Org-mode Basics

Ben Maughan

October 9, 2015

Contents

1 org-mode structure

Text in org is structured by headings, denoted by lines starting with one or more `*` so we are currently in a section!

1.1 A subheading

Starts with an extra `*` and so on

1.2 navigation

Headings can be expanded or collapsed by moving to the (sub)heading and pressing `TAB`. `S-TAB` cycles all headings. You can jump to next and previous headings with `C-c C-n` and `C-c C-p` respectively.

You can move headings up and down to reorder them with the arrow keys, using `M-up` or `M-down`. You can change the level of headings with `M-left` and `M-right` (and use `M-S-left` and `M-S-right` to also change the levels of and subheadings).

1.3 lists

1.3.1 bullet lists

- bullet lists can be created like this (start a line with one or more space and a `-`
- pressing M-RET gives you a new bullet

- we might also like nested bullets
  - like this one (I pressed `M-RET TAB` to indent it)
  - and another (`M-RET` now indents to the new level)
- the nice thing is that for long lines of text, emacs wraps them so that they line up with the bullet
- you can also reorder list items and change indentation using M-up or M-down just like with section headings
- you can change bullet style using `S-left` and `S-right`

1.3.2 numbered lists

1. numbered lists are also possible
2. `M-RET` gives me a new number

1.3.3 checklists [/]

- ☐ we can even have check lists
- ☐ `M-S-RET` gives a new item with a check box
- ☐ `C-c C-c` check/unchecks a box
- ☐ you can have sub items
  - ☐ like this
  - ☐ that can be checked off individually
- ☐ and you can track the number of items by adding [/] to the end of a line above a checklist - this updates when you check items off

1.3.4 definition lists

definition lists these are useful sometimes

item 2 `M-RET` again gives another item, and long lines wrap in a tidy way underneath the definition

2 Tables

Hopefully you can see straight away that the simple structure provided by org-mode gives a nice way to keep an electronic note book.

Often it is nice to include tables in our notes. Org handles this by using | to separate columns, and a line of — (inserted with `C-c -`) to add horizontal lines.

Exercise: start typing in this table below; type the first line in verbatim

1. when you get to the "s" of comments, press `TAB` to go to the next line
2. go up to the previous line and use `C-c -` to add the row of dashes
3. next enter a few lines of data, using `TAB` to go through the cells - you should notice the columns changing width as needed

ID	x	y	comments
A	2	4	blah
B	3	9	blah
C	4	16	blah blah blah
D	5	25	blah

Now, you can move rows and columns around using `M-arrow` and insert or delete rows and columns using `M-S-arrow`. Try this out now.

2.1 Creating and exporting tables

You can create an empty table using `C-c |` to run the command `org-table-create-or-convert-from-region`, which will prompt for table dimensions if no region is selected.

The same command can easily convert some text to a table; select the following text and use `C-c |` to run the command `org-table-create-or-convert-from-region` again to convert the text to a table

```
ID  x   y
A   2   4
B   3   9
C   4  16
D   5  25
```

You can also save tables to their own files by putting the cursor in the table and using `M-x org-table-export`. You'll be asked for a file name and a format. For the format, type `orgtbl-to` and press TAB to see the available options (e.g. `orgtbl-to-csv` will convert to csv in the output file).

2.2 Formulae

You can use formulae to do arithmetic on tables, and use them like a spreadsheet. This is something I keep meaning to use more often, but don't generally find I need it. One useful command is `C-c +` which runs `org-table-sum` to sum the numbers in the current column.

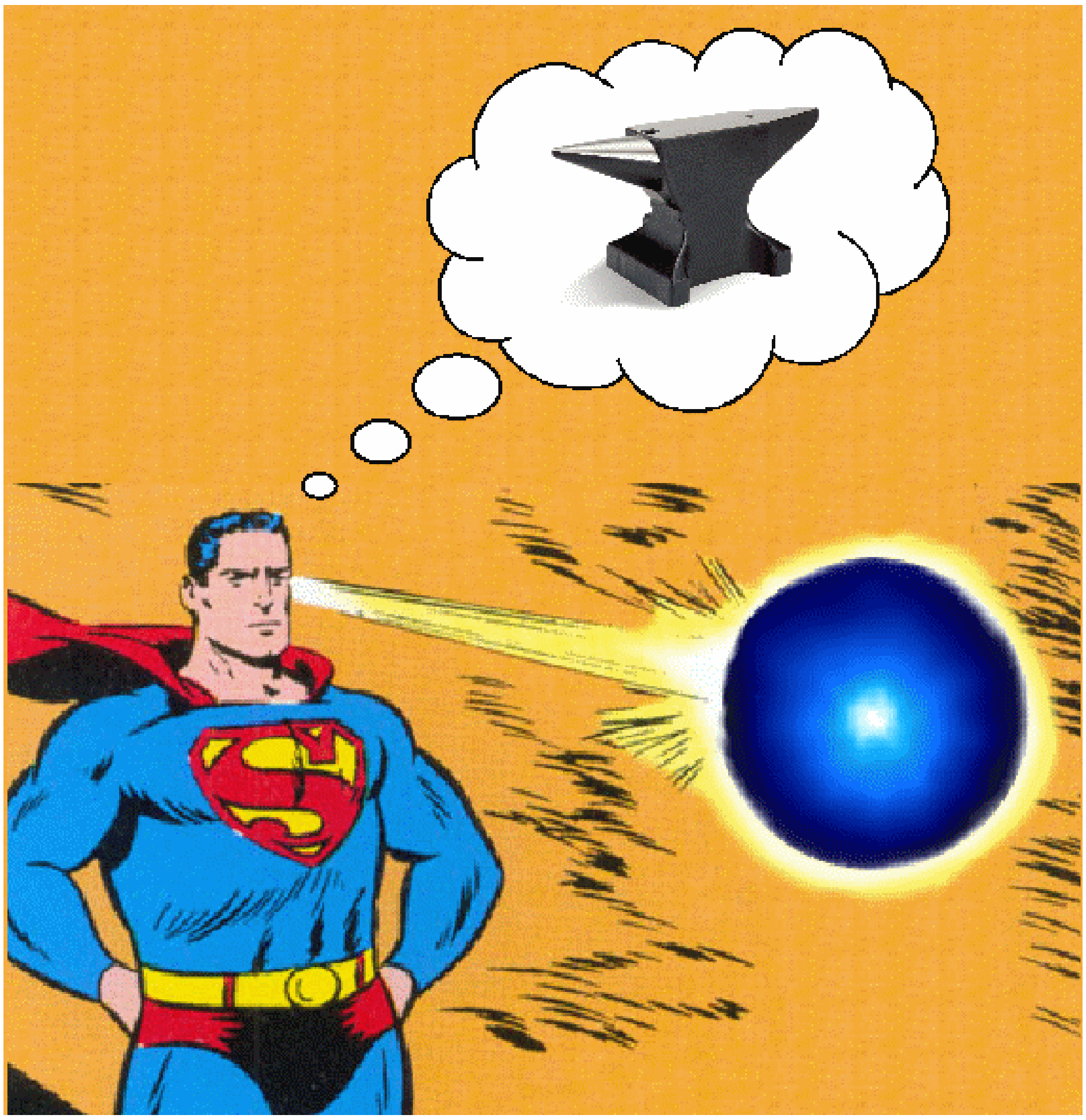
For more on this, see e.g. this introduction. Notice that we just added a link in our org-mode file - this is a teaser for what we will cover next!

3 Links and images

Org mode supports links to files, URLs, and to other points in the org file. In this example let's use an image from my website. First copy it to the current directory. You can do this within emacs but for now just run this command in your terminal.

```
curl http://www.star.bris.ac.uk/bjm/superman_cluster.png -o superman_cluster.png
```

To add a link to a file use `C-u C-c C-l` and type the name of a file. Use tab-completion to select the image we just copied and you will then be asked for a description - you can press enter to leave this blank. This will create a link that looks like this



If you do this in your org file, you won't see the `[[ ]]` above, instead you'll see the text as a clickable link.

Since the file we have linked to is an image, we can tell emacs to the image in the document using C-c C-x C-v and use the same command to turn the image off again.

You can also click the link with the mouse, or use C-c C-o to follow it, which might open your web browser, an image viewer or open a file in emacs depending on the target of the link.

The structure of a link in org mode looks like this

```
[[link address][description]]
```

(I've enclosed the link in an example block which prevents org-mode from trying to interpret as a real link, for the purpose of showing its structure - we'll come back to blocks like this later.)

The link address is the URL or file name, and the description is the text that is displayed, so we can replace our superman link with something tidier like this.

Links to web pages are easy - just put the http address in as the link address. Use `C-c C-l` as a quick way to add such a link (remember we used `C-u C-c C-l` for adding a link to a file).

Links to other parts of the org file are added easily like this link. Because the address part of the link matches a headline in this document, then org-mode points the link to that part of the file. Clicking it will move the cursor there.

Finally, we can add a caption and a label to our image like this

which means we can refer to our image later with a link like this one 1

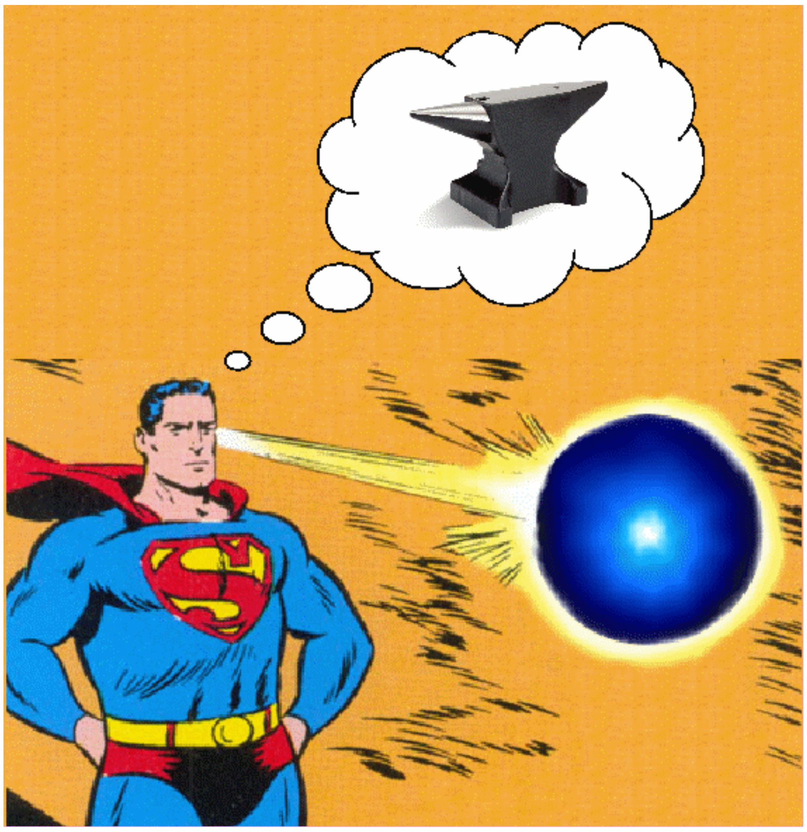


Figure 1: Superman and a galaxy cluster

4 Formatting text

4.1 Simple formatting

You can apply simple formatting to your text by enclosing words in special characters. These include

- *italicised text*
- **bold text**
- underlines
- `literal text`
- `code` (generally appears the same as literal text)

4.2 Formatted blocks of text

For longer pieces of text you can enclose the text in blocks marking it as a specific sort of text. I commonly use these ones

```
This is an example block into which you can type text that you don't want org to mess with like a [[
```

> This block encloses text that you want to appear as a quotation.

This text will be centred when it is exported.

You can save time typing out the block wrapper by using shortcuts. Go to the start of a new line and type `<e` and press `TAB` and it will expand to an example block. The same works for `<q` for quote and `<c` for centre.

4.3 LaTeX

Org-mode does a good job of understanding snippets of LaTeX (a powerful typesetting language used in scientific and other technical documents). For example, it will correctly export simple superscripts x^2 or subscripts x_0 or symbols like α, β, γ.

Org also understands more complex LaTeX like this

$$x^2 + \left(\frac{y}{z}\right)^4 = 0 \tag{1}$$

but for longer bits of LaTeX it is better to use a LaTeX block. You start one with `<l` and `TAB`

LaTeX code goes here

4.4 Source code blocks

It is also handy to include source code in your notes - on a new line type `<s` and `TAB` to create a source block. You can tell org what type of code is contained - in this case we'll put in some simple shell code, so well put "sh" at the top of the block.

```
echo "Hello $USER! Today is ‘date‘"
exit
```

You can get org to syntax highlight the text in the block by adding the following to your emacs config file (without the source block wrapper of course).

```
;;syntax highlight code blocks
(setq org-src-fontify-natively t)
```

What is more, when the cursor is inside a SRC block, you can use `C-c '` to create a new temporary buffer in the major mode of the programming language you have specified. Type some code in, and then type `C-c '` again to come back to this buffer.

4.5 Executing source code blocks

Org-mode can execute your source code blocks and add the output to your file. This part of org-mode is called babel. I'll write more about this later, but it is too cool not to mention here.

For example, take the simple code block we had above:

```
echo "Hello $USER! Today is ‘date‘"
exit
```

Put the cursor inside the block and hit `C-c C-c` to execute it. You will be asked to confirm and then you should see the output appear like this:

```
Hello bjm! Today is Fri 25 Sep 2015 15:03:12 BST
```

You can do much more with this, like reading input data from a table in the same file, creating images that appear in the file, extracting (tangling) all the code snippets into one or more files to be executed separately, and much more. Here are some nice examples.

You can tell org-mode which programming languages to support by adding something like the following to your emacs config file:

```
;; Some initial languages we want org-babel to support
(org-babel-do-load-languages
 'org-babel-load-languages
 '(
   (sh . t)
   (python . t)
   (R . t)
   (ditaa . t)
   (perl . t)
   (gnuplot t)
   ))
```

5 Exporting

One strength of org-mode is the ability to export to multiple formats. Probably the most useful to begin with are web pages and pdf (via latex) but more are available; to quote the org manual

> ASCII export produces a readable and simple version of an Org file for printing and sharing notes. HTML export allows you to easily publish notes on the web, or to build full-fledged websites. LaTeX export lets you use Org mode and its structured editing functions to create arbitrarily complex LaTeX files for any kind of document. OpenDocument Text (ODT) export allows seamless collaboration across organizational boundaries. Markdown export lets you seamlessly collaborate with other developers. Finally, iCal export can extract entries with deadlines or appointments to produce a file in the iCalendar format.

To export your org file to a web page, type `C-c C-e` to start the exporter and then press `h` to select html and `o` to select open. A new web page should now open in your browser.

Similarly, typing `l` and `o` in the exporter will convert the org file to latex and then compile it to produce a pdf and display that. Try both of these.

It is possible to add many customisations to the export process. For example, go to the top of the buffer (using `M-<`) and use `C-c C-e` and then `#` to insert an export template. You can then choose to add html or latex (or other) templates (press `TAB` to see the list).

As an example, add the following to the top of your org file to tweak the appearance of the exported documents.

```
#+LaTeX_CLASS: bjmarticle
#+TITLE:     Org-mode Basics
#+AUTHOR: Ben Maughan
#+OPTIONS: html-link-use-abs-url:nil html-postamble:auto
#+OPTIONS: html-preamble:t html-scripts:t html-style:t
#+OPTIONS: html5-fancy:nil tex:t
#+HTML_DOCTYPE: xhtml-strict
#+HTML_CONTAINER: div
#+DESCRIPTION:
#+KEYWORDS:
#+HTML_LINK_HOME:
#+HTML_LINK_UP:
#+HTML_MATHJAX:
#+HTML_HEAD: <link rel="stylesheet" type="text/css" href="http://www.star.bris.ac.uk/bjm/css/bjm.css"
#+HTML_HEAD_EXTRA:
#+SUBTITLE:
#+INFOJS_OPT:
#+CREATOR: <a href="http://www.gnu.org/software/emacs/">Emacs</a> 24.4.1 (<a href="http://orgmode.org
#+LATEX_HEADER:
```

This is the default html export template with a couple of tweaks.

- I have added a link to a style sheet to style the html
- I have added a latex class `bjmarticle` to control the appearance of the generated pdf

The latex class is defined in my emacs config file with the following

```
(add-to-list 'org-latex-classes
             '("bjmarticle"
               "\\documentclass{article}
\\usepackage[utf8]{inputenc}
\\usepackage[T1]{fontenc}
\\usepackage{graphicx}
\\usepackage{longtable}
\\usepackage{hyperref}
\\usepackage{natbib}
\\usepackage{amssymb}
\\usepackage{amsmath}
\\usepackage{geometry}
\\geometry{a4paper,left=2.5cm,top=2cm,right=2.5cm,bottom=2cm,marginparsep=7pt, marginparwidth=.6in}"
               ("\\section{%s}" . "\\section*{%s}")
               ("\\subsection{%s}" . "\\subsection*{%s}")
               ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
               ("\\paragraph{%s}" . "\\paragraph*{%s}")
               ("\\subparagraph{%s}" . "\\subparagraph*{%s}")))
```

You'll need some experience of LaTeX to make significant changes here, but the sky is the limit.