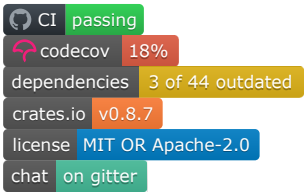
cargo-compete



English

競技プログラミングのためのCargoコマンドです。

AtCoder、Codeforces、yukicoderをサポートしています。 その他のサイトもonline-judge-tools/api-clientを使うことで利用可能です。

機能

- サイトへのログイン
- (自動で)コンテストへの参加登録
- サンプルテストケース/システムテストケースを取得し、YAMLで保存
- コードのテスト
- 提出
- 提出一覧のストリーミング (AtCoderのみ)

	参加登録	サンプルテストケース	システムテストケース	提出	提出一覧をwatch	提出の詳細
AtCoder	✓	✓	✓	✓	?	✗
Codeforces	✗	✓	N/A	✓	✗	✗
yukicoder	N/A	✓	✓	✓	✗	✗
その他のサイト	✗	online-judge-tools次第	online-judge-tools次第	online-judge-tools次第	✗	✗

インストール

Crates.ioからインストール

```
$ cargo install cargo-compete
```

ビルドが失敗するなら、 `--locked` を付けると成功する場合があります。

`master` ブランチからインストール

```
$ cargo install --git https://github.com/qryxip/cargo-compete
```

GitHub Releasesからバイナリをダウンロード

バイナリでの提供もしています。

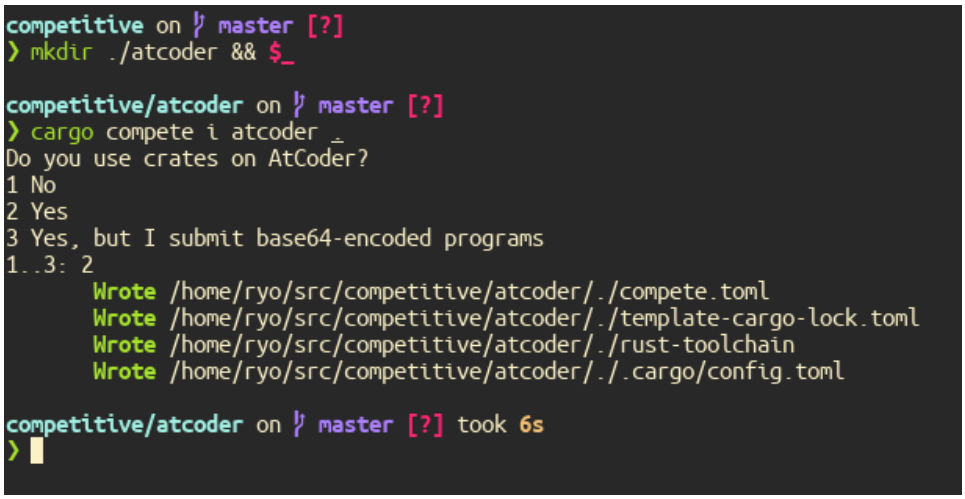
使い方

`cargo compete init`

他のコマンドのためにいくつかのファイルを生成します。

最初に実行してください。 生成するファイルは以下の通りです。

- `compete.toml`
  他のコマンドに必要です。cargo-atcoderのように自動で生成しません。
- `rust-toolchain`
  ツールチェインのバージョンを指定するテキストファイルまたはTOMLファイルです。 例えば `1.42.0` と書けば、 `rust-toolchain` を置いたディレクトリ下で `~/.cargo/bin/cargo(.exe)` を起動したときに1.42.0の `cargo` と `rustc` が呼ばれるようになります。
- `.cargo/config.toml`
  `build/target-dir` を設定し、 `target` ディレクトリを共有するようにします。
- `template-cargo-lock.toml`
  `cargo compete new` に使う `Cargo.lock` のテンプレートです。 質問に「AtCoderでクレートを使用するがバイナリ提出はしない」と回答した場合のみ生成されます。 生成された場合、 `compete.toml` の `new.template.lockfile` にこのファイルへのパスが追加されます。



`cargo compete migrate cargo-atcoder`

`cargo-atcoder` で作ったパッケージをそれぞれ `cargo-compete` 用にマイグレートし、 `compete.toml` 等のファイルも追加します。

TODO: ↓のスクショをアップデート。今 `package.metadata.cargo-compete.bin.*.problem` はURLの文字列です。

```
atcoder on  master [?]
❯ ll
drwxr-xr-x - ryo ryo 2020-08-30 06:26 -N abc170
drwxr-xr-x - ryo ryo 2020-08-30 06:26 -N abc171
drwxr-xr-x - ryo ryo 2020-08-30 06:26 -N abc172

atcoder on  master [?]
❯ cargo compete m cargo-atcoder
       Found `/home/ryo/src/atcoder/abc170/Cargo.toml`
       Found `/home/ryo/src/atcoder/abc171/Cargo.toml`
       Found `/home/ryo/src/atcoder/abc172/Cargo.toml`
    Modified /home/ryo/src/atcoder/abc170/Cargo.toml
    Modified /home/ryo/src/atcoder/abc171/Cargo.toml
    Modified /home/ryo/src/atcoder/abc172/Cargo.toml
    Updating /home/ryo/src/atcoder/abc170/Cargo.lock
    Updating /home/ryo/src/atcoder/abc171/Cargo.lock
    Updating /home/ryo/src/atcoder/abc172/Cargo.lock
       Wrote /home/ryo/src/atcoder/compete.toml
    Finished migrating

atcoder on  master [?]
❯ bat ./abc170/Cargo.toml --style plain
[package]
name = "abc170"
version = "0.1.0"
authors = ["Ryo Yamashita <qryxip@gmail.com>"]
edition = "2018"

[package.metadata.cargo-compete]
config = "../compete.toml"

[package.metadata.cargo-compete.bin]
a = {name = "abc170-a",problem = {platform = "atcoder",contest = "abc170",index = "A"}}
b = {name = "abc170-b",problem = {platform = "atcoder",contest = "abc170",index = "B"}}
c = {name = "abc170-c",problem = {platform = "atcoder",contest = "abc170",index = "C"}}
d = {name = "abc170-d",problem = {platform = "atcoder",contest = "abc170",index = "D"}}
e = {name = "abc170-e",problem = {platform = "atcoder",contest = "abc170",index = "E"}}
f = {name = "abc170-f",problem = {platform = "atcoder",contest = "abc170",index = "F"}}

# dependencies added to new project
[dependencies]
proconio = { version = "0.4.1", features = ["derive"] }

[[bin]]
name = "abc170-a"
path = "./src/bin/a.rs"

[[bin]]
name = "abc170-b"
path = "./src/bin/b.rs"

[[bin]]
name = "abc170-c"
path = "./src/bin/c.rs"
```

`cargo compete login`

サイトにログインします。

パッケージを対象に取りません。 引数で与えられた `platform` に対してログインします。

ただし `new` コマンド等ではログインが必要になった場合でも認証情報を聞いてログインし、続行するため事前にこのコマンドを実行しなくてもよいです。

`cargo compete participate`

コンテストに参加登録します。

パッケージを対象に取りません。 引数で与えられた `platform` と `contest` に対して参加登録します。

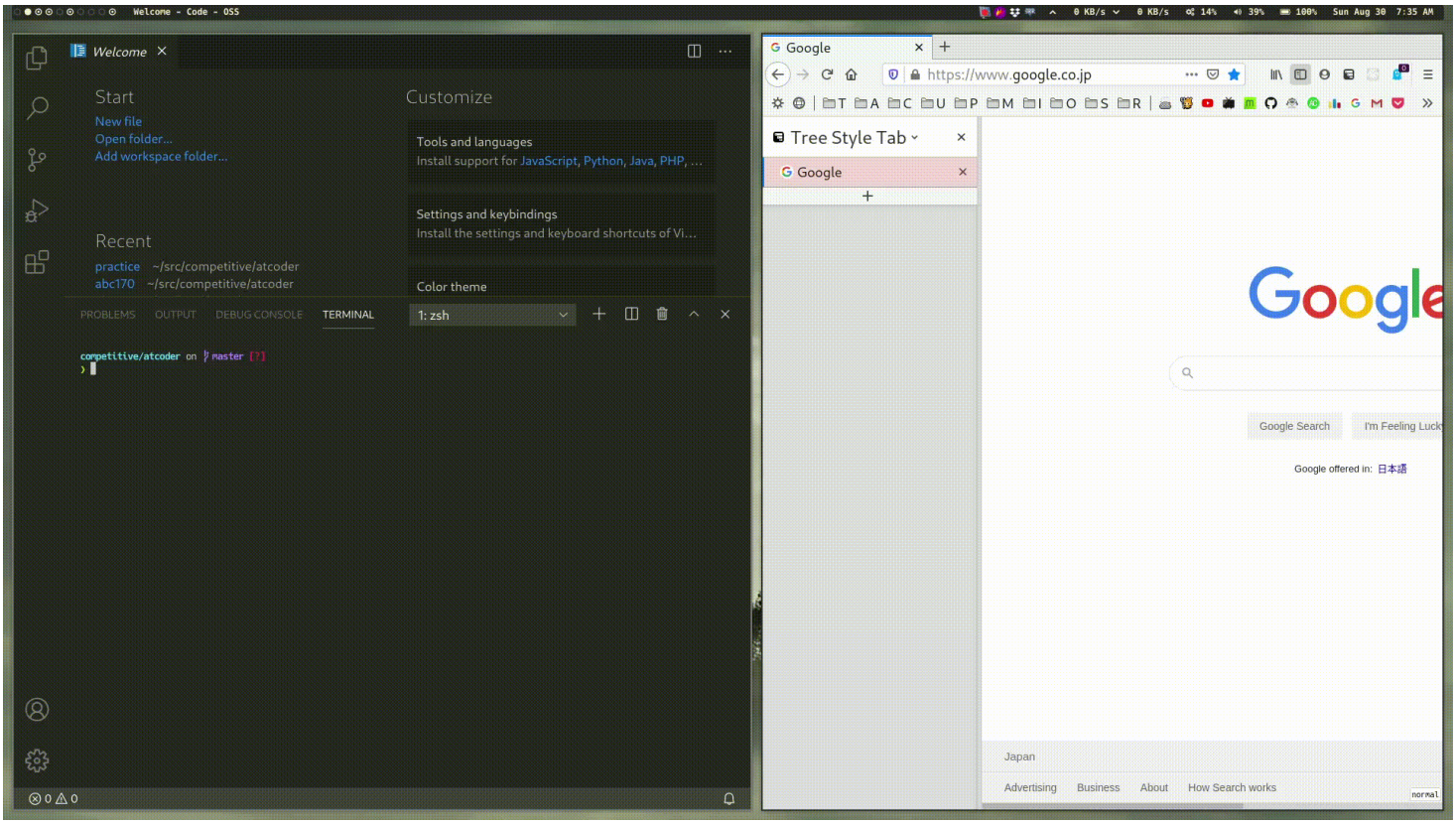
(`login` コマンドと同様に、) `new` コマンド等で自動で参加登録するため事前にこのコマンドを実行しなくてもよいです。

`cargo compete new`

テストケースを取得し、コンテストに応じたパッケージを作ります。

`compete.toml` が必要です。 最初に `cargo compete init` で生成してください。

`--open` で問題のページをブラウザで開きます。 また `compete.toml` の `open` を設定することで、ソースコードとテストケースのYAMLをエディタで開くことができます。 `--open` を付け忘れた場合は生成されたパッケージに `cd` した後に `cargo compete open` で開いてください。



`.cargo/config.toml` によりtarget directoryが共有されるので、クレートを使う場合も初回を除いて"warmup"は不要です。

`cargo compete add`

コンテストもしくは問題に対して `bin` ターゲットを生成し、テストケースをダウンロードします。

`compete.toml` が必要です。 最初に `cargo compete init` で生成してください。

設定は `compete.toml` の `add` で行ってください。

```
# for yukicoder
[add]
url = '{% case args[0] %}{% when "contest" %}https://yukicoder.me/contests/{{ args[1] }}{% when "problem" %}https
is-contest = ["bash", "-c", '[[ $(cut -d / -f 4) == "contests" ]]'] # optional
bin-name = '{% assign segments = url | split: "/" %}{{ segments[5] }}'
#bin-alias = '{% assign segments = url | split: "/" %}{{ segments[5] }}' # optional
#bin-src-path = './src/bin/{{ bin_alias }}.rs' # optional
```

```
❯ cargo compete a contest 296
    Added `1358` (bin) for https://yukicoder.me/problems/no/1358
    Added `1359` (bin) for https://yukicoder.me/problems/no/1359
    Added `1360` (bin) for https://yukicoder.me/problems/no/1360
    Added `1361` (bin) for https://yukicoder.me/problems/no/1361
    Added `1362` (bin) for https://yukicoder.me/problems/no/1362
    Added `1363` (bin) for https://yukicoder.me/problems/no/1363
    Added `1364` (bin) for https://yukicoder.me/problems/no/1364
    Added `1365` (bin) for https://yukicoder.me/problems/no/1365
    Saved 1 test case to /home/ryo/src/competitive/yukicoder/testcases/1358.yml
    Saved 3 test cases to /home/ryo/src/competitive/yukicoder/testcases/1359.yml
    Saved 3 test cases to /home/ryo/src/competitive/yukicoder/testcases/1360.yml
    Saved 3 test cases to /home/ryo/src/competitive/yukicoder/testcases/1361.yml
    Saved 3 test cases to /home/ryo/src/competitive/yukicoder/testcases/1362.yml
    Saved 1 test case to /home/ryo/src/competitive/yukicoder/testcases/1363.yml
    Saved 3 test cases to /home/ryo/src/competitive/yukicoder/testcases/1364.yml
    Saved 3 test cases to /home/ryo/src/competitive/yukicoder/testcases/1365.yml
❯ cargo compete a problem 9001
    Added `9001` (bin) for https://yukicoder.me/problems/no/9001
    Saved 1 test case to /home/ryo/src/competitive/yukicoder/testcases/9001.yml
```

`cargo compete retrieve testcases` / `cargo compete download`

テストケースの再取得を行います。

パッケージを対象に取ります。 パッケージに `cd` して実行してください。

```
competitive/atcoder/practice on  master [?] is 📦 v0.1.0 via ⚙ v1.42.0
❯ cargo compete d
       Saved 2 test cases to /home/ryo/src/competitive/atcoder/practice/testcases/a.yml
       Saved no test cases (interactive problem) to /home/ryo/src/competitive/atcoder/practice/testcases/b.yml
```

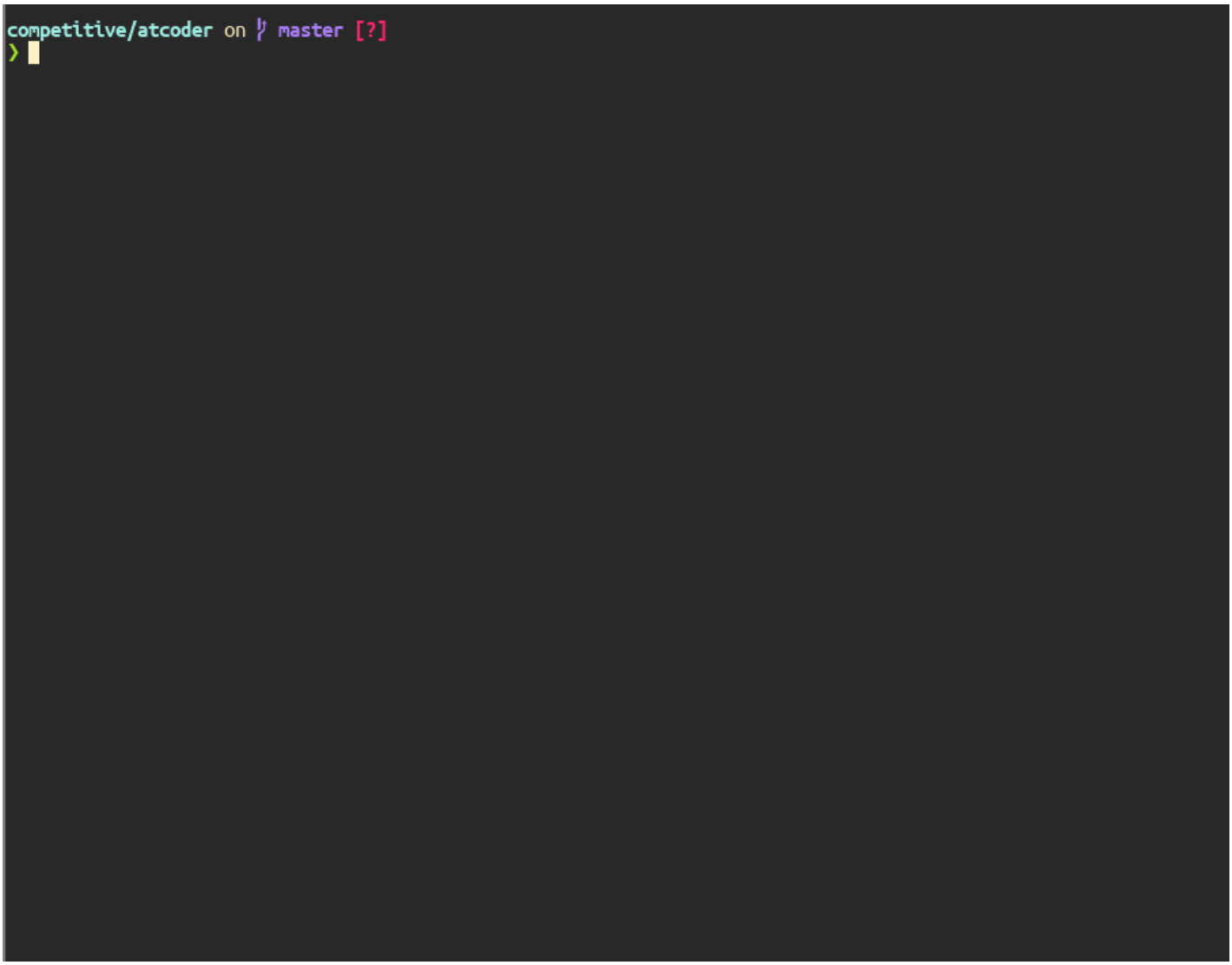
プラットフォームが使っているテストケースを公開している場合、 `--full` を指定することでそちらをダウンロードすることができます。

AtCoderの場合、テストケースはDropboxにアップロードされているのでそちらからダウンロードします。ただしDropbox APIを使用するため

- `files.metadata.read`
- `sharing.read`

の2つのパーミッションが有効なアクセストークンが必要です。 何らかの方法でアクセストークンを取得し、以下の形式のJSONファイルを `{local data directory}/cargo-compete/tokens/dropbox.json` に保存してください。 (この辺はなんとかしたいと考えてます)

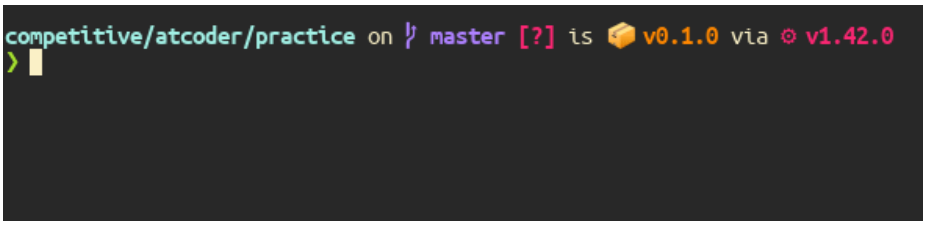
```
{
  "access_token": "<access token>"
}
```



cargo compete retrieve submission-summaries

自分の提出の一覧を取得し、JSONで出力します。

パッケージを対象に取ります。 パッケージに `cd` して実行してください。



例えばAtCoderであれば(AtCoderしか実装してませんが) `| jq -r '.summaries[0].detail` とすることで「最新の提出の詳細ページのURL」が得られます。

```
$ # 最新の提出の詳細ページをブラウザで開く（Linuxの場合）
$ xdg-open "$(cargo compete r ss | jq -r '.summaries[0].detail')"
```

cargo compete open

`new` の `--open` と同様に問題のページをブラウザで、コードとテストファイルをエディタで開きます。

パッケージを対象に取ります。 パッケージに `cd` して実行してください。

`cargo compete test`

テストを行います。

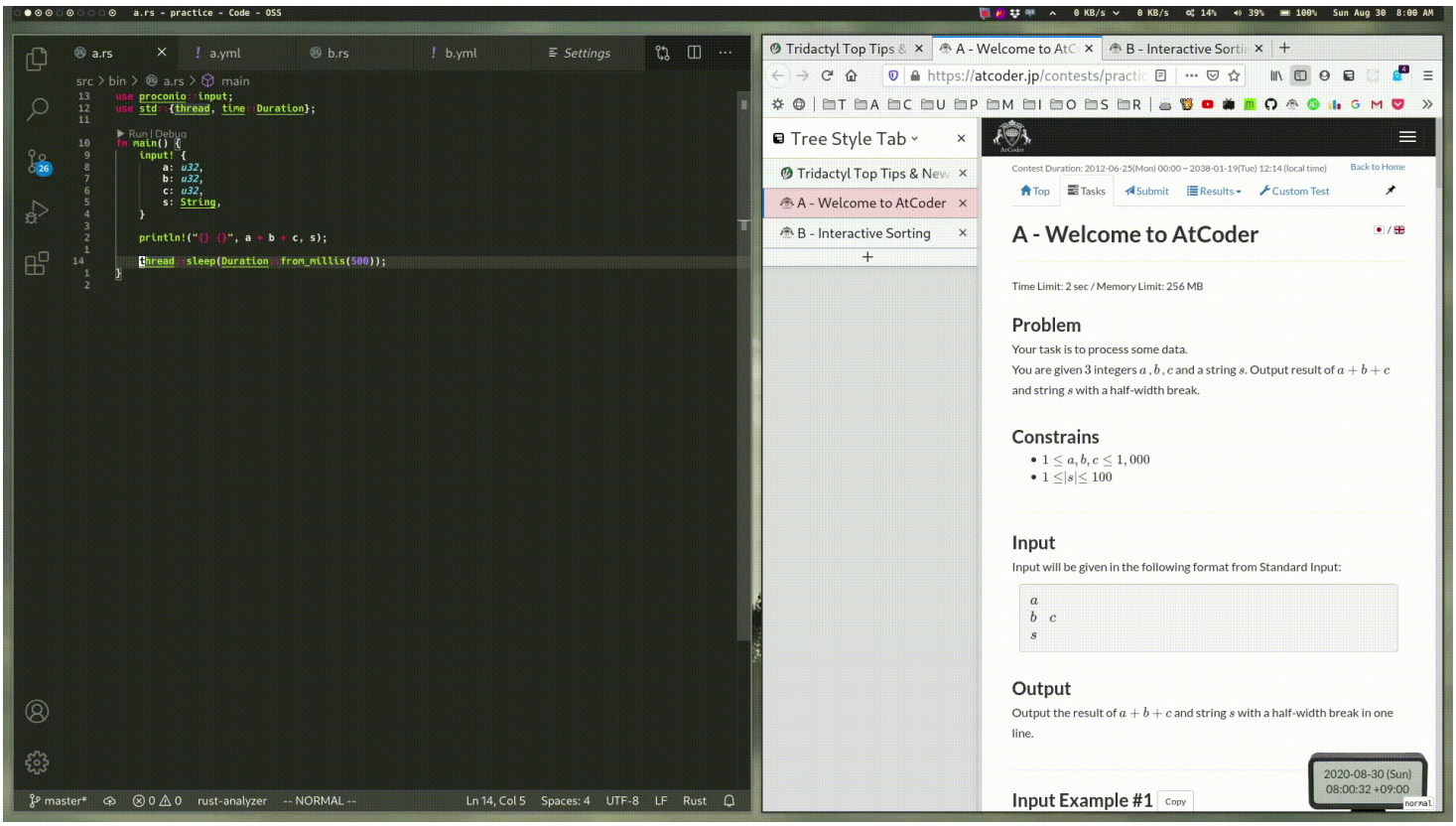
パッケージを対象に取ります。 パッケージに `cd` して実行してください。

`submit` 時にも提出するコードをテストするため、提出前にこのコマンドを実行しておく必要はありません。

`cargo compete submit`

提出を行います。

パッケージを対象に取ります。 パッケージに `cd` して実行してください。



`compete.toml` の `submit.transpile` を設定することで、cargo-equipやcargo-executable-payload等のコード変換ツールを使って提出するコードを変換できます。

```
[submit.transpile]
kind = "command"
args = ["cargo", "equip", "--resolve-cfgs", "--remove", "docs", "--minify", "libs", "--rustfmt", "--check", "--bi
#language_id = ""


[submit.transpile]
kind = "command"
args = ["cargo", "executable-payload", "--bin", "{{ bin_name }}"]
#language_id = ""
```

設定

設定は各ワークスペース下にある `compete.toml` にあります。

```
# Path to the test file (Liquid template)
#
# Variables:
#
# - `manifest_dir`: Package directory
# - `contest`:      Contest ID (e.g. "abc100")
# - `bin_name`:     Name of a `bin` target (e.g. "abc100-a")
# - `bin_alias`:    "Alias" for a `bin` target defined in `pacakge.metadata.cargo-compete` (e.g. "a")
# - `problem`:      Alias for `bin_alias` (deprecated)
#
# Additional filters:
#
# - `kebabcase`: Convert to kebab case (by using the `heck` crate)
test-suite = "{{ manifest_dir }}/testcases/{{ bin_alias }}.yml"

# Open files with the command (`jq` command that outputs `string[] | string[][]`)
#
# VSCode:
#open = '[["code", "-a", .manifest_dir], ["code"] + (.paths | map([.src, .test_suite]) | flatten)]'
# Emacs:
#open = '["emacsclient", "-n"] + (.paths | map([.src, .test_suite]) | flatten)'

[new]
kind = "cargo-compete"
# Platform
#
# - atcoder
# - codeforces
# - yukicoder
platform = "atcoder"
# Path (Liquid template)
#
# Variables:
#
# - `contest`:      Contest ID. **May be nil**
# - `package_name`: Package name
path = "./{{ contest }}"

#[new]
#kind = "oj-api"
#url = "https://atcoder.jp/contests/{{ id }}"
#path = "./{{ contest }}"

[new.template]
lockfile = "./template-cargo-lock.toml"
# `profile` for `Cargo.toml`.
#
# By setting this, you can run tests with `opt-level=3` while enabling `debug-assertions` and `overflow-checks`.
#profile = '''
#[dev]
#opt-level = 3
#'''

[new.template.dependencies]
kind = "inline"
content = '''
num = "=0.2.1"
num-bigint = "=0.2.6"
num-complex = "=0.2.4"
num-integer = "=0.1.42"
num-iter = "=0.1.40"
num-rational = "=0.2.4"
num-traits = "=0.2.11"
num-derive = "=0.3.0"
ndarray = "=0.13.0"
```

```
nalgebra = "=0.20.0"
alga = "=0.9.3"
libm = "=0.2.1"
rand = { version = "=0.7.3", features = ["small_rng"] }
getrandom = "=0.1.14"
rand_chacha = "=0.2.2"
rand_core = "=0.5.1"
rand_hc = "=0.2.0"
rand_pcg = "=0.2.1"
rand_distr = "=0.2.2"
petgraph = "=0.5.0"
indexmap = "=1.3.2"
regex = "=1.3.6"
lazy_static = "=1.4.0"
ordered-float = "=1.0.2"
ascii = "=1.0.0"
permutohedron = "=0.2.4"
superslice = "=1.0.0"
itertools = "=0.9.0"
itertools-num = "=0.1.3"
maplit = "=1.0.2"
either = "=1.5.3"
im-rc = "=14.3.0"
fixedbitset = "=0.2.0"
bitset-fixed = "=0.1.0"
proconio = { version = "=0.3.6", features = ["derive"] }
text_io = "=0.1.8"
whiteread = "=0.5.0"
rustc-hash = "=1.1.0"
smallvec = "=1.2.0"
'''

[new.template.src]
kind = "inline"
content = '''
fn main() {
    todo!();
}
'''

# for Library-Checker
#[add]
#url = "https://judge.yosupo.jp/problem/{{ args[0] }}"
##is-contest = ["false"] # optional
#bin-name = '{{ args[0] }}'
##bin-alias = '{{ args[0] }}' # optional
##bin-src-path = './src/bin/{{ bin_alias }}.rs' # optional

# for yukicoder
#[add]
#url = '{% case args[0] %}{% when "contest" %}https://yukicoder.me/contests/{{ args[1] }}{% when "problem" %}http
#is-contest = ["bash", "-c", '[[ $(cut -d / -f 4) == "contests" ]]'] # optional
#bin-name = '{% assign segments = url | split: "/" %}{{ segments[5] }}'
##bin-alias = '{% assign segments = url | split: "/" %}{{ segments[5] }}' # optional
##bin-src-path = './src/bin/{{ bin_alias }}.rs' # optional

[test]
# Profile for `cargo build`. ("dev" | "release")
#
# Defaults to `"dev"`.
#profile = "dev"

#[submit.transpile]
#kind = "command"
#args = ["cargo", "equip", "--resolve-cfgs", "--remove", "docs", "--minify", "libs", "--rustfmt", "--check", "--b
##language_id = ""
```

各 `bin` targetに紐付くサイト上の問題は、パッケージの `Cargo.toml` の `[package.metadata]` に記述されます。

```
[package]
name = "practice"
version = "0.1.0"
authors = ["Ryo Yamashita <qryxip@gmail.com>"]
edition = "2018"

[package.metadata.cargo-compete.bin]
practice-a = { alias = "a", problem = "https://atcoder.jp/contests/practice/tasks/practice_1" }
practice-b = { alias = "b", problem = "https://atcoder.jp/contests/practice/tasks/practice_2" }

[[bin]]
name = "practice-a"
path = "src/bin/a.rs"

[[bin]]
name = "practice-b"
path = "src/bin/b.rs"
[dependencies]
num = "=0.2.1"
num-bigint = "=0.2.6"
num-complex = "=0.2.4"
num-integer = "=0.1.42"
num-iter = "=0.1.40"
num-rational = "=0.2.4"
num-traits = "=0.2.11"
num-derive = "=0.3.0"
ndarray = "=0.13.0"
nalgebra = "=0.20.0"
alga = "=0.9.3"
libm = "=0.2.1"
rand = { version = "=0.7.3", features = ["small_rng"] }
getrandom = "=0.1.14"
rand_chacha = "=0.2.2"
rand_core = "=0.5.1"
rand_hc = "=0.2.0"
rand_pcg = "=0.2.1"
rand_distr = "=0.2.2"
petgraph = "=0.5.0"
indexmap = "=1.3.2"
regex = "=1.3.6"
lazy_static = "=1.4.0"
ordered-float = "=1.0.2"
ascii = "=1.0.0"
permutohedron = "=0.2.4"
superslice = "=1.0.0"
itertools = "=0.9.0"
itertools-num = "=0.1.3"
maplit = "=1.0.2"
either = "=1.5.3"
im-rc = "=14.3.0"
fixedbitset = "=0.2.0"
bitset-fixed = "=0.1.0"
proconio = { version = "=0.3.6", features = ["derive"] }
text_io = "=0.1.8"
whiteread = "=0.5.0"
rustc-hash = "=1.1.0"
smallvec = "=1.2.0"
```

テストファイルのYAML

テストケースは以下のような形でYAMLに保存されます。

```yaml
# https://atcoder.jp/contests/practice/tasks/practice_1
---
type: Batch
timelimit: 2s
match: Lines

cases:
  - name: sample1
    in: |
      1
      2 3
      test
    out: |
      6 test
  - name: sample2
    in: |
      72
      128 256
      myonmyon
    out: |
      456 myonmyon

extend:
  - type: Text
    path: "./a"
    in: /in/*.txt
    out: /out/*.txt
```

```yaml
# https://atcoder.jp/contests/ddcc2019-final/tasks/ddcc2019_final_a
---
type: Batch
timelimit: 2s
match:
  Float:
    relative_error: 1e-8
    absolute_error: 1e-8

cases:
  - name: sample1
    in: |
      5
      -->--
    out: |
      3.83333333333333
  - name: sample2
    in: |
      7
      -------
    out: |
      6.5
  - name: sample3
    in: |
      10
      -->>>-->--
    out: |
      6.78333333333333

extend:
  - type: Text
    path: "./a"
    in: /in/*.txt
    out: /out/*.txt


# https://judge.yosupo.jp/problem/sqrt_mod
---
type: Batch
timelimit: 10s
match:
  Checker:
    cmd: ~/.cache/online-judge-tools/library-checker-problems/math/sqrt_mod/checker "$INPUT" "$ACTUAL_OUTPUT" "$E
    shell: Bash

cases: []

extend:
  - type: SystemTestCases
```

形式は以下のスキーマにおける `TestSuite` です。

TestSuite

`type` をタグとしたinternally taggedのADTです。

- `TestSuite::Batch`
- `TestSuite::Interactive`
- `TestSuite::Unsubmittable`

TestSuite::Batch

通常の問題に対するテストスイートです。

フィールド	型	デフォルト	説明
timelimit	Duration | null	~	実行時間制限
match	Match		出力の判定方法
cases	Case[]	[]	入出力のセット
extend	Extend[]	[]	入出力のセットの追加

Duration

humantime::format_duration でパースできる文字列です。

Match

untaggedなADTです。

- Match::Exact
- Match::Lines
- Match::Float
- Match::Checker

Match::Exact = "Exact"

文字列全体の一致で判定します。

Match::Lines = "Lines"

各行の一致で判定します。

Match::Float

空白区切りでの単語の一致で判定します。

この際数値として読める単語は浮動小数点数とみなし、誤差を許容します。

フィールド	型	デフォルト	説明
relative_error	PositiveFiniteFloat64 | null	~	相対誤差
absolute_error	PositiveFiniteFloat64 | null	~	絶対誤差

PositiveFiniteFloat64

正かつ inf ではない64-bitの浮動小数点数です。

Match::Checker

フィールド	型	デフォルト	説明
cmd	str		コマンド

フィールド	型	デフォルト	説明
shell	Shell		シェル

Shell

untaggedなADTです。

- Shell::Bash

Shell::Bash = "Bash"

Bashです。

Case

フィールド	型	デフォルト	説明
name	str	""	名前
in	str		入力
out	str | null	~	出力
timelimit	Duration | null	~	timelimit をオーバーライド
match	Match | null	~	match をオーバーライド

Extend

type をタグとしたinternally taggedのADTです。

- Extend::Text
- Extend::SysTemTestCases

Extend::Text

フィールド	型	デフォルト	説明
path	str		ディレクトリ
in	Glob		入力のテキストファイル
out	Glob		出力のテキストファイル
timelimit	Duration | null	~	timelimit をオーバーライド
match	Match | null	~	match をオーバーライド

Glob

globを示す文字列です。

Extend::SystemTestCases

システムテストケースです。

システムテストケースは `{ cache directory }/cargo-compete/system-test-cases` 下に保存されます。 `test` 時に見つからない場合、自動でダウンロードされます。

フィールド	型	デフォルト	説明
`problem`	`Url` | `null`	~	問題のURL

`Url`

URLを示す文字列です。

`TestSuite::Interactive`

フィールド	型	デフォルト	説明
`timelimit`	`Duration` | `null`	~	実行時間制限

`TestSuite::Unsubmittable`

APG4bの問題のようなもののためのダミーのテストスイートです。

フィールド	型	デフォルト	説明

Cookieとトークン

Cookieと各トークンは `{ local data directory }/cargo-compete` 下に保存されています。

```
.
├── cookies.jsonl
└── tokens
    ├── codeforces.json
    ├── dropbox.json
    └── yukicoder.json
```

環境変数

cargo-competeは以下の環境変数が存在する場合、それらを読んで使います。

- `$DROPBOX_ACCESS_TOKEN`
- `$YUKICODER_API_KEY`
- `$CODEFORCES_API_KEY`
- `$CODEFORCES_API_SECRET`

online-judge-toolsの利用

`download` 時と `submit` 時に対象のURLがサポートされていないサイトを指しているのなら、 `$PATH` 内にある `oj-api(.exe)` が使われます。

```toml
[package]
name = "library-checker"
version = "0.0.0"
edition = "2018"
publish = false

[package.metadata.cargo-compete.bin]
aplusb = { problem = "https://judge.yosupo.jp/problem/aplusb" }
```

cargo-atcoderとの対応

cargo atcoder new

`cargo compete new` でパッケージを作成します。

`compete.toml` を起点とします。 `cargo compete init` か `cargo compete migrate cargo-atcoder` で作成してください。

なお、開始前のコンテストには使えません。 `target` ディレクトリを共有する限り"warmup"が不要なためです。 ブラウザとエディタを開くのも `--open` で自動で行えます。

cargo atcoder submit

`cargo compete submit` でソースコードを提出します。

他のコマンドと同様に、ワークスペース下に `compete.toml` がある必要があります。

「バイナリ提出」を行う場合、cargo-executable-payloadを使うように `compete.toml` の `submit.transpile` を設定してください。

cargo atcoder test

`cargo compete test` でテストを実行します。

cargo-atcoderと同様にパッケージを対象に取ります。

一部のテストのみを実行する場合は、 `<case-num>...` の代わりに `--testcases <NAME>...` で `"sample1"` 等の「名前」で絞ります。

cargo atcoder login

`cargo compete login` でログインします。

cargo atcoder status

`cargo compete watch submission-summaries` で提出一覧をwatchします。

cargo-competeの方はブラウザ上の表示に近い挙動をするため、実行時点で「ジャッジ待ち」/「ジャッジ中」のものが無い場合には直近20件を表示だけして終了します。

cargo atcoder result

今のところありません。 `cargo compete watch submission-summaries` の出力を `| jq -r ".summaries[$nth].detail"` して得たURLをブラウザで開いてください。

`cargo atcoder clear-session`

今のところありません。 local data directory下の `cargo-compete` を削除してください。

`cargo atcoder info`

今のところありません。 ログインしているかを確認する場合、practice contestのテストケースをダウンロードしてください。
practice contestの場合問題の閲覧にログインが必要です。

`cargo atcoder warmup`

今のところありません。 上で述べた通り、 `target` ディレクトリを共有する場合初回を除きwarmupは不要です。

`cargo atcoder gen-binary`

cargo-executable-payloadを使ってください。

ライセンス