1. Comprehensive Example

Ado is a simple command-line todo system. Ado lets you create notes, projects and tasks, and organize them into portfolios corresponding to the different areas of responsibility in your life. Ado also lets you track the time you spend on tasks so you can better understand your commitments and manage your time better. Because ado uses Dexy as its reporting engine, you can create powerful reports (or use the defaults) to really understand and plan how you spend your productive time.

Ado uses a sqlite file to store data. By default, this sqlite file is named ado.sqlite3 and is stored in a `.ado` directory in the user's home directory, e.g. /home/ana/.ado

You can override either of these locations by setting environment constants:

```
$ export ADO_DIR=`pwd`
$ export ADO_DB_FILE=example.sqlite
```

The `setup` command creates the `ADO_DIR` directory (if necessary) and creates the tables in the database:

```
$ ado setup
```

You can check that this has worked by using the `sqlite3` command directly:

```
$ sqlite3 -line $ADO_DIR/$ADO_DB_FILE ".dump"
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE Portfolio (id INTEGER PRIMARY KEY AUTOINCREMENT, alias text, created_at timestamp, description text, name text, archived_at timestamp);
CREATE TABLE Project (id INTEGER PRIMARY KEY AUTOINCREMENT, completed_at timestamp, parent_project_id integer, portfolio_id integer, description text,
CREATE TABLE Task (id INTEGER PRIMARY KEY AUTOINCREMENT, waiting_for_task_id integer, description text, completed_at timestamp, recipe_id integer, est
CREATE TABLE Note (id INTEGER PRIMARY KEY AUTOINCREMENT, note text, archived_at timestamp, created_at timestamp, linked_to_id integer, linked_to_type
CREATE TABLE Timer (id INTEGER PRIMARY KEY AUTOINCREMENT, finished_at timestamp, started_at timestamp, description text, task_id integer);
CREATE TABLE Recipe (id INTEGER PRIMARY KEY AUTOINCREMENT, frequency integer, portfolio_id integer, description text, context text, based_on integer,
CREATE TABLE Metric (id INTEGER PRIMARY KEY AUTOINCREMENT, archived_at timestamp, created_at timestamp, description text, name text);
CREATE TABLE MetricData (id INTEGER PRIMARY KEY AUTOINCREMENT, created_at timestamp, metric_id integer, value float);
COMMIT;
```

To manage your todo items, you can create notes, tasks and projects.

1.1. Notes.

A note can stand alone, or it can be assigned to a task or a project. Create notes by using the `note` command.

```
$ ado note -note "do some stuff"
Created note 1
$ echo "do some stuff" | ado note
Created note 2
```

Each note is assigned an ID number, which is printed to STDOUT when the note is created. You can list notes with the `notes` command.

```
$ ado notes
Note 0001. do some stuff                 (0 days old)
Note 0002. do some stuff                 (0 days old)
```

This displays the note id, the first 30 characters of the note, and how old the note is (in days). You can use the `show` command to show the full text of a note:

```
$ ado show -n 1
do some stuff
```

Notes can be deleted using the `delete` command, which removes them from the database:

```
$ ado delete -n 1
```

```
$ ado notes
Note 0002. do some stuff                 (0 days old)
```

You can also use the `archive` command which keeps the content in the database, but removes it from lists and reports.

```
$ ado archive -n 2
```

```
$ ado notes
No notes found.
```

1.2. **Tasks.** Tasks have a name and a context, and an optional description.

Create a task:

```
$ ado task -name "This is a task" -context "@home" -due "2012-02-02" -p 1
Created task 1
$ ado task -name "This is another task" -context "@car" -description "This is a more detailed description of this task." -p 1
Created task 2
```

And to list all tasks:

```
$ ado tasks
Task 0001) This is a task                                    @home          [This is a project.]  (due in -340 days)
Task 0002) This is another task                              @car           [This is a project.]
$ ado tasks -by context
Task 0002) This is another task                              @car           [This is a project.]
Task 0001) This is a task                                    @home          [This is a project.]  (due in -340 days)
$ ado tasks -by due_at
Task 0002) This is another task                              @car           [This is a project.]
Task 0001) This is a task                                    @home          [This is a project.]  (due in -340 days)
```

To show detail on a specific task:

```
$ ado show -t 1
This is a task @home
Created Sunday 06 January 2013
Due Thursday 02 February 2012
```

When you have finished a task, you can mark it as complete:

```
$ ado complete -t 1
Task 1 marked as complete!
```

1.3. **Projects.** To create a new project:

```
$ ado project -name "This is a project." -p 1
1
```

To list all projects:

```
$ ado projects
Project 0001        This is a project.                       (in Example)
```

1.4. **Notes with Projects and Tasks.** Notes can be assigned to projects or tasks, either when they are created, or afterwards.

You can pass a project or task id when creating a new note, use `-p` to pass a project id and `-t` to pass a task id, the new note will be assigned to the project or task in question.

```
$ ado note -note "This is a note in a project" -p 1
Created note 3
Assigned to project 1
```

When you show the project, you see that the note is linked to it:

```
$ ado show -p 1
Project 1: This is a project.
Elapsed days 0, Created at 2013-01-06 15:17:30.679429
Notes for project:
Note 0003. This is a note in a project    (0 days old)
```

You can also use the `assign` command to assign a note to a project or task:

```
$ ado task -name "task with a note" -context "@anywhere" -p 1
Created task 3
$ ado assign -n 3 -t 3
$ ado show -t 3
task with a note @anywhere
Created Sunday 06 January 2013
Notes for task:
Note 0003. This is a note in a project    (0 days old)
```

Let's change the note's content to reflect the fact that it is now a part of a task rather than a project, using the `update` command:

```
$ ado update -n 3 -note "This is a note in a task."
$ ado show -t 3
task with a note @anywhere
Created Sunday 06 January 2013
Notes for task:
Note 0003. This is a note in a task.      (0 days old)
```

You can use the update command to change any attribute of a note, project or task.

1.5. **Workflow.** This section talks about how these elements work together for a GTD-style workflow.

The `inbox` command lists all tasks that aren't assigned to a project, and all notes that aren't assigned to a project or a task:

```
$ ado inbox
No notes in the inbox!
No tasks in the inbox!
```

To process this 'inbox', the `assign` command is used to assign notes to tasks and projects, or tasks to projects. Use the `complete` command to mark tasks and projects as complete.

So, you can create a note or a task any time so that it's in your system, and later you can assign it to a project, or create a task for the note to be attached to.

Tasks have contexts, which traditionally start with the `@` symbol. You can pass the `-by` option with 'context' to the `tasks` command to sort your tasks by context.

The tasks, notes and projects commands also take a 'search' option which lets you find objects that have the search string. The `search` command lets you search across notes, tasks and projects.

2. Commands

2.1. **Archive Command.**

```
======================
Help for 'ado archive'
======================
   Archive the note, project or task specified.

   Arguments:
      n
            [optional, defaults to '-1'] e.g. 'ado archive --n -1'


      p
            [optional, defaults to '-1'] e.g. 'ado archive --p -1'


      t
            [optional, defaults to '-1'] e.g. 'ado archive --t -1'
```

2.2. **Assign Command.**

```
====================
Help for 'ado assign'
====================
   Assign a note to a project or task, or a task to a project.

   Arguments:
      n
            [optional, defaults to '-1'] e.g. 'ado assign --n -1'


      p
            [optional, defaults to '-1'] e.g. 'ado assign --p -1'


      t
            [optional, defaults to '-1'] e.g. 'ado assign --t -1'
```

2.3. Complete Command.

```
======================
Help for 'ado complete'
======================
   Mark the project or task as completed.

   Arguments:
      p
            [optional, defaults to '-1'] e.g. 'ado complete --p -1'


      t
            [optional, defaults to '-1'] e.g. 'ado complete --t -1'
```

2.4. Completion Command.

```
========================
Help for 'ado completion'
========================
   Prints a bash script that can be saved to generate bash autocompletion for ado commands.
```

2.5. Delete Command.

```
====================
Help for 'ado delete'
====================
   Delete the note, project or task specified.

   Arguments:
      n
            [optional, defaults to '-1'] e.g. 'ado delete --n -1'


      p
            [optional, defaults to '-1'] e.g. 'ado delete --p -1'


      t
            [optional, defaults to '-1'] e.g. 'ado delete --t -1'
```

2.6. Dump Command.

```
==================
Help for 'ado dump'
==================
   Dumps your data to console in sqlite format (data only, not structure, so you can preserve data while resetting your db schema).
```

2.7. Help Command.

```
==================
Help for 'ado help'
==================
   Prints this help.

   Arguments:
      on
            [optional, defaults to 'False'] e.g. 'ado help --on False'
```

2.8. Inbox Command.

```
====================
Help for 'ado inbox'
====================
   Lists all notes and tasks that are still in the 'inbox', i.e. not assigned to projects, tasks or other elements.
```

2.9. Load Command.

```
==================
Help for 'ado load'
==================
   Loads a data file previously created by saving the output of 'dump'.

   Arguments:
      filename ,
            [required] e.g. 'ado load --filename <value>'
```

2.10. Note Command.

```
==================
Help for 'ado note'
==================
   Create a new note.

   Arguments:
      note
            [optional, defaults to ''] e.g. 'ado note --note '


      p
            [optional, defaults to '-1'] e.g. 'ado note --p -1'


      t
            [optional, defaults to '-1'] e.g. 'ado note --t -1'
```

2.11. Notes Command.

```
====================
Help for 'ado notes'
====================
   Lists all notes.
```

2.12. Portfolio Command.

```
=======================
Help for 'ado portfolio'
=======================
   Creates a new portfolio.

   Arguments:
      description ,
            [required] e.g. 'ado portfolio --description <value>'


      name ,
            [required] e.g. 'ado portfolio --name <value>'
```

2.13. Portfolios Command.

```
========================
Help for 'ado portfolios'
========================
```

2.14. Project Command.

```
======================
Help for 'ado project'
======================
   Create a new project.

   Arguments:
      description - an optional description for this project
            [optional, defaults to ''] e.g. 'ado project --description '


      name - the name of the project (required),
            [required] e.g. 'ado project --name <value>'


      p - portfolio id for this project (required unless parent project specified)
            [optional, defaults to '-1'] e.g. 'ado project --p -1'
```

```
      parent - parent project id, if this is a subproject
            [optional, defaults to '-1'] e.g. 'ado project --parent -1'
```

2.15. Projects Command.

```
======================
Help for 'ado projects'
======================
   List all projects.
```

2.16. Recipe Command.

```
====================
Help for 'ado recipe'
====================
   Create a new recipe.

   Arguments:
      context - context for this recipe,
            [required] e.g. 'ado recipe --context <value>'


      description - optional description of this recipe
            [optional, defaults to ''] e.g. 'ado recipe --description '


      frequency - how often this recipe should be done (in days)
            [optional, defaults to '-1'] e.g. 'ado recipe --frequency -1'


      name - Name  of this recipe,
            [required] e.g. 'ado recipe --name <value>'


      p - portfolio id,
            [required] e.g. 'ado recipe --p <value>'


      recipe - The instructions for how to perform this recipe.,
            [required] e.g. 'ado recipe --recipe <value>'
```

2.17. Recipes Command.

```
======================
Help for 'ado recipes'
======================
```

2.18. Reset Command.

```
====================
Help for 'ado reset'
====================
   Deletes user dir and recreates database tables. DESTROYS ALL YOUR DATA!
```

2.19. Search Command.

```
====================
Help for 'ado search'
====================
   Lists all items which meet the search criteria.

   Arguments:
      search ,
            [required] e.g. 'ado search --search <value>'
```

2.20. Setup Command.

```
====================
Help for 'ado setup'
====================
   Run this command to initialize all database tables. Can be run multiple times safely.
```

2.21. **Show Command.**

```
==================
Help for 'ado show'
==================
   Print detailed information for a project, task or note.

   Arguments:
      n
            [optional, defaults to '-1'] e.g. 'ado show --n -1'


      p
            [optional, defaults to '-1'] e.g. 'ado show --p -1'


      t
            [optional, defaults to '-1'] e.g. 'ado show --t -1'
```

2.22. **Start Command.**

```
====================
Help for 'ado start'
====================
   Start a timer for a recipe.

   Arguments:
      r ,
            [required] e.g. 'ado start --r <value>'
```

2.23. **Stop Command.**

```
==================
Help for 'ado stop'
==================
   Arguments:
      t
            [optional, defaults to '-1'] e.g. 'ado stop --t -1'
```

2.24. **Task Command.**

```
==================
Help for 'ado task'
==================
   Create a new task.

   Arguments:
      context - The @context in which task can be done,
            [required] e.g. 'ado task --context <value>'


      description - optional longer description for this task
            [optional, defaults to ''] e.g. 'ado task --description '


      due - due date in YYYY-MM-DD format
            [optional, defaults to '-1'] e.g. 'ado task --due -1'


      estimate - estimate of time this will take, in minutes
            [optional, defaults to '-1'] e.g. 'ado task --estimate -1'


      name - The name for this task,
            [required] e.g. 'ado task --name <value>'


      p - project id this task is part of
            [optional, defaults to '-1'] e.g. 'ado task --p -1'


      waiting - the id of another task that must be completed first
            [optional, defaults to '-1'] e.g. 'ado task --waiting -1'


      worktype - type of work this is
            [optional, defaults to 'adhoc'] e.g. 'ado task --worktype adhoc'
```

2.25. Tasks Command.

```
====================
Help for 'ado tasks'
====================
   List all tasks.

   Arguments:
      by
            [optional, defaults to 'id'] e.g. 'ado tasks --by id'


      search
            [optional, defaults to '-1'] e.g. 'ado tasks --search -1'
```

2.26. Tasktime Command.

```
======================
Help for 'ado tasktime'
======================
   Arguments:
      t ,
            [required] e.g. 'ado tasktime --t <value>'
```

2.27. Time Command.

```
==================
Help for 'ado time'
==================
   Starts a timer, optionally give a description and specify the task id you are working on.

   Arguments:
      description
            [optional, defaults to ''] e.g. 'ado time --description '


      t
            [optional, defaults to '-1'] e.g. 'ado time --t -1'
```

2.28. Update Command.

```
====================
Help for 'ado update'
====================
   Update a project, task or note with the supplied kwargs.

   Arguments:
      n
            [optional, defaults to '-1'] e.g. 'ado update --n -1'


      p
            [optional, defaults to '-1'] e.g. 'ado update --p -1'


      r
            [optional, defaults to '-1'] e.g. 'ado update --r -1'


      t
            [optional, defaults to '-1'] e.g. 'ado update --t -1'
```