TodX Documentation

Release a

AdiUltra

July 12, 2016

CHAPTER 1

Welcome to TodX's documentation!

The Stupid Simple Plain ToDo List App

TodX is a simple and to the point ToDo list app which enhances productivity and doesn't come in between you.

1.1 Quick Start

TodX is a great Todo app. But you need to first learn a tool to use it. Learn it below.

For installation, refer to the *Installation* for install instructions.

1.1.1 Basic Commands

These commands are used in the app to do the root or basic actions like open, append etc. These are entered in the `*>` prompt.

New

Command : `new` or `n`

Create a new list by entering title and tags(*optional*).

Open

Command : `open` or `o`

Open a list from previously created lists. The list that is opened with this command is said as *opened*.

Note: Commands marked * work only when a list is open.

Stats

Command : `stats` or `stat`

View wether a list is opened or not.

View *

Command : `view` or `v`

View the Todos of currently opened list.

Append *

Command : `append` or `a`

Append a new Todo in the currently *opened* list.

Mark *

Command : `mark` or `m`

Mark a Todo as done using charcters like `*` or `#`.

Search

Command : `search` or `srch`

Search the data for a search term.

Export

Command : `export` or `exp`

Export the data to a readable text file.

Quit

Command : `quit` or `q`

Close the app. It also saves the notes to the data file.

1.1.2 Entry Modes

These are the prompts used in the app

Prompt	Usual Meaning	Answer Format
`*>`	Basic commands	commands like `open`
`+>`	Entry or Input for Content	Sentence or one word
`#>`	Usually index of Lists etc	Numbers `0`, `2` etc
`?>`	Question Input	`yes` or `no`

1.2 User Guide

This guide helps to provide a extended documentation for using the app.

1.2.1 Installation

Currently Source code is available on github and can be compiled with any modern c++ compiler. although only g++ is tested and recommended.

Note: The users of TC++ should know that thier compiler is older than me, and is supported neither by Borland nor by me. Please switch to g++.

Linux

1. Clone the repo using `git clone https://github.com/arcxon/todx.git` or Download the Source zip.
2. Use the command `g++ -o main.cpp todx.out` to compile the executable.
3. Now the `todx.out` file can be used anywhere. Just place it in your desired folder and execute in the command-line : `./todx.out` to enjoy the app.

Note: You should also copy `help.txt` so that *Help* command works.

Windows

1. Clone the repo using `git clone https://github.com/arcxon/todx.git` or Download the Source zip.
2. Ensure `g++` is installed using the bellow command in the pallete :

   g++ -v

If it prints some strange stuff that relates to version number etc then all is well and good. If not, and it shows command not found, then download and install MinGW.

3. Now, We compile the code :

   g++ main.cpp fabric.cpp -o todx.exe

4. The binary `todx.exe` is ready to be executed. Move it to your desired folder to store the todo's. Don't forget to move the `help.txt`, since the program requires it. Then simply execute `todx.exe` to enjoy it.

Binary Executables

There is a plan to offer stable binary builds for both Linux and Windows. Please Remain Calm and use compilation till then.

1.2.2 Basic Commands

There are various Basic commands in the app. These commands are entered with the *> prompt.

New

Command

`new` or `n`

Function

Adds/Creates a new List.

Explanation

It creates a new List, and when executed asks for the **title** (mandatory) and **tags**.

The Tags can be skipped by entereing *no*.

After creation the list gets added to the Main Lists of List. You can open it by the command *open*.

Note: The current limitation for the number of lists is 20. So You *cannot* create more than 20 lists.

Open

Command

`open` or `o`

Function

This opens a List for further editing

Explanation

It shows the currently available lists, that can be opened. Enter the Number/Index in the prompt to open that list.

After Opening other commands like `append` and `view` can be used on the opened list.

Stats

Command

`stats` or `stat`

Function

Tells if a list is open or not.

Explanation

Tells wether a list is _opened_ or not. Useful before executing commands like `append` and `qdelete`.

View

Command

`view` or `v`

Warning: You can use this command only when a list is open, use `open` to open a list.

Function

View the contents of Opened List.

Explanation

This shows the **Title**, **Tags** and **ToDos** that belong to the currently _opened_ list.

Append

Command

`append` or `a`

Warning: You can use this command only when a list is open, use `open` to open a list.

Function

This adds a Todo to the opened list.

Explanation

This adds a Todo to the currently _opened_ list. The todo is added to the end/last of the list.

Mark

Command

`mark` or `m`

Warning: You can use this command only when a list is open, use `open` to open a list.

Function

This marks status of specified Todo to the input character.

Explanation

This modifies the status, (the thing between `[]`) to the character provided.

AddTag

Command

`addtag` or `addt`

Warning: You can use this command only when a list is open, use `open` to open a list.

Function

This adds a new tag to the opened list.

Explanation

It adds a new Tag to the currently opened list. The tag is added to the end of the tag-list.

Search

Command

`search` or `srch`

Function

This Searches the database for the string in Title, Tags and Todos

Explanation

It searches and displays the Lists which have the given search term in thier **Title**, **tag** or **ToDo**.

Hacks

1. You can surround the search term with spaces to limit search to the exact word.

```
" cool " instead of "cool"
```

Save

Command

`save` or `s`

Function

This saves the data to the binary data file.

Explanation

It saves the contents of Lists in internal binary file. The extension is `.tdx`. This file can be reread by the program.

Note: It is often prefered to not use `save` command since, The `quit` command already saves the data before exiting. It is meant for developing only.

Export

Command

`export` or `exp`

Function

This saves the data to a readable text file.

Explanation

It saves the contents of Lists in an external human readable text file. The extension is `.tdexp`. This file cannot be reread by the program.

Warning: If a file exists by the same name It is **Over Written**, Please later save your exported files to different location.

Delete

Command

`delete` or `del`

Function

This deletes the List/Todo/Tag specified

Explanation

It deletes the **Title** , **Todo** or **Tag** specified by the user. It takes input of index and asks confirmation (*yes*/*no*) before deleting.

Warning: This command can not be reversed. After deletion, the data is lost. Please use the command carefully.

Qdelete

Command

`qdelete` or `qdel`

Warning: You can use this command only when a list is open, use `open` to open a list.

Function

This deletes Todo specified

Explanation

Like *delete* but skips the step for asking for **Todo**. Used to **Quickly** (thus qdelete) Delete a Todo.

It deletes the **Todo** specified by the user. It takes input of index and asks confirmation (*yes*/*no*) before deleting.

Warning: This command can not be reversed. After deletion, the data is lost. Please use the command carefully.

Help

Command

`help` or `h`

Function

Displays Help file.

Explanation

Displays the commonly used commands and the lisence info with link to the documentation.

Warning: For Help to display, there should be `help.txt` in the directory of executable.

Quit

Command

`quit` or `q`

Function

This saves the data to the binary data file, and exits the program.

Explanation

It saves the contents of Lists in internal binary file, and exits the program. The extension is `.tdx`. This file can be reread by the program.

1.2.3 Entry Modes

The following Entry modes are used throughout the program

Prompt	Color	Usual Meaning	Answer Format
`*>`	yellow	Basic commands	commands like `open`
`+>`	green	Entry or Input for Content	Sentence or one word
`#>`	blue	Usually index of Lists etc	Numbers `0`, `2` etc
`?>`	red	Question Input	`yes` or `no`

1.2.4 Data Concepts

The understanding of data structures is not necessary to acomplish trivial daily usage tasks. But if you truly want to understand what the program refers to when it says Todo or List then you will find them here.

Data formats/structures used in the program are :

ToDo

This is the basic unit of the program. It stores the sentence that you would like to be reminded of with a status.

Status

This represents the *status* or what is the position of your Todo. This is set and maintained by the user using the *Mark* command. It is a charater, Any character on your keyboard. Usually it is `*` or `o` for done. `!` for important, `?` for unknown. and = or − for doing.

But here the app shines the most. Rather than using these, You can use your own markers for status. It could be a letter a number etc.

Lists

Lists are the collection of similar *ToDo* s that are grouped together. Every List has a **Title**, A group of **Tags** and a list of ToDo's.

During Creation, You can specify the Title(*mandatory*) and Tags(*Optional*). When Created a list can be opened by using *Open* command.

To add a new ToDo to an *open* List, use the *Append* command.

Warning: The developer documentation is **Not Complete**. the User Guide, although, is usable.

1.3 Developer's Guide

This documentation is for the people who want to contribute to TodX by improving the codebase. Or to those who would like to know how things happen in the App.

1.3.1 Installation from Source

This step is critical for helping development in TodX. It will allow you to get the source-code, publish changes, and ask them to be included.

Getting Source

If you have already thought of contributing by editing the code, It is much better to fork the Repo and then clone it.

```
$ git clone https://github.com/{user name}/todx
```

where `{user name}` should be your user name.

If you want the repo for just seeing the code.

```
$ git clone https://github.com/arcxon/todx
```

Compiling and running

You should have `make` and `g++` installed, although I don't think the compilation is dependent on platform since the code uses mostly standard c++. To compile the source use `$ make`. To run it use `make run`. As given below

```
$ make
$ make run
```

Note: Makefile already has a command for `make clean` so run it to remove the object files.

After successfull installation and running you can try out the program.

The binary and object files are located at *build* directory.

To clean/remove build directory eith object and executable binary, run

```
$ make clean
```

Building the Docs

Note: For Building Docs, You need *pip* installed.

After installation, you might want to build the docs, for this you need to install the following:

Sphinx

Install the Documentation generator.

```
$ pip install sphinx sphinx-autobuild
```

Sphinx RTD theme

Install the cool theme for Documentation.

```
$ pip install sphinx_rtd_theme
```

Now we can build the docs. Switch to the folder/Directory `docs` in the repo and execute

```
$ make html
```

This will build the docs. You can visit the index file at `docs\_build\html\index.html`, open it in your browser and enjoy the docs.

1.3.2 Fabric

Note: Before proceding make sure you have read the *Data Concepts*

This is the header file which contains the data members used in the program.

They are as follows:

Todo

Todo is the basic unit of the application it contains the following public data members

- `content` a character array of 200 char elements.
- `status` user defined char that stores the *Status*.

List

It is the collection of Todos. Most operations are performed throught this class. All data members and functions are public.

It contains the following data members -

- `Todo list[20]`
- `char title[100]`
- `char tags[10][20]`
- **`int _listIndex`** Variable to count the filling of list
- **`int _tagIndex`** Variable to count the filling of tags
- `int _hasTags`

It contains the following functions -

- **`List()`** Asks for title(required) and tags(can be skipped).
- **`List(char Title[100])`** Initialize the list with title
- **`void enter()`** Enter the info given by user
- **`void addTag(char Tag[20])`** Add the tag to Tags
- **`void setTitle(char Title[100])`** Set the Title given
- **`void view()`** View the list's title tags and #content
- **`void indexView()`** View the contents with the index
- **`void tagView()`** View only Title and tags
- **`void todoView(int)`** View the todo of index passed
- **`void tagIndexView()`** View the Title and Tags with the index
- **`void append()`** append a new ToDo in the list
- **`void removeTodo(int)`** delete the list at index passed
- **`void removeTag(int)`** delete the tag at index passed
- **`void changeStatus(int index, char status)`** Changes Status of the Item of index to status