Contents

1 PDFsync for org files

$e^x = 3.14$

First, we need a way to check if the current buffer has changed since the last time we built so we can avoid unnecessary builds. We can use an md5 sum for this. We will just store the md5 sum in a global variable, so we can compare it at any point in time.

```
(md5 (current-buffer))
```

```
203af8310c9cff1c4aba54e922d94d79
```

We want our build to happen asynchronously, so we can continue typing. First, we develop the build function.

```
(defvar *last-md5* nil "md5 of current buffer")
(defvar *async-building* nil "is a build happening")

(setq *async-building* nil)
(defun async-build ()
  (interactive)
  (cond
   (*async-building*
    (message "Build in progress"))
   ((equal *last-md5* (md5 (current-buffer)))
    (message "No change to build"))
   (t
    (message "building")
    (setq *last-md5* (md5 (current-buffer)))
    (setq *async-building* t)
    (save-buffer)
    (org-latex-export-to-pdf t))))
```

```
async-build
```

That last function will launch an asynchronous build process. We can see the contents and progress of this in this variable.

```
org-export-stack-contents
```

Org Export Process	nil	org-export-process
/Users/jkitchin/blogofile-jkitchin.github.com/_blog/blog.pdf	latex	(21752 50592 42735 0)
#<killed buffer>	nil	org-export-process

When the export is done,

```
org-export-stack-contents
```

/Users/jkitchin/blogofile-jkitchin.github.com/_blog/pdfsync.tex	latex	(21752 57725 897453 0)
#<killed buffer>	nil	org-export-process
/Users/jkitchin/blogofile-jkitchin.github.com/_blog/pdfsync.pdf	latex	(21752 57668 576629 0)
#<killed buffer>	nil	org-export-process
/Users/jkitchin/blogofile-jkitchin.github.com/_blog/blog.pdf	latex	(21752 52220 238830 0)
#<killed buffer>	nil	org-export-process

We can view the result here in docview.

```
(find-file-other-window (caar org-export-stack-contents))
```

```
#<buffer blog.pdf>
```

The last little idea is to create an idle timer to launch the build whenever we are idle. There are some tricks we want to use. First, we need to save the timer so we can cancel it later. Second, we need two timers, one to start the build, and one to check when it is done.

```
(defvar async-message-timer1 nil
  "Variable to store the timer in.")

(defvar async-message-timer2 nil
  "Variable to store the timer in.")

;; timer that starts builds when we are idle
(setq async-message-timer1 (run-with-idle-timer 0.5 t 'async-build))

(setq async-message-timer2
      (run-with-idle-timer
       0.5 t
       (lambda ()
         (when (and (stringp (caar org-export-stack-contents))
                    (file-exists-p (caar org-export-stack-contents))
                    (string= "pdf" (f-ext (caar org-export-stack-contents))))
           (setq *async-building* nil)
           (switch-to-buffer (get-file-buffer (caar org-export-stack-contents)))
           (find-file (caar org-export-stack-contents))))))
```

```
[nil 0 0 500000 t (lambda nil (when (and (stringp (caar org-export-stack-contents)) (file-exists-p (caa
```

```
(cancel-timer async-message-timer1)
(cancel-timer async-message-timer2)
```