(
                CalendarYearFullStandardRange::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::MonthNumerical(
                MonthNumerical::default()
            ),
            FormatDescriptionV3Inner::BorrowedLiteral("-"),
            FormatDescriptionV3Inner::Day(Day::default())
        ]
    })
    .into_opaque()
}
```

Pushing the Compiler

Pushing the Compiler

- Simulating `const impl Default`

```
impl T {
    pub const fn default() → Self { /* … */ }
}
```

Pushing the Compiler

- Simulating `const impl Default`

Pushing the Compiler

- Simulating `const impl Default`
- Eliminating magic constants

```
Second::per_t::<u32>(Week) // 604,800 as a constant
//                  ———   ————
//                named / anonymous
```

Pushing the Compiler

- Simulating `const impl Default`
- Eliminating magic constants

Pushing the Compiler

- Simulating `const impl Default`
- Eliminating magic constants
- Type-level assertions

```
Time::MIDNIGHT.format(&Iso8601::DATE) // compile error
```

Pushing the Compiler

- Simulating `const impl Default`
- Eliminating magic constants
- Type-level assertions

Pushing the Compiler

- Simulating `const impl Default`
- Eliminating magic constants
- Type-level assertions
- const generic ADTs

```
pub type EncodedConfig = u128; // explicitly unspecified

impl Config {
    pub const fn encode(self) → EncodedConfig { /* … */ }
    pub const fn decode(_: u128) → Self { /* … */ }
}
```

Finding Relevant Algorithms

Finding Relevant Algorithms

- Most problems have solutions! Research first and foremost.

Finding Relevant Algorithms

- Most problems have solutions! Research first and foremost.
- When gaps exist, design from first principles.

Finding Relevant Algorithms

- Most problems have solutions! Research first and foremost.
- When gaps exist, design from first principles.
  - day of year → month/day conversion
  - 2.35× throughput with new algorithm

Optimizing with Novel Calendrical Algorithms

Finding Relevant Algorithms

- Most problems have solutions! Research first and foremost.
- When gaps exist, design from first principles.

Finding Relevant Algorithms

- Most problems have solutions! Research first and foremost.
- When gaps exist, design from first principles.
- solved does not mean done
  - `is_leap_year` seems trivial, but keeps improving

```
year % 4 == 0 && (year % 100 =/= 0 || year % 400 == 0)
```

Optimizing with Novel Calendrical Algorithms

Finding Relevant Algorithms

- Most problems have solutions! Research first and foremost.
- When gaps exist, design from first principles.
- solved does not mean done
  - `is_leap_year` seems trivial, but keeps improving

```
year % 4 == 0 && (year % 100 =/= 0 || year % 400 == 0)
```

```
(year as i64)
    .unsigned_abs()
    .wrapping_mul(0×4000_0000_28F5_C28F)
    & 0×C000_000F_8000_000F
    ≤  0×F_8000_0000
```

Optimizing with Novel Calendrical Algorithms

Finding Relevant Algorithms

- Most problems have solutions! Research first and foremost.
- When gaps exist, design from first principles.
- solved does not mean done
  - `is_leap_year` seems trivial, but keeps improving

```
year % 4 == 0 && (year % 100 =/= 0 || year % 400 == 0)
```

```
(year as i64)
    .unsigned_abs()
    .wrapping_mul(0×4000_0000_28F5_C28F)
    & 0×C000_000F_8000_000F
    ≤  0×F_8000_0000
```

  - similar story with date ⟷ integer mapping

Optimizing with Novel Calendrical Algorithms

Finding Relevant Algorithms

- Most problems have solutions! Research first and foremost.
- When gaps exist, design from first principles.
- solved does not mean done
  - `is_leap_year` seems trivial, but keeps improving

```
year % 4 == 0 && (year % 100 =/= 0 || year % 400 == 0)
```

```
(year as i64)
    .unsigned_abs()
    .wrapping_mul(0×4000_0000_28F5_C28F)
    & 0×C000_000F_8000_000F
    ≤  0×F_8000_0000
```

  - similar story with date ⟷ integer mapping

- Resources: *Baum*, *Hinnant*, *Neri*, *Schneider*, *Drepper*, *Joffe*

Optimizing with Novel Calendrical Algorithms

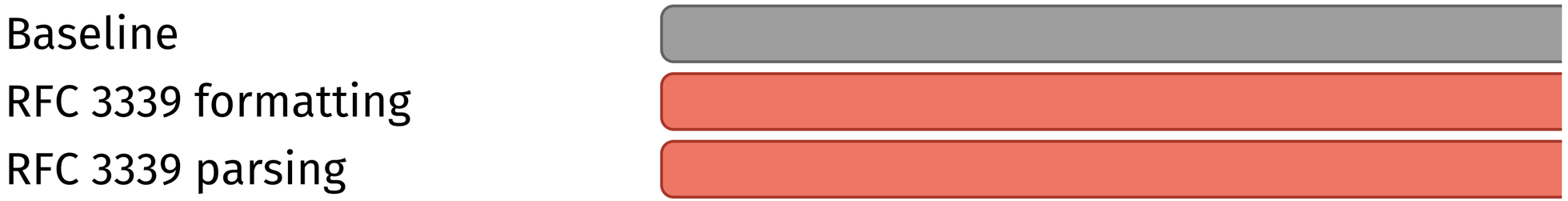
Performance

(elapsed time, smaller is better)

Baseline

Performance

(elapsed time, smaller is better)



Performance

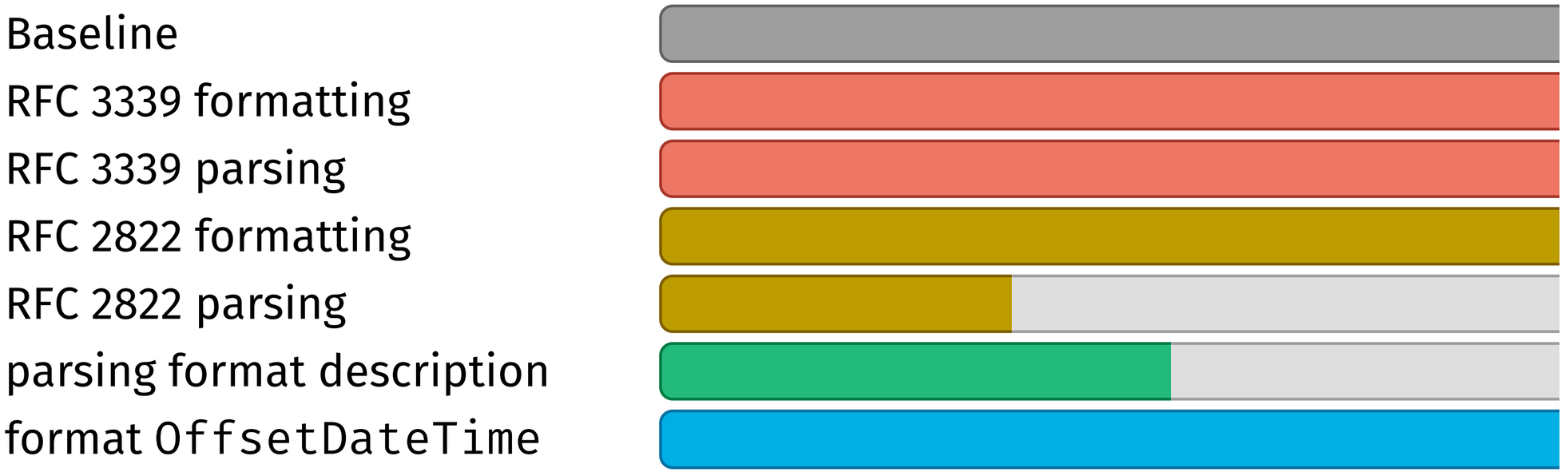
(elapsed time, smaller is better)

Performance

(elapsed time, smaller is better)

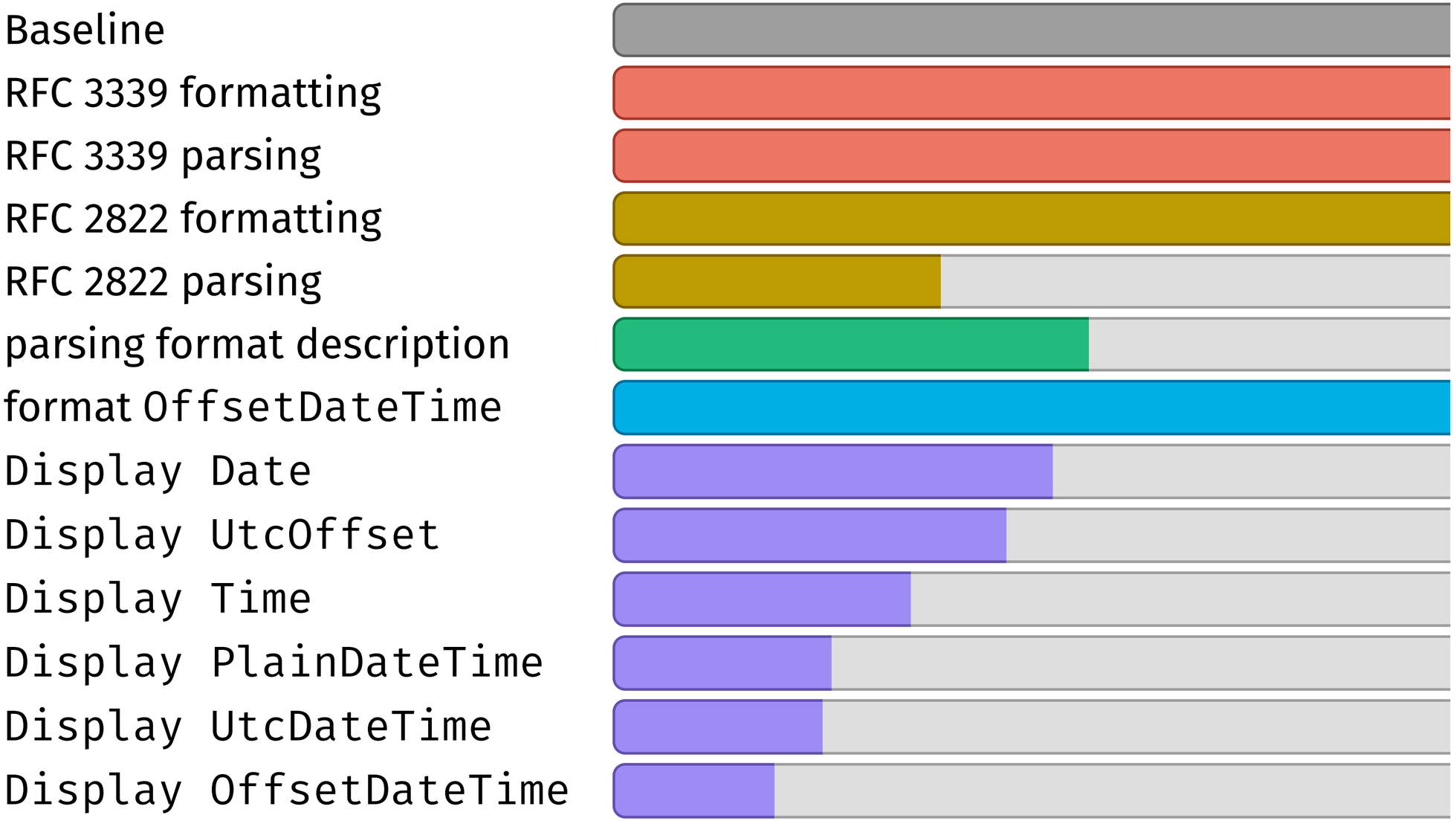
Performance

(elapsed time, smaller is better)



Performance

(elapsed time, smaller is better)



Performance

```rust
#[repr(C, align(8))]
#[derive(Clone, Copy)]
struct Digits {
    _padding: MaybeUninit<[u8; 7]>,
    digit_1: u8,
    digits_2_thru_9: [u8; 8],
}

let [
    digit_1,
    digits_2_and_3, // b'static str
    // …
] = subsecond_from_nanos(n);

let buf = Digits {
    _padding: MaybeUninit::uninit(),
    digit_1: digit_1.as_bytes()[0],
    digits_2_thru_9: [
        digits_2_and_3.as_bytes()[0],
        digits_2_and_3.as_bytes()[1],
        // …
    ],
};
```

```rust
let bitmask =
    u64::from_le_bytes(buf.digits_2_thru_9)
    ^ u64::from_le_bytes([b'0'; 8]);

let digits_to_truncate =
    bitmask.leading_zeros() / 8;

let len = 9 - digits_to_truncate as usize;

unsafe {
    StackStr::new(
        *(&raw const buf)
            .byte_add(
                offset_of!(Digits, digit_1)
            ).cast(),
        len,
    )
}
```

Performance

```rust
#[repr(C, align(8))]
#[derive(Clone, Copy)]
struct Digits {
    _padding: MaybeUninit<[u8; 7]>,
    digit_1: u8,
    digits_2_thru_9: [u8; 8],
}
```

Performance

```rust
#[repr(C, align(8))]
#[derive(Clone, Copy)]
struct Digits {
    _padding: MaybeUninit<[u8; 7]>,
    digit_1: u8,
    digits_2_thru_9: [u8; 8],
}

let [
    digit_1,
    digits_2_and_3, // &'static str
    // …
] = subsecond_from_nanos(n);
```

Performance

```
#[repr(C, align(8))]
#[derive(Clone, Copy)]
struct Digits {
    _padding: MaybeUninit<[u8; 7]>,
    digit_1: u8,
    digits_2_thru_9: [u8; 8],
}

let [
    digit_1,
    digits_2_and_3, // &'static str
    // …
] = subsecond_from_nanos(n);

let buf = Digits {
    _padding: MaybeUninit::uninit(),
    digit_1: digit_1.as_bytes()[0],
    digits_2_thru_9: [
        digits_2_and_3.as_bytes()[0],
        digits_2_and_3.as_bytes()[1],
        // …
    ],
};
```

Performance

```rust
#[repr(C, align(8))]
#[derive(Clone, Copy)]
struct Digits {
    _padding: MaybeUninit<[u8; 7]>,
    digit_1: u8,
    digits_2_thru_9: [u8; 8],
}

let [
    digit_1,
    digits_2_and_3, // &'static str
    // …
] = subsecond_from_nanos(n);

let buf = Digits {
    _padding: MaybeUninit::uninit(),
    digit_1: digit_1.as_bytes()[0],
    digits_2_thru_9: [
        digits_2_and_3.as_bytes()[0],
        digits_2_and_3.as_bytes()[1],
        // …
    ],
};
```

```rust
let bitmask =
    u64::from_le_bytes(buf.digits_2_thru_9)
    ^ u64::from_le_bytes([b'0'; 8]);

let digits_to_truncate =
    bitmask.leading_zeros() / 8;

let len = 9 - digits_to_truncate as usize;

unsafe {
    StackStr::new(
        *(&raw const buf)
            .byte_add(
                offset_of!(Digits, digit_1)
            ).cast(),
        len,
    )
}
```

Performance

```rust
#[repr(C, align(8))]
#[derive(Clone, Copy)]
struct Digits {
    _padding: MaybeUninit<[u8; 7]>,
    digit_1: u8,
    digits_2_thru_9: [u8; 8],
}

let [
    digit_1,
    digits_2_and_3, // &'static str
    // …
] = subsecond_from_nanos(n);

let buf = Digits {
    _padding: MaybeUninit::uninit(),
    digit_1: digit_1.as_bytes()[0],
    digits_2_thru_9: [
        digits_2_and_3.as_bytes()[0],
        digits_2_and_3.as_bytes()[1],
        // …
    ],
};
```

```rust
let bitmask =
    u64::from_le_bytes(buf.digits_2_thru_9)
    ^ u64::from_le_bytes([b'0'; 8]);

let digits_to_truncate =
    bitmask.leading_zeros() / 8;

let len = 9 - digits_to_truncate as usize;

unsafe {
    StackStr::new(
        *(&raw const buf)
            .byte_add(
                offset_of!(Digits, digit_1)
            ).cast(),
        len,
    )
}
```

Performance

```rust
#[repr(C, align(8))]
#[derive(Clone, Copy)]
struct Digits {
    _padding: MaybeUninit<[u8; 7]>,
    digit_1: u8,
    digits_2_thru_9: [u8; 8],
}

let [
    digit_1,
    digits_2_and_3, // &'static str
    // …
] = subsecond_from_nanos(n);

let buf = Digits {
    _padding: MaybeUninit::uninit(),
    digit_1: digit_1.as_bytes()[0],
    digits_2_thru_9: [
        digits_2_and_3.as_bytes()[0],
        digits_2_and_3.as_bytes()[1],
        // …
    ],
};
```

```rust
let bitmask =
    u64::from_le_bytes(buf.digits_2_thru_9)
    ^ u64::from_le_bytes([b'0'; 8]);

let digits_to_truncate =
    bitmask.leading_zeros() / 8;

let len = 9 - digits_to_truncate as usize;

unsafe {
    StackStr::new(
        *(&raw const buf)
            .byte_add(
                offset_of!(Digits, digit_1)
            ).cast(),
        len,
    )
}
```

Performance

```rust
#[repr(C, align(8))]
#[derive(Clone, Copy)]
struct Digits {
    _padding: MaybeUninit<[u8; 7]>,
    digit_1: u8,
    digits_2_thru_9: [u8; 8],
}

let [
    digit_1,
    digits_2_and_3, // &'static str
    // …
] = subsecond_from_nanos(n);

let buf = Digits {
    _padding: MaybeUninit::uninit(),
    digit_1: digit_1.as_bytes()[0],
    digits_2_thru_9: [
        digits_2_and_3.as_bytes()[0],
        digits_2_and_3.as_bytes()[1],
        // …
    ],
};
```

```rust
let bitmask =
    u64::from_le_bytes(buf.digits_2_thru_9)
    ^ u64::from_le_bytes([b'0'; 8]);

let digits_to_truncate =
    bitmask.leading_zeros() / 8;

let len = 9 - digits_to_truncate as usize;

unsafe {
    StackStr::new(
        *(&raw const buf)
            .byte_add(
                offset_of!(Digits, digit_1)
            ).cast(),
        len,
    )
}
```

Niche Value Optimization

```
size_of::<Date>() == size_of::<Option<Date>>()
```

Niche Value Optimization

```
size_of::<Date>() == size_of::<Option<Date>>()

size_of::<Time>() == size_of::<Option<Time>>()
```

Niche Value Optimization

```
size_of::<Date>() == size_of::<Option<Date>>()

size_of::<Time>() == size_of::<Option<Time>>()

struct Time {
    hour: u8,
    minute: u8,
    second: u8,

    nanosecond: u32,
}
```

Niche Value Optimization

```
size_of::<Date>() == size_of::<Option<Date>>()

size_of::<Time>() == size_of::<Option<Time>>()

struct Time {
    hour: u8,
    minute: u8,
    second: u8,
    _padding: Padding,
    nanosecond: u32,
}

#[repr(u8)]
enum Padding {
    Optimization,
}
```

CI/CD

CI/CD

- CI is needed to avoid breakage

CI/CD

- CI is needed to avoid breakage
- split one runner into multiple statically-known shards

CI/CD

- CI is needed to avoid breakage
- split one runner into multiple statically-known shards
  - became unbalanced with more feature flags

CI/CD

- CI is needed to avoid breakage
- split one runner into multiple statically-known shards
  - became unbalanced with more feature flags
- change to dynamic sharding

CI/CD

- CI is needed to avoid breakage
- split one runner into multiple statically-known shards
  - became unbalanced with more feature flags
- change to dynamic sharding
  - CI dropped from 20 minutes to 6½

CI/CD

- CI is needed to avoid breakage
- split one runner into multiple statically-known shards
  - became unbalanced with more feature flags
- change to dynamic sharding
  - CI dropped from 20 minutes to 6½
- miscellaneous improvements

CI/CD

- CI is needed to avoid breakage
- split one runner into multiple statically-known shards
  - became unbalanced with more feature flags
- change to dynamic sharding
  - CI dropped from 20 minutes to 6½
- miscellaneous improvements
  - limit of 20 concurrent runners raised to 30

CI/CD

- CI is needed to avoid breakage
- split one runner into multiple statically-known shards
  - became unbalanced with more feature flags
- change to dynamic sharding
  - CI dropped from 20 minutes to 6½
- miscellaneous improvements
  - limit of 20 concurrent runners raised to 30
  - native parallel jobs within a single runner

CI/CD

- CI is needed to avoid breakage
- split one runner into multiple statically-known shards
  - became unbalanced with more feature flags
- change to dynamic sharding
  - CI dropped from 20 minutes to 6½
- miscellaneous improvements
  - limit of 20 concurrent runners raised to 30
  - native parallel jobs within a single runner
  - time reduced further to 4½ minutes

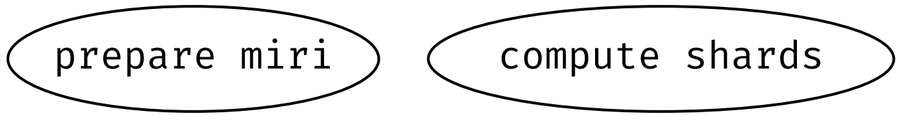
CI/CD

- CI is needed to avoid breakage
- split one runner into multiple statically-known shards
  - became unbalanced with more feature flags
- change to dynamic sharding
  - CI dropped from 20 minutes to 6½
- miscellaneous improvements
  - limit of 20 concurrent runners raised to 30
  - native parallel jobs within a single runner
  - time reduced further to 4½ minutes
- sharding publishing reduced time by a full hour

CI/CD

CI/CD

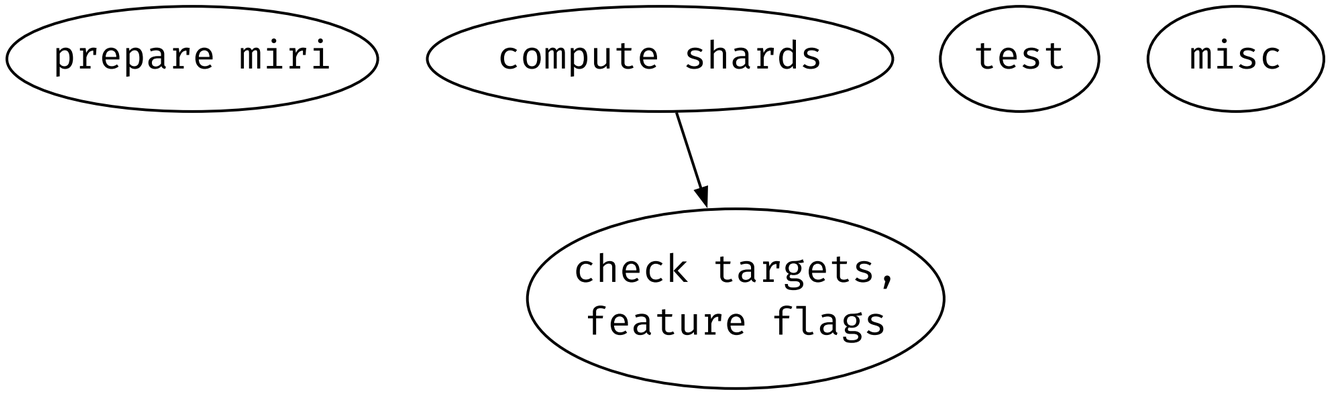
CI/CD



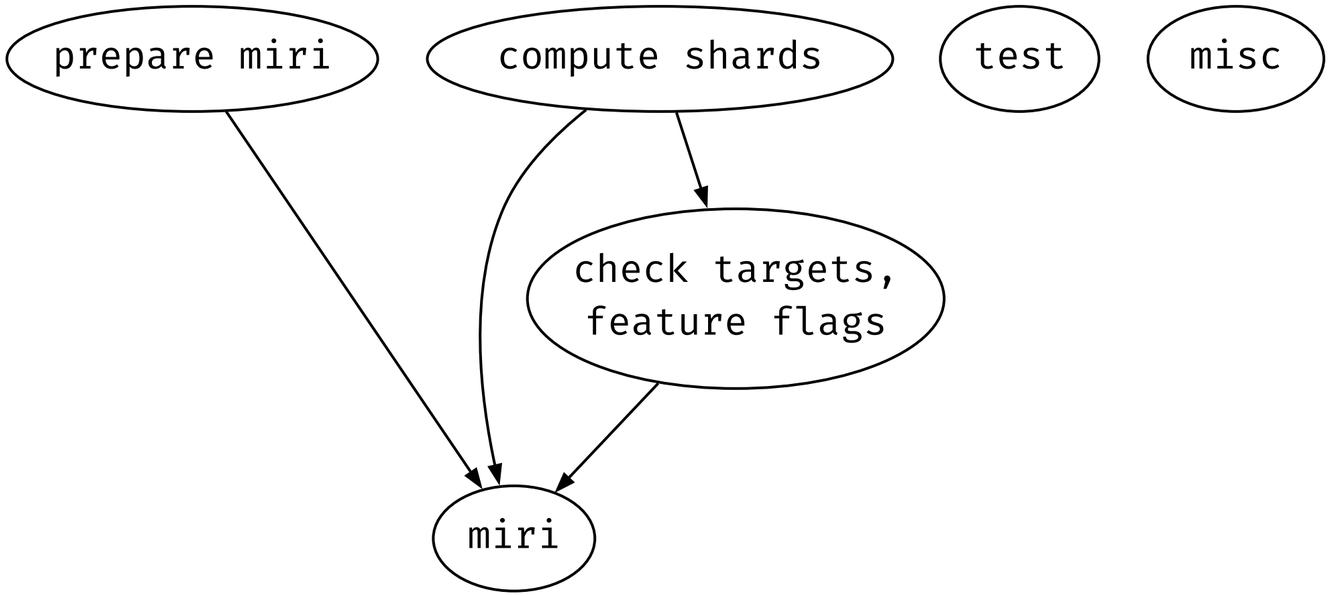
CI/CD

CI/CD

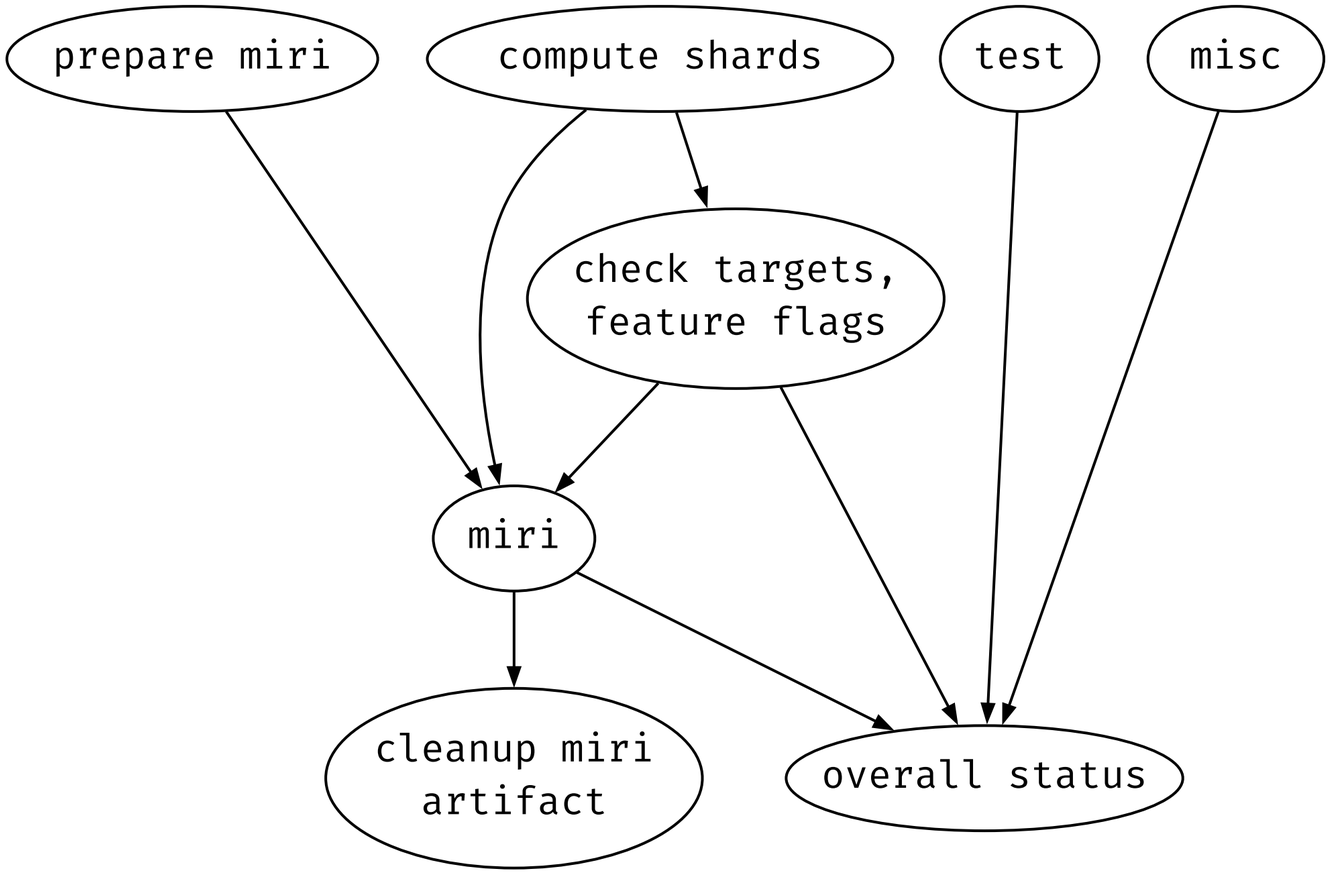
CI/CD


prepare miri
compute shards
test
misc
check targets,
feature flags

CI/CD


prepare miri
compute shards
test
misc
check targets,
feature flags
miri

CI/CD


prepare miri
compute shards
test
misc
check targets,
feature flags
miri
cleanup miri
artifact
overall status

It's bug-free...right?

It's bug-free...right?

- CI can only catch what is tested.

It's bug-free…right?

- CI can only catch what is tested.
  - `SignedDuration` division was quietly wrong for years because of `std`.

It's bug-free…right?

- CI can only catch what is tested.
  - `SignedDuration` division was quietly wrong for years because of `std`.
- The largest breakages were from *compiler* bugs or *process* failures.

It's bug-free…right?

- CI can only catch what is tested.
  - `SignedDuration` division was quietly wrong for years because of `std`.
- The largest breakages were from *compiler* bugs or *process* failures.
- Security reports are the top priority.

It's bug-free…right?

- CI can only catch what is tested.
  - `SignedDuration` division was quietly wrong for years because of `std`.
- The largest breakages were from *compiler* bugs or *process* failures.
- Security reports are the top priority.
- Undefined behavior is inherently difficult to detect.

Language Changes

- Experience maintaining `time` feeds back into Rust itself

Language Changes

- Experience maintaining `time` feeds back into Rust itself
- Multiple RFCs

Language Changes

- Experience maintaining `time` feeds back into Rust itself
- Multiple RFCs
  - `#[derive(Default)]` on enums

Language Changes

- Experience maintaining `time` feeds back into Rust itself
- Multiple RFCs
  - `#[derive(Default)]` on enums
  - `impl` and `mut` restrictions

Language Changes

- Experience maintaining `time` feeds back into Rust itself
- Multiple RFCs
  - `#[derive(Default)]` on enums
  - `impl` and `mut` restrictions
    - ***Please*** test this! We need more feedback.

Language Changes

- Experience maintaining `time` feeds back into Rust itself
- Multiple RFCs
  - `#[derive(Default)]` on enums
  - `impl` and `mut` restrictions
    - ***Please*** test this! We need more feedback.
  - `unsafe` fields

Language Changes

- Experience maintaining `time` feeds back into Rust itself
- Multiple RFCs
  - `#[derive(Default)]` on enums
  - `impl` and `mut` restrictions
    - ***Please*** test this! We need more feedback.
  - `unsafe` fields
  - default field values

```
struct Foo {
    field: Type = value,
}
let _ = Foo { .. };
```

Language Changes

- Experience maintaining `time` feeds back into Rust itself
- Multiple RFCs
  - `#[derive(Default)]` on enums
  - `impl` and `mut` restrictions
    - ***Please*** test this! We need more feedback.
  - `unsafe` fields
  - default field values

    ```
    struct Foo {
        field: Type = value,
    }
    let _ = Foo { .. };
    ```

  - `&&`, `||`, `!` in `cfg` — needs T-lang decision

Near Future

Near Future

- Thorough performance audit

Near Future

- Thorough performance audit
- Improved documentation

Near Future

- Thorough performance audit
- Improved documentation
- `TimeSpan` type

Near Future

- Thorough performance audit
- Improved documentation
- `TimeSpan` type
- `Interval` type

Near Future

- Thorough performance audit
- Improved documentation
- `TimeSpan` type
- `Interval` type
- `ZonedDateTime`, including ahead-of-time codegen

Near Future

- Thorough performance audit
- Improved documentation
- `TimeSpan` type
- `Interval` type
- `ZonedDateTime`, including ahead-of-time codegen
  - needs funding or it'll be licensed under AGPL

Near Future

- Thorough performance audit
- Improved documentation
- `TimeSpan` type
- `Interval` type
- `ZonedDateTime`, including ahead-of-time codegen
  - needs funding or it'll be licensed under AGPL
- Specification for format descriptions

Near Future

- Thorough performance audit
- Improved documentation
- `TimeSpan` type
- `Interval` type
- `ZonedDateTime`, including ahead-of-time codegen
  - needs funding or it'll be licensed under AGPL
- Specification for format descriptions

1.0 release

Tick Tock: Maintaining Time

Jacob Pratt

@jhpratt

@jhpratt@mastodon.social

jhpratt.dev

sponsor.jhpratt.dev

2026–09–10
Montréal

RustConf
2026