it-edit-3.0-doc Documentation

Release latest

Apr 19, 2017

Contents

Contents:

CHAPTER 1

it-edit (Integrated Terminal Editor)

program it-edit

version 3.0

author Brüggemann Eddie

contact <mrcyberfighter@gmail.com>

license GPLv3

website <http://www.open-source-projects.net/it-edit/it-edit>

release Apr 19, 2017

Motivation for writing it-edit

I think they are 2 sort of programmers in their habits of writing programs:

- They which use an I.D.E (Integrated Development Environment) with full integrated functionalities, and master it to respond to most of their requirements.
- They which use an text editor to write their programs and a terminal to compile, debug or launch them and some other tools separately.

Because I'm an programmer from the second category and because i remark that i often use additional tools than the editor and the terminal.

I decide to write my own text editor program which provide me all the functionalities that i need to get a development environment fully adapt to my requirement.

So in fact **`it-edit`** is more than a basic programming text editor, but a powerful tool which I hope you will agree the concept.

Note: In fact in conjunction with my project generator mk-project which generate me a big **do all** `Makefile`.

I enjoy using it-edit every time !

Even when I only to type few targets like (make, make exec, make ddebug, make gdb,...) **it-edit** is useful for all task to do in a terminal.

All this with an accompanied editor.

note By the way mk-project can be useful for every **vim** or T.U.I (Terminal User Interface) editor user.

Requirement of it-edit

it-edit requires

- gtk-3 as G.U.I (Graphical User Interface)
- gtksourceview-3.
- vte-2.91 for the virtual terminals emulation as a gtk-3 widget.
- Optionally gspell-1 as *spell checker*.

  warning You must install the development packages of all required library because it will be compiled and installed on the target host.

What it-edit does for me !

it-edit consist of a basic programming text editor with all basic functionalities with

intelligently integrated terminals the best positioned so that they are `easy to reach` into the main interface of the program.

With many others practices functionalities.

—

it-edit integrated terminals in the best way I have thought for me.

A side bar terminal which you can add and remove items, a big full-screen terminal, the same divided into 4 terminals and as many top-level terminals as you want.

—

it-edit provide an easy file access...

In fact it was thought for registering **HTML** documentation and so accessing it **easily**.

But you can register your winner song if you want when you have terminate all your assertions well,

to celebrate your victory **!**

it-edit will open the registered file with the default program for it if any available.

Saving and restoring your file(s) list.

```
$ cp /usr/local/share/it-edit/Files_handler/Files_handler.conf $HOME
```

To restore your file(s) list after upgrade per example.

```
$ cp $HOME/Files_handler.conf /usr/local/share/it-edit/Files_handler/Files_
↪handler.conf
```

—

You can launch graphical applications with **it-edit** very easily:

- At first it-edit will check your system, at first start, for some predefined applications and register it into their related category.

  note You can change the application(s), if found or not, as you want.

- Then you can choose to register and delete the applications you want, into the category `Other`.

—

The version **3**.**0** of **it-edit** embedded a configurable spell checker utility which you can use to write documentations

reachable as

- **in-line spell-check** (*the misspelled words are underline*)

or

- through launching a **dialog spell-check** window which permit you to correct all misspelled words of the current document.

it-edit spirit

it-edit offers:

A programming editor

A text-editor without favoring any language.

With all the basic text editor functionalities and overall useful *shortcuts* which you didn't have to know all.

And some unusual like **duplicate text** or **copy to clipboard** the current edited absolute **file-path** (`Ctrl + Shift + Y`).

You will be able to open, open a recent, save, save as, save all, file(s).

Or reload your last session file(s) as documentation and launched applications.

it-edit provides some informations like:

- Line
- Column
- Total lines
- Total characters
- The filename (hold the mouse over the notebook tab, you will know the absolute file-path).

By opening the file informations you will get more informations and can do some basic functions on (and strictly over) the file on disk (**not** the current edited buffer):

- The file base-name.
- Language, Mime-type and Extension.
- Lines, Words and Chars count.
- MD5, SHA1, SHA256 and SHA512 checksum in hexadecimal notation.

You will be able to see and to modify:

- The file permissions.

  warning By saving the `configured mask` will overwrite your changes.

- The last
  - Access time
  - Modification time (can be useful with make and system clock disturbing).
  - Last status change time (this one isn't modifiable).
- *Rename*, *Copy*, *Compress* (*using the gzip algorithm*), *Move file*.

Terminal integration

it-edit provides the best terminals integration for everyone:

- The **sidebar terminals** can be useful by editing and requiring immediately a terminal in the same window.
- The **single "Big Terminal"** can be useful if you need space: you can mask the button bar and made it full-screen.
- The **4 divided "Big Terminal"** can be useful if you need severals terminals on the screen.
- The **top level terminal** is re-sizable and in a separate window.

  note It's recommended to use **it-edit** with a `Makefile` (easy self-build or not) for compiled languages, else enjoy the terminals for launching your scripts and commands.

`make makes the life easier !`

All this terminals have a good contextual menu and I had problems to add some items, as `sync current directory` in every terminal, and so on.

it-edit writing spirit

it-edit is written in C (-std=c99) using gtk-3 and related libraries.

it-edit writing style takes cares of:

- structure variables order: so that no unneeded padding is added from the compiler.
- cache optimizing: so that we get the best cache-hits we can.

it-edit make strong usage of the gtk-3 types.

it-edit provide an optional spell checker useful for writing documentation or simply comments.

it-edit make usage of `/** **/` comments so for commenting out a code section, by hacking it-edit, use the preprocessor: `#if 0 ... #endif`.

note The spirit of **it-edit** says that you have to do a thing a single time then it's automatize.

Conclusion

Control all your system with **it-edit** and show us that you can dialog with it through the best medium: the terminal.

CHAPTER 2

it-edit's menu

Files

- → *New file* (`Ctrl + n`)
- → *Open file* (`Ctrl + o`)
- → *Recent file(s)*

  ... List of recent files
- → *Save file* (`Ctrl + s`)
- → *Save file as*
- → *Save all file(s)* (`Ctrl + Shift + N`)
- → *Close file* (`Ctrl + Alt + c`)
- → *Close all files* (`Ctrl + Shift + C`)
- → *Reload :ref:'session <session>*
  - → *Reload entire session*
  - → *Reload last file(s)*
  - → *Reload last documentation*
  - → *Reload last application(s)*
  - → *Clear session*
- → *Reload current file* (`Ctrl + r`)
- → *File informations* (`Ctrl + i`)
- → *Quit* (`Ctrl + q`)

Edition

- → *Undo* (`Ctrl + z`)
- → *Redo* (`Ctrl + Shift + Z`)
- → *Find text* (`Ctrl + f`)
- → *Find all* (`Ctrl + Shift + F`)
- → *Find previous* (`Ctrl + -`)
- → *Find next* (`Ctrl + +`)
- → *Replace text* (`Ctrl + Enter (KP)`)
- → *Replace all* (`Ctrl + Shift + Enter (KP)`)
- → *Replace all in all files* (`Ctrl + Shift + R)`)
- → *Go to line number* (`Ctrl + g`)
- → *Cut* (`Ctrl + x`)
- → *Copy* (`Ctrl + c`)
- → *Paste* (`Ctrl + v`)
- → *Duplicate text* (`Ctrl + d`)
- → *Use space instead of tabs* (`Ctrl + p`)
- → *In-line spell check* (`Ctrl + w`)

Actions

- → *Execute command* (`Ctrl + e`)
- → *Order pages* (`Ctrl + Alt + y`)
- → *Copy file-path to clipboard* (`Ctrl + y`)
- → *Copy folder-path to clipboard* (`Ctrl + Shift + Y`)
- → *Sidebar terminals add tab* (`Ctrl + Shift + T`)
- → *Big terminals(s) switch* (`Ctrl + Shift + B`)
- → *File(s) handler* (`Ctrl + h`)
- → *Application launcher* (`Ctrl + a`)
- → *Spell check dialog* (`Ctrl + Shift + W`)

Applications

- → *Programming*
  - → *Diff G.U.I (Graphical User Interface) tool*
  - → *Debugger G.U.I (Graphical User Interface) tool*
  - → *Python smart interpreter G.U.I (Graphical User Interface) tool*

  - → *G.U.I (Graphical User Interface) designer tool*
  - → *devhelp*
- → *Utilities*
- → *Calculator*
- → *Color picker*
- → *Dictionary*
- → *File manager*
- → *Browser*
- → *User defined applications*

View

- → *Big terminal show*
- → *Sidebar terminals show*
- → *Button bar show*
- → *Full screen*

Settings

- → *Syntax highlight*
- → *Editor scheme*
- → *Configure spell check language*
- → *Configure program*

About

- → *Uptime*
- → *Notice*
- → *License*
- → *About*

CHAPTER 3

Editor

The editor has as functionalities :

Files management.

- New file.
- Open file(s) or a recent file.
- Save file(s).
- Close file.
- Close all file(s).
- File informations.
- Reload file (from disk).

Text edition functionalities.

- Undo/Redo.
- Search and Replace.
- Go to line number.
- Cut/Copy/Paste.
- Duplicate text.
- Use spaces instead of tabs.
- Enable/Disable in-line spell-check.

Contextual menu from editor

- Undo
- Redo

—

- Cut
- Copy
- Paste
- Erase

—

- Select all
- Change case
  - All to uppercase
  - All to lowercase
  - Invert case
  - First letter uppercase

  note If you're using the `In-line spell check` functionality an item with a *sub-menu* of **suggestions** is added to the **contextual menu** of the editor.

Spell-check functionality

- In-line spell-check (`Ctrl + w`) : the misspelled words are highlight.
- Spell-check dialog (`Ctrl + Maj + w`) : scan all the text buffer for misspelled words, with a dialog window for correct them or not.

  warning Install the optional **`gspell-1`** `-dev` or `-devel` package, before installing **`it-edit`**.

Go to line number

- Use the menu item, the button or the shortcut: `Ctrl + g`.

  A window will appear asking you for a valid line number.

  Enter a valid line number and press `Enter` or the *Apply* button to move the editor to the wanted line,

  which will be highlight according to the current scheme.

  note This is very useful to go to a specific line.

Find and replace

- You can *show* | *hide* the `Find and replace` bar by using the toggle button.

  The `Search and Replace` bar has following functionalities:

- **Search** button: this will highlight all the matching occurrences from the search term and moving the editor to the first occurrence position.
- **Next** button: highlight the next matching occurrence from the search term.

  Pressing **Enter** when the search entry field has the focus has the same effect.

  The search will start at selection or at the cursor position.

  If you select some text with the mouse and using the shortcut `Ctrl + f`.

  The search field will toggle on if not visible and will contains the selected text as search term.

  note You can use the shortcut `Ctrl + + (KP)` to activate the **Next** functionality.
- **Previous** button: highlight the previous matching occurrence from the search term.

  The search will start at selection or at the cursor position.

  If you select some text with the mouse and use the shortcut `Ctrl + f`.

  The search field will toggle on if not visible and will contains the selected text as search term.

  note You can use the shortcut `Ctrl + - (KP)` to activate the **Previous** functionality.
- **Replace** button: replace the current matching occurrence, which is highlight, with the content of the replace field.

  note You can use the shortcut `Ctrl + Enter (KP)`: instead of the button.
- **Replace all** button: replace all the matching occurrence in the current file.

  note You can use the shortcut `Ctrl + Shift + Enter (KP)` instead of the button.
- **Mode**: You can select how your search term(s) will be interpreted.
  - **Raw text**: all search terms matching.
  - **Word boundary**: The search term must be a complete word, not a part but an variable with separators like underscores or points will work too.
  - **Regular expression**: *Perl* compatible regular expression (**REGEX**).

    note For **REGEX** read the **GLib Regex** documentation which is contains into the `gtk-doc` directory.
- **Close** button: hide the search and replace bar and clear the highlight.

note If you select some text with the mouse and use the `Ctrl + f` shortcut, then:

- The search and replace bar will be show.
- The search field will be filled with your selection.

  note The search will begin at your selection position if you hit the **Next** or **Previous** button.

Note: The search terms history:

Every search term you make a search for will be register into the history.

- You can use the `Up` key to start the history search from the beginning.
- You can use the `Down` key to start the history search from the end.

It will flow through the search terms history but not wrap around, simply end at the other end.

Replace in all files

You can use the button, menu item or shortcut (`Ctrl + Shift + R`),

to replace all occurrence(s) from a pattern according the settings:

- Case sensitive
- Mode
  - Raw text
  - Word boundary
  - REGEX

Which are all settable, like the pattern and the replacement text, into the appearing top-level window.

Copy to clipboard

You can copy to clipboard either :

- The current absolute file-path (`Ctrl + y`).
- The current absolute folder-path (`Ctrl + Shift + Y`).

Order page

You can use menu item or the shortcut (`Ctrl + Shift + O`) to reorder all the pages lexicographically.

Editor usage more

- You can use the page up and page down to scroll faster than with the arrows.
- You can use the key `Insert` for changing the form of the cursor.
- You can use the combination `Control + Left | Right` to move to previous | next sequence of characters.
- You can use the Erase key to erase the current selection.
- You can use the shortcut `Ctrl + Backspace` to remove an entire characters sequence.
- You can use the shortcut `Ctrl + Start` and `Ctrl + End` to move the cursor to the beginning or end of the document.
- The search-replace shortcuts are clever set on the keypad:
  - Find next : `Ctrl + + (KP)`.
  - Find previous : `Ctrl + - (KP)`.
  - Replace : `Ctrl + Enter (KP)`.
  - Replace all : `Ctrl + Shift + Enter (KP)`.

  For a better and faster search and replace feature.
- By all top-level windows you can use the shortcuts:

- `Escape` to close the window.
- `Enter` to confirm.

CHAPTER 4

Terminals

it-edit provides:

A sidebar terminals

it-edit provides **terminals in the same window as the editor**, as a sidebar, which can easily **shown**, **hidden** and **pull** as you want.

You can **add** and **remove** as many terminals as you want to the sidebar.

A full-screen terminal

it-edit provides a **big terminal** occupying the full interface, **dividable** into **4 re-sizable terminals**, to which you can **easy toggle** from the main window.

Top-level terminals

it-edit provides a **top-level terminal window** (so it can be **resize**, **minimize**, **maximize** and **closed**),

At first you will be prompt to enter a command.

After the execution of the command the top-level terminal is yours and can continue to enter commands.

Contextual menu from the terminals

- Copy from terminal
- Paste to terminal

—

- Decrease font
- Increase font

—

- Reset terminals

Note: By the sidebar terminals the items:

- Open new tab
- Close tab

Are added to the contextual menu.

CHAPTER 5

Files

New file

You can open a New file, this will create a random named `New_XXXXXX` file into your **TEMPDIR** folder, and erase it immediately but the file-path is kept.

You will surely write inside the new create buffer and surely save it after (surely not into the **TEMPDIR** folder).

Use the menu item *Files* → *New file* or the shortcut `Ctrl + N` to create a new buffer as describe above.

File(s) opening

They are several ways for opening file(s) into **`it-edit`**:

- By using the open file(s)
  - The menu item: *Files* → *Open file*.
  - The button.
  - The shortcut `Ctrl + O`.

This will present you a file selector to open the wanted file(s).

Note: The current tab influate the file-selector behaviour:

- The active editor page will influence into which folder the file-selector gets open.

  Because the file-selector will be launched into the folder from the current edited file location.

  > You can held the mouse over the tab (which content the current filename) to sea the absolute file path in a tool-tip.
  >
  > The same mechanic is by placing the mouse over the filename into the bottom bar.

- You can configure to get the current edited file selected (highlighted into the file-selector) into the editor or not.

note This can be practice if per example you want to open the header file from a source file or inversed.

- By using the *recent file* menu-item to open a recent used file.
- You can open the last registered files, most often the files opened in the last session, by using the the menu item *Reload Session* → *Reload last files*.

  This will open automatically all the last registered files.

  note This can be practice if you work on a single project during some time.

Saving files

They are several ways of saving files into **it-edit**:

1. You can save the current file simply using:
   - The menu item: *Files* → *Save file*.
   - The button.
   - The shortcut `Ctrl + S`

   This will save the file at is current location.
2. You can save a "`New`" file or the edited file into another location by using:
   - The menu item: *Files* → *Save file as*.
   - The button.

   warning They is no shortcut for this purpose, but if you save (*Save file* `Ctrl + S`) simply a "`New`" file this will act as a *Save file as*.
3. You can save all the open files using:
   - The menu item: *Files* → *Save all files*.
   - The button.
   - The shortcut `Ctrl + Shift + S`

   This will save all the unsaved files at their current location.

Note: You can distinguish if a file is currently save or modified by looking at the **name** *in the tab*:

if their is an **asterisk** '⋆' *before* the **file name** this mean that the *file* is currently **not saved** on the *disk*.

note If enabled **it-edit** will remove all the trailing spaces from the document after having save it.

Reload file

You can reload a file from disk with **it-edit** by using:

- The menu item: *Files* → *Reload current file*.
- The shortcut `Ctrl + R`.

note This can be practice if per example you have redirect you compilation process to a file for debugging compilation errors.

File informations

You can get and change files informations by using the menu item *Files* → *File informations* `Ctrl + I`.

This will display a **top-level window** presenting following **informations** and **action** to *process* on the *file*:

1. A frame named: `Main informations` will display:
   - A nice image from the **mime type** of your current edited file.
   - The programming **language** of the file.
   - The **mime type** verbatim.
   - The file **extension**.
2. A frame named: `Mode` will display a file **permissions** table like this:

```
+---+---+---+---+
|   | R | W | X |
+---+---+---+---+
| U | * | * |   |
+---+---+---+---+
| G | * | * |   |
+---+---+---+---+
| O | * |   |   |
+---+---+---+---+

U -> User.      R -> Read.

G -> Group.     W -> Write.

O -> Others.    X -> Execute.
```

   The cells of the table contains **check-boxes** representing the current **permissions** of the file.

   By simply (un)checking the **check-boxes** you change the **permissions** of the file on disk.

   warning By saving your file you will set the permissions according to your configuration into **`it-edit`** for files saving.

3. A frame named: `File counts` display some few statistics of the file:
   - *The number* of **lines**.
   - *The number* of **characters**.
   - *The number* of **words**.

   Of the file on the disk.

   note **`it-edit`** use the program **`wc`** to gets this informations.

   warning The number of line(s) and character(s) into your current edited buffer is visible into the bottom bar.

4. A frame named: `Timestamps` display the:
   - **Last status change** *date and time*.

- **Last access** *date and time*.
- **Last modification** *date and time*.

Of the file on the disk.

Near of every information is a button named `Modify` which permit you to change the timestamps.

Which will present you a **calendar** for the date

and **3** *spin buttons*:

- **Hours**
- **Minutes**
- **Seconds**

Which permit you to change the timestamps easily.

note This can be useful per example if you have change your system clock and you use the **`make`** tool,...

5. A frame named: `Checksum` will display the:

   - **MD 5** hash.
   - **SHA 1** hash.
   - **SHA 256** hash.
   - **SHA 512** hash.

   of your file.

   note The *checksums* are displayed into **hexadecimal** values.

6. A frame named: `File actions` will present you:

   The file name and 4 buttons, named:

   - Rename file
   - Copy file
   - Compress file (using gzip)
   - Move file.

   The functionalities of this buttons are clear as their name.

CHAPTER 6

Spell check

`it-edit` spell check features

`it-edit` provide 2 different spell check methods:

- In-line spell check:

  The in-line spell check mechanism is to underline the misspelled words and to provides corrections by setting the cursor over the misspell word and opening the contextual menu from **`it-edit`**.

  The contextual menu include then a *Spelling suggestion* menu item, from which you can choose to correct the misspelled word: the word is automatically replaced with the word you've chosen.

- Spell check dialog:

  `it-edit` provide a dialog window which will check the entire current edited file buffer.

Using `it-edit` spell check

You can enable or disable the in-line spell check by using:

- The menu item: *Edition* → *Inline spell check*.
- The button.
- The shortcut: `Ctrl + W`.

You can display the spell check dialog window using the:

- The menu item: *Actions* → *Spell check dialog*.
- The button.
- The shortcut: `Ctrl + Shift + W`.

it-edit spell check support

it-edit use the **gspell-1** library for providing spell check.

See also:

it-edit and **gspell-1** library:

Actually the **gspell-1** library is relative young, so not available in every repository.

So where ever you get the spell check functionality into **it-edit** depends on what version

of **gtk-3** you get.

Because the **gspell-1** library is only available with **>= gtk-+3.20**.

> **note** You can get gtk+-3.22 and gspell-1 currently with the *ppa* `gnome3-staging` for debian packages or by debian distributions through the "sid" repository.

gspell-1 has the advantages:

- To be compatible with the gtksourceview-3 library contextual menu.
- To provide a spell check dialog has widget.
- To have a good language selection mechanism.

See also:

gspell-1 library

```
gspell provides a flexible API to add spell checking to a GTK+ application. It
features:
* GObject wrappers around Enchant
* An inline spell checker for GtkTextView (enhanced version of GtkSpell)
* A spell checker dialog for GtkTextView
* Support of the no-spell-check tag defined by GtkSourceView
* Language choosers (button and dialog)
```

So we know that gspell is based on enchant:

```
Enchant is a generic spell checking library which uses existing spell checker
engines such as ispell, aspell and myspell as its backends.

Enchant steps in to provide uniformity and conformity on top of these libraries,
and implement certain features that may be lacking in any individual provider
library.
```

So for getting dictionaries compatibles with the **gspell-1** library simply download either or:

- aspell
- ispell
- myspell

dictionnaries in the wanted language(s).

Thanks

Big Thanks to the author of the gpsell-1 library **Sébastien Wilmet** which I get some form of familiarity within.

Sébastien Wilmet is the author of the **`Texilla`** Latex editor and maintainer of the gtksourceview-3 library and participate in many other projects like **`gedit`**.

CHAPTER 7

Sessions

Configure your sessions recovering

Their are 3 different modes for registering your session at your convenience.

Files, documentation, applications registering:

- Asked for registering the current session at quitting.
- Automatic registering.
- Disable session registering.

The automatic `session` mechanism

How does it work ?

Registering:

- Every time you open or close a file into the editor the registered files list is recomputed.

  note The path is registered.
- Every time you launch a file through the *File handler* (`Ctrl + H`), the file is registered.

  note The URI is registered.
- Every time you launch an application, the application is registered.

  note The application path is registered.
- Nothing is erase if you don't clear the session *Files* → *Reload session* → *Clear session*.

- If you reload something it's automatically stored again for next session.

Reloading a session

You can reload the entire session by activating the menu item:

- *Reload all last session*

  This will activate the reloading of the last registered:

  - Files.
  - Documentation files.
  - Applications.

Or reload the different items singular.

Clear a session

> **warning** To know that every **non-empty** registered list can be relaunch at the next session or into the same session.

The only way to clear all the list is to activate the menu item *Files* → *Reload session* → *Clear session*.

Into a session.

This permit to clear the lists and

if you want to reconstruct a new session by reactivating the concern session registering mechanism.

> **Warning:** Application launching Note:
>
> When you launch an application per the *Actions* → *Application launch* it won't be registered as applications
>
> because I consider that we need the application only now not very often.
>
> But if you open an application per the menu items *Applications* where you can register your personal applications you use often.
>
> They are are registered because you will use them often with the **`it-edit`** easy application access menus.

CHAPTER 8

Shortcuts table

Application shortcuts:

Shortcut	functionality	Mnemonic
Ctrl + n	New file	n = new
Ctrl + o	Open file	o = open
Ctrl + s	Save file	s = save
Ctrl + Shift + S	Save all files	S = Save
Ctrl + Alt + c	Close file	C = Close
Ctrl + Shift + c	Close all file(s)	C = Close
Ctrl + r	Reload file	r = reload
Ctrl + i	File informations	i = Informations
Ctrl + z	Undo	None
Ctrl + Shift + Z	Redo	None
Ctrl + f	Search	f = find
Ctrl + Enter	Replace	None
Ctrl + Shift + Enter	Replace all	None
Ctrl + +	Next	None
Ctrl + -	Previous	None
Ctrl + g	Go to line number	g = go to
Ctrl + x	Cut	None
Ctrl + c	Copy	c = copy
Ctrl + v	Paste	None
Ctrl + d	Duplicate text	d = duplicate
Ctrl + p	Use tabs	None
Ctrl + w	In-line spell-check	None
Ctrl + Shift + W	Spell-check dialog	None
Ctrl + e	Execute command	e = execute
Ctrl + Alt + o	Order pages	o = order
Ctrl + y	Copy file-path to clipboard	None

Continued on next page

Table 8.1 – continued from previous page

Shortcut	functionality	Mnemonic
Ctrl + Shift + y	Copy folder-path to clipboard	None
Ctrl + b	Show|Hide big term	b = big term
Ctrl + Shift + B	big term switch	B = Big term
Ctrl + t	Show | Hide terminal	t = terminal
Ctrl + Shift + T	Add new terminals	T = Terminals
Shift + Copy	Copy from terminal	None
Shift + Insert	Paste to terminal	None
Ctrl + h	File handler	h = handler
Ctrl + a	Application launcher	a = application
Ctrl + q	Quit application	q = quit

Terminals shortcuts:

Shortcut	functionality	Mnemonic
Shift + Copy (KP 1)	Copy from terminal	None
Shift + Insert (KP 0)	Paste to terminal	None
Shift + Ctrl + T	Open new tab	t = tab
Shift + Ctrl + -	Decrease font-scale	- = decrease
Shift + Ctrl + +	Increase font-scale	+ = increase

note Else you can close a tab and reset the terminal from the terminals contextual menu.

CHAPTER 9

Supported charset

Unicode

- UTF-8

—

Western

- ISO-8859-1

—

Central European

- ISO-8859-2

—

South European

- ISO-8859-3

—

Baltic

- ISO-8859-4

—

Cyrillic

- ISO-8859-5

—

Arabic

- ISO-8859-6

—

Greek

- ISO-8859-7

—

Hebrew Visual

- ISO-8859-8

—

Turkish

- ISO-8859-9

—

Nordic

- ISO-8859-10

—

Baltic

- ISO-8859-13

—

Celtic

- ISO-8859-14

—

Western

- ISO-8859-15

—

Romanian

- ISO-8859-16

—

Unicode

- UTF-7

—

Unicode

- UTF-16

—

Unicode

- UTF-16BE

—

Unicode

- UTF-16LE

—

Unicode

- UTF-32

—

Unicode

- UCS-2

—

Unicode

- UCS-4

—

Armenian

- ARMSCII-8

—

Chinese Traditional

- BIG5

—

Chinese Traditional

- BIG5-HKSCS

—

Cyrillic/Russian

- CP866

—

Japanese

- EUC-JP

—

Japanese

- EUC-JP-MS

—

Japanese

- CP932

—

Korean

- EUC-KR

—

Chinese Traditional

- EUC-TW

—

Chinese Simplified

- GB18030

—

Chinese Simplified

- GB2312

—

Chinese Simplified

- GBK

—

Georgian

- GEORGIAN-ACADEMY

—

Western

- IBM850

—

Central European

- IBM852

—

Cyrillic

- IBM855

—

Turkish

- IBM857

—

Hebrew

- IBM862

—

Arabic

- IBM864

—

Japanese

- ISO-2022-JP

—

Korean

- ISO-2022-KR

—

Cyrillic

- ISO-IR-111

—

Korean

- JOHAB

—

Cyrillic

- KOI8R

—

Cyrillic

- KOI8-R

—

Cyrillic/Ukrainian

- KOI8U

—

Japanese

- SHIFT_JIS

—

Vietnamese

- TCVN

—

Thai

- TIS-620

—

Korean

- UHC

—

Vietnamese

- VISCII

—

Central European

- WINDOWS-1250

—

Cyrillic

- WINDOWS-1251

—

Western

- WINDOWS-1252

—

Greek

- WINDOWS-1253

—

Turkish

- WINDOWS-1254

—

Hebrew

- WINDOWS-1255

—

Arabic

- WINDOWS-1256

—

Baltic

- WINDOWS-1257

—

Vietnamese

- WINDOWS-1258

CHAPTER 10

Supported languages

ActionScript:

text/x-actionscript

- *.as

—

Ada:

text/x-ada, text/x-adasrc

- *.adb
- *.ads

—

ANS-Forth94:

text/x-forth

- *.4th
- *.forth

—

ASP:

text/x-asp, application/x-asp, application/x-asap

- *.asp

—

Automake:

- Makefile.am
- GNUmakefile.am

—

awk:

application/x-awk

- *.awk

—

BennuGD:

- *.prg

—

BibTeX:

text/x-bibtex

- *.bib

—

Bluespec SystemVerilog:

- *.bsv

—

Boo:

text/x-boo

- *.boo

—

C:

text/x-c, text/x-csrc, image/x-xpixmap

- *.c

—

C#:

text/x-csharpsrc, text/x-csharp

- *.cs

—

C++:

text/x-c++, text/x-cpp, text/x-c++src

- *.cpp
- *.cxx
- *.cc
- *.C
- *.c++

—

CG Shader Language:

- *.cg

—

ChangeLog:

text/x-changelog

- ChangeLog*

—

C++ Header:

text/x-c++hdr

- *.hh
- *.hp
- *.hpp
- *.h++

—

CMake:

- CMakeLists.txt
- *.cmake
- *.cmake.in
- *.ctest
- *.ctest.in

—

C/ObjC Header:

text/x-chdr

- *.h

—

COBOL:

- *.cbl
- *.cob
- *.cbd
- *.cdb
- *.cdc

—

CSS:

text/css

- *.css
- *.CSSL

—

CSV:

text/csv

- *.csv

—

CUDA:

- *.cu
- *.cuh

—

D:

text/x-dsrc

- *.d

—

Defaults:

.desktop:

application/x-gnome-app-info, application/x-desktop

- *.desktop
- *.kdelnk

—

Diff:

text/x-diff, text/x-patch, text/x-reject

- *.diff
- *.patch
- *.rej

—

DocBook:

application/docbook+xml

- *.docbook

—

DOS Batch:

- *.bat
- *.cmd
- *.sys

—

DPatch:

text/x-dpatch

- *.dpatch

—

DTD:

text/x-dtd

- *.dtd

—

Eiffel:

text/x-eiffel

- *.e
- *.eif

—

Erlang:

text/x-erlang

- *.erl
- *.hrl

—

F#:

text/x-fsharp

- *.fs

—

FCL:

- *.fcl

—

Forth:

text/x-forth

- *.frt
- *.fs

—

Fortran 95:

text/x-fortran

- *.f
- *.f90
- *.f95
- *.for
- *.F
- *.F90

—

GAP:

text/x-gap

- *.g
- *.gd
- *.gi
- *.gap

—

GDB Log:

- *.gdb

—

Genie:

text/x-genie

- *.gs

—

gettext translation:

text/x-po, text/x-pot, text/x-pox, text/x-gettext-translation, text/x-gettext-translation-template

- *.po
- *.pot

—

Go:

- *.go

—

Graphviz Dot:

text/vnd.graphviz

- *.dot
- *.gv

—

gtk-doc:

GtkRC:

text/x-gtkrc

- gtkrc
- .gtkrc
- gtkrc-*
- .gtkrc-*

—

Haddock:

Haskell:

text/x-haskell

- *.hs

—

HTML:

text/html

- *.html
- *.htm

—

IDL:

text/x-idl

- *.idl

—

IDL-Exelis:

- *.pro

—

ImageJ:

- *.ijm

—

.ini:

text/x-ini-file, application/x-ini-file

- *.ini

—

J:

- *.ijs

—

Jade:

- *.jade

—

Java:

text/x-java

- *.java

—

JavaScript:

application/javascript, application/x-javascript, text/x-javascript, text/javascript, text/x-js

- *.js
- *.node

—

JSON:

application/json

- *.json
- *.geojson
- *.topojson

—

Julia:

- *.jl

—

LaTeX:

text/x-tex

- *.tex
- *.ltx
- *.sty
- *.cls
- *.dtx
- *.ins
- *.bbl

—

Lex:

- *.l
- *.lex
- *.flex

—

libtool:

text/x-libtool

- *.la
- *.lai
- *.lo

—

Literate Haskell:

text/x-literate-haskell

- *.lhs

—

LLVM IR:

- *.ll

—

Lua:

text/x-lua

- *.lua

—

m4:

application/x-m4

- *.m4
- configure.ac
- configure.in

—

Makefile:

text/x-makefile

- [Mm]akefile
- GNUmakefile
- *.make
- *.mak
- *.mk

—

Mallard:

- *.page

—

Markdown:

text/x-markdown

- *.markdown
- *.md
- *.mkd

—

Matlab:

text/x-matlab

- *.m

—

MediaWiki:

Meson:

text/x-meson

- meson.build
- meson_options.txt

—

Modelica:

text/x-modelica

- *.mo
- *.mop

—

MXML:

- *.mxml

—

Nemerle:

text/x-nemerle

- *.n

—

NetRexx:

text/x-netrexx

- *.nrx

—

NSIS:

- *.nsi
- *.nsh

—

Objective-C:

text/x-objcsrc

- *.m

—

Objective-J:

text/x-objective-j

- *.j

—

OCaml:

text/x-ocaml

- *.ml
- *.mli
- *.mll
- *.mly

—

OCL:

text/x-ocl

- *.ocl

—

Octave:

text/x-octave

- *.m

—

OOC:

- *.ooc

—

Opal:

- *.sign
- *.impl

—

OpenCL:

- *.cl

—

OpenGL Shading Language:

- *.glslv
- *.glslf

—

Pascal:

text/x-pascal

- *.p
- *.pas

—

Perl:

text/x-perl, application/x-perl

- *.pl

- *.pm
- *.al
- *.perl
- *.t

—

PHP:

text/x-php, application/x-php, text/x-php-source, application/x-php-source

- *.php
- *.php3
- *.php4
- *.phtml

—

Pig:

- *.pig

—

pkg-config:

text/x-pkg-config

- *.pc

—

Prolog:

text/x-prolog

- *.prolog

—

Protobuf:

text/x-protobuf

- *.proto

—

Puppet:

- *.pp

—

Python:

text/x-python, application/x-python

- *.py

—

Python 3:

- *.py3

—

R:

text/x-R

- *.R
- *.Rout
- *.r
- *.Rhistory
- *.Rt
- *.Rout.save
- *.Rout.fail

—

reStructuredText:

text/x-rst

- *.rst

—

RPM spec:

text/x-rpm-spec

- *.spec

—

Ruby:

application/x-ruby, text/x-ruby

- *.rb
- *.rake
- *.gemspec
- Rakefile
- Capfile
- Gemfile

—

Rust:

text/rust

- *.rs

—

Scala:

text/x-scala

- *.scala

—

Scheme:

text/x-scheme

- *.scm

—

Scilab:

- *.sce
- *.sci

—

sh:

text/x-shellscript, application/x-shellscript, text/x-sh

- *.sh
- *bashrc
- .profile
- .bash_profile

—

SPARQL:

application/sparql-query

- *.rq

—

SQL:

text/x-sql

- *.sql

—

Standard ML:

- *.sml
- *.sig

—

Sweave:

- *.rnw
- *.Rnw
- *.snw
- *.Snw

—

SystemVerilog:

- *.sv

- *.svh

—

Tcl:

text/x-tcl, application/x-tcl

- *.tcl
- *.tk

—

Texinfo:

text/x-texinfo

- *.texi
- *.texinfo

—

Thrift:

- *.thrift

—

txt2tags:

- *.t2t

—

Vala:

text/x-vala

- *.vala
- *.vapi

—

VB.NET:

text/x-vbnet, text/x-vb

- *.vb

—

Verilog:

text/x-verilog-src

- *.v

—

VHDL:

text/x-vhdl

- *.vhd

—

XML:

application/xml, text/xml

- *.xml
- *.xspf
- *.siv
- *.smil
- *.smi
- *.sml
- *.kino
- *.xul
- *.xbel
- *.abw
- *.zabw
- *.glade
- *.jnlp
- *.xhtml
- *.svg
- *.mml
- *.rdf
- *.rss
- *.wml
- *.xmi
- *.fo
- *.xslfo

—

XSLT:

application/xslt+xml

- *.xslt
- *.xsl

—

Yacc:

text/x-yacc, text/x-bison

- *.y
- *.yacc

—

YAML:

application/x-yaml

- *.yaml
- *.yml

CHAPTER 11

it-edit smart widgets

it-edit implement some few self build widget:

Note: If you get interest into Gtk-3 widget building

you can broad the source to understand How-To build GtkWidget(s) for Gtk-3 from your own.

GtkSmartMenuItem

A simple menu item containing:

- An icon.
- A label.
- An universal shortcut text.

Constructors

GtkWidget* gtk_smart_menu_item_new_all(const gchar *label, const gchar *icon_filepath, GtkA

Parameters

- **label** (const gchar *) – The label to display into the menu item.
- **icon_filepath** (const gchar *) – The menu item icon file-path.
- **accel_group** (GtkAccelGroup *) – The shortcut accelerator group.
- **accel_modifier** (const GdkModifierType) – The shortcut modifier.
- **accel_key** (const guint) – The shortcut accelerator key.

Return type GtkWidget *

Returns A pointer to the `GtkSmartMenuItem`.

GtkWidget* gtk_smart_check_menu_item_new_all(const gchar *label, const gboolean draw_as_ra

Parameters

- **label** (`const gchar *`) – The label to display into the menu item.
- **draw_as_radio** (`const gboolean`) – draw_as_radio
- **icon_filepath** (`const gchar *`) – The menu item icon file-path.
- **accel_group** (`GtkAccelGroup *`) – The shortcut accelerator group.
- **accel_modifier** (`const GdkModifierType`) – The shortcut modifier.
- **accel_key** (`const guint`) – The shortcut accelerator key.

Return type `GtkWidget *`

Returns A pointer to the `GtkSmartMenuItem` check button.

Note: You can pass a `NULL` pointer or `0` to the parameters :

- icon_filepath
- accel_group
- accel_modifier
- accel_key.

---

note You can build others constructors if you have understand How-To build this kind of widgets.

Getters

GtkWidget* gtk_smart_menu_item_get_image(GtkWidget *smart_menu_item) ;

Parameters

- **smart_menu_item** (`GtkWidget *`) – The return value from the constructors.

Return type `GtkWidget *`

Returns A pointer to the `GtkImage` widget.

GtkWidget* gtk_smart_menu_item_get_menuitem(GtkWidget *smart_menu_item) ;

Parameters

- **smart_menu_item** (`GtkWidget *`) – The return value from the constructors.

Return type `GtkWidget *`

Returns A pointer to the `GtkMenuItem` widget.

GtkWidget* gtk_smart_menu_item_get_label(GtkWidget *smart_menu_item) ;

Parameters

- **smart_menu_item** (`GtkWidget *`) – The return value from the constructors.

Return type `GtkWidget *`

Returns A pointer to the `GtkLabel` widget.

GtkWidget* gtk_smart_menu_item_get_accel_label(GtkWidget *smart_menu_item) ;

Parameters

- **smart_menu_item** (GtkWidget *) – The return value from the constructors.

Return type GtkWidget *

Returns A pointer to the GtkAccelLabel widget.

GtkSmartIconButton

A simple button with an icon without label and tool-tip which embed an universal short-cut text.

Constructors

GtkWidget* gtk_smart_icon_button_new_all(const gchar *filepath, const gchar *tooltip_text,

Parameters

- **filepath** (const gchar *) – The filepath to the image to use as icon.
- **tooltip_text** (const gchar *) – The tool-tip text without the accelerator label.
- **accel_key** (const guint) – The shortcut accelerator key.
- **accel_modifier** (const GdkModifierType) – The shortcut modifier.

Return type GtkWidget *

Returns A pointer to the GtkSmartIconButton widget.

GtkWidget* gtk_smart_icon_toggle_button_new_all(const gchar *filepath, const gchar *toolti

Parameters

- **filepath** (const gchar *) – The filepath to the image to use as icon.
- **tooltip_text** (const gchar *) – The tool-tip text without the accelerator label.
- **accel_key** (const guint) – The shortcut accelerator key.
- **accel_modifier** (const GdkModifierType) – The shortcut modifier.

Return type GtkWidget *

Returns A pointer to the GtkSmartIconButton toggle button widget.

Getters

GtkWidget* gtk_smart_icon_button_get_image(GtkWidget *smart_icon_button) ;

Parameters

- **smart_icon_button** (GtkWidget *) – The return value from the constructor of a GtkSmartIconButton.

Return type GtkWidget *

Returns A pointer to the GtkImage widget.

GtkItTerm

Not reusable like this.

Note: I've learned how to implement self builded Gtk-3 widgets

If you get interest in building widgets take a look at the source for basics.

CHAPTER 12

Indices and tables

- genindex
- modindex
- search