# Good practices in Rust

This is not exhaustive

#### Why having good practices

- They help you
  - You avoid mistakes
- They help other getting up to speed on your project
  - It makes your code more readable
- They create a de facto standard
  - Some cargo commands emit a warning if your code is not commited
- They cost some time at the beginning, but they help a lot in the long run
  - The bigger the project, the bigger the reward, e.g. git is very useful if you're working alone, but a must-have for a team

#### But they do not solve everything

- They cannot cancel the effects of a poor design
- They are not all mandatory (especially on small projects)
- Don't forget your goal: solving a problem, publishing a paper, etc.
  - you are not paid for writing test, documenting or even writing code

# OK, let's go

## Ease of life rules

#### Respect the standards of your team

They take precedence over what you prefer.

If you want to change them:

- You have a good reason? You can change them, but don't it alone.
- You don't? You are bike-shedding.

#### Respect the standards of the community

They (still) take precedence over what you prefer.

Even if you're working alone. Mostly.

### Use git

Git allows you to collaborate and to keep track of your changes

https://learngitbranching.js.org

Protect the main branch, use git hooks, etc.



#### Name your variables

Avoid name like a, drvr\_clnt\_ctrl, etc.

Exception: iteration variables.

Actually, avoid variables when possible.

#### Use functions

- They document your code for free
- They make you code faster
- They provide context for naming your variables
- No out parameters
- They have a cost, but unless proven otherwise, it is negligible.

#### Document your functions

Place

///

Before your functions.

#### Example:

/// Returns the arguments that this program was started with (normally passed

/// via the command line).

///

/// The first element is traditionally the path of the executable, but it can be

/// set to arbitrary text, and might not even exist. This means this property should

/// not be relied upon for security purposes.

///

/// Note that the returned iterator will not check if the arguments to the

/// process are valid Unicode. If you want to panic on invalid UTF-8,

/// use the [`args`] function instead.

/// # Examples
///
/// ```
/// use std::env;
///
/// // Prints each argument on a separate line
/// for argument in env::args_os() {
/// println!("{argument:?}");
/// }
/// ```

#### Organize your code into modules

A single file, e.g. module\_name.rs.

Or a folder, module\_name, containing submodules and a file mod.rs.

#### Document your modules

Same as functions:

//!

#### Example

//! Inspection and manipulation of the process's environment.

#### //!

//! This module contains functions to inspect various aspects such as

//! environment variables, process arguments, the current directory, and various

//! other important directories.

#### //!

//! There are several functions and structs in this module that have a

//! counterpart ending in `os`. Those ending in `os` will return an [`OsString`]

//! and those without will return a [`String`].

## Correctness

#### Write unit tests

(See code)

#### Use assertions

- assert!(boolean) panics if boolean is false
- debug\_assert!(boolean) does the same, but is removed in release mode

#### Avoid unsafe code

Unsafe code disables some of the compiler's guarantees

#### If you're not careful, it can lead to undefined behavior

There is a reason we didn't see unsafe code yet

- There is usually a better alternative

#### Run your unsafe code with miri

https://github.com/rust-lang/miri

Miri is able to detect undefined behavior at runtime

### More effort: run your unsafe code with kani

https://model-checking.github.io/kani/install-guide.html

kani tests all possible input for your code to detect undefined behavior

#### Setup a CI/CD pipeline

They run the tests in a standardize environment

They can protect a git branch

They can send emails when they are broken

#### Setup a CI/CD pipeline

name: tests	jobs:	tests_stable:	tests_nightly:			
	check:	strategy:	strategy:			
	strategy:	matrix:	matrix:			
on:	matrix:	os: [ubuntu-latest]	os: [ubuntu-latest]			
push:	os: [ubuntu-latest]	toolchain: [stable]	toolchain: [nightly]			
branches: [main]	toolchain: [stable, nightly]	runs-on: \${{ matrix.os }}	runs-on: \${{ matrix.os }}			
pull_request:	runs-on: \${{ matrix.os }}	steps:	steps:			
	steps:	- name: Checkout code	- name: Checkout code			
branches: [main]	- name: Checkout code	uses: actions/checkout@v4	uses: actions/checkout@v4			
	uses: actions/checkout@v4	- name: Install Rust	- name: Install Rust			
	- name: Install Rust	uses: dtolnay/rust-toolchain@stable	uses: dtolnay/rust-toolchain@stable			
	uses: dtolnay/rust-toolchain@stable	with:	with:			
	with:	toolchain: \${{ matrix.toolchain }}	toolchain: \${{ matrix.toolchain }}			
	toolchain: \${{ matrix.toolchain }}	- name: Run tests	- name: Run tests			
	- name: Run check	run: cargo test	run: cargo testall-features			

#### Check out the code coverage

Check that critical functions are covered

It also prevent you from forgetting to write tests

# Generate test coverage: # \* report on terminal # \* lcov file # \* html # # USAGE # -----# ./coverage.sh # or # ./coverage.sh <name of the module to test> module name=\$1 LCOV FILE=.lcov.info # Clean files rm \$LCOV FILE 2>/dev/null

cargo llvm-cov clean --workspace

# Terminal visual report
cargo llvm-cov nextest \$module name

# Generate lcov file in .lcov.info
cargo llvm-cov report --lcov --output-path \$LCOV\_FILE

# Generate html file in target/llvm-cov/html
genhtml \$LCOV\_FILE --output-directory=coverage/html
mv .lcov.info coverage/lcov.info

#### LCOV - code coverage report

Current view: top level		Hit	Total	Coverage
Test: .lcov.info	Lines:	5114	7481	68.4 %
Date: 2025-05-13 14:14:30	Functions:	496	665	74.6 %

Directory	i i i i i i i i i i i i i i i i i i i	Line Coverage 🖨	Functions 🗢		
STC		54.4 %	834 / 1534	66.1 %	115 / 174
src/codec		99.7 %	686 / 688	94.7 %	36 / 38
src/codec/bmi		99.5 %	1136 / 1142	100.0 %	65 / 65
<u>src/index</u>		55.2 %	315 / 571	27.5 %	11/40
<pre>src/index/components</pre>		96.2 %	126 / 131	83.3 %	10 / 12
<pre>src/index/components/hyperkmer_parts</pre>		82.4 %	98 / 119	70.6 %	12/17
<pre>src/index/components/hyperkmer_parts/typical_hyperkmer_parts</pre>		97.5 %	306 / 314	77.8 %	28 / 36
<pre>src/index/components/hyperkmers_counts</pre>		36.4 %	368 / 1010	57.1 %	36 / 63
<pre>src/index/computation</pre>		0.0 %	0 / 586	0.0 %	0/19
<pre>src/index/computation/cache</pre>		90.3 %	112 / 124	88.2 %	15 / 17
src/serde		18.5 %	20 / 108	16.7 %	2/12
<u>src/simd</u>		96.0 %	984 / 1025	96.3 %	155 / 161
<pre>src/simd/intrinsics</pre>		100.0 %	129 / 129	100.0 %	11 / 11

Generated by: LCOV version 1.16



## Performance

#### Release and debug mode

- Run tests in debug mode.
- Work in debug mode if possible.
- Test performance in release mode.

#### Do NOT optimize before benchmarking it

"We should forget about small efficiencies, say about 97% of the time: **premature optimization is the root of all evil**. Yet we should not pass up our opportunities in that critical 3 %."

Structured Programming with goto Statements

Donald E. Knuth

#### How to benchmark

Do not do it "manually", e.g. with print(time) everywhere

- does not work
- time your time

Use a profiler, e.g. <u>https://www.intel.com/content/www/us/en/developer/tools/oneapi/vtune-profiler-do</u> wnload.html

(You may need to run "echo -1 | sudo tee /proc/sys/kernel/perf\_event\_paranoid" before usage) (do it at every reboot)





#### Look at your CPU usage

Q: + = ⊭	1	)s 20s	40s	60s	82.857s	100s	120s	140s	160s	180s	200s	220s	
std::sys::pal::unix::thread::Tl	hr	os 20s		and a superior of the supervised of the	10 marine and a second		and the second second	and have been a been	- Brankerka kikana antakan	a faire descent setting a stat	Marth	THE REAL PROPERTY AND IN COMPANY	
std::sys::pal::unix::thread::Ti	hr		of the sector which have	A PARTY & MINIS PARTY	in the second	a of the River of the other River	and a subscription of the second second	and a line of the second second	and the section of th	And the second sec	And the second second	Automation Street	
std::sys::pal::unix::thread::Tl	hr	and her many of her second		and the distribution of the second	معمدهم فيعمد ستساللا هي		عريبه سال العليب	Large Management	and a south a fille works	And the second se	name (M) provide balan	and the second se	·
std::sys::pal::unix::thread::Tl	hr	المحمد والمرتب المؤتم والمراجع والمرتب	al status and the state of a discuss	and product the first sector of	California (California)	and the second second	And the Although the	Indiana providence and the	y produkti da dan da abbi das	manifer Jule Poly di	APPRILIT PROPERTY AND	All and a state of the	
std::sys::pal::unix::thread::Tl	hr			and the local product of the									
std::sys::pal::unix::thread::Tl	hr			and by the still state in the state of the s	100 December 10			and the second second		all a share of the second	Intel Lawrence	a frances of the second se	
std::sys::pal::unix::thread::Tl	hr		and the second	Man Andre Manager Marson	All and the second s		alla Mala panta a	and the second sec		and the second states of the	Could proceed	and and and a	
std::sys::pal::unix::thread::Tl	hr			al all constant and	And the second second	and the first of the second second	and the line of the second	KILVEN CONTRACTOR	and the product of the product of the	Man Marine and Construction	Nextlineerneit	and the state of the	<u> </u>
std::sys::pal::unix::thread::Tl	hr			and the first state of the second state of the									
std::sys::pal::unix::thread::Tl	hr	Constant of the second second second	And constant of some data dataset	and the state of Mark and states	And the second second second	States and the second second	and subbracks			and the property of the	and him the second	ALCONOMIC STREET, ST.	1. A. 1
std::sys::pal::unix::thread::Tl	hr		alles sands illus as decisioned data in a	and the second second second	W Manual Manual	All sectors in the local sectors		A PARTY CONTRACTOR	Adamatic Contractor Contractor	and a state of the	(he de la company de la compan	and a blocker of	
std::svs::pal::unix::thread::TI CPU Utilization	hr			na na antara di kina ni ang mana ang ma		a la an de mar and discase po			an ant an An-Malan dia and a faile and	dan on All Bilded, Man			