Sioyek

Release 2.0.0

Ali Mostafavi

Mar 04, 2023

CONTENTS

Sioyek is a PDF viewer designed for reading research papers and technical books.

CHAPTER

ONE

CONTENTS

1.1 Installation

1.1.1 Pre-built Binaries

You can find the latest pre-built binaries of sioyek here: https://github.com/ahrm/sioyek/releases

1.1.2 Homebrew Cask

There is a homebrew cask available here: https://formulae.brew.sh/cask/sioyek. Install by running:

```
brew install --cask sioyek
```

1.1.3 Third-party packages for Linux

If you prefer to install sioyek with a package manager, you can look at this list. Please note that they are provided by third party packagers. USE AT YOUR OWN RISK! If you're reporting a bug for a thrid-party package, please mention which package you're using.

Distro	Link
Flathub	https://flathub.org/apps/details/com.github.ahrm.sioyek
Alpine	https://pkgs.alpinelinux.org/packages?name=sioyek
AUR Sioyek-git	https://aur.archlinux.org/packages/sioyek-git/
AUR sioyek-appimage	https://aur.archlinux.org/packages/sioyek-appimage/
NixOS	https://search.nixos.org/packages?channel=unstable&show=sioyek&from=0&size=50&sort=relevance&type=packages&query=sioyek

1.1.4 Building from source

Linux

1. Install Qt 5 and make sure qmake is in `PATH`.
2. Install `libharfbuzz`:

```
$ apt install libharfbuzz-dev
```

3. Clone the repository and build:

```
$ git clone --recursive https://github.com/ahrm/sioyek
$ cd sioyek
$ ./build_linux.sh
```

Windows

1. Install Visual Studio (tested on 2019, other relatively recent versions should work too)
2. Install Qt 5 and make sure qmake is in `PATH`.
3. Clone the repository and build using 64 bit Visual Studio Developer Commnand Prompt:

```
$ git clone --recursive https://github.com/ahrm/sioyek
$ cd sioyek
$ build_windows.bat
```

Mac

1. Install Xcode and Qt 5.
2. Clone the repository and build:

```
$ git clone --recursive https://github.com/ahrm/sioyek
$ cd sioyek
$ chmod +x build_mac.sh
$ ./build_mac.sh
```

1.2 Usage

Note that all the keybindings in this section are the default keybindings that can be changed by editing `keys_user.config`. Each key executes a command. In order to add a keybinding that executes a command, you add the following to `keys_user.config`:

```
command key
```

For example suppose you want to use the `j` key to move down. You can add the following to your config file:

```
move_down j
```

You can press `:` to open a searchable list of all available commands.

1.2.1 Opening Files

- Press `o` to open the select file menu. (`open_document` command)
- Press <`Shift o`> (`O`) to open a searchable list of recently opened files. (`open_prev_doc` command)
- Press <`Ctrl o`> to open an embedded file system browser. (`open_document_embedded` command)
- Press <`Ctrl Shift o`> to open an embedded file system browser rooted in the current document folder. (`open_document_embedded_from_current_path` command)
- You can also drag files into sioyek window to open them.
- You can press `delete` in the list of recently opened files to remove a file from the list (doesn't remove the file on the filesystem).
- If you want to open a file in a new sioyek window, you can pass the `--new-window` command line option. Or you could open a new window from within sioyek by pressing <`Ctrl t`>.
- You can switch between opened sioyek windows using `goto_window` command.

1.2.2 Basic Movement

- You can move the screen using the arrow keys. (corresponding to the following commands: `move_down`, `move_up`, `move_left` and `move_right`)
- You can also use the mouse wheel to scroll the screen.
- Press `gg` to go to the first page and `G` to go to the last page. (`goto_begining` and `goto_end` commands)
- In order to go to a specific page you can enter the page number and then press `gg`. For example in order to go to page 42, you should enter `42gg`.
- To go to a specific page, you can also press the `HOME` button which opens a menu where you can enter the page number. (`goto_page_with_page_number` command)
- Press `space` to move the screen down and <`Shift space`> to move the screen up. (`screen_down` and `screen_up` commands)
- Press <`Ctrl PageDown`>/<`Ctrl PageUp`> to go to the next/previous page. (`next_page` and `previous_page` commands)
- Press `t` to open a searchable table of contents. You can jump to table of content entries by selecting. (`goto_toc`)
- Press `gc` to go to the next chapter and `gC` to go to the previous chapter. (`next_chapter` and `prev_chapter` commands)
- We don't have a scrollbar by default. If you want to enable it, you can run `toggle_scrollbar`

1.2.3 Zoom

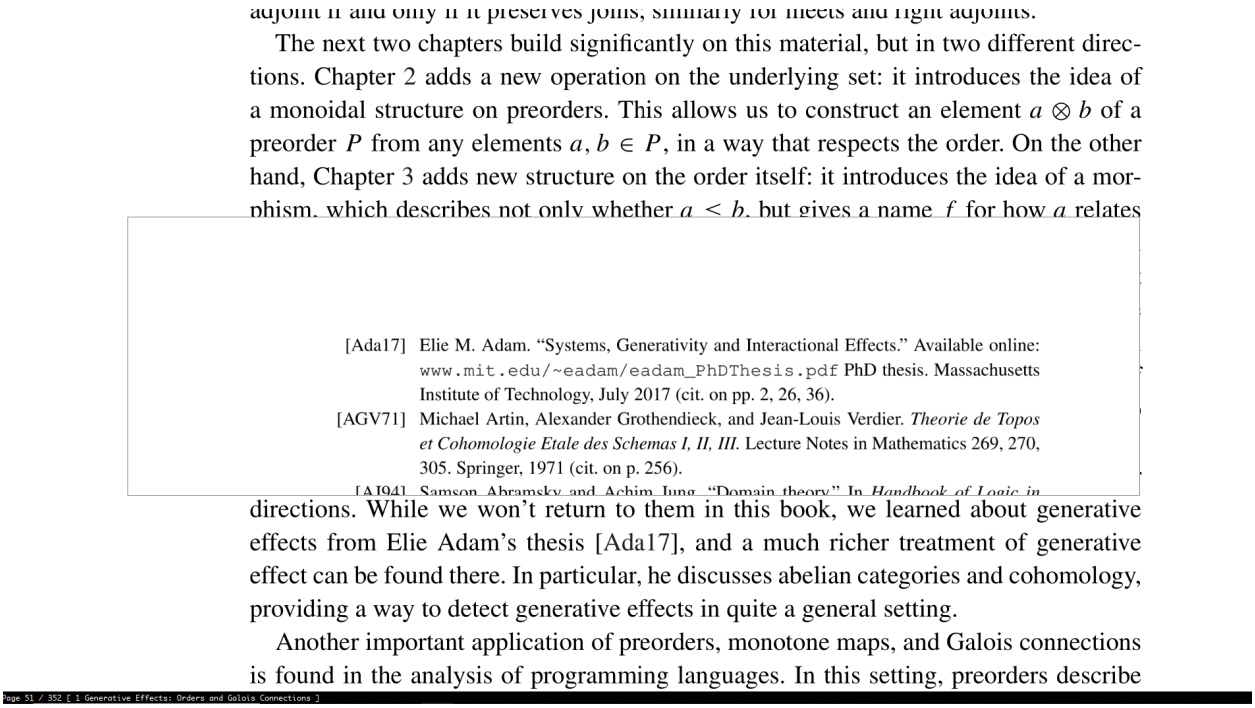
- Press + (<`Shift =`>) to zoom in and - to zoom out. (`zoom_in` and `zoom_out` commands)
- You can also use the mouse wheel while holding control to zoom in/out.
- Press `f9` to fit the page to window width. (`fit_to_page_width` command)
- Press `f10` to fit the page to window width ignoring white page margins. (`fit_to_page_width_smart` command)

1.2.4 History Navigation

Sioyek keeps a full, browser-like history of your locations. For example suppose you click on a link in the PDF file to go to a figure. You can now return back to the location of the link by issuing the `prev_state` command. Now, you can again return to the figure by issuing `next_state` command. Note that this history even works across multiple documents. For example if I am viewing document `A.pdf` and then open `B.pdf` I can return back to `A.pdf` by going back in history.

- You can go back/forward in history by pressing `backspace`/`<Shift backspace>` or `Ctrl LeftArrow`/`Ctrl RightArrow`. (`prev_state` and `next_state` commands)

1.2.5 Overview and SmartJump


adjoint if and only if it preserves joins, similarly for meets and right adjoints.

The next two chapters build significantly on this material, but in two different directions. Chapter 2 adds a new operation on the underlying set: it introduces the idea of a monoidal structure on preorders. This allows us to construct an element $a \otimes b$ of a preorder P from any elements $a, b \in P$, in a way that respects the order. On the other hand, Chapter 3 adds new structure on the order itself: it introduces the idea of a morphism, which describes not only whether $a \leq b$, but gives a name f for how a relates

[Ada17] Elie M. Adam. "Systems, Generativity and Interactional Effects." Available online: `www.mit.edu/~eadam/eadam_PhDThesis.pdf` PhD thesis. Massachusetts Institute of Technology, July 2017 (cit. on pp. 2, 26, 36).

[AGV71] Michael Artin, Alexander Grothendieck, and Jean-Louis Verdier. *Theorie de Topos et Cohomologie Etale des Schemas I, II, III.* Lecture Notes in Mathematics 269, 270, 305. Springer, 1971 (cit. on p. 256).

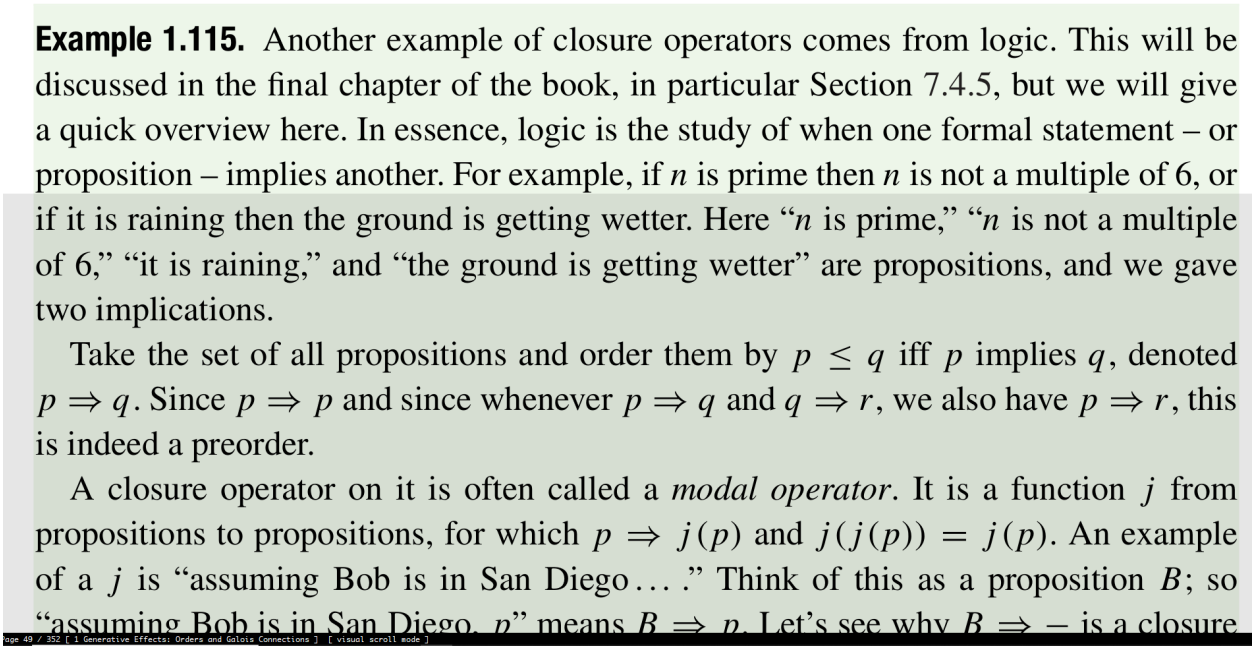
[AJ94] Samson Abramsky and Achim Jung. "Domain theory." In *Handbook of Logic in*

directions. While we won't return to them in this book, we learned about generative effects from Elie Adam's thesis [Ada17], and a much richer treatment of generative effect can be found there. In particular, he discusses abelian categories and cohomology, providing a way to detect generative effects in quite a general setting.

Another important application of preorders, monotone maps, and Galois connections is found in the analysis of programming languages. In this setting, preorders describe

- You can right click on PDF links to open a quick overview of the link destination. You can navigate in this overview using mouse wheel.
- Even if the PDF document has no links, you can right click on items and sioyek tries to open an overview of item destination. For example right clicking on the text "Figure 2.19" opens an overview showing (hopefully) Figure 2.19 . This also works with equations, tables, references, etc.
- You can also middle click on items to directly jump to their location instead of opening an overview.

1.2.6 Visual Mark



Example 1.115. Another example of closure operators comes from logic. This will be discussed in the final chapter of the book, in particular Section 7.4.5, but we will give a quick overview here. In essence, logic is the study of when one formal statement – or proposition – implies another. For example, if n is prime then n is not a multiple of 6, or if it is raining then the ground is getting wetter. Here "n is prime," "n is not a multiple of 6," "it is raining," and "the ground is getting wetter" are propositions, and we gave two implications.

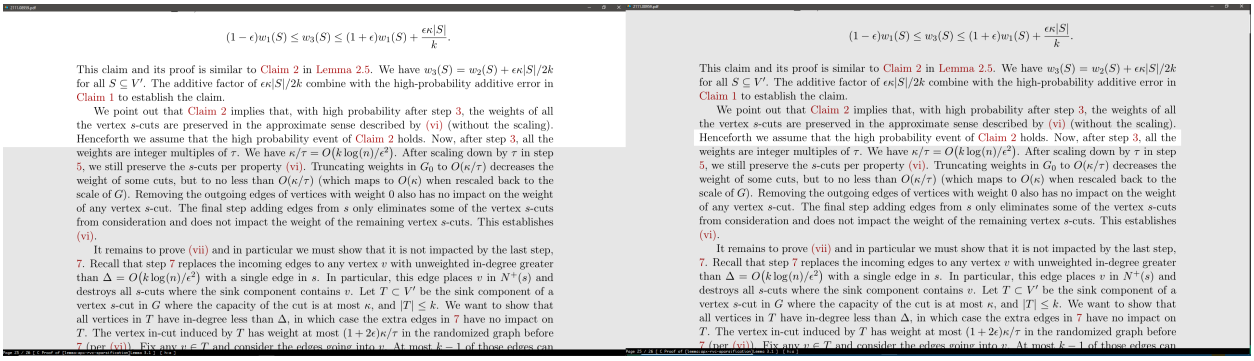
Take the set of all propositions and order them by $p \leq q$ iff p implies q, denoted $p \Rightarrow q$. Since $p \Rightarrow p$ and since whenever $p \Rightarrow q$ and $q \Rightarrow r$, we also have $p \Rightarrow r$, this is indeed a preorder.

A closure operator on it is often called a *modal operator*. It is a function j from propositions to propositions, for which $p \Rightarrow j(p)$ and $j(j(p)) = j(p)$. An example of a j is "assuming Bob is in San Diego . . . ." Think of this as a proposition B; so "assuming Bob is in San Diego, p" means $B \Rightarrow p$. Let's see why $B \Rightarrow -$ is a closure

If you right click on a line of text in a PDF file, sioyek displays a visual highlight below that line (we refer to this as the "visual mark" from now on). This visual mark has multiple use-cases.

1. It can act as a simple mark (see *Marks*). You can always return to the last location of visual mark by pressing and then right-clicking or pressing again. This can be useful when you want to quickly check previous pages and jump back to where you were.
2. It highlights the current line being read which reduces eye strain and prevents losing your place in PDF files.

- You can move the visual mark to the next/previous line by pressing `j/k` (`move_visual_mark_down` and `move_visual_mark_up` commands).
- You can toggle visual scroll mode by pressing `f7` (`toggle_visual_scroll` command). In this mode, mouse wheel moves the visual mark up and down.
- If you set `ruler_mode 0` in your `prefs_user.config`, we just highlight below the current line, if you set `ruler_mode 1` we draw a rectangle around the current line. The following figure shows the difference between these two settings.



- While a line is highlighted, you can press `l` (`overview_definition` command) to create an overview to the

reference in current line (for example if there is a `Figure 2.3` in the current highlighted line, we create an overview to the figure location. If there are multiple references in the same line, we use the first reference)

- Similarly you can press `]` (`portal_to_definition`) and `ctrl+]` (`goto_definition`) to create a portal or jump to the definition.

1.2.7 Search

- Press `/` or `<Ctrl f>` to open the search menu. (`search` command)
- Once a search is initiated, you can press `n` to go to the next match or `N` to go to the previous match. (`next_item` and `previous_item` commands)
- You can enter `c/` to search only the current chapter. (`chapter_search` command).
- You can limit the search range to a specific page range using the following syntax in the search bar `<begin, end>search term`. For example, if you want to search for the term 'Figure' in pages 20 to 30, you can enter the following:

```
<20,30>Figure
```

- Using `overview_next_item` and `overview_prev_item` commands, you can open an overview to search results instead of jumping to them.

1.2.8 Marks

Sometimes while reading a document, you need to glance at the contents of previous pages and quickly return to the original location. For example, maybe you forgot the definition of a term that was defined a few pages ago, or perhaps the current paragraph is referencing a previous figure. Using marks, you can mark your location before glancing at previous content and then quickly return to the location of the mark.

- In order to create a mark, first press `m` (`set_mark` command) followed by an alphabet symbol (this symbol will be the name of the mark, you can have multiple marks with different names). For example, in order to create a mark named `a` for your current location, enter `ma`.
- You can go to a mark by pressing followed by the name of the mark (note that is different from single quote `'`. usually is on the same key as ~). For example, in order to go back to the mark created in the previous example, enter . (`goto_mark` command)
- Marks are persistent (they are saved even when sioyek is closed).
- Lower case marks are local to the current document but upper case marks are global across sioyek.

1.2.9 Bookmarks

Bookmarks are similar to marks except they use a textual description instead of an alphabet letter.

- You can add a bookmark by pressing `b` (`add_bookmark` command) and then entering a textual description of the bookmark in the opened menu.
- You can open a searchable list of the bookmarks in the current document by entering `gb` (`goto_bookmark` command).
- You can open a searchable list of all the bookmarks in all documents by entering `gB` (`goto_bookmark_g` command).

- You can delete the closest bookmark to the current location by entering `db` (`delete_bookmark` command). You can also delete bookmarks directly in the bookmarks list by selecting the bookmark and then pressing the `delete` button on keyboard.

1.2.10 Highlights

for every s and adding $s \leq u$ whenever $s \leq t$ and $t \leq u$.

Exercise 1.117. Let $S = \{1, 2, 3\}$. Let's try to understand the adjunction discussed above.

1. Come up with any preorder relation $\leq$ on S, and define $U(\leq)$ to be the subset $U(\leq) := \{(s_1, s_2) \mid s_1 \leq s_2\} \subseteq S \times S$, i.e. $U(\leq)$ is the image of $\leq$ under the inclusion $\mathbf{Pos}(S) \to \mathbf{Rel}(S)$, the relation "underlying" the preorder.
2. Come up with any two binary relations $Q \subseteq S \times S$ and $Q' \subseteq S \times S$ such that $Q \subseteq U(\leq)$ but $Q' \not\subseteq U(\leq)$. Note that your choice of Q, Q' do not have to come from preorders.

We now want to check that, in this case, the closure operation Cl is really left adjoint to the "underlying relation" map U.

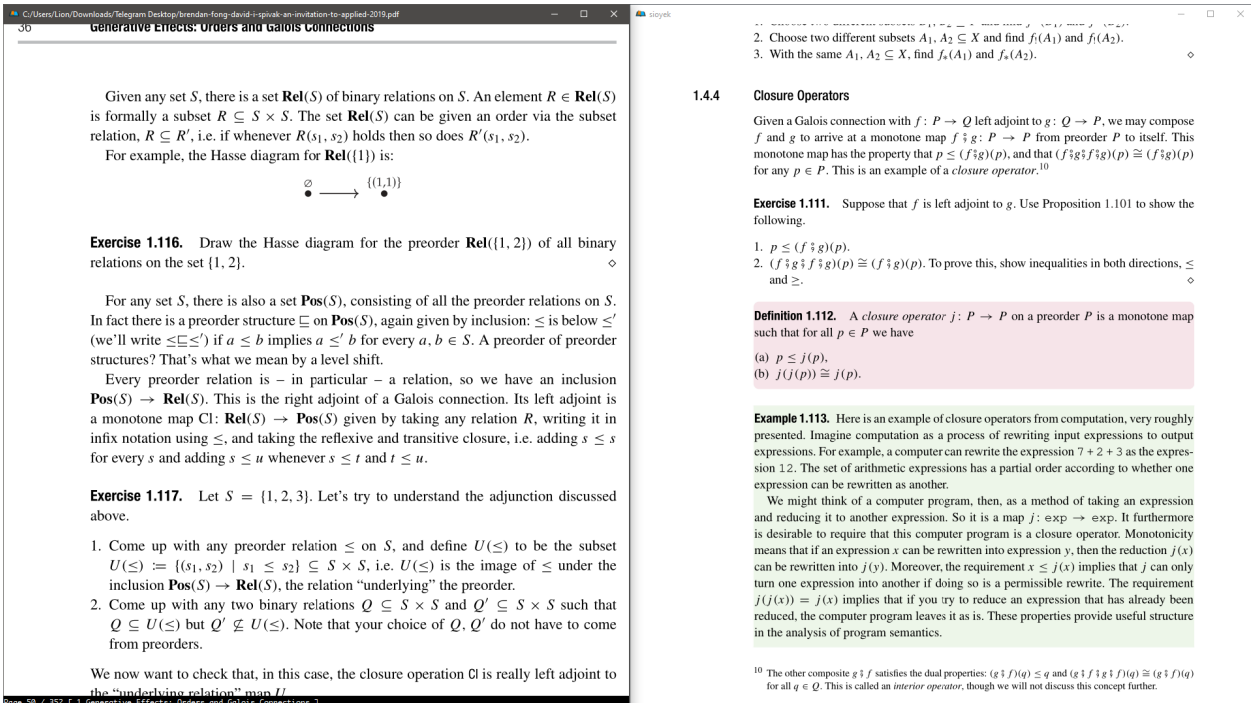
3. Concretely (without using the assertion that there is some sort of adjunction), show that $\mathsf{Cl}(Q) \sqsubseteq \leq$, where $\sqsubseteq$ is the order on $\mathbf{Pos}(S)$, defined immediately above this exercise.
4. Concretely show that $\mathsf{Cl}(Q') \not\sqsubseteq \leq$. ◇

1.5 Summary and Further Reading

You can highlight text in sioyek. Similar to marks, you can assign a "type" to each highlight using a lower case alphabet letter. Highlights of different types are distinguished by different colors.

- Select a piece of text and then press `h` followed by a letter to create a highlight of that letter type. For example in order to create a highlight of type "a" enter `ha`. (`add_highlight` command)
- Alternatively, if you don't want to specify highlight type every time, you can execute `add_highlight_with_current_type` command which uses the current highlight type to highlight the selected text. You can change the current highlight type by executing `set_select_highlight_type` command.
- You can toggle select highlight mode by executing `toggle_select_highlight` command. While in this mode, all selected text will be highlighted using the selected highlight type.
- Press `gh` to open a searchable list of the highlights in the current document. (`goto_highlight` command)
- Press `gH` to open a searchable list of the highlights in all documents. (`goto_highlight_g` command)
- You can execute `goto_next_highlight` and `goto_prev_highlight` to navigate the highlights in current document. If you want to navigate the highlights of selected type, you can use `goto_next_highlight_of_type` and `goto_prev_highlight_of_type` commands.
- In order to delete a highlight, first left click on the highlight and then enter `dh` (`delete_highlight` command). Alternatively, you can press the `delete` button on the keyboard in the highlight list.

1.2.11 Portals



Sometimes, a paragraph is entirely about a previous part of the document. For example maybe the paragraph is explaining the data in a previous table or describing a previous figure or answering a previous exercise. In such situations, it is usually very annoying to keep alternating between the paragraph and the referenced material. Using portals, you can link the location of the paragraph with the location of the referenced material and whenever you scroll to the paragraph, the referenced content is automatically displayed in a secondary window.

- In order to create a portal, press `p` (`portal` command, note that portals are called "links" in the sioyek source code, however, in the documentation we refer to them as portal so as not to confuse them with PDF links). This creates an incomplete portal with the current location as the source. Now, navigate to the location of the referenced material and press `p` again. This completes the portal with the second location as the destination.
- Alternatively, you can create portals by pressing `p` and then clicking on a PDF link. This automatically creates a portal from the location of the link to the link destination. Also you can middle click after pressing `p` which uses *SmartJump* to determine the destination.
- To view portal destinations, you need to open the helper window. You can open/close the helper widow by pressing `f12` (`toggle_window_configuration` command). This window automatically shows the destination of the portal with the closest source to the current location. This window is more useful for users with multiple monitors.
- You can delete the closest portal by entering `dp` (`delete_portal` command)
- You can go to the destination of the closest link by executing `goto_portal` command.
- In order to edit the destination of the current link, press `Shift p` (`edit_portal` command). This automatically jumps you to the destination of the link. Now you can adjust the screen or zoom level. When you are done, return to the source location by going back in history (`backspace` by default).
- You can also edit the link destination by directly panning the helper window using mouse or scrolling using the mouse wheel. You can also adjust the zoom level using the mouse wheel while holding `Ctrl`.

1.2.12 Command Menu

You can open the command menu by pressing `:` (`command` command). This is a searchable list of all the commands available in sioyek (including all the commands described above) along with their current keybindings. Note that some commands do not have any keybindings. The only way to execute these commands is using the commands menu.

1.2.13 External Search

- Select a piece of text and enter `ss/sl` (`external_search` command followed by a letter `a-z`) to search it in google scholar/library genesis.
- You can also middle click or shift-middle click on the name of papers/books to quickly search them in google scholar or other search engines. You don't need to select the text, sioyek will automatically try to guess the paper name on which you middle clicked.
- You can configure the search engines using `search_url_*` configurations in `prefs_user.config`. The `*` stands for any letter `a-z`. For example if I have `search_url_g https://www.google.com/search?q=` in my `prefs_user.config`, then sioyek will use google to search selected texts when I press `sg`.
- You can configure which search engines to use for middle click or shift-middle click searches using `middle_click_search_engine` and `shift_middle_click_search_engine` configurations in `prefs_user.config`. The value of these configurations should be a letter corresponding to `search_url_*` configs. For example if I want to use the google search from previous example when middle clicking, I should add the following to my `prefs_user.config`:

```
middle_click_search_engine g
```

1.2.14 Synctex

- Press `f4` to toggle synctex mode (`toggle_synctex` command). While in this mode, right clicking on any text opens the corresponding `tex` file in the appropriate location.
- You can configure the synctex inverse search command using `inverse_search_command` config in `prefs.config` . Here is an example for VS Code (%1 stands for the name of the file and %2 stands for the line number in the file):

```
inverse_search_command "C:\path\to\vscode\Code.exe" "C:\path\to\vscode\resources\
↪app\out\cli.js" --ms-enable-electron-run-as-node -r -g "%1:%2"
```

- Here is a sample configuration for latex workshop in VS Code:

```
"latex-workshop.view.pdf.external.synctex.command": "C:\\path\\to\\sioyek.exe",
"latex-workshop.view.pdf.external.synctex.args": [
"--inverse-search",
 "\"C:\\path\\to\\vscode\\Code.exe\" \"C:\\path\\to\\vscode\\resources\\app\\out\\
↪cli.js\" --ms-enable-electron-run-as-node -r -g \"%1:%2\"",
 "--reuse-instance",
 "--forward-search-file",
 "%TEX%",
 "--forward-search-line",
 "%LINE%",
 "%PDF%" ]
```

- Here is a sample configuration for VimTeX:

```
let g:vimtex_view_method = 'sioyek'
```

1.2.15 Data and Synchronization

Sioyek stores your data in two database files named `local.db` and `shared.db`. As the name suggests, `local.db` stores system-specific data (for example the location of PDF files in your filesystem) while `shared.db` stores all other data including marks, bookmarks, portals, etc. As the name suggests, `shared.db` can be shared across machines. There is also a `shared_database_path` config which you can set in your `prefs_user.config` which specifies the path of this shared database file. For example you can set this path to be a file in your dropbox directory and this way all your data will automatically be synchronized across your machines.

You can also export/import your data into/from a `json` file by executing the `export/import` command (see *Commands*).

1.2.16 Window Configuration

- Toggle fullscreen mode using `f12` (`toggle_window_configuration` command)
- You can save your current window configuration *copy_window_size_config* command.

1.2.17 Miscellaneous

- Copy text by first selecting it and pressing `<Ctrl c>`. (`copy` command)
- You can open the links in PDF files from keyboard by pressing `f` and entering the number next to the desired link. (`open_link` command)
- Press `f8` to toggle dark mode. (`toggle_dark_mode` command)
- Press `f5` to toggle presentation mode (`toggle_presentation_mode` command).
- You can use `toggle_horizontal_scroll_lock` command to prevent the document from being scrolled horizontally (can be useful for touchpad users).
- Use `toggle_custom_color` to toggle the custom color scheme configured in *Configuration*.
- Use `execute` to open a command line where you can execute shell commands. In this command line `%1` expands to the full path of the current file, `%2` expands to current file name and `%3` expands to current selected text. You can also predefine commands and execute them quickly using `execute_predefined_command`. See this post for examples of what is possible.
- You can export a version of current PDF file with all bookmarks/highlights embedded in it (so it is available to other PDF software) by executing `embed_annotations` command.
- You can rotate the page by executing `rotate_clockwise` and `rotate_counterclockwise` commands.

1.3 Commands

All sioyek keybindings execute some command. You can open a searchable list of all sioyek commands by pressing `:` by default. Below is a list of all available sioyek commands.

1.3.1 goto_begining

Go to the beginning of the document. If prefixed with a number, it will go to that page number.

1.3.2 goto_end

Go to the end of the document. If prefixed with a number, it will go to that page number.

1.3.3 goto_definition

While the visual ruler is highlighting a line, executing `goto_definition` will jump to the location of reference in that line. For example, executing this command while a line containing "Figure 2.1" is highlighted will jump to the location of the figure in the document.

1.3.4 overview_definition

Similar to `goto_definition`, but instead of jumping to the location of the reference, it will show a small overview of the reference in the overview window.

1.3.5 portal_to_definition

Similar to `goto_definition`, but instead of jumping to the location of the reference, it will create a portal from current location to the location of the definition.

1.3.6 next_item

While a search is in progress, this command will jump to the next search result.

1.3.7 previous_item

While a search is in progress, this command will jump to the previous search result.

1.3.8 set_mark

Mark the location with the symbol entered following the `set_mark` command.

1.3.9 goto_mark

Go to the mark previously set using `set_mark`.

1.3.10 goto_page_with_page_number

Go to the page number corresponding to entered text.

1.3.11 search

Search for a text in document.

1.3.12 regex_search

Search for a regular expression in the document.

1.3.13 move_down, move_up, move_left, move_right

Move the document.

1.3.14 zoom_in and zoom_out

Change the zoom level.

1.3.15 fit_to_page_width and fit_to_page_height

Fit the current page to window width/height.

1.3.16 fit_to_page_width_smart and fit_to_page_height_smart

Fit the current page (ignoring white margins) to window width/height.

1.3.17 next_page and previous_page

Navigate to the next/previous page.

1.3.18 open_document

Open the file browser to open a document.

1.3.19 add_bookmark

Add a bookmark to the current location at the document with the entered text.

1.3.20 add_highlight

Add the selected text as a highlight with the type of following symbol.

1.3.21 goto_toc

Open the table of contents.

1.3.22 goto_bookmark

Open bookmark list of current document.

1.3.23 goto_highlight

Open highlight list of current document.

1.3.24 goto_bookmark_g

Open bookmark list of all documents.

1.3.25 goto_highlight_g

Open highlight list of all documents.

1.3.26 portal

Set the current location as the portal source. If a portal source is already set, then use the current location as the portal destination and create the portal.

1.3.27 next_state, prev_state

Go forward/backward in history.

1.3.28 delete_bookmark

Delete the closest bookmark to the current location.

1.3.29 delete_highlight

Delete the last highlight clicked on.

1.3.30 delete_portal

Delete the closest portal to the current location.

1.3.31 goto_portal

Go to the destination of the closest portal source to the current document location.

1.3.32 edit_portal

Jump to the destination of the closest portal. All the edits you do to the location/zoom is applied to the portal when you go back by executing `prev_state`.

1.3.33 open_prev_doc

Open a list of all opened document using sioyek.

1.3.34 open_document_embedded

Open an embedded file browser in sioyek.

1.3.35 open_document_embedded_from_current_path

Open an embedded file browser in sioyek. Initially, we are in the directory of current document.

1.3.36 copy

Copy the selected text into clipboard.

1.3.37 toggle_fullscreen

Toggle fullscreen mode.

1.3.38 toggle_one_window

Open/close the helper window.

1.3.39 toggle_highlight

Toggle whether we highlight PDF links.

1.3.40 toggle_synctex

Toggle synctex mode. In synctex mode, right clicking on a PDF file opens the corresponding location in the latex file using the configured text editor.

1.3.41 command

Open a list of all sioyek commands.

1.3.42 external_search

Search the selected text in the extenal search engine corresponding to the following symbol.

1.3.43 open_selected_url

Open the selected URL in a browser.

1.3.44 screen_up and screen_down

Move the screen up/down.

1.3.45 next_chapter and prev_chapter

Go to the next/previous chapter in the current document.

1.3.46 toggle_dark_mode

Switch between light/dark colorschemes.

1.3.47 toggle_presentation_mode

Toggle presentation mode. In presentation mode we only show one page at a time and the current page fills the entire window.

1.3.48 toggle_mouse_drag_mode

Toggle mouse drag mode. In mouse drag mode, clicking and dragging using mouse moves the document (instead of selecting text).

1.3.49 close_window

Close the current window.

1.3.50 quit and q

Quit sioyek (closes all windows).

1.3.51 open_link

Open the PDF links using keyboard. Each link will be shown with a label and the link corresponding to the entered label text will be opened.

1.3.52 keyboard_select

Select text using keyboard. Each word in the document will be shown with a label. You should enter the labels corresponding to the beginning and end of the selection (separated by a space).

1.3.53 keyboard_smart_jump and keyboard_overview

Perform a smartjump (or open an overview) using keyboard. Each word in the document will be shown with a label and we will smart jump to the location of the reference corresponding to the entered label.

1.3.54 keys and prefs

Open the `keys.config` / `prefs.config` file.

1.3.55 keys_user and prefs_user

Open the `keys_user.config` / `prefs_user.config` file.

1.3.56 export and import

Export/import sioyek data to/from a json file.

1.3.57 enter_visual_mark_mode

Place a visual mark at the center of screen.

1.3.58 move_visual_mark_down and move_visual_mark_up

Move the visual mark up and down which highlights the previous/next line.

1.3.59 toggle_visual_scroll

Toggle visual scroll mode. In visual scroll mode, mouse wheel moves the ruler (the visual mark) instead of scrolling the document.

1.3.60 toggle_horizontal_scroll_lock

Toggles the horizontal scroll lock which prevents scrolling in horizontal direction (useful for touchpad users).

1.3.61 toggle_custom_color

Toggle custom color mode. In this mode, we change the text/background color of the document to the colors specified in `prefs_user.config`.

1.3.62 execute

Execute the entered text as a shell command.

1.3.63 execute_predefined_command

Execute the predefined command (in `prefs_user.config`) corresponding to the following symbol.

1.3.64 embed_annotations

Export a new version of the current document with all sioyek annotations embedded so that they are visible in other software.

1.3.65 copy_window_size_config

Copy the current window size configuration (useful when you want to set the configs in `prefs_user.config`).

1.3.66 toggle_select_highlight

Toggle select highlight mode. In this mode, just selecting the text automatically highlights it.

1.3.67 set_select_highlight_type

Set the highlight type used in select highlight mode.

1.3.68 open_last_document

Open the previous document.

1.3.69 toggle_window_configuration

Toggle between one window/two window configuration settings.

1.3.70 prefs_user_all and keys_user_all

Show a list of all `prefs_user.config` / `keys_user.config` files discovered by sioyek.

1.3.71 fit_to_page_width_ratio

Similar to `fit_to_page_width` but instead of fitting to the entire screen width, we fit to a ratio configured in `prefs_user.config`.

1.3.72 smart_jump_under_cursor and overview_under_cursor

Perform a smart jump (or open an overview) to the reference under the mouse cursor.

1.3.73 close_overview

Close the overview window.

1.3.74 visual_mark_under_cursor

Place a visual mark (ruler) under the mouse cursor.

1.3.75 close_visual_mark

Exit the visual mark (ruler) mode.

1.3.76 zoom_in_cursor and zoom_out_cursor

Zoom in/out on the mouse cursor.

1.3.77 goto_left, goto_right, goto_top_of_page and goto_bottom_of_page

Go to the left/right/top/bottom of the current page.

1.3.78 goto_left_smart and goto_right_smart

Go to the left/right side of the current page ignoring white margins.

1.3.79 rotate_clockwise and rotate_counterclockwise

Rotate the document.

1.3.80 goto_next_highlight and goto_prev_highlight

Go to the next/previous highlight in the current document.

1.3.81 goto_next_highlight_of_type and goto_prev_highlight_of_type

Go to the next/previous highlight in the current document with following symbol's type.

1.3.82 add_highlight_with_current_type

Add a highlight with the type specified using `set_select_highlight_type`.

1.3.83 enter_password

Enter the password for password-protected documents.

1.3.84 toggle_fastread

Highlight the first few characters of each word. Supposedly it may increase reading speed (unconfirmed).

1.3.85 new_window

Open a new sioyek window.

1.3.86 toggle_statusbar

Toggle the statusbar at the bottom of the screen.

1.3.87 reload

Reload the current document.

1.3.88 synctex_under_cursor

Perform a synctex search for the location under mouse cursor.

1.3.89 set_status_string

Set a message to be displayed in sioyek's statusbar.

1.3.90 clear_status_string

Clear the message in sioyek's statusbar.

1.3.91 toggle_titlebar

Toggle the window titlebar.

1.3.92 next_preview and previous_preview

If there are multiple possible previews for the overview window, move to the next/previous preview.

1.3.93 goto_overview and portal_to_overview

Go to / create a portal to the location displayed in the overview window.

1.3.94 goto_selected_text

Jump to the location of current selected text.

1.3.95 focus_text

If in visual mark (ruler) mode, focus the ruler on the line corresponding to the entered text.

1.3.96 goto_window

Open a searchable list of sioyek windows.

1.3.97 toggle_smooth_scroll_mode

Toggle smooth scroll mode. In this mode, scrolling is done smoothly.

1.3.98 toggle_scrollbar

Toggle the scrollbar.

1.3.99 overview_to_portal

Open the overview window to the closest portal to current document location.

1.3.100 select_rect

Select a rectangle (can be used in extensions using `%{selected_rectangle}` variable).

1.3.101 donate

Open donation page.

1.3.102 overview_next_item and overview_prev_item

Open overviews to previous/next search items.

1.4 Configuration

Sioyek uses four config files:

- `keys.config` which stores the default keybindings
- `keys_user.config` which stores the modified keybindings
- `prefs.config` which stores the default preferences
- `prefs_user.config` which stores the modified preferences

The location of these files are OS-dependent. You can open these files by executing `keys`, `keys_user`, `prefs` and `prefs_user` commands using the sioyek command line (see *Command Menu*).

1.4.1 Syntax of `keys.config` files

- Lines starting with # are comments and are ignored

```
command        k              (command is executed when k is pressed)
command        <C-k>          (command is executed when k is pressed while holding␣
↪control)
command        K              (command is executed when K is entered, which is shift+k)
command        <A-k>          (command is executed when k is pressed while holding alt)
command        +              (command is executed when = is pressed while holding shift)
command        <C-S-k>        (command is executed when k is pressed while holding␣
↪control and shift)
command        gg             (command is executed when g is pressed twice)
command        gt             (command is executed when g is pressed and then t is␣
↪pressed)
command        g<C-n>Dt       (command is executed when g is pressed and then n is␣
↪pressed while holding\
                               control and then d is pressed while holding shift and then␣
↪t is pressed)

                               you can execute multiple commands using the following␣
↪syntax:
command1;command2;command3         <keybinding>
```

1.4.2 Prefrences in `prefs.config` file

`background_color`

Specifies the background color of the app (this is different from the background color of PDF page which is configured using *custom_background_color and custom_text_color*, this color is only shown when the displayed page is smaller than the srceen). The syntax to set colors is:

```
background_color  r  g  b
```

where r,g and b red, green and blue values between 0 and 1. For example in order to set the background color to gray we can add the following to our `prefs_user.config`:

```
background_color  0.5  0.5  0.5
```

dark_mode_background_color

Specifies the background color when dark mode is enabled.

dark_mode_contrast

White text in dark mode can be annoying for the eyes. This option allows us to dim the white colors when dark mode is enabled. Allowed values are between 0.0 and 1.0.

text_highlight_color

Highlight color when text is selected using mouse.

visual_mark_color

This is the color of the transparent visual mark explained in *Visual Mark* (this feature originally had an entirely different functionality which is why it is called vertical line even though there is nothing vertical about it!). Allowed values are RGBA colors between 0.0 and 1.0. For example, to set the color to a transparent red we add the following to our `prefs_user.config`:

```
visual_mark_color  1.0  0.0  0.0  0.1
```

ruler_mode

If it is 1, we highlight a rectangle around the current line in visual mark mode. Otherwise, we highlight below the current line.

```
ruler_mode  1
```

ruler_padding and ruler_x_padding

Additional padding for ruler. Makes the ruler a little larger and more readable.

```
ruler_padding 1.0
ruler_x_padding 5.0
```

visual_mark_next_page_fraction

When we go to the next page while in visual mark mode, this setting determines which location of screen the new line should be located at. The values are between -1 and 1. With 0 being the middle of the screen and 1 and -1 being the top and bottom of the screen respectively.

```
visual_mark_next_page_fraction  0.5
```

visual_mark_next_page_threshold

Determines at which point in screen we move to the next page. Acceptable range is between 0 and `visual_mark_next_page_fraction`.

search_highlight_color

The color used to highlight search results.

link_highlight_color

The color used to highlight links in PDF files.

synctex_highlight_color

Highlight color for synctex forward search highlights.

search_url_a to search_url_z

The web addresses used for performing `external_search` command. (see *External Search*). Example:

```
search_url_g https://www.google.com/search?q=
```

middle_click_search_engine and shift_middle_click_search_engine

The letter corresponding to `search_url_*` configs to use when middle clicking/shift-middle clicking on text. Example:

```
middle_click_search_engine          g
```

This causes the search engine configures using `search_url_g` to be used when middle clicking on text.

zoom_inc_factor

The fraction by which we enlarge the page when zooming in/out.

wheel_zoom_on_cursor

If set, when using mouse wheel to zoom we zoom in on mouse cursor instead of middle of screen.

vertical_move_amount and horizontal_move_amount

How many inches we move vertically/horizontally when performing move_* commands.

move_screen_ratio

The fraction of screen by which we move when executing `screen_down` and `screen_up` commands. (note that despite the name, the values are fractions between 0 and 1, not percentages)

flat_toc

Displays a simplified flat table of contents instead of a hierarchial one. This can improve performance for documents with very large number of table of contents entries (thousands). Acceptable values are 0 and 1.

collapsed_toc

If set, we initially collapse all table of content entries.

should_use_multiple_monitors

If it is 1, when launching the application if we detect multiple monitors, we automatically launch the helper window in second monitor. Acceptable values are 0 and 1.

should_load_tutorial_when_no_other_file

If the last opened document is empty, load the tutorial pdf instead.

should_launch_new_instance

> **Warning:** This is deprecated. Use *should_launch_new_window* instead.

If it is 0, then we use the previous instance of sioyek when launching a new file, otherwise a new instance is launched every time we open a new file.

should_launch_new_window

If it is 0, then we use the previous window of sioyek when opening a new file, otherwise a new window is opened every time we open a new file.

inverse_search_command

The command to use when trying to do inverse search into a LaTeX document. %1 expands to the name of the file and %2 expans to the line number. For example:

```
inverse_search_command                "C:\path\to\vscode\Code.exe" -r -g %1:%2
```

highlight_color_a to highlight_color_z

The color to use for highlights of type a to z.

should_draw_unrendered_pages

If set, we display a checkerboard pattern for unrendered pages (by default we display nothing).

rerender_overview

Normally we reuse the rendered page for overview window. This may cause the overview page to be blurry or too sharp if there is a significant difference between the zoom levels of the main window and overview window. If rerender_overview is set, we rerender overview which solves this issue at the cost of some additional computation.

```
rerender_overview             1
```

default_dark_mode

Use dark mode by default.

sort_bookmarks_by_location

If set, we sort the bookmarks by their location instead of their creation time.

shared_database_path

The path of shared.db database file. You can set this path to be in a synchronized folder (for example a dropbox folder) and sioyek data will be automatically synchronized across your devices.

font_size

Size of the UI font.

ui_font

The font to use for UI text.

```
ui_font              Segoe UI Emoji
```

item_list_prefix

A prefix character to use before list of items (for example can be used to display a checkmark before each of the bookmarks).

```
item_list_prefix     ✓
```

check_for_updates_on_startup

If set, sioyek checkes for new versions on startup and notifies the user if a new version if available.

```
check_for_updates_on_startup 1
```

custom_background_color and custom_text_color

Specify the background and text color when using custom color mode (by executing `toggle_custom_color` command).

```
custom_background_color              0.18 0.20 0.25
custom_text_color                    1.0 1.0 1.0
```

startup_commands

Semicolon-separated list of commands to execute on startup. For example, to start in custom color mode and in visual scroll mode, you can add the following (the command names are the same as the ones displayed when opening the command window using `:`):

```
startup_commands             toggle_custom_color;toggle_visual_scroll
```

status_bar_color, status_bar_text_color and status_bar_font_size

Allow you to customize the appearance of status bar.

```
status_bar_color        0 0 0
status_bar_text_color   1 1 1
status_bar_font_size    10
```

execute_command_a to execute_command_z

Predefined shell commands to be executed using `execute_predefined_command`. `%1` expands to the path of the current file, `%2` expands to name of the current file and `%3` expands to current selected text. For example, suppose you have a command named `ocr` which takes a file path and produces an OCR'd version of the document. You can add the following to you `prefs_user.config`:

```
execute_command_o    ocr "%1"
```

You can later quickly invoke this command by executing `execute_predefined_command` and then pressing `o`.

Warning: The command parsing code in sioyek is not very good. For example it can not handle multiple commands like `command1 args;command2` or commands that include spaces. If you want to run a complex command, first put all commands in a script file and then run the script file using using sioyek like this: `/path/to/script.sh %1 %2 %3`.

papers_folder_path

Path to a directory on your computer. Sioyek monitors the changes in this directory and if a new file is added to this directory while we have a pending portal, this file is automatically used as the destination of the portal. This is useful when creating a portal from a reference in a paper to the actual reference file.

display_resolution_scale

Manual resolution scaling. Can be useful for some very high resolution displays which report the wrong resolution.

linear_filter

If set, we use linear texture filtering instead of the normal nearest neighbour filtering. This is useful when using manual display resolution scale which causes the nearest neighbour filter to look bad.

main_window_size, main_window_move, helper_window_size, helper_window_move, single_main_window_size and single_main_window_move

Configures the size and position of the main window and the helper window. `single_main_window_*` is used when helper window is closed and the other configs are used when both windows are opened. These values are automatically written to `auto.cong` file when sioyek exits but you can manually override them by setting them in your `prefs_user.config`.

```
single_main_window_size    1824 988
single_main_window_move     22 21
main_window_size    1824 988
main_window_move     18 44
helper_window_size    1891 1033
helper_window_move     1951 0
```

touchpad_sensitivity

Can be used to adjust the sensitivity of the touchpad.

```
touchpad_sensitivity    0.1
```

page_separator_width and page_separator_color

Used to adjust the appearance of page separator.

```
page_separator_width 2
page_separator_color 0.5 0.5 0.5
```

fit_to_page_width_ratio

Ratio of screen width to use when using `fit_to_screen_width_ratio` command. Can be useful for very wide screens.

```
fit_to_page_width_ratio 0.75
```

create_table_of_contents_if_not_exists

If set and the file doesn't have a table of contents, we use heuristic methods to create a table of contents. You can use `max_created_toc_size` to prevent creating very large table of contents.

```
create_table_of_contents_if_not_exists 1
max_created_toc_size 5000
```

force_custom_line_algorithm

Use legacy line detection algorithm instead of the mupdf one.

overview_size and overview_offset

Adjust the size of overview window. The values are in normalized window coordinates between -1 and 1.

```
overview_size 0.852604 0.597729
overview_offset -0.0119792 0.120151
```

show_doc_path

Show the full document path instead of just the file name in list of recently opened documents.

should_warn_about_user_key_override

If set, we warn the user in command line when overriding already degined keybinds.

single_click_selects_words

If set, single clicking and dragging mouse selects entire words rather than characters.

shift_click_command, control_click_command, alt_click_command, shift_right_click_command, control_right_click_command, and alt_right_click_command

Custom commands to run when mouse click is pressed while modifier keys are held down. For example:

```
control_click_command overview_under_cursor
```

If set, single clicking and dragging mouse selects entire words rather than characters.

use_legacy_keybinds

By default we use legacy keybindings which have some problems. For example to bind the % key, you would have to enter something like this:

```
command <S-%>
```

which is a little weird. Also legacy keybinds don't work well with some keyboard layouts. If you set `use_legacy_keybinds` to 0, then we use a new method for keybind parsing which allows you to do something like this:

```
command %
```

which also works with all keyboard layouts. Since this is a backwards incompatible change, `use_legacy_keybinds` is activated by default.

multiline_menus

If set, we show long menu items in multiple lines rather than truncating them.

start_with_helper_window

Open helper window when sioyek starts.

prerender_next_page_presentation

When in presentation mode, we pre-render the next page to remove flickering when moving between pages.

highlight_middle_click

If enabled, highlights the selected text by middle clicking on it.

smooth_scroll_speed and smooth_scroll_drag

When smooth scrolling is enabled (using `toggle_smooth_scroll` command), these control the speed and slowdown of movement.

```
smooth_scroll_speed 3
smooth_scroll_drag 3000
```

super_fast_search

If enabled, indexes the document text and considerably speeds of searching. It also enables `regex_search` command which uses the index to search for regular expressions.

show_closest_bookmark_in_statusbar

If enabled, displays the text for closest bookmark in the statusbar.

show_close_portal_in_statusbar

If enabled, it displays a statusbar message when we are close to a portal.

prerender_page_count

The number of pages to prerender. The larger it is, the more memory we use but there will be less flickering when quickly moving pages.

case_sensitive_search

It is enabled by default. If disabled, we use case-insensitive searching.

show_document_name_in_statusbar

Displays the document name in sioyek's statusbar.

ui_background_color and ui_text_color

The background and text color of (unselected) UI elements.

ui_selected_background_color and ui_selected_text_color

The background and text color of selected UI elements.

numeric_tags

If enabled we use numeric (instead of alphabetical) tags when executing `keyboard_*` commands.

source

Includes another config file, which is useful for themes and extensions.

```
# includes file.config in this configuration file
source /path/to/file.config
```

1.5 Scripting and Extensions

1.5.1 Using existing extensions

Before writing your own extensions, you should check out some existing useful extensions in this repository, along with instructions on how to enable them in sioyek.

1.5.2 Defining custom commands

Sioyek allows you to define your own custom commands and extensions. For example adding this

```
new_command _read_selected_text espeak "%{selected_text}"
```

to your `prefs_user.config` adds a `_read_selected_text` command to sioyek that read the selected text aloud using espeak (of course you have to have espeak executable in your path). Note that custom command names must begin with an underscore so as not to be confused with built-in sioyek commands. You can use custom commands the same way you would use sioyek commands. For example you can bind them to keys in your `keys_user.config` like this:

```
_read_selected_text <C-r>
```

or you could search and run them using sioyek's command palette.

Warning: The way we parse `new_command` is very simple: we split the text following the command name by spaces and treat the first part as the name of executable and the rest as parameters. So first of all, you have to escape the spaces in command using backslash. Secondly something like this would not work:

```
new_command _some_command echo "%{file_path}" > somefile.txt
```

because we would be executing the `echo` command with three parameters: `%{file_path}`, > and `somefile.txt`. So if you want to do a complex command using piping, you would have to first create a script that does the piping and call the script from sioyek.

1.5.3 Input parameters

Sioyek provides the following possible inputs to custom commands:

- `%{file_path}`: expands to current document's full path
- `%{file_name}`: expands to current document's file name
- `%{selected_text}`: expands to current selected text
- `%{selection_begin_document}`: if some text is selected expands to the page number and x and y coordinates in the page's coordinate system of the start of the selection
- `%{selection_end_document}`: if some text is selected expands to the page number and x and y coordinates in the page's coordinate system of the end of the selection
- `%{line_text}`: expands to text of the current selected line when in sioyek's visual line mode
- `%{page_number}`: expands to current page number (zero-indexed)
- `%{command_text}`: If this argument is present in the commands, sioyek prompts the user to enter a text and expands `command_text` to that text.
- `%{mouse_pos_window}`: exapnds to mouse x and y position in window coordinates in pixels
- `%{mouse_pos_document}`: expands to the page number and x and y coordinates of the mouse in the page's coordinate system
- `%{paper_name}`: expands to the paper name under mouse cursor
- `%{sioyek_path}`: expands to the path of sioyek's executable
- `%{local_database}`: expands to the path of sioyek's local database file
- `%{shared_database}`: expands to the path of sioyek's shared database file
- `%{zoom_level}`: expands to the current document's zoom level
- `%{selection_begin_document}` and `%{selection_end_document}`: expands to the current text selection in document space
- `%{selected_rect}`: expands to the current selected rectangle using `select_rect` command

Warning: In some builds of sioyek (for example AppImages), `sioyek_path` does not expands to the correct path of the AppImage. In such cases you would have to just manually replace `%{sioyek_path}` with the absolute path of sioyek's executable.

1.5.4 Controlling sioyek

Scripts can also communicate back to sioyek by executing commands. You can execute all sioyek commands from the command line by running:

```
sioyek --execute-command command_name
```

for example to zoom in, you could run:

```
sioyek --execute-command zoom_in
```

You can also run commands that require text/symbol by specifying `execute-command-data`. For example:

```
sioyek --execute-command add_bookmark --execute-command-data "this is a bookmark made␣
↪from command line"
```

One of the most useful commands for extensions is `set_status_string` which shows the given text in sioyek's statusbar. For example:

```
sioyek --execute-command set_status_string --execute-command-data "this is a status␣
↪message"
```

you can clear the status message by running `clear_status_string` command:

```
sioyek --execute-command clear_status_string
```

Of course, instead of running these commands manually, you could automate the process by using any programming language capable of executing command line programs. For example, here is a simple translator in python which shows the translated selected text in sioyek's statusbar:

```
import sys
from googletrans import Translator
import subprocess

if __name__ == '__main__':
   sioyek_path = sys.argv[1]
   text = sys.argv[2]
   translator = Translator()
   translation = translator.translate(text, dest='en')
   subprocess.run([sioyek_path, '--execute-command', 'set_status_string', '--execute-
↪command-data', translation.text])
```

and the corresponding config in `prefs_user.config`:

```
new_command _translate python /path/to/translate/script.py "%{sioyek_path}" "%{selected_
↪text}"
```

We have made a python wrapper around sioyek which makes writing extensions a little easier. You can download it by running:

```
pip install sioyek
```

Using the wrapper, the previous script can be simplified like this:

```
import sys
from googletrans import Translator

from sioyek import Sioyek, clean_path

if __name__ == '__main__':
    sioyek_path = clean_path(sys.argv[1])
    text = sys.argv[2]
    sioyek = Sioyek(sioyek_path)
    translator = Translator()
    translation = translator.translate(text, dest='en')
    sioyek.set_status_string(translation.text)
```

1.5.5 Coordinate spaces

All coordinates in database files are in "absolute document space", which might be a little confusing. MuPDF, (the PDF engine that we use) uses something that I call "document space" to specify positions in documents which is the following triplet:

- Page number (zero-indexed)
- x-offset in points relative to the top left of page (1 point = 1/72 inch)
- y-offset in points relative to the top left of page (1 point = 1/72 inch)

In absolute document space, we conceptually view the document as a list of pages stacked vertically. So we don't have page numbers anymore but the y-offset of previous pages are added, so for example, the following page in document space: `(2, 100, 200)` is translated to the following coordinate in absolute document space (note that in this example pages are zero-indexed, so page 2 is the third page of the document): `(100 - page_width[2] / 2, page_height[0] + page_heights[1] + 200)`. In order to convert between absolute document space and document space, you can use `to_absolute` and `to_document` functions in https://github.com/ahrm/sioyek-python-extensions/blob/main/src/sioyek/sioyek.py .

1.5.6 Database files

Sioyek stores all your data in two simple sqlite database files: `local.db` and `shared.db`. Using `%{local_database}` and `%{shared_database}`, you can pass the file path of these database files to your scripts, which are then allowed to read/write data directly to these files.

> **Warning:** Access to sioyek's local and shared database file is a classic great power/responsibility situation. You could easily wipe out your data if you are not careful. I recommend only adding to database files and deleting only when you know what you are doing.

local database file includes a single table named `document_hash` which maps file paths to their `md5` hash. We later use this hash to reference files. This allows us to keep bookmarks/highlights even when the document is moved to another location or even another machine.

Shared database files stores all your bookmarks, highlights, etc. The tables in `shared.db` are:

- **`bookmarks`: stores the bookmarks. Fields:**
  - `document_path`: `md5`-hash of the document (from `document_hash` table in `local.db`)
  - `desc`: the text description of the bookmark

  - `offset_y`: the y-offset of the bookmark in the absolute document space

- **`highlights`: stores the highlights. Fields:**
  - `document_path`: `md5`-hash of the document
  - `desc`: highlighted text
  - `type`: the type of highlight (the symbol used to create the highlight)
  - `begin_x`: the x-offset of first character of highlight in the absolute document space
  - `begin_y`: the y-offset of first character of highlight in the absolute document space
  - `end_x`: the x-offset of last character of highlight in the absolute document space
  - `end_y`: the y-offset of last character of highlight in the absolute document space
- **`links`: stores the portals (they used to be called links but changed their name so as not to be confused with PDF links). Fields:**
  - `src_document`: `md5`-hash of the source document
  - `dst_document`: `md5`-hash of the destination document
  - `src_offset_y`: the y-offset of the source document in absolute document space where portal is located
  - `dst_offset_x`: the x-offset of the destination of the portal in absolute document space
  - `dst_offset_y`: the y-offset of the destination of the portal in absolute document space
  - `dst_zoom_level`: the zoom level of the destination of the portal
- **`marks`: stores the marks:**
  - `document_path`: `md5`-hash of the document
  - `symbol`: The type of the mark (symbol used to create the mark)
  - `offset_y`: The y-offset of the mark in absolute document position
- **`opened_books`: stores a list of all opened books along with the current position and zoom level. Fields:**
  - `path`: `md5`-hash of the document
  - `offset_x`: current x-offset in the document (in absolute document space)
  - `offset_y`: current x-offset in the document (in absolute document space)
  - `last_access_time`: last time we accessed this document

For an example of how to use database files in extensions, see this script which extracts the highlights of the current document into a new document and creates portals from this new document to the corresponding locations in the original document.