`cargo-leet` - A leetcode local development assistant

A program that given the link or slug to a leetcode problem, creates a local file where you can develop and test your solution before post it back to leetcode.

Usage

The `cargo leet` program helps you streamline your workflow with LeetCode problems by generating local files for solution development and testing before submitting them back to LeetCode. Below is a summary of how to use the various commands and options available in `cargo leet`.

General Usage

```
cargo leet [OPTIONS] <COMMAND>
```

Commands

- **new**
  Creates a new pre-configured project based on a template, which can be used with `cargo-leet`.

  ```
  cargo leet new [OPTIONS] [NAME]
  ```

- **generate, -g, gen**
  Generates a module for the specified problem, allowing you to start working on the solution locally. You can provide a LeetCode problem slug or URL, or leave it blank to use the daily challenge.

  ```
  cargo leet generate [OPTIONS] [PROBLEM]
  ```

- **active**
  Prints the currently active problem or sets the active problem to the provided problem slug.

  ```
  cargo leet active [OPTIONS] [PROBLEM_SLUG]
  ```

- **test**
  Runs tests on the currently active problem to verify your solution.

  ```
  cargo leet test [OPTIONS]
  ```

Options

- **-p, –path <FOLDER>**
  Specify the path to the project root. If not provided, the current working directory is used.
- **-l, –log-level <LOG_LEVEL>**
  Set the logging level. Default is `warn`. Available levels:
  - `off`: No logging
  - `error`
  - `warn`
  - `info`
  - `debug`
  - `trace`
- **-h, –help**
  Displays help information.
- **-V, –version**
  Prints the version of `cargo leet`.

Examples

- **Create a new project**:

  ```
  cargo leet new my-leetcode-project
  ```

- **Change into the directory**

  ```
  cd my-leetcode-project
  ```

- **Generate a module for a specific problem**:

  ```
  cargo leet generate two-sum
  ```

- **Set the active problem (done automatically by `cargo leet gen`)**:

  ```
  cargo leet active two-sum
  ```

- **Run tests on the active problem**:

  ```
  cargo leet test
  ```

Installation

Note: If `cargo-leet` is already installed and you install it again, the existing installation will be replaced, even if it was originally installed from a different source. For instance, if you first install it from a local clone and then reinstall it from a Git repository, the new installation will overwrite the previous one (you won't have both versions installed).

Build from Source

You can build `cargo-leet` from source using two different channels:

- **Stable (main)**

  ```
  cargo install --git https://github.com/rust-practice/cargo-leet.git --branch main -F tool
  ```

  This installs the stable version of `cargo-leet` from the `main` branch.

- **Development (develop)**

  ```
  cargo install --git https://github.com/rust-practice/cargo-leet.git --branch develop -F tool
  ```

  This installs the latest development version from the `develop` branch, which may include new features or changes that are still being tested.

Install from crates.io

You can also install `cargo-leet` directly from crates.io. However, please note that the crates.io release may not always reflect the latest updates.

```
cargo install cargo-leet -F tool
```

Running Directly from Source without Installation (For Development)

When developing the tool, you can run it directly from the source code without needing to install it. By default, these commands will execute the tool within the current working directory, meaning it will interact with the current project folder for `cargo-leet`.

Running in the Current Directory

Running the tool this way is useful for testing but may not be ideal if you need to target a specific project or repository. In such cases, you can specify the path to the desired repository using the `--path` option.

For example, to run the tool against a specific test repository:

```
cargo run -F tool -- leet gen --path $TEST_REPO
```

Using an Alias

If you have an alias configured in `.cargo/config.toml`, you can simplify the command:

```
cargo g
```

For additional options and usage details, refer to the generate help in commands.

Using as a Library

You can use `cargo-leet` as a library to mimic the LeetCode environment in your own projects. To do so, add it as a dependency in your `Cargo.toml` file using

```
cargo add cargo-leet
```

For more information, see the documentation.

License