Validation and Conversion of Physical Units at Compile Time

Leveraging the power of the C++11 type system in your simulation model

32C3

2015-12-29

About me

name/alias kamikaze / Dominic Fandrey

About me

name/alias kamikaze / Dominic Fandrey

objectives researcher at -censored- in Karlsruhe

About me

name/alias	kamikaze / Dominic Fandrey
objectives	researcher at -censored- in Karlsruhe FreeBSD hacker

About me

name/alias	kamikaze / Dominic Fandrey
objectives	researcher at -censored- in Karlsruhe FreeBSD hacker unicyclist

About me

name/alias	kamikaze / Dominic Fandrey
objectives	researcher at -censored- in Karlsruhe
	FreeBSD hacker
	unicyclist
	FSAE enthusiast (i.e. I used to build race cars)

Bio (chronological)

- Europäische Teams bei internationalem Hacking-Wettbewerb vorne

  `http://www.heise.de/security/meldung/Europaeische-Teams-bei-internationalem-Hacking-Wettbewerb-vorne-158175.html`
- BSDA:OBJ - Real World OO for Shell-Scripting

  `http://sourceforge.net/p/bsdadminscripts/bsda2/ci/default/tree/src/bsda_obj.md`
- High Speed Karlsruhe

  `http://highspeed-karlsruhe.de/`
- hsk-libs: XC878 µC library code

  `http://sourceforge.net/p/hsk/libs/ci/default/tree/`
- Weird stuff about shell and AWK scripting

  `http://angryswarm.blogspot.de/`

Why?

- Unit mismatch is expensive and dangerous

Why?

- Unit mismatch is expensive and dangerous
  - See Mars Climate Orbitter loss (worth $125,000,000)

    `https://www.youtube.com/watch?v=ZD4jPnk49PE`

Why?

- Unit mismatch is expensive and dangerous
  - See Mars Climate Orbitter loss (worth $125,000,000)

    `https://www.youtube.com/watch?v=ZD4jPnk49PE`
- Also, unit conversions are annoying

What Do We Want?

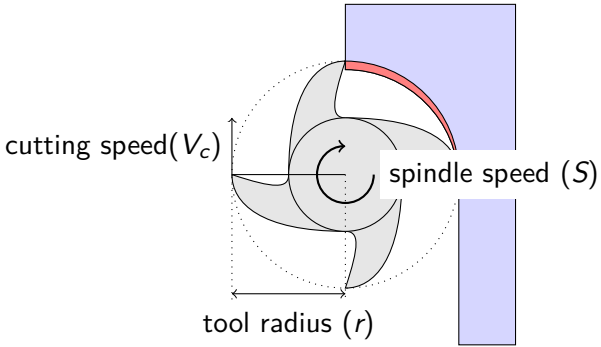
```
5000._m == 5._km; // true

typedef decltype(1._m/1._s) mps;
mps velocity = 100._km/1._h; // 27.778 m/s

double sin(rad angle) {
     // map angle to ±180° for max precission
     angle = ((angle + 180._deg) % 360._deg) - 180._deg;
     ...
}

42._min + 23._s; // compiles
42._min + 23._m; // doesn't
```

Example: Top view cross section of a milling process



Example: Problems from the CAD/CAM Domain

- The *diameter* is commonly provided in millimetre, while the *cutting speed* is provided in $^{\text{metre}}/_{\text{minute}}$.

Example: Problems from the CAD/CAM Domain

- The *diameter* is commonly provided in millimetre, while the *cutting speed* is provided in $^{\text{metre}}/_{\text{minute}}$.
- The *spindle speed* is an angular velocity, expected to be represented in $^{\text{revolution}}/_{\text{minute}}$ instead of $^{\text{radian}}/_{\text{minute}}$.

Example: Problems from the CAD/CAM Domain

- The *diameter* is commonly provided in millimetre, while the *cutting speed* is provided in $^{\text{metre}}/_{\text{minute}}$.
- The *spindle speed* is an angular velocity, expected to be represented in $^{\text{revolution}}/_{\text{minute}}$ instead of $^{\text{radian}}/_{\text{minute}}$.
- OR you get imperial units

Example: Weakly typed code with explicit conversions

```
/**
 * Update the angular velocity of the spindle, according to the desired
 * cutting speed and tool radius.
 *
 * @param speed
 *   The cutting speed in m/min
 * @param radius
 *   The tool radius in mm
 */
void Spindle::setCuttingSpeed(double const speed, double const radius) {
    this->av = speed * 1000 / (2 * PI * radius);
}
```

Example: Strongly typed code with implicit conversions

```
/**
 * Update the angular velocity of the spindle, according to the desired
 * cutting speed and tool radius.
 *
 * @param speed,radius
 *     The cutting speed and tool radius
 *
 *
 */
void Spindle::setCuttingSpeed(mmpmin const speed, mm const radius) {
    this->av = speed / radius * 1._rad;
}
```

So, Why C++11?

- `constexpr`

So, Why C++11?

- `constexpr`
- `static_assert()`

So, Why C++11?

- `constexpr`
- `static_assert()`
- *user-defined literals*

So, Why C++11?

- `constexpr`
- `static_assert()`
- *user-defined literals*
- *template aliases*

So, Why C++11?

- `constexpr`
- `static_assert()`
- *user-defined literals*
- *template aliases*
- *variadic templates*

So, Why C++11?

- `constexpr`
- `static_assert()`
- *user-defined literals*
- *template aliases*
- *variadic templates*
- `auto`, `decltype`

Design: A system of units is defined as ...

$$n, m \in \mathbb{Z} \tag{1}$$

$$\mathit{bases} \in \{\mathrm{m}^{n_0}, \mathrm{s}^{n_1}, \mathrm{rad}^{n_2}\} \tag{2}$$

$$\mathit{factor} \in \frac{\mathbb{N}}{\mathbb{N}}, \text{ the positive subset of } \mathbb{Q} \tag{3}$$

$$\mathit{constants} \in \{\pi^{m_0}\} \tag{4}$$

$$\mathit{system} \in \{\mathit{bases}, \mathit{factor}, \mathit{constants}\} \tag{5}$$

Design: Base Units

$$\mathit{metre} = \left\{ \left\{ \mathrm{m}^1, \mathrm{s}^0, \mathrm{rad}^0 \right\}, \frac{1}{1}, \left\{ \pi^0 \right\} \right\} \tag{6}$$

$$\mathit{second} = \left\{ \left\{ \mathrm{m}^0, \mathrm{s}^1, \mathrm{rad}^0 \right\}, \frac{1}{1}, \left\{ \pi^0 \right\} \right\} \tag{7}$$

$$\mathit{rad} = \left\{ \left\{ \mathrm{m}^0, \mathrm{s}^0, \mathrm{rad}^1 \right\}, \frac{1}{1}, \left\{ \pi^0 \right\} \right\} \tag{8}$$

Design: Derived Units

$$\textit{millimetre} = \left\{ \left\{ \mathrm{m}^1, \mathrm{s}^0, \mathrm{rad}^0 \right\}, \frac{1}{1000}, \left\{ \pi^0 \right\} \right\} \tag{9}$$

$$\textit{minute} = \left\{ \left\{ \mathrm{m}^0, \mathrm{s}^1, \mathrm{rad}^0 \right\}, \frac{60}{1}, \left\{ \pi^0 \right\} \right\} \tag{10}$$

$$\textit{revolution} = \left\{ \left\{ \mathrm{m}^0, \mathrm{s}^0, \mathrm{rad}^1 \right\}, \frac{2}{1}, \left\{ \pi^1 \right\} \right\} \tag{11}$$

Design: Convert Units

$$\text{expr: } x_0\,\text{rad} + x_1\,\text{revolution} \tag{12}$$

(17)

Design: Convert Units

$$\text{expr: } x_0\,\text{rad} + x_1\,\text{revolution} \tag{12}$$

$$\Longrightarrow \text{expr: } (x_0 + x_1 \cdot c)\,\text{rad} \tag{13}$$

(17)

Design: Convert Units

$$\text{expr: } x_0\,\text{rad} + x_1\,\text{revolution} \tag{12}$$

$$\implies \text{expr: } (x_0 + x_1 \cdot c)\,\text{rad} \tag{13}$$

$$c = \left|\frac{\text{revolution}}{\text{rad}}\right| \tag{14}$$

(17)

Design: Convert Units

$$\text{expr: } x_0\,\mathrm{rad} + x_1\,\mathrm{revolution} \tag{12}$$

$$\Longrightarrow \text{expr: } (x_0 + x_1 \cdot c)\,\mathrm{rad} \tag{13}$$

$$c = \left| \frac{\mathrm{revolution}}{\mathrm{rad}} \right| \tag{14}$$

$$\Longrightarrow c = \left| \frac{\left\{ \left\{ \mathrm{m}^0, \mathrm{s}^0, \mathrm{rad}^1 \right\}, \frac{2}{1}, \left\{ \pi^1 \right\} \right\}}{\left\{ \left\{ \mathrm{m}^0, \mathrm{s}^0, \mathrm{rad}^1 \right\}, \frac{1}{1}, \left\{ \pi^0 \right\} \right\}} \right| \tag{15}$$

(17)

Design: Convert Units

$$\text{expr: } x_0 \,\mathrm{rad} + x_1 \,\mathrm{revolution} \tag{12}$$

$$\Longrightarrow \text{expr: } (x_0 + x_1 \cdot c)\,\mathrm{rad} \tag{13}$$

$$c = \left| \frac{\mathrm{revolution}}{\mathrm{rad}} \right| \tag{14}$$

$$\Longrightarrow c = \left| \frac{\left\{\left\{\mathrm{m}^0, \mathrm{s}^0, \mathrm{rad}^1\right\}, \frac{2}{1}, \left\{\pi^1\right\}\right\}}{\left\{\left\{\mathrm{m}^0, \mathrm{s}^0, \mathrm{rad}^1\right\}, \frac{1}{1}, \left\{\pi^0\right\}\right\}} \right| \tag{15}$$

$$\Longrightarrow c = \frac{\frac{2}{1} \cdot \pi^1}{\frac{1}{1} \cdot \pi^0} = 2\pi \tag{16}$$

$$\tag{17}$$

Design: Convert Units

$$\text{expr: } x_0 \,\mathrm{rad} + x_1 \,\text{revolution} \tag{12}$$

$$\implies \text{expr: } (x_0 + x_1 \cdot c)\,\mathrm{rad} \tag{13}$$

$$c = \left| \frac{\text{revolution}}{\mathrm{rad}} \right| \tag{14}$$

$$\implies c = \left| \frac{\left\{ \left\{ \mathrm{m}^0, \mathrm{s}^0, \mathrm{rad}^1 \right\}, \frac{2}{1}, \left\{ \pi^1 \right\} \right\}}{\left\{ \left\{ \mathrm{m}^0, \mathrm{s}^0, \mathrm{rad}^1 \right\}, \frac{1}{1}, \left\{ \pi^0 \right\} \right\}} \right| \tag{15}$$

$$\implies c = \frac{\frac{2}{1} \cdot \pi^1}{\frac{1}{1} \cdot \pi^0} = 2\pi \tag{16}$$

$$\implies \text{expr: } (x_0 + x_1 \cdot 2\pi)\,\mathrm{rad} \tag{17}$$

Design: Compose Units (1/2)

$$\text{expr: } \frac{10000\,\text{revolution}}{1\,\text{minute} \cdot 2\,\text{second}} \tag{18}$$

(22)

Design: Compose Units (1/2)

$$\text{expr: } \frac{10000\,\text{revolution}}{1\,\text{minute} \cdot 2\,\text{second}} \tag{18}$$

$$\Longrightarrow \text{expr: } \frac{10000\,\text{revolution}}{(1 \cdot 2)\,(\text{minute} \cdot \text{second})} \tag{19}$$

(22)

Design: Compose Units (1/2)

$$\text{expr: } \frac{10000\,\text{revolution}}{1\,\text{minute} \cdot 2\,\text{second}} \tag{18}$$

$$\implies \text{expr: } \frac{10000\,\text{revolution}}{(1 \cdot 2)\,(\text{minute} \cdot \text{second})} \tag{19}$$

$$\implies \text{expr: } \frac{10000\,\text{revolution}}{(1 \cdot 2)\left(\begin{array}{c}\left\{\left\{\mathrm{m}^0,\mathrm{s}^1,\mathrm{rad}^0\right\},\frac{60}{1},\left\{\pi^0\right\}\right\} \\ \cdot\left\{\left\{\mathrm{m}^0,\mathrm{s}^1,\mathrm{rad}^0\right\},\frac{1}{1},\left\{\pi^0\right\}\right\}\end{array}\right)} \tag{20}$$

(22)

Design: Compose Units (1/2)

$$\text{expr: } \frac{10000\,\text{revolution}}{1\,\text{minute} \cdot 2\,\text{second}} \tag{18}$$

$$\Longrightarrow \text{expr: } \frac{10000\,\text{revolution}}{(1 \cdot 2)\,(\text{minute} \cdot \text{second})} \tag{19}$$

$$\Longrightarrow \text{expr: } \frac{10000\,\text{revolution}}{(1 \cdot 2)\left(\begin{array}{c}\left\{\left\{\mathrm{m}^0,\mathrm{s}^1,\mathrm{rad}^0\right\},\frac{60}{1},\left\{\pi^0\right\}\right\} \\ \cdot\left\{\left\{\mathrm{m}^0,\mathrm{s}^1,\mathrm{rad}^0\right\},\frac{1}{1},\left\{\pi^0\right\}\right\}\end{array}\right)} \tag{20}$$

$$\Longrightarrow \text{expr: } \frac{10000\,\text{revolution}}{(1 \cdot 2)\left\{\left\{\mathrm{m}^0,\mathrm{s}^2,\mathrm{rad}^0\right\},\frac{60}{1},\left\{\pi^0\right\}\right\}} \tag{21}$$

(22)

Design: Compose Units (2/2)

$$\implies \text{expr: } \frac{10000\left\{\left\{\mathrm{m}^0, \mathrm{s}^0, \mathrm{rad}^1\right\}, \frac{2}{1}, \left\{\pi^1\right\}\right\}}{(1 \cdot 2)\left\{\left\{\mathrm{m}^0, \mathrm{s}^2, \mathrm{rad}^0\right\}, \frac{60}{1}, \left\{\pi^0\right\}\right\}} \tag{23}$$

(24)

Design: Compose Units (2/2)

$$\Longrightarrow \text{ expr: } \frac{10000\left\{\left\{\mathrm{m}^0, \mathrm{s}^0, \mathrm{rad}^1\right\}, \frac{2}{1}, \left\{\pi^1\right\}\right\}}{(1\cdot 2)\left\{\left\{\mathrm{m}^0, \mathrm{s}^2, \mathrm{rad}^0\right\}, \frac{60}{1}, \left\{\pi^0\right\}\right\}} \tag{23}$$

$$\Longrightarrow \text{ expr: } \left(\frac{10000}{1\cdot 2}\right)\left\{\left\{\mathrm{m}^0, \mathrm{s}^{-2}, \mathrm{rad}^1\right\}, \frac{2}{60}, \left\{\pi^1\right\}\right\} \tag{24}$$

For Reference

- https://people.freebsd.org/~kami/2015-32C3/
- https://github.com/lonkamikaze/units