pace user documentation

the pace authors

Contents

Chapter 1

Introduction

pace is a mindful productivity tool designed to help you keep track of your activities with ease and intention.

Born from the desire to blend simplicity with effectiveness, pace offers a command-line interface (CLI) that encourages focused work sessions, thoughtful reflection on task durations, and a harmonious balance between work and rest.

Whether you're a developer, a writer, or anyone who values structured time management, pace provides the framework to log activities, reflect on your progress, and optimize how you spend your time.

With features like the first activity wizard for onboarding new users, real-time configuration validation (upcoming), and personalized activity logs, pace is more than a time tracker — it's your partner in crafting a productive and mindful routine.

Contact

You can ask questions in the Discussions or have a look at the FAQ.

Contact	Where?
Issue Tracker	GitHub Issues
Discord	pace 4 members
Discussions	GitHub Discussions

Chapter 2

Contributing

Thank you for your interest in contributing to the `pace` ecosystem!

We appreciate your help in making this project better.

Table of Contents

Code of Conduct

Please review and abide by the general Rust Community Code of Conduct when contributing to this project. In the future, we might create our own Code of Conduct and supplement it at this location.

How to Contribute

Reporting Bugs

If you find a bug, please open an issue on GitHub and provide as much detail as possible. Include steps to reproduce the bug and the expected behavior.

Issue and Pull Request labels

Our Issues and Pull Request labels follow the official Rust style:

```
A - Area
C - Category
D - Diagnostic
E - Call for participation
F - Feature
I - Issue e.g. I-crash
```

```
M - Meta
O - Operating systems
P - priorities e.g. P-{low, medium, high, critical}
PG - Project Group
perf - Performance
S - Status e.g. S-{blocked, experimental, inactive}
T - Team relevancy
WG - Working group
```

Suggesting Enhancements

If you have an idea for an enhancement or a new feature, we'd love to hear it! Open an issue on GitHub and describe your suggestion in detail.

Developer's documentation

For more information about developing around `pace`, see the developer's documentation.

License

By contributing to `pace` or any crates contained in this repository, you agree that your contributions will be licensed under:

- **AGPL-3.0-or-later**

Unless you explicitly state otherwise, any contribution intentionally submitted for inclusion in the work by you, as defined in the AGPL-3.0-or-later license, shall be licensed as above, without any additional terms or conditions.

Chapter 3

User Guide

This section provides a comprehensive guide to using the `pace` tool. It covers the following topics:

- Getting Started
- Installation
- Configuration
- Usage Examples
- FAQ

We recommend starting with the Getting Started guide to learn how to install and configure pace. Once you have pace set up, you can explore the other topics to learn how to use the tool effectively.

3.1 Installation

- Official Binaries
  - Stable Releases
- From Source

Official Binaries

Stable Releases

cargo-binstall

```
cargo binstall pace-rs
```

Windows

Scoop

```
scoop bucket add pace https://github.com/pace-rs/pace/
scoop install pace
```

MacOS

Homebrew You can use our tap:

```
brew install pace-rs/homebrew-tap/pace-rs
```

From GitHub You can download the latest stable release versions of pace from the pace release page. These builds are considered stable and releases are made regularly in a controlled manner.

There's both pre-compiled binaries for different platforms as well as the source code available for download.

From source

Beware: This installs the latest development version, which might be unstable.

```
cargo install --git https://github.com/pace-rs/pace.git pace-rs --locked
```

crates.io

```
cargo install pace-rs --locked
```

3.2 Configuration

The configuration file is a TOML file that is used to set up the pace CLI. The configuration file is used to set up the location of the activity file, the storage kind, the category separator, the default priority, and the most recent count, among other settings.

Specification of the Configuration File

A full list of settings and their default values can be found in the Pace Configuration Specification.

Guided Setup Process

You can use the `pace setup config` command to create a configuration file. This command will open an interactive setup process that will ask you a few questions about where to store your data and how to store it. Once you've answered all the questions, pace will create a configuration and an activity log file for you.

Manually Setting up the Configuration File

The configuration file is a TOML file. This means it's a text file that contains key-value pairs. You can use this file to adjust the settings of pace to your liking. Here's an example of a configuration file:

```
[general]
path = "C:\\Users\\YourName\\pace\\activities.pace.toml"
storage-kind = "file"
category-separator = "::"
default-priority = "medium"
most-recent-count = 12
```

If you want to use a different location for the config file, you can specify the location with the `--config` flag. For example:

```
pace --config "C:\\Users\\YourName\\pace\\config.toml" begin "Creating new
↪ assets" --category "MyProject::MyAreaOfFocus" --tags "design,assets"
```

Showing Settings

You can show the current settings with the `pace setup show` command. This will show you the currently loaded settings and the location of the activity file.

3.3 Getting started

Installation

Please refer to the installation guide for instructions on how to install pace.

Quickstart

Pace is a command-line tool, this means it runs in your terminal. To get started, open your terminal. Depending on your operating system, you can use the following commands to open your terminal:

- **Windows**: Press `Win + R`, type `cmd` and press `Enter`.
- **MacOS**: Press `Cmd + Space`, type `terminal` and press `Enter`.
- **Linux**: Press `Ctrl + Alt + T`.

Once you have your terminal open, you can run the following command to check if pace is installed:

```
pace --version
```

If you see a version number, you're good to go! If not, please refer to the installation guide for instructions on how to install pace. Or ask for help in our Discord Chat.

Opening The Documentation

You can open the use documentation website with the following command. This will open the documentation in your default web browser.

```
pace docs
```

If you want to open the developer documentation, you can use the following command:

```
pace docs --dev
```

The documentation for the configuration file can be opened with the following command:

```
pace docs --config
```

Setting up Pace

Before you can start tracking your time, you need to set up pace. This means telling pace where to store your time tracking data. You can do this by running the following command:

```
pace setup config
```

This will open an interactive setup process. You will be asked a few questions about where to store your data and how to store it. Once you've answered all the questions, pace will create a configuration and an activity log file for you.

Tracking Time

Now that you have pace set up, you can start tracking your time. In the next sections, we'll go over the basic workflow of pace. This will show you how to start, pause, and conclude your activities, alongside generating insightful summaries.

3.4 Usage Examples

- Basic Usage - The Pace Workflow

Basic Usage - The Pace Workflow

Principle ”Do one thing and do it good.”

Pace is designed to support this principle. It will allow only one active activity at a time. This means you can only start, pause, resume, or end one activity at a time. This is to ensure that the time tracking is as accurate as possible and you can actually focus on the task at hand. That's why Pace will automatically end the current activity when you start a new one. pace is called a mindful time tracking tool for that reason.

With pace you can easily manage your day. Here's how to start, pause, and conclude your activities, alongside generating insightful summaries.

Starting an Activity

Kick off any task with ease. Simply begin your activity now:

```
pace begin "Creating new assets" --category "MyProject::MyAreaOfFocus" --tags
 ↪  "design,assets"
```

Note: If you don't specify a category or tags, Pace will use reasonable defaults. Which is no tags and the 'Uncategorized' category. You can also specify a starting time with the `--at` flag, your time should be in the format `HH:MM`. If you don't specify a time, pace will use the current time. Using a different starting time than the current time is useful if you forgot to start the activity at the right time, but you want to make sure the time is logged correctly. You cannot start an activity with a starting time in the future.

Switching Activities Move seamlessly to a new task. Pace automatically concludes the previous one:

```
pace begin "Reviewing PR #43" --category "MyOtherProject::MyOtherAreaOfFocus"
 ↪  --tags "web,design"
```

Ending an Activity

Wrap up your current task:

```
pace end
```

Note: When ending an activity, you can also specify the ending time with the `--at` flag, your time should be in the format `HH:MM`. If you don't specify a time, Pace will use the current time. Using a different ending time than the current time is useful if you forgot to end the activity at the right time, but you want to make sure the time is logged correctly. You cannot end an activity with an ending time in the future.

Pausing an Activity

Need a break or a shift in focus? Pause your current activity:

```
pace hold
```

And resume it when you're ready:

```
pace resume
```

Note: When resuming an activity, you can also specify the resuming time with the `--at` flag, your time should be in the format `HH:MM`. If you don't specify a time, Pace will use the current time. Using a different resuming time than the current time is useful if you forgot to resume the activity at the right time, but you want to make sure the time is logged correctly. You cannot resume an activity with a resume time in the future.

Showing the Current Activity

You can show the current activity with the `pace now` command. This will show you the current activity, if there is one.

Gain Insights with Summaries

You can generate summaries of your activities with the `pace reflect` command. It will show you a summary of your activities for the current day, week, or month. You can also specify a custom date range to reflect. If you want to see a summary of your activities for the current day, you can just run:

```
pace reflect
```

It defaults to the current day. If you want to see a summary of your activities for the current week, you can run:

```
pace reflect --current-week
```

If you want to see a summary of your activities for a specific date, you can run:

```
pace reflect --date "2021-12-31"
```

While you can also specify a date range to reflect upon:

```
pace reflect --from "2021-12-01" --to "2021-12-23"
```

If you only specify the `--from` flag, pace will show you a summary of your activities from the specified date to the current date. If you only specify the `--to` flag, pace will show you a summary of your activities from the beginning of the month to the specified date.

Advanced Functionality

Listing Activities

Listing resumable activities You can list all activities that are resumable with the `pace resume --list` command. This will show you all activities that are ended but which you can resume by creating a new activity with the same contents. Resuming an activity will automatically end all the other and the currently active activities.

Note: In case `pace resume` cannot find any active activity, it will list the last `X` activities that are ended and which you can resume by creating a new activity with the same contents. You can specify the number of activities to list by setting:

```
[general]
most_recent_count = 12 # Default: 9
```

Adjusting Activities

Need to correct a mistake? You can adjust the contents of the currently active activity:

```
pace adjust --category "MyProject::MyAreaOfFocus" --tags "design,assets"
 ↪  --description "Creating new assets" --start "08:00"
```

Note: When adjusting an activity, you can also specify the starting time with the `--start` flag, your time should be in the format `HH:MM`. If you don't specify a time, the time will not be changed. Using a different starting time than the current time is useful if you forgot to start the activity at the right time, but you want to make sure the time is logged correctly.

3.5 Frequently asked questions

What is pace?

`pace` is a mindful productivity tool designed to help you keep track of your activities with ease and intention.

Where can I contact the pace community?

Getting help

If you need help with pace, you can ask for help in our Discord Chat or in our GitHub Discussions forum.

Found a bug?

If you found a bug in pace, please report it in our GitHub Issues tracker.

Contributing

If you want to contribute to pace, please refer to our Contributing Guide.

Chapter 4

Command Line Interface (CLI)

The `pace` command-line interface (CLI) is the primary way to interact with the `pace` application. It provides a set of commands to log activities, review progress, and optimize how you spend your time.

This section of the documentation provides an overview of the `pace` CLI, including its command structure, command reference, and usage examples.

4.1 Command Structure

Command Syntax

The Pace CLI uses a simple and intuitive syntax for executing commands. The basic structure of a command is as follows:

```
pace [GLOBAL_OPTIONS] <COMMAND> [OPTIONS] [ARGUMENTS]
```

- `<COMMAND>`: The primary action or task to be performed, such as `begin`, `end`, `hold`, `resume` or `reflect`.
- `[GLOBAL_OPTIONS]`/`[OPTIONS]`: Optional flags or settings that modify the behavior of the command. Options are typically preceded by a hyphen (`-`) or double hyphen (`--`).

- `[ARGUMENTS]`: Additional information or data required to complete the command, such as activity descriptions, time ranges, or category names.

Subcommands

The Pace CLI organizes its functionality into subcommands, each serving a specific purpose or task. For example, the `begin` command is used to start tracking time for an activity, while the `reflect` command provides insights on your activities.

For a complete list of available commands, you can run:

```
pace help
```

Options and Flags

Options and flags are used to modify the behavior of commands or provide additional information. They are typically specified using a hyphen (`-`) or double hyphen (`--`) followed by a keyword or abbreviation.

For example, the `--category`/`-c` option can be used to specify the category of an activity, while the `--tags`/`-t` option can be used to add tags to an activity.

You can view the available options and flags for each command by running:

```
pace help <COMMAND>

# or within the command itself using the --help flag
pace <COMMAND> --help
```

Argument Formats

Arguments are additional pieces of information or data required to complete a command. They can take various formats, such as:

- **Activity descriptions**: Text that describes the activity being logged. For example, `"Writing blog post"` or `"Debugging code"`. Descriptions are mandatory for the `begin` command. For other commands, they are optional and can be specified using the `--description`/`-d` option.
- **Time**: A start or end time for an activity. For example, `"9:00"` or `"24:00"`. Where applicable, time can be specified using the `--at`/`-a` option.
- **Category names**: The name of a category to which an activity belongs. For example, `"Development"` or `"Writing"`. Where applicable, category names can be specified using the `--category`/`-c` option.
- **Tags**: Keywords or labels that provide additional context for an activity. For example, `"important"` or `"urgent"`. Where applicable, tags can be specified using the `--tags`/`-t` option. They can be specified as a comma-separated list, such as `"important,urgent"`.

4.2 Command Reference

- List of Commands
- Command Descriptions
  - adjust
  - begin
  - end
  - hold

List of Commands

```
Commands:
  adjust   □ Adjust the details of an activity, such as its category,
↪  description, or tags [aliases: a]
  begin    □ Starts tracking time for an activity [aliases: b]
  end      □ Stops time tracking for the most recent or all activities
↪  [aliases: e]
  hold     □ Pauses the time tracking for the most recent active activity
↪  [aliases: h]
  now      □ Shows you at a glance what you're currently tracking [aliases: n]
  resume   □ Resumes a previously paused activity, allowing you to continue
↪  where you left off [aliases: r]
  reflect   □ Get sophisticated insights on your activities [aliases: rev]
  setup    □ Set up a pace configuration, a new project, or generate shell
↪  completions [aliases: s]
  docs     □ Open the online documentation for pace [aliases: d]
  help     Print this message or the help of the given subcommand(s)
```

Command Descriptions

adjust

Adjust the current activity's start time, description, category, or tags. This is useful for correcting mistakes or adding more detail to your activities.

Usage: `pace adjust --category <Category> --description <Description> --start <Start Time>`

begin

Starts tracking time for the specified task. You can optionally specify a category or project to help organize your tasks.

Usage: `pace begin "Design Work" --category "Freelance" --start 10:00`

end

Stops time tracking for all tasks, marking them as completed or finished for the day.

Usage: `pace end --end 11:30`

hold

Pauses the time tracking for the specified task. This is useful for taking breaks without ending the task.

Usage: `pace hold --reason <Reason>`

now

Displays the currently running task, showing you at a glance what you're currently tracking.

Usage: `pace now`

resume

Resumes time tracking for a previously paused task, allowing you to continue where you left off.

Usage: `pace resume` or `pace resume --list`

reflect

Gain insight in your activities and tasks. You can specify the time frame for daily, weekly, or monthly insights.

Usage: `pace reflect --last-week` or `pace reflect --from 2024-02-10 --to 2024-03-06` or `pace reflect --today -o json -e ./data/data.json`

setup

Create configuration files for pace, including the main configuration file and any additional settings. This is useful for setting up pace for the first time or when you need to change your settings. You can also generate shell completions for your shell of choice. And generate a project configuration file.

Usage: `pace setup config` or `pace setup completions`

docs

Opens the documentations for users, developers, and the configuration file in your default browser.

Usage: `pace docs` or `pace docs --dev` or `pace docs --config`

help

Prints a help message or the help of the given subcommand(s).

Usage: `pace help` or `pace help begin` or `pace help adjust`

Options and Flags

Only available on `adjust`

- **--override-tags**: Overrides the current tags with the specified tags.

  Usage: `--override-tags "design,assets"` or `--override-tags "design,assets,client-x"`
- **--start**: Specifies the start time of the activity.

  Usage: `--start 10:00` or `--start 10:00:00`

`adjust`, `begin`, `end`, `hold`, `resume`

- **--category**: Specifies the category of the activity.

  Usage: `--category "Freelance"` or `--category "Freelance::Design"`
- **--description**: Specifies the description of the activity.

  Usage: `--description "Design Work"` or `--description "Design Work for Client X"`
- **--at**: Specifies the start, end or resuming time of the activity.

  Usage: `--at 10:00` or `--at 10:00:00`
- **--tags**: Specifies the tags for the activity, they will be deduplicated.

  Usage: `--tags "design,assets"` or `--tags "design,assets,client-x"`

Only available on `resume`

- **--reason**: Specifies the reason for holding the activity.

  Usage: `--reason "Lunch Break"` or `--reason "Meeting"`
- **--list**: Lists all activities that are resumable.

  Usage: `--list`

Only available on `reflect`

```
Options:
  -a, --activity-kind <Activity Kind>  Filter by activity kind (e.g.,
↪  activity, task)
  -c, --category <Category>            Filter by category name, wildcard
↪  supported
      --case-sensitive                 Case sensitive category filter
  -o, --output-format <Output Format>  Specify output format (e.g., text,
↪  markdown, pdf) [possible values: console, json, html, csv, markdown,
↪  plain-text]
  -e, --export-file <Export File>      Export the reflection report to a
↪  specified file
```

```
  -h, --help                           Print help
  -V, --version                        Print version

Flags for specifying time periods:
      --today          Show the reflection for the current day
      --yesterday      Show the reflection for the previous day
      --current-week   Show the reflection for the current week
      --last-week      Show the reflection for the previous week
      --current-month  Show the reflection for the current month
      --last-month     Show the reflection for the previous month

Date flags for specifying custom date ranges or specific dates:
      --date <Specific Date>  Show the reflection for a specific date,
 ↪  mutually exclusive with `from` and `to`. Format: YYYY-MM-DD
      --from <Start Date>     Start date for the reflection period. Format:
 ↪  YYYY-MM-DD
      --to <End Date>         End date for the reflection period. Format:
 ↪  YYYY-MM-DD

Expensive flags for detailed insights:
      --detailed         Include detailed time logs in the reflection
      --comparative      Enable comparative insights against a previous period
      --recommendations  Enable personalized recommendations based on
 ↪  reflection data
```

Only available on docs

- **--dev**: Opens the developer documentation.

  Usage: `--dev`

- **--config**: Opens the configuration documentation.

  Usage: `--config`

Usage Syntax

adjust

```
Usage: pace adjust <--category <Category>|--description <Description>|--start
 ↪  <Start Time>|--tags <Tags>|--override-tags>
```

begin

```
Usage: pace begin [OPTIONS] <Activity Description>
```

end

```
Usage: pace end [OPTIONS]
```

hold

```
Usage: pace hold [OPTIONS]
```

now

```
Usage: pace now
```

resume

```
Usage: pace resume [OPTIONS]
```

reflect

```
pace reflect [OPTIONS]
```

setup

```
Usage: pace setup <config|completions|show>
```

docs

```
Usage: pace docs [--dev|--config]
```

help

```
Usage: pace help [COMMAND]
```

4.3 Completions

Pace can generate shell completions for `bash`, `zsh`, `fish`, and `powershell`. To generate completions, run the following command:

```
pace setup completions <SHELL>
```

Replace `<SHELL>` with the name of the shell you are using. For example, to generate completions for `bash`, run the following command:

```
pace setup completions bash
```

Chapter 5

Data Management

This section covers the data management features of pace, including activity log storage, importing and exporting data, and editing activity logs.

- Activity Log Storage
- Editing Activity Logs

5.1 Activity Log Storage

The activity log is the central data structure in pace. It is a record of all activities, intermissions, and tasks that you have tracked. This document describes how pace stores and manages activity logs.

Storage Format

When using `pace` offline the activity log is stored in a file in the TOML format. The file is named `activities.pace.toml` and is located in the pace data directory. The pace data directory is platform-specific and is typically located in the user's home directory. The file is a list of activity log entries, each of which is a TOML table.

Here is an example of an activity log entry with an activity and an intermission:

```toml
[01HQR3QRXEYK51T6N0D85G5668]
description = "My Task"
begin = "2024-02-28T16:01:13"
end = "2024-02-28T16:02:40"
duration = 87
kind = "activity"
tags = ["test", "my", "lawn"]
status = "ended"

[01HQR3RTH6XJKMW4PFD2XFNK71]
description = "My Task"
begin = "2024-02-28T16:01:48"
end = "2024-02-28T16:02:17"
duration = 29
kind = "intermission"
parent-id = "01HQR3QRXEYK51T6N0D85G5668"
status = "ended"
```

The activity log entry is identified by a unique identifier. The identifier is a ULID (Universally Unique Lexicographically Sortable Identifier). The ULID is used to uniquely identify the activity log entry and to establish relationships between activity log entries.

Storage Location

The pace data directory is platform-specific and is typically located in the user's home directory. The pace data directory is used to store the activity log and other pace data. The pace data directory is determined by the following rules:

- On Linux, the pace data directory is `$XDG_DATA_HOME/pace` or `$HOME/.local/share/pace`.
- On Windows, the pace data directory is `{FOLDERID_LocalAppData}/pace`.
- On MacOS, the pace data directory is `$HOME/Library/Application Support/org.pace-rs.pace`.

The pace data directory is used to store the activity log and other pace data. The activity log is stored in the file `activities.pace.toml` in the pace data directory.

Pace also uses a configuration directory to store configuration files. The configuration directory is platform-specific and is typically located in the user's home directory. The configuration directory is determined by the following rules:

- On Linux, the configuration directory is `$XDG_CONFIG_HOME/pace` or `$HOME/.config/pace/pace.toml`.
- On Windows, the configuration directory is `{FOLDERID_RoamingAppData}/pace/config/pace.toml`.
- On MacOS, the configuration directory is `$HOME/Library/Application Support/org.pace-rs.pace/config/pace.toml`.

The configuration directory is used to store configuration files. The configuration file is named `pace.toml` and is located in the configuration directory.

PACE_HOME

You can override the pace data directory and the configuration directory by setting the `PACE_HOME` environment variable. The `PACE_HOME` environment variable is used to override the default pace data directory and the configuration directory.

5.2 Editing Activity Logs

`pace` currently doesn't provide a lot of commands for editing activity logs. However, you can edit the activity log file directly using a text editor. The activity log file is named `activities.pace.toml` and stored in the TOML format. Please refer to the Activity Log Storage document for more information about the storage format and location of the activity log file.

NOTE: The plan is to provide a web-interface, synchronization with cloud storage, and a mobile and desktop application for managing activity logs, tasks and even projects in the future. So it is unlikely, that we will provide a lot of commands for editing activity logs in the command line interface.

Chapter 6

Time Tracking Concepts

This section covers the time tracking concepts of pace.

6.1 Activity, Intermission, Session

This document explains the concepts of activity, intermission, and session in pace.

Activity

An activity is a period of time during which you are working on something. It may be a task, a project, or any other kind of work or break.

An activity always has a GUID, a start time, an end time and a description. The GUID is a unique identifier for the activity. The start time is the time when you start working on the activity, and the end time is the time when you stop working on the activity. A description is a short text that describes the activity.

Activities may have a status, such as `active`, `held`, `finished`, et cetera. The status of an activity may change over time. Activities can also have categories, tags, and other optional metadata.

An activity may be interrupted by one or many intermissions. An activity may also be part of a session.

Intermission

An intermission is a period of time during which you are not working on anything. It may be a break, a rest, or any other kind of pause.

An intermission is a special kind of activity. So overall it has the same properties as an activity. An intermission always has a GUID, a start time, an end time and a reason. The GUID is a unique identifier for the intermission. The start time is the time when you start the intermission, and the end time is the time when you stop the intermission. An intermission may interrupt an activity. An intermission may also be part of a session.

In addition to the properties of an activity, an intermission has a reason which internally is a description of the intermission. Also an intermission has a `parent-id` field which is the GUID of the activity that the intermission interrupts.

Session

A session is a period of time during which you are working on a series of activities that can be interrupted by intermissions. It may be a work session, a study session, or any other kind of session. A session is basically a list of activities and intermissions that are combined under the same category and description.

A session has no unique identifier and not really an own start or end time, but it inherits the duration of the activities and intermissions that are part of the session. A session may have some optional metadata, such as a description and other properties like duration amount and total duration lengths.

6.2 Activity Status and Lifecycle

This document explains the concepts of activity status and lifecycle in pace.

Activity Status

An activity may have a status, such as `active`, `held`, `finished`, et cetera.

An activity may have the following statuses:

- `active`
- `ended`
- `inactive`
- `held`

Note: In the future there may also be `archived`, `unarchived` and `deleted` statuses. But for now, these are not implemented.

Activity Lifecycle

When an activity is created, it is in the `inactive` state. As we are always calling a wrapper function to begin an activity, the activity is automatically set to `active` when it is created.

When you start working on the activity, the status remains `active`. When you stop working on the activity with `end` or begin another activity the status changes to `ended`.

When you pause the activity with `hold`, the status changes to `held`. When you resume the activity with `resume`, the status changes back to `active`.

NOTE: There can be only one `active` activity at a time. If you start a new activity, the previous one and all other active activities are automatically concluded. This makes sure that there is no overlap between activities, as there is no way to work on two activities at the same time.

One could say: ”But multi-tasking!” - But that's not how pace works. Pace is about focus and concentration. If you want to track multi-tasking, you can create a new activity for each task and switch between them with `resume`. In the future, we might add a feature to track multi-tasking more explicitly, e.g. by being able to assign multiple tasks to an activity, and switch between these tasks or something similar to that.

Chapter 7

Advanced Features

This section covers advanced features of pace, such as generation of reflections and future ones like billing and invoicing.

- Reflections Generation

7.1 Reflections Generation

This feature is designed to help you generate a reflection of your activities, which can be useful for various purposes, such as:

- Preparing for a meeting with your manager or team.
- Writing a report or article about your work.
- Preparing for a performance review or job interview.
- Reflecting on your work and identifying areas for improvement.
- Sharing your progress with others.
- Keeping a record of your work for future reference.
- and more.

A reflection is generated based on the activities you have logged, and it can be customized to include different types of information, such as:

- A summary of your activities, including the time spent on each one.
- A breakdown of your activities by category, project, or other criteria.

Output can be done to different formats, such as plain text, markdown, HTML, or PDF, and it can be customized to include different types of information, such as:

- Recommendations for future work.
- Highlights of your achievements.
- Areas for improvement.

An important setting for reflection generation is your choice of a time period, you can chose from a variety of options, such as:

- The current day.
- Yesterday.
- A specific date range.
- The current week, month, or year.
- The entire history of your activities.

The reflection generation feature is designed to be flexible and customizable, so you can generate a reflection that meets your specific needs and preferences.

Usage

To generate a reflection, you can use the `pace reflect` command, which has various options to customize the output. For example, you can specify the time period, format, and other settings. The generated reflection can be saved to a file or printed to the console.

Templates

You can customize the appearance and content of the generated reflection by using templates. Templates are written in a simple markup language that allows you to specify the structure and formatting of the output. You can create your own templates or use predefined ones.

pace uses the Tera template engine, which provides a powerful and flexible way to create templates. You can use variables, loops, conditionals, and other features to control the content and appearance of the output. You can also use filters and functions to transform and format the data. It is inspired by Jinja2 and Django templates.

Chapter 8

Integration and Extensibility

This section covers the various ways in which pace can be integrated with other systems and extended to suit specific needs.

8.1 Database Integration

pace is designed to be an offline-first application, meaning that it does not require an internet connection to function. However, it is possible to integrate pace with a database to enable features such as data synchronization across multiple devices, data backup, and data recovery.

Note: This feature is not yet implemented. It is planned for a future release. So far, pace only supports local storage of activity logs in a file-based format. This section provides an overview of the planned database integration feature and how it will work.

Vision

The vision for database integration is to provide a way to store activity logs in a database, such as SQLite, PostgreSQL, or MySQL. This will enable the following features:

- **Data Synchronization**: Activity logs can be synchronized across multiple devices, so that you can access your data from anywhere and keep it up to date.
- **Data Backup**: Activity logs can be backed up to a remote server or cloud storage service, so that you can recover your data in case of loss or corruption.
- **Data Recovery**: Activity logs can be recovered from a backup, so that you can restore your data to a previous state.
- **Data Sharing**: Activity logs can be shared with other users, so that you can collaborate on projects and activities.
- **Data Security**: Activity logs can be encrypted and protected with access controls, so that your data is safe from unauthorized access.
- **Data Privacy**: Activity logs can be stored in compliance with privacy regulations, so that your data is protected and respected.
- **Data Integrity**: Activity logs can be verified and validated, so that you can trust that your data is accurate and reliable.

Implementation

The implementation of database integration will involve the following components:

- **Database Adapter**: The `Storage` trait will be implemented for a database storage. We will start out with SQLite to provide a simple and portable solution. The `Storage` trait already has a `FileStorage` and `InMemoryStorage` implementation, so we will add a `DatabaseStorage` implementation. The `DatabaseStorage` will be a thin wrapper around the `diesel` ORM, providing a simple interface for storing and retrieving activity logs in a database. This adapter will support different database backends, such as SQLite, PostgreSQL, and MySQL.
- **Data Synchronization**: A synchronization service will be developed to enable data synchronization across multiple devices. This service will use a client-server architecture, with a server component for storing and managing activity logs, and a client component for interacting with the server. We will use `axum` and `tokio` to develop the server and client components.
- **Data Backup and Recovery**: A backup and recovery service will be developed to enable data backup to a remote server or cloud storage service, and data recovery from a backup. This service will use a client-server architecture, similar to the synchronization service. We will use `reqwest` and `opendal` to interact with remote servers and cloud storage services.
- **Data Sharing**: A sharing service will be developed to enable sharing of activity logs with other users. For this feature, we will use a combination of the synchronization service and the backup and recovery service. We will also implement access controls to manage sharing permissions. This will be a more advanced feature, and we will build the team collaboration feature on top of it.
- **Data Security and Privacy**: Security and privacy features will be implemented to protect activity logs from unauthorized access and ensure compliance with privacy regulations.
- **Data Integrity**: Integrity features will be implemented to verify and validate activity logs, and to ensure that they are accurate and reliable.

Chapter 9

Best Practices and Tips

This section provides best practices and tips for using pace effectively and efficiently.