How I use org-mode

Yann Esposito

[2020-10-29 Thu]

In this article I'll try to give an overview of my current use of org mode. I use org mode for:

- tasks management & time tracking
- writing documents (articles, book, etc...)
- note taking ; which I consider slightly different from just writing documents

TL;DR:

- `SPC y a` ☒ Show agenda view for today
- `SPC X` ☒ Capture a new task, write a description, then `C-c C-c`, save that in `tracker.org` (or `inbox.org` depending of the capture template)
- `SPC n o` ☒ jump to the current time tracked tasks
- `SPC m c o` ☒ stop the clock on that task; if you capture a new time tracking tasks you don't need to clock-out
- `SPC y o r` ☒ `org-refile`, meaning move that task somewhere else
- `SPC q` ☒ add/remove tags to that task

In this article I would like to share a tool that was a real life changer to me: org mode.

In my opinion emacs is worth learning just for org-mode. This is by far the best solution I ever used to manage my tasks. I tried a lot of differents tools before it, and this is the only one I really stick with. It is so versatile that it can adapt to your very specific needs.

The major difficulty faced by tasks management application is the wrong level of complexity facing the user. This is a very hard problem to tackle.

If your system is too simple, the users will not be able to manage the tasks how they would like. If your system is too complex, the user will be faced with too much details. So most successful systems have a way to adapt their apparent complexity to the need of their users. And org mode is exactly like that. Not only starting with org mode can be extremely simple but also there are mostly no complexity limit.

Either org mode already handle one of your need, or most of the time you will find a package to fulfill your need. And if not, it is easy to write your own.

Here is the result of a few years of improving my use of org mode. Today I can say that org mode is part of my day to day life. I still invest a bit of time to improve minor details of my workflow time to time. But now my daily workflow is mostly stable. So I think I can share it.

Overview

daily routine

The first thing I do in the morning is to open `org-agenda` view for today. It shows me the tasks planned for today. What are urgent tasks, deadlines, tasks that have deadlines in a few days, etc... I also have a glimpse of my habits, tasks that I should start at some hour in the day, etc...

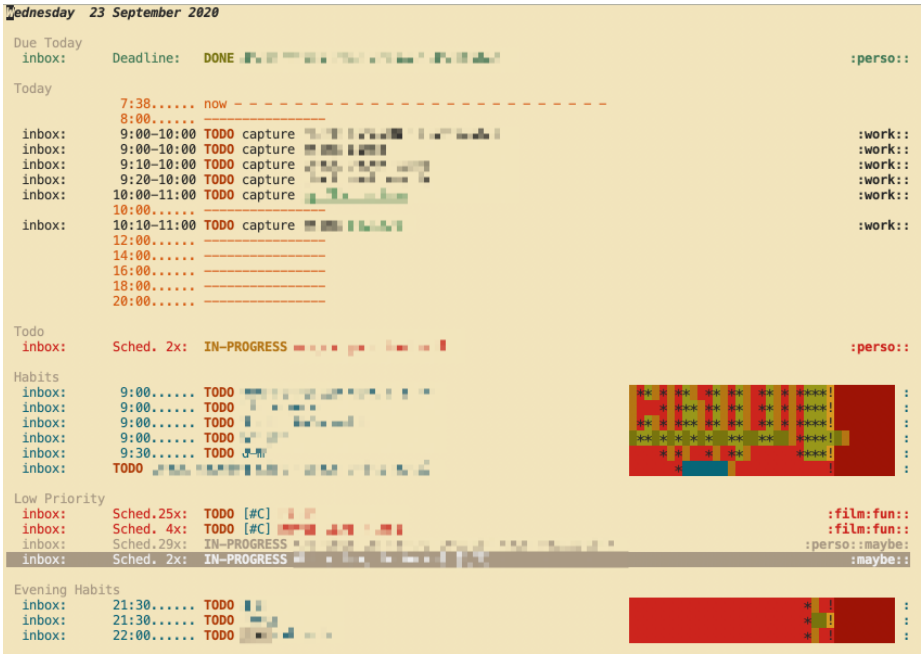

Figure 1: Org super calendar view

I then start to track (*clock*) the tasks I'm currently working on.

Often during the day, I need to create new tasks. Most of the time I create a task and I add either a deadline or a schedule date.

Sometime I also need to deal with interruptions. In that case, I *capture* the interruption that will also create a new task being clocked.

At the end of the day, every tasks I worked on are saved in a `tracker.org` file. That file look like a date tree. And I generally generate a *report* that tell me how much hours I worked today. Some tasks are tagged `work`. The report filter only on the `work` tagged tasks.

Also I have some repeating tasks like review memory cards using a spaced repetition plugin. I start it, and it shows me a few `cards` with questions that I review. So mainly those cards contain info I want to keep in my mind and not only in my notes.

document writing

Writing documents with org mode and in particular technical document is just incredible. Org mode feels a lot like markdown.

But org mode shine with its use of **org-babel**. **org-babel** is used to execute code inside your document. So you can execute block of code and get their result in block of code. For technical writing this is extremely useful.

For example, I wrote most part of an OAuth2 provider in Clojure. And to generate a documentation to some of our advanced users it is very nice to provide the full HTTP request along the response.

But I also often need to play a few tricks in the doc and directly use our Clojure code to generate JWT for example. The great part is the ability to use those JWT generated from Clojure code in the following code block making HTTP call.

That plus the natural ability to fold/unfold the tree structure of the org mode file is great.

note taking

Time to time, I need to really take the time write note on a technical subject or sometime about articles I read about anything. For that I use org roam. I only started to use it a few months ago. But this is a great addition to my previous workflow that used `deft` (that I still use). But I must say, this is pretty perfect as a note taking app.

Mainly you capture notes quite easily and put links about the subject, but also tags. In the end that generate a graph of notes that you could use later to dig into your own notes.

journal

Along with note taking. I also try to write a journal note everyday. For that I use org-journal (another org mode related package). I have a default template which take care of a few metrics I want to focus on. And I guess it is different for anyone of us.

Task Management

So here is a more detailed description about my org mode usage.

Workflow 1; planned tasks : org-agenda + clock

1. look at the current tasks planned for today
2. select a task, clock it
3. work on the task
4. back to the task and clock it out.

I work most of my using emacs[1]. Generally the first thing I do in the morning is opening ‘org-calendar‘. It looks like this:

Pretty brutalist interface which is a great thing to me. Distraction free interface going to the essential.

With this view, I see what I planned to do today. I also see a few ”Due Soon” tasks in case I have the time to handle those.

When I start working on a task I start a clock on it (I simply type `I` when my cursor is on the TODO line). When I finished some task I change its status from TODO to something else. Mainly I'm prompted when doing so:

```
{ [t] TODO   [p] IN-PROGRESS   [h] HOLD   [w] WAITING
  [d] DONE   [c] CANCELLED     [l] HANDLED }
```

And that's it. The time spent on the task as been clocked I can work on another task.

Looking at the agenda view you could notice habits. They start to become green when you are doing them correctly.

But generally, I don't use much direct clocking from the agenda. Most of the time I prefer the capture mechanism. Which bring us to ”Workflow 2”.

[1] *Short digression*: Historically, I coded using different IDEs. Then I worked for a company that forced me to use terrible keyboards and after just a few weeks I started to have serious wrist issues. So to minimize that pain I switched to vim. And it was *awesome*. Once you're use to the power of vim keybinding forever your soul will bound to them. So learning vim is a bit like learning a new music instrument. You need to construct some muscle memory and integrate one after one new tricks. Once learned your personal editing power start to become overwhelming.

After a few years of vim, I wanted to try to explore new editor tooling. So I switched to emacs using the spacemacs distribution. So mainly it's vim but with even better keybindgs, helpers and within emacs. The main reason for the switch was that vimscript is a really bad language to configure your editor. Emacs use emacs-LISP. For editor customization a LISP looked perfect to me. LISP is still one of the most powerful and easy to use programming language to date.

And recently, as my personal configuration started to grow so much I switched to doom-emacs. I was quite hesitant to do the switch but so far its been a pleasure. IMHO using doom-emacs is a lot better than using my own personal configuration from scratch because I wouldn't be able to end up with so much configuration quality.

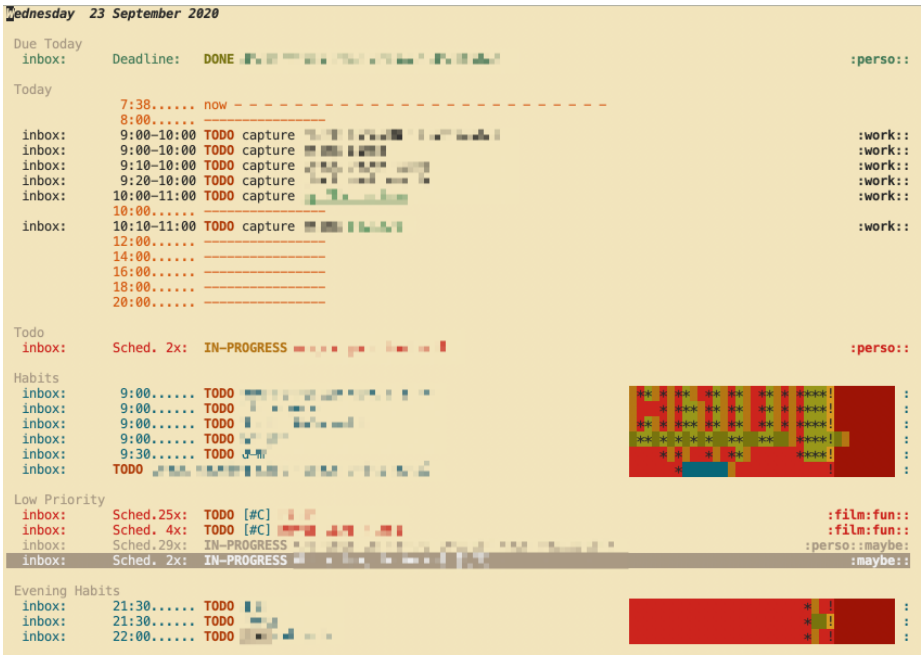

Figure 2: Org super calendar view

Workflow 2: Tracking; org-capture

Most of the tasks I perform on the day are not planned. I have a generic routine + some prepared events and tasks to performs. But during the day you have multiple interruptions, and part of my job is to write code reviews too. I cannot plan those.

In that case I use `org-capture` along `org-refile`. Mainly `org-capture` helps you create a new TODO entry. And `org-refile` will help you move that TODO entry to the correct place.

So let say I get a direct message in the chat asking me to do something. I generally start org capture (for me it's `SPC X`). I am presented with the following choice:

```
Select a capture template
=========================

[t] todo
[c] chat
[e] email
[m] meeting
[p] pause
```

```
[r] review
[w] work
[i] interruption
[f] chore
------------------------------------------------------------------------
[q] Abort
```

In my example it was a chat interruption. So I type `i` that presents me with this

```
**** IN-PROGRESS |  :interruption:
:LOGBOOK:
[2020-09-23 Wed 08:01]
ref :: [link-to-where-I-was-in-emacs-when-captured]
```

My cursor placed where the `|` is displayed. Here I add the tag `chat` and a small description, "dm from John about X" for example. Then I type `C-c C-c` and the TODO is placed in a `tracker.org` file under a date tree that looks like this:

```
* 2020
** 2020-W39
*** 2020-09-21 Monday
*** 2020-09-22 Tuesday
*** 2020-09-23 Wednesday
**** IN-PROGRESS Chat with John about X                    :interruption:chat:
:LOGBOOK:
:END:
[2020-09-23 Wed 17:58]
ref ::
...
```

So the clock for this task started at the moment at made the capture. In my workflow, I prefer to finish the capture and stop clock later. So after I finished the capture, the clock is still running while the task is put in my tracker file.

Once I finished with that task. I can:

1. Jump to the tasks with `SPC n o` (`org-clock-goto`), and stop the clock `SPC m c o` (`clock-out`).
2. Jump to the task and change its status to `DONE` which will stop the clock.
3. Capture another tasks which will stop the clock on the current task and will start on the new one.

By the end of the day, my tracker file will contain a date tree with all the tasks I done in the day. All tasks nicely clocked. I generally create a clock report that look like this:

```
#+BEGIN: clocktable :scope subtree :maxlevel 4 :timestamp t :narrow 36! :match "work"
#+CAPTION: Clock summary at [2020-09-23 Wed 08:20]
| Timestamp           | Headline                        | Time   |   |     |     |
```

```
|----------------------+------------------------------------+--------+---+---
---+------|
|                      | *Total time*                       | *6:40* |   |     |     |
|----------------------+------------------------------------+--------+---+---
---+------|
|                      | \_   2020-09-21 Monday             |        |   | 7:40 |     |
| [2020-09-21 Mon 08:54] | \_     check chat                |        |   |     | 0:36 |
| [2020-09-21 Mon 09:30] | \_     check reviews             |        |   |     | 0:41 |
| [2020-09-21 Mon 10:11] | \_     check emails              |        |   |     | 0:07 |
| [2020-09-21 Mon 10:37] | \_     review PR about xxx       |        |   |     | 0:44 |
| [2020-09-21 Mon 11:21] | \_    update my PR from feedbacks |      |   |     | 0:36 |
| [2020-09-21 Mon 12:08] | \_    review John's PR about Foo  |      |   |     | 0:12 |
| [2020-09-21 Mon 13:41] | \_     review M's PR about Bar    |      |   |     | 0:11 |
| [2020-09-21 Mon 13:53] | \_     another thing             |      |   |     | 0:16 |
| [2020-09-21 Mon 14:09] | \_     review PR                 |      |   |     | 0:51 |
| [2020-09-21 Mon 15:00] | \_     work on PR                |      |   |     | 1:30 |
| [2020-09-21 Mon 16:49] | \_     check another PR          |      |   |     | 0:33 |
| [2020-09-21 Mon 17:03] | \_     answer email              |      |   |     | 0:55 |
| [2020-09-21 Mon 17:58] | \_     Chat John about X         |      |   |     | 0:28 |
```

And that's mostly it for TODOs and tasks handling.

Workflow 3: Add new tasks; org-capture / org-refile

Another thing I do quite often. I need to add new task to be done. Be it for today or another day.

In that case, I generally use org-capture again. This time I choose `t` for TODO and I generally detail the task to be done. I add either a SCHEDULE (when I plan to start) or a DEADLINE (when this must be finished) and I refile it.

So refile will start a fuzzy search to put this task under some subtree. So instead of going to my `tracker.org` file, this goes to my `inbox.org` file.

And it will appear in my agenda.

Configuration

So to have all of that, I added a lot of configuration over time. But here is the most important part.

Most of that config is what I personally think are better defaults. And a minor part of it only is about how I organize myself.

```
(defun org-mode-config ()
  "Org-mode."
  (setq org-extend-today-until 4
        org-use-effective-time t)
```

```
(setq org-todo-keywords
      '((sequence "TODO(t)"
                  "IN-PROGRESS(p)"
                  "|"
                  "DONE(d)"
                  "HOLD(h@/!)"
                  "CANCELED(c@/!)"
                  "HANDLED(l@/!)")
        (sequence "|" "PAUSE(p)" "CHAT(c)" "EMAIL(e)" "MEETING(m)" "REVIEW(r)" "GEEK(g)")))

;;; Look & Feel

;; I like to have something different than ellipsis because I often use them
;; myself.
(setq org-ellipsis " [+]")
(custom-set-faces '(org-ellipsis ((t (:foreground "gray40" :underline nil)))))

(defun my-org-settings ()
  (org-display-inline-images)
  (setq fill-column 75)
  (abbrev-mode)
  (org-indent-mode)
  nil)

(add-hook 'org-mode-hook #'my-org-settings)

(setq org-tags-column 69)

;; src block indentation / editing / syntax highlighting
(setq org-src-fontify-natively t
      org-src-window-setup 'current-window ;; edit in current window
      org-src-preserve-indentation t ;; do not put two spaces on the left
      org-src-tab-acts-natively t)

;; *** Templates
;; the %a refer to the place you are in emacs when you make the capture
;; that's very neat when you do that in an email for example.
(setq org-capture-templates
      '(("t" "todo"         entry (file "~/.org/inbox.org")
         "* TODO %?\n%U\n- ref :: %a\n")
        ;; time tracker (clocked tasks)
        ("g" "geek"         entry (file+olp+datetree "~/.org/tracker.org")
         "* GEEK %?         :perso:\n%U\n- ref :: %a\n"
         :prepend t :tree-type week :clock-in t :clock-keep t)
        ("c" "chat"         entry (file+olp+datetree "~/.org/tracker.org")
         "* CHAT %?         :work:chat:\n%U\n- ref :: %a\n"
```

```
         :prepend t :tree-type week :clock-in t :clock-keep t)
        ("e" "email"        entry (file+olp+datetree "~/.org/tracker.org")
         "* EMAIL %?        :work:email:\n%U\n- ref :: %a\n"
         :prepend t :tree-type week :clock-in t :clock-keep t)
        ("m" "meeting"      entry (file+olp+datetree "~/.org/tracker.org")
         "* MEETING %?      :work:meeting:\n%U\n- ref :: %a\n"
         :prepend t :tree-type week :clock-in t :clock-keep t)
        ("r" "review"       entry (file+olp+datetree "~/.org/tracker.org")
         "* REVIEW %?       :work:review:\n%U\n- ref :: %a\n"
         :prepend t :tree-type week :clock-in t :clock-keep t)
        ("w" "work"         entry (file+olp+datetree "~/.org/tracker.org")
         "* IN-PROGRESS %?  :work:\n%U\n- ref :: %a\n"
         :prepend t :tree-type week :clock-in t :clock-keep t)
        ("p" "pause"        entry (file+olp+datetree "~/.org/tracker.org")
         "* PAUSE %?        :pause:\n%U\n- ref :: %a\n"
         :prepend t :tree-type week :clock-in t :clock-keep t)
        ("i" "interruption" entry (file+olp+datetree "~/.org/tracker.org")
         "* IN-PROGRESS %?  :interruption:work:\n%U\n- ref :: %a\n"
         :prepend t :tree-type week :clock-in t :clock-keep t)
        ("f" "chore"        entry (file "~/.org/inbox.org")
         "* IN-PROGRESS %?  :chore:\n%U\n"
         :clock-in t :clock-keep t)))

;; How to create default clocktable
(setq org-clock-clocktable-default-properties
      '(:scope subtree :maxlevel 4 :timestamp t :link t :tags t :narrow 36! :match "work"))

;; How to display default clock report in agenda view
(setq org-agenda-clockreport-parameter-plist
      '(:lang "en" :maxlevel 4 :fileskip0 t :link t :indent t :narrow 80!))

;; *** Projectile; default TODO file to create in your projects
(setq org-projectile-file "inbox.org")

;; *** Refile mapped to SPC y o r
(map! :leader :desc "org-refile" "y o r" #'org-refile)

;; Refile to either the =refile.org= file or to =agenda.org= org =standup.org=
(setq org-refile-target-files
      '("~/.org/tracker.org"
        "~/.org/inbox.org"))

(setq org-refile-targets
      '((nil :maxlevel . 5)
        (org-refile-target-files :maxlevel . 5)))
```

```
;; *** Agenda
(setq org-log-into-drawer t) ;; hide the log state change history a bit better
(setq org-agenda-files org-refile-target-files)
(setq org-deadline-warning-days 7)
(setq org-agenda-skip-scheduled-if-deadline-is-shown t)
(setq org-habit-show-habits-only-for-today nil)
(setq org-habit-graph-column 65)
(setq org-duration-format 'h:mm) ;; show hours at max, not days
(setq org-agenda-compact-blocks t)
;; default show today
(setq org-agenda-span 'day)
(setq org-agenda-start-day "-0d")
(setq org-agenda-start-on-weekday nil)
(setq org-agenda-custom-commands
      '(("d" "Done tasks" tags "/DONE|CANCELED")
        ("g" "Plan Today"
         ((agenda "" ((org-agenda-span 'day)))
          (org-agenda-skip-function '(org-agenda-skip-deadline-if-not-today))
          (org-agenda-entry-types '(:deadline))
          (org-agenda-overriding-header "Today's Deadlines ")))))
(setq org-agenda-window-setup 'only-window)

(defun y/go-to-today-agenda ()
  (interactive)
  (org-agenda nil "a"))
;; Faster jump to agenda today keybinding shortcut (SPC y a)
(map! :leader
      :desc "Today's agenda"
      "y a" #'y/go-to-today-agenda)

;; ** Org Annotate

;; Ability to take annotate some files, can of double usage with org-capture.
;; Still, I keep that keyboard shortcut here.
;; (evil-leader/set-key "oa" 'org-annotate-file)
(setq org-annotate-file-storage-file "~/.org/annotations.org")


;; ** Org colums
;; Can be nice sometime to have that column view
;; give a felling of Excel view
(setq org-columns-default-format
      "%TODO %3PRIORITY %40ITEM(Task) %17Effort(Estimated Effort){:} %CLOCKSUM %8TAGS(TAG)")
(map! :leader "y o c" #'org-columns)

;; ** Deft
```

```
  ;; useful to find files and jump to them
  (setq deft-extensions '("org" "gpg" "md" "txt"))
  (setq deft-recursive t)
  (setq deft-use-filter-string-for-filename t)
  (setq deft-default-extension "org")
  (setq deft-directory "~/.org")


  ;; Org Babel
  (org-babel-do-load-languages
   'org-babel-load-languages
   '(;; other Babel languages
     (shell . t)
     (http . t)
     (clojure . t)
     (haskell . t)
     (plantuml . t) ;; UML graphs
     (gnuplot . t)))
  (setq org-plantuml-jar-path "~/bin/plantuml.jar"))

(use-package! org
  :config (org-mode-config))
```

And also

```
(use-package! org-super-agenda
  :after org-agenda
  :custom (org-super-agenda-groups
           '( ;; Each group has an implicit boolean OR operator between its selectors.
             (:name "Overdue" :deadline past :order 0)
             (:name "Evening Habits" :and (:habit t :tag "evening") :order 8)
             (:name "Habits" :habit t :order 6)
             (:name "Today" ;; Optionally specify section name
              :time-grid t  ;; Items that appear on the time grid (scheduled/deadline with time)
              :order 3)     ;; capture the today first but show it in order 3
             (:name "Low Priority" :priority "C" :tag "maybe" :order 7)
             (:name "Due Today" :deadline today :order 1)
             (:name "Important"
              :and (:priority "A" :not (:todo ("DONE" "CANCELED")))
              :order 2)
             (:name "Due Soon" :deadline future :order 4)
             (:name "Todo" :not (:habit t) :order 5)
             (:name "Waiting" :todo ("WAITING" "HOLD") :order 9)))
  :config
  (setq org-super-agenda-header-map nil)
  (org-super-agenda-mode t))
```

Conclusions

That article is already quite long. But if you intend to dig into org mode, this can be a nice default starting point.

I haven't really dig into some details but only given you the ability to start not completely from scratch and with decent default values for an already advanced usage.

To resume:

- `SPC y a` ☒ Show agenda view for today
- `SPC X` ☒ Capture a new task, write a description, then `C-c C-c`, save that in `tracker.org` (or `inbox.org` depending of the capture template)
- `SPC n o` ☒ jump to the current time tracked tasks
- `SPC m c o` ☒ stop the clock on that task; if you capture a new time tracking tasks you don't need to clock-out
- `SPC y o r` ☒ `org-refile`, meaning move that task somewhere else
- `SPC q` ☒ add/remove tags to that task

Footnotes