README.markdown · master · David Beniamine / todo.txt-vim · GitLab

gitlab.com/dbeniamine/todo.txt-vim/-/blob/master/README.markdown

Todo.txt-vim

Hierarchical sort

This fork provides a hierarchical sorting function designed to do by project and/or by context sorts and a priority sort.

- `<LocalLeader>sc` : Sort the file by context then by priority
- `<LocalLeader>scp` : Sort the file by context, project then by priority
- `<LocalLeader>sp` : Sort the file by project then by priority
- `<LocalLeader>spc` : Sort the file by project, context then by priority

For more information on the available flags see `help :sort`

Recurrence

By adding a `rec:` tag to your task, when you complete (`<LocalLeader>x`) or postpone (`<LocalLeader>p`) the task, a new recurrence will be created due after the specified amount of time.

The format is: `rec:[+][count][d|w|m|y]`

Where: d = days, w = weeks, m = months, y = years The optional `+` specifies strict recurrence (see below)

Examples: * `rec:2w` - Recurs two weeks after the task is completed. * `rec:3d` - Recurs three days after the task is completed. * `rec:+1w` - Recurs one week from the due date (strict)

Sort

- `<LocalLeader>s` : Sort the file by priority
- `<LocalLeader>s+` : Sort the file on `+Projects`
- `<LocalLeader>s@` : Sort the file on `@Contexts`
- `<LocalLeader>sc` : Sort the file by context then by priority
- `<LocalLeader>scp` : Sort the file by context, project then by priority
- `<LocalLeader>sp` : Sort the file by project then by priority
- `<LocalLeader>spc` : Sort the file by project, context then by priority
- `<LocalLeader>sd` : Sort the file on due dates. Entries with a due date appear sorted by at the beginning of the file, completed tasks are moved to the bottom and the rest of the file is not modified.

Priorities

- `<LocalLeader>j` : Lower the priority of the current line
- `<LocalLeader>k` : Increase the priority of the current line
- `<LocalLeader>a` : Add the priority (A) to the current line
- `<LocalLeader>b` : Add the priority (B) to the current line
- `<LocalLeader>c` : Add the priority (C) to the current line

Dates

- `<LocalLeader>d` : Insert the current date
- `<LocalLeader>p` : Postpone the due date (accepts a count)
- `<LocalLeader>P` : Decrement the due date (accepts a count)
- `date<tab>` : (Insert mode) Insert the current date
- `due:` : (Insert mode) Insert `due:` followed by the current date
- `DUE:` : (Insert mode) Insert `DUE:` followed by the current date

If you would like the creation date (today) prefixed on new lines, add the following to your vimrc:

```
let g:Todo_txt_prefix_creation_date=1
```

With insert mode maps on, typing `date<Tab>` or `due:` can feel like glitches This is because vim wait for mappings before inserting the words to the buffer. To prevent the glitches, abbreviations can be used instead of mappings. To turn it on, add the following to your vimrc:

```
let g:TodoTxtUseAbbrevInsertMode=1
```

Abbreviations uses word separator to expand the abbreviations, thus `<Tab>` is unavailable on abbreviations. Turning abbreviations mode will change `date<Tab>` mapping into `date:`. The resulting abbreviations would be:

- `date:` : (Insert mode) Insert the current date
- `due:` : (Insert mode) Insert `due:` followed by the current date
- `DUE:` : (Insert mode) Insert `DUE:` followed by the current date

Done

- `<LocalLeader>x` : Toggle mark task as done (inserts or remove current
- date as completion date)
- `<LocalLeader>C` : Toggle mark task cancelled
- `<LocalLeader>X` : Mark all tasks as completed
- `<LocalLeader>D` : Move completed tasks to done file, see TodoTxt Files

Format

`<LocalLeader>ff` : Try to fix todo.txt format