Grimoire



Version 1.20 (05-Nov-2026)

© 2026 R. F. Windley & The Really Small Software Company

Grimoire is public domain

This program may be freely distributed for no more than the cost of the media on which it is distributed.
It remains at all times © Copyright 2026 R. F. Windley & The Really Small Software Company.

Disclaimer:

This software is supplied as is. No responsibility will be taken by the author for any loss of data through the use or misuse of this application.

Contents

Introduction 1
  Assumptions 1
  Requirements 1
Getting started. 2
  Loading Grimoire 2
  Creating a new file 2
  Loading an existing file 3
The Grimoire window 4
  Toolbar 5
  Status line 6
Menus. 6
Saving 10
Languages 11
  Tokenised BBC BASIC 11
  !Languages and !LangMan 11
Editing 12
  Editing keys 12
Brackets and quotes 13
Column selection 13
Comments 14
Snippets 14
Keystroke macros 14
Completion and declarations 15
Parameter hints. 16
Going to a declaration 17
The function list. 17
Class diagrams. 18

Finding text 19
Regular expressions 20
Find in files 21
Building and running 22
Throwback 22
Run script 23
Task windows 23
Files 24
Line ends 24
Files changed by other programs 24
Autosave and backups 24
Comparing two files 24
Choices 25
Using Grimoire with Sourcery 33

Introduction

Grimoire is an editor for programmers. Some of the features it offers include:

- Syntax colouring
- Completion of symbols and names
- Control-click to follow link to file with recognised symbol
- Highlighting of simple errors while you type
- Support for over twenty different languages via plugins
- Throwback support
- Simple code formatting
- TaskWindow support
- Tabbed window support

Assumptions

This manual assumes you know your way round the RISC OS desktop: the Select, Menu and Adjust buttons, menus, and saving a file by dragging its icon to a directory display.

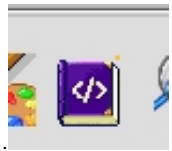
Requirements

Grimoire has been tested on RISC OS 5.30 and 6.16. It requires the 32 bit C library to be installed and the latest set of Toolbox modules.

Getting started

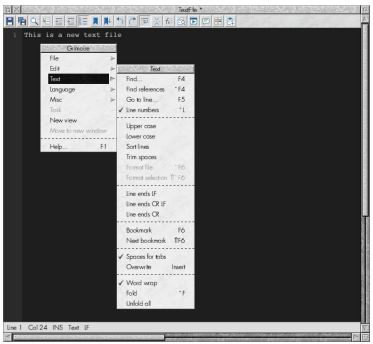
Loading Grimoire

Double-click on ***!Grimoire*** and once loaded an the ***Grimoire*** icon will appear on the iconbar in the standard way



Creating a new file

Click Select on the icon bar icon to open a new, untitled text file. This will give the core functionality. If you want to edit a different type of file you can choose the type of file from the Language menu.



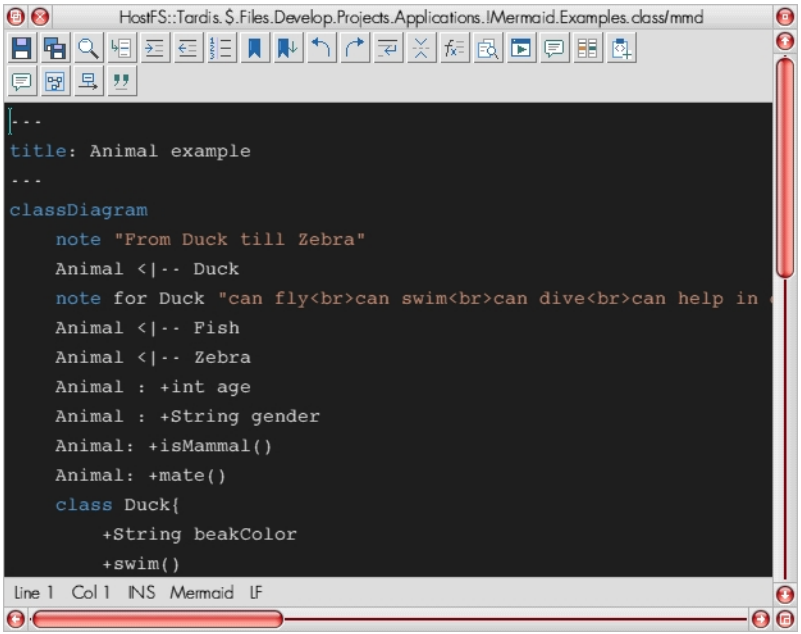
Loading an existing file

Drag a file to the icon bar icon or double click a text file (shift double click other types like BASIC to load them without running). If the text file lives within a recognised directory like ***c*** or ***h*** or ***c++***, or has a recongised suffix like ***/md*** or ***/json*** or ***/mermaid***, then the appropriate language plugin will be loaded and used.

The exact combination of suffix and directory mappings available depend on the language plugins installed.

The Grimoire window

By default each file has its own window, with a toolbar across the top and a status line along the bottom. A tabbed mode is also supported which is covered later.



Toolbar

The top row of the toolbar is the same for every file:

Save	Save the file
Save all	Save all open files
Find	Open the find and replace dialog
Go to	Open the Go to dialog to go to a specific line
Indent	Indent the selected lines
Outdent	Remove indent from the selected lines
Lines	Display or hide line numbers in the left margin
Mark	Set a bookmark
Next	Move to the next bookmark
Undo	Undo the last operation
Redo	Redo the previously undone operation
Wrap	Toggle wordwrap
Fold	Fold the highlighted section
Funcs	Display a list of functions in the current file
Files	Find across files
Run	Run the current file if a runner is available
Snips	Display a list of templates and snippets
Comment	Comment out the selected lines
Compare	Compare two files

The second row belongs to the file's language. ***Markdown*** has ***Bold*** and ***Link*** there, for instance, while plain text has no second row at all. If you are not sure what a button does, rest the pointer on it for a moment and its name and short-cut key appear. If ***!Help*** is loaded you get a longer description.

Status line

The status line shows the line and column of the caret, whether you are inserting (**INS**) or overwriting (**OVR**), the file's language and the kind of line ends it uses. The ***Insert*** key switches between **INS** and **OVR**.

Menus

Click Menu over a window for the ***Grimoire*** menu:

File

Info

Display information about the current file.

Save

Save the file.

Print

Print the file using the currently selected printer.

Edit

Common editing features:

Cut

Cut the currently selected text.

Copy

Copy the currently selected text.

Paste

Paste the contents of the clipboard.

Undo
Undo the last operation.

Redo
Redo the last undone operation.

Select all
Select all text.

Clear
Unselect the current selection.

Text
Find...
Open the find and replace dialog.

Find references
For the highlighted symbol, finds all references in all currently open files and display a list. This can also be actived by Controle-Shift clicking a symbol.

Go to line
Open the Go to line dialog.

Line numbers
Toggle line numbers in the left margin.

Upper case
Convert selected text to upper case.

Lower case
Convert selected text to lower case.

Sort lines
Sort the currently selected block alphabetically

Trim spaces
Trim whitespace from the end of lines.

Format
If available, performs simple language specific formatting on the file. Some of the rules can be changed in ***Choices***.

Format selection
Performs formatting just on the highlighted text.

Line ends LF
Line ends CR LF
Line ends CR
Change the line ending character.

Bookmark
Toggle a bookmark on the current line.

Next bookmark
Move to the next bookmark.

Spaces for tabs
Replace tabs with spaces

Overwrite
Toggle insert and overwrite mode.

Word wrap
Toggle word wrap.

Fold
Hides the lines below the caret that are indented further than it; use it again to show them.

Unfold all
Unfold all folded blocks.

Language
Displays a menu to allow choosing the language the file is coloured as.

Misc
Class diagram....
Where available, create a mermaid file that can be used to render a class diagram, picking the class to create the diagram for.

Task
When used with a task window offers control of the running task.

Suspend
Stops the task, and starts it again. It is ticked while the task is stopped.

Kill
Ends the task.

Reconnect
Starts a new command line once the task has ended.

Link
While it is ticked, keys go to the task. Untick it to edit the text.

Send selection
Types the selected text to the task.

New view
Opens another window on the same file.

Help...
Shows the help text.

Saving

Press F3, or choose ***File > Save***, then drag the file icon to a directory display or click ***Save***.

Languages

Grimoire works out the language from the file's directory or extension (**c**, **h**, **c++**, **s**, **hdr**, **py**, **php**, **js**, **css**, **pl**, **lua**, **sh**, **sql**, **json**, **xml**, **html**, **yaml**, **tex**, **diff** and others), and from its file type.

Some programs pass files over with no helpful name or type. If such a file is Text or HTML, ***Grimoire*** looks inside it for clues and picks the language from those.

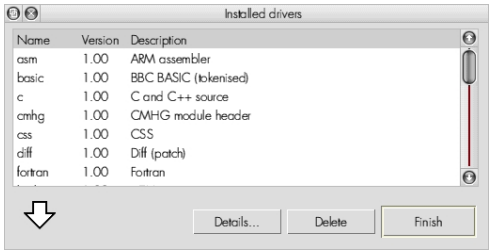
If ***Grimoire*** still picks the wrong language, choose the right one from the **Language** submenu of the window's menu. It lists every language in ***!Languages***, with the file's own ticked. The colouring, the language's toolbar row and the status line change in every window on the file, and the file is not marked as changed.

Tokenised BBC BASIC

BASIC programs (file type &FFB) are shown as text and tokenised again when saved. Before letting you edit one, ***Grimoire*** checks that it would save back exactly as it was loaded. Crunched programs often fail this test, and they are opened read-only with a message saying why.

!Languages and !LangMan

Grimoire gets all its languages from the drivers in ***!Languages***, so the Filer must have seen ***!Languages*** before ***Grimoire*** is run. ***!LangMan*** shows which drivers are installed and their versions. New drivers can be added by dragging them to ***!LangMan*** to install.



Editing

Typing, selecting with Select and extending a selection with Adjust all work as they do in any RISC OS editor. Double-click to select a word, and click a third time, straight after, to select the whole line. You can drag a selection to move it. Pressing Return starts the new line with the same indent as the one above.

Editing keys

Short-cut keys are defined as follows:

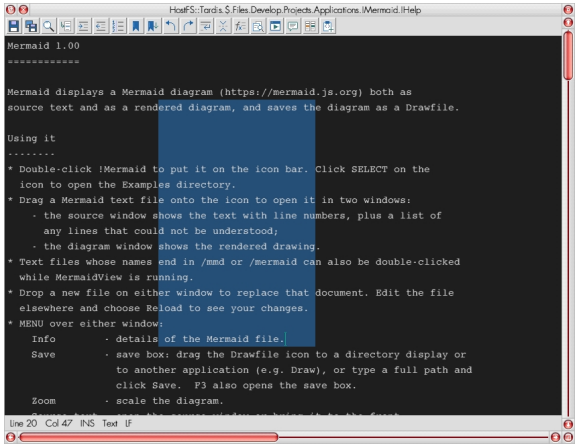
F1	Help
F3	Save
F4	Find and replace
Ctril F4	Find reference to symbol in open files
F5	Go to line
F6	Set or clear a bookmark on this line
Shift-F6	Go to the next bookmark
Ctrl F6	Format file
Ctril Shift F6	Format highligted text
F7	List the file's functions
Shift-F7	Put in a snippet
F9	Play the keystroke macro
Shift-F9	Start or stop recording a macro
Ctrl-L	Show or hide line numbers
Ctrl-B	Jump to the matching bracket
Ctrl-F	Fold or unfold the more indented lines below
Tab, Shift-Tab	Indent or outdent the selected lines
Insert	Switch between inserting and overwriting
Ctrl-X, C, V	Cut, copy and paste
Ctrl-Z, Ctrl-Y	Undo and redo
Ctrl-A	Select all
Ctrl-click	Go to a name's declaration, or open an included file
Alt-drag	Select a column

Brackets and quotes

When you type an opening bracket, the closing one is put in after the caret for you. Quotes are paired in the same way, except in plain text, **Markdown** and **TeX**, and in **BASIC** and **Obey** only " is paired. Nothing is added if the caret is just before a word, or (for a quote) just after one. Typing the closing character steps over the one already there, and pressing **Delete** straight after an opening bracket removes the pair. None of this happens in overwrite mode.

Column selection

Hold down **Alt** while you drag to select a rectangle of text. It can run past the ends of short lines.



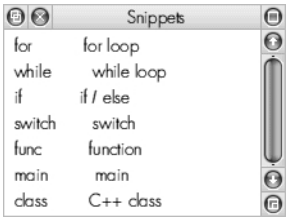
Anything you type, and **Delete** and **Backspace**, then act on every line of the rectangle at once; after the first key the rectangle narrows to a thin bar, so you can carry on typing down the column. **Ctrl-C** and **Ctrl-X** copy or cut the rectangle, and **Ctrl-V** pastes a single line onto every row, or several lines one row at a time. Any other key, or a click, ends the column selection. It isn't available while **Wrap** is on.

Comments

Comment comments out the selected lines, or just the caret's line, in whatever way suits the language. If the lines are all comments already, it turns them back into code. In HTML, XML, Markdown, CSS and Pascal the selection is wrapped in a block comment instead.

Snippets

A snippet is a ready-made piece of code. Type its name, for example for, and press **Shift-F7** (or click **Snips)**: the name is replaced by the snippet, indented to match the line. Press **Shift-F7** anywhere else to see a list of the language's snippets and pick one.



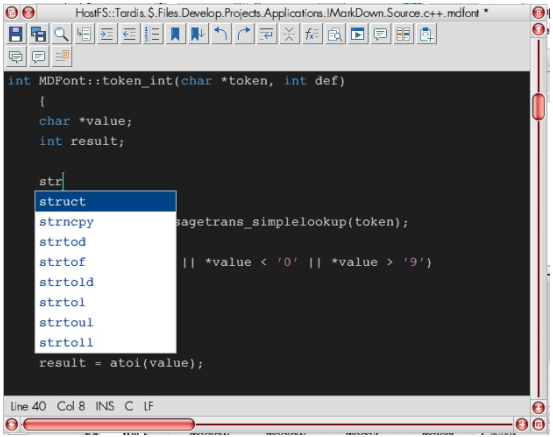
The snippets for each language are kept in a text file of that name in ***!Grimoire.Snippets***. To use your own, put a file with the same name in ***Choices:Grimoire.Snippets***. In the file, a line starting **@@** begins a new snippet and gives its name and a short description; the snippet's text follows on the next lines. Use **$|** to mark where the caret should end up, and **$$** for a dollar sign.

Keystroke macros

To record a macro, press **Shift-F9**, type the keys you want repeated (cursor and editing keys included), then press **Shift-F9** again. **F9** plays the keys back into whichever window has the caret. Up to 1024 keys are kept, until you record another macro or quit ***Grimoire***. Toolbar buttons and menu choices are not recorded.

Completion and declarations

As you type a name, ***Grimoire*** lists the words that would complete it. These come from the file itself and, in C and C++, from the functions, types, macros, variables and structure members declared in the headers the file **#includes**, and the headers they include in turn, up to 8 deep. Click a word or press **Return** to put it in. Functions are shown with their arguments and return type.



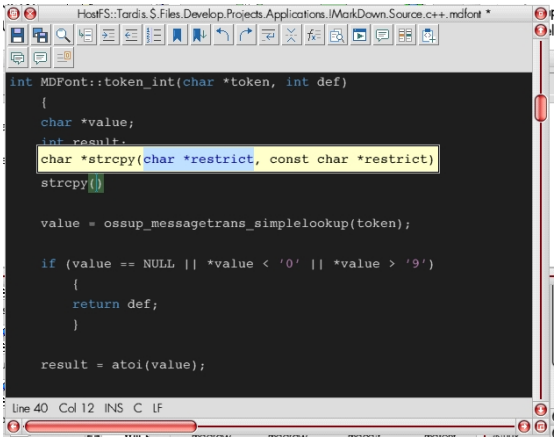
Each header is read once, and read again only when it changes. ***Grimoire*** also remembers where each header was found, or that it wasn't. Headers found through a path such as **C:** or **OSLib:** are system headers, which seldom change, so they are checked again only when a file is loaded; other headers are checked at most every 3 seconds. The list of words is gathered once you have typed two letters of a name, and narrowed down as you type more, so typing stays quick however many headers there are.

Other languages are followed in the same way:

PHP	include and require of a constant name, beside the page then on PHP$Path.
Python	import a.b and from a.b import x: a.b/py, then on Python$Path.
Lua	require "a.b": a.b/lua, then on Lua$Path.
Perl	use and require Foo::Bar: Foo.Bar/pm, then on Perl$Path.
BASIC	LIBRARY and INSTALL; DEF PROC and DEF FN names.
Assembler	GET and INCLUDE; labels and MACRO names.

Parameter hints

When you type **(** after the name of a known function, its full declaration appears above the caret, with the argument you are typing highlighted. The highlight moves on at each comma, and the hint closes at the) or when you press **Escape**.



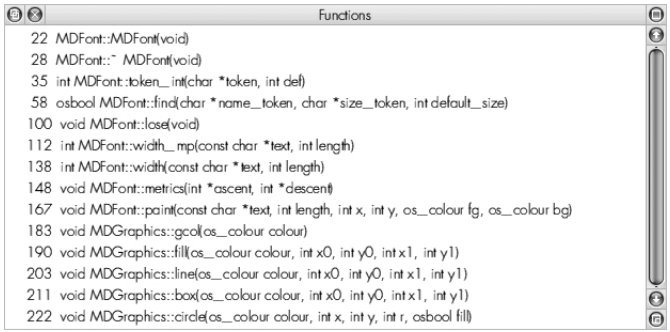
Going to a declaration

Hold down **Ctrl** and click on a function, type, macro or variable to jump to where it is declared. If that is in a header, the header is opened for you. In BBC BASIC, Ctrl-clicking a **PROC** or **FN** name takes you to its **DEF**. Ctrl-click the file name in an include statement, such as **#include "name.h"**, to open that file; it is looked for in the same places as for completion.

While you hold down **Ctrl**, a name or file name that Ctrl-click can go to is underlined when the pointer is over it. Nothing is underlined if Ctrl-click would go nowhere.

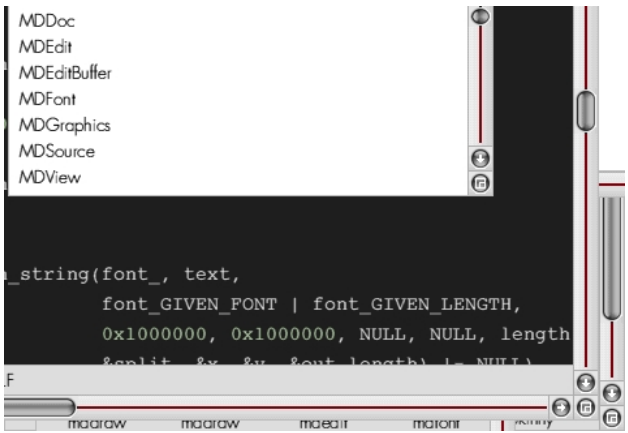
The function list

Funcs (F7) lists the functions, procedures and types in the file (or the labels, in assembler) with their line numbers. Click one to go to it. The list is rebuilt each time you open it, so it is always up to date.



Class diagrams

Misc > Class diagram... lists the classes and structs defined in the file, in the files it includes and, for a file in a c, c++ or h directory, in the headers in the h directory beside it.

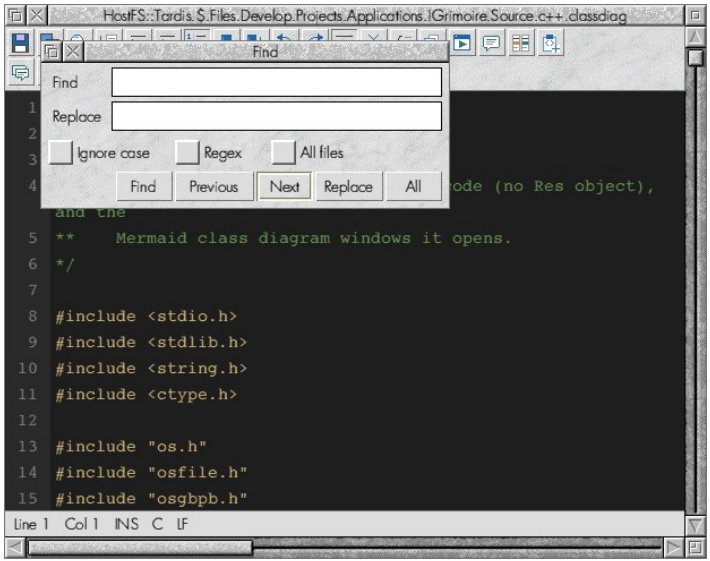


Click one to open a new window holding a **Mermaid** class diagram of it: its base classes, the classes derived from it and the classes its members hold, with --> for a pointer or reference and *-- for a value. The window's language is Mermaid, so the diagram can be saved and viewed in ***!Mermaid***. Destructors are left out, because Mermaid reads ~ as the start of a generic type.

Finding text

Find (F4) opens the Find window. Type what you are looking for and, if you want to change it, what to replace it with. Tick **Ignore case** to match regardless of case, and **Regex** to use a regular expression. A **Replace all** can be undone in one go.

The **Find** button finds the next match, like **Next**. Tick **All files** to work on every open file: Find then lists every match in the Throwback window, where a click goes to it, and **All** replaces in every open file. Task windows are left out, and a file open in more than one window is searched once.



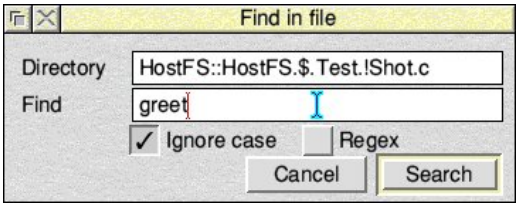
Regular expressions

Pattern	Matches
.	Any character except a line end
[abc] [^a-z]	Any of a set, or any not in it
\d \w \s	A digit, word character or space; \D \W \S are the opposite
* + ? {n,m}	Repeats; add ? to match as little as possible
^ $	The start and end of a line
\b	A word boundary
a|b	Either
( )	A group, matched again by \1 to \9; (?:) is not one

In the replacement, \1 to \9 put back a group, \0 the whole match, \n a new line and \t a tab. A pattern that would take too long, such as (a*)b, is stopped with an error rather than hanging the machine.

Find in files

Files opens the Find in files window. Give it a directory (it starts with the current file's) and the text to look for, tick **Ignore case** or **Regex** if you need them, and click **Search**. Every text file in that directory, and in the directories inside it, is searched, and the matching lines are listed in the Throwback window. Click a line to open the file there. Files over 2MB, and files that look like binary, are skipped, and the search gives up after 2000 matches.

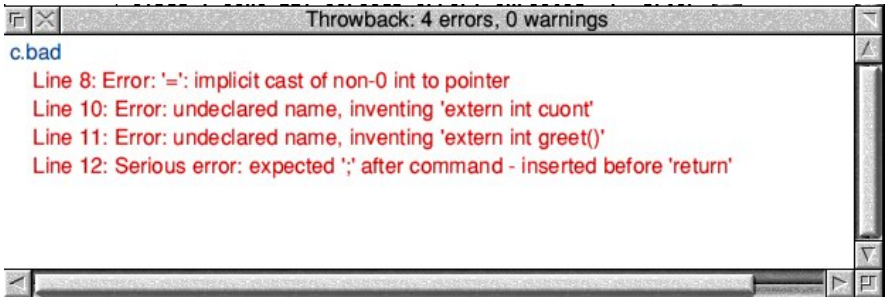




Building and running

Throwback

When a compiler, AMU or ObjAsm is run with **-throwback**, as it is in ***Sourcery***'s builds for example, its errors and warnings are sent to ***Grimoire*** and listed in the Throwback window, grouped by file. The window's title gives the number of errors and warnings, and each new build replaces the list. Click an error to go straight to the line in question, opening the file first if necessary.



Only one program at a time can receive throwback. If another editor got there first, quit it and start ***Grimoire*** again.

Run script

Run runs the file as a script in a task window. **BBC BASIC**, **Obey**, **Lua**, **Perl** and **Python** need nothing setting up. For anything else, set **Grimoire$Run_<language>** to the command to use, with **%f** where the file name goes. For example:

```
*Set Grimoire$Run_Shell "bash %f"
```

The file must be saved first. Anything the script prints is shown in the Throwback window, and a line that mentions a line of the script, such as "at line 20", is treated as an error: click it to go there. Click Run again to stop a script that is still running.

Task windows

Task window on the icon bar menu opens a window running a command line: type a * command and its output appears in the window. Keys go to the task as they are typed, so a command line editor running in the task gets every key, including Tab for completion.

The cursor keys, Copy and Tab, with Shift or Ctrl, go as the keyboard buffer has them, and the window follows the task's cursor moves, so a line editor such as LineEditor can recall earlier commands with Up and Down and edit the line with Left, Right and Delete.

Copy, Cut and Ctrl-C (while text is selected) copy the selection without changing the window, and Paste, Ctrl-V and text dropped on the window go to the task as if typed.

While **Open task windows for Ctrl-F12** is on in ***Choices*** (it starts on), **Ctrl-F12** and the Task Manager's **Task window** entry open a ***Grimoire*** task window instead of an ***Edit*** one. The window's **Task** menu is covered in the **Menus** section above.

Files

Line ends

Grimoire reads files with LF, CR LF or CR line ends and saves them back the way they came. The status line shows which kind a file has, and the Text menu converts between them.

Files changed by other programs

If another program changes a file you have open, ***Grimoire*** notices the next time you click in its window and asks whether to load the new version. It warns you first if you have unsaved changes. Reloading keeps the caret where it was but clears the undo history. Saving from ***Grimoire*** doesn't count as a change.

Autosave and backups

Every 5 minutes (configurable in **Choices**), ***Grimoire*** saves a copy of each file with unsaved changes in ***<Choices$Write>.Grimoire.Recover***. The copy is deleted when you save or close the file, or quit. If ***Grimoire*** is stopped by an error or a reset before it can tidy up, it offers you the copies the next time it starts, under their original names and marked as modified.

The first time you save over a file, the previous version is kept under the same name with ~ on the end.

Comparing two files

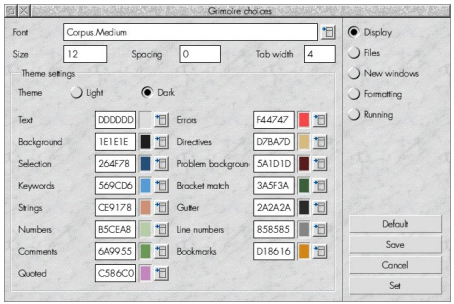
Click **Compare** and choose another open file from the menu. Its window opens beside this one, and the lines that differ are coloured in both. If there are no differences, a message tells you so. Editing either file clears its colouring, and clicking **Compare** again clears both.

Choices

Choices..., on the icon bar menu, opens the ***Choices*** window which allows a number of different settings to be controlled:

Display

The Display panel sets the font the editor uses, its size and any extra space between lines as well as tab spacing, and lets you choose a Light or Dark theme and the colour of each part of the syntax in each theme.



Font details

Type the name of a font, or click the arrow beside the field and choose one from the font menu. The current font is ticked, and clicking with ADJUST keeps the menu open.

The font size and space can also be specified, as well as the tab width.

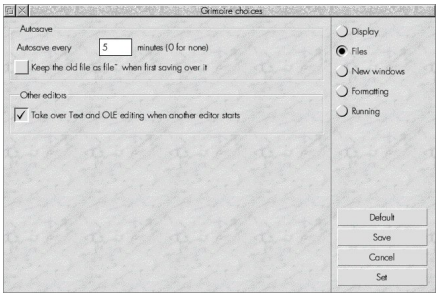
Theme settings

You can choose between ***light*** and ***dark*** themes.

Each colour is shown as six hex digits, RRGGBB, with a swatch beside it. Type a new value and the swatch changes to match, or click the arrow or the swatch to choose the colour in the colour picker.

Files

The Files pane offers some options controlling how files are saved and loaded.



Autosave

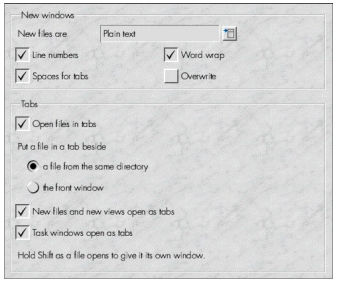
You can set the ***Autosave frequency*** and whether a backup file is created when the file is saved.

Other editors

Choosing this option ensures that Grimoire will handle editing of file types it knows about.

New Windows

New Windows enables the way new windows are opened to be specified.

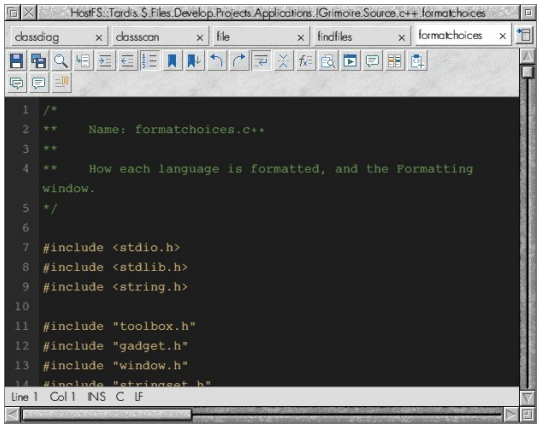


New windows

New files are sets the language a new file starts in (Text to start with). A file you load still gets the language that matches it. **Line numbers**, **Word wrap**, **Spaces for tabs** and **Overwrite** set whether each new window starts with them on. The menu and toolbar still turn them on and off in each window.

Tabs

Grimoire allows tabbed windows to be used rather than opening a new window for each file.



Tabs appear at the top of the window. Clicking a tab displays that files content. A tab can be dragged off of the main window into a window of it's own, and tabs can be dragged between windows.

Open files in tabs

With **Open files in tabs** on (it starts off), a file you open goes in a tab of a window that is already open instead of getting its own window: beside a file from the same directory, or in the front window (the one that last had the caret), whichever you choose.

A directory that gives its files their language, such as c or h, counts as the directory above it, so ***Proj.c.main*** and ***Proj.h.main*** go in the same window.

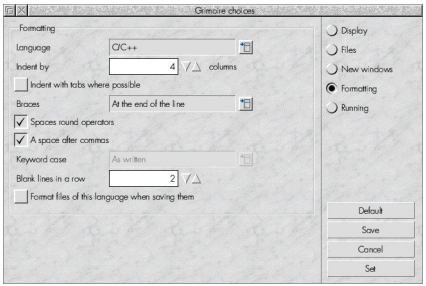
New files and new views open as tabs
New files, new views and task windows open as tabs if this option is on.

Task windows open as tabs
This allows Task Windows to also open as tabs rather than in their own window.

Even if tabs are on, it is possible to force a file into it's own window by holding Shift as a file opens.

Formatting

Clicking the ***Formatting*** button opens a window which allows a number of language specific formatting options to be set:



Language

Pick the language to configure. Not all options are available for all languages, some will be disabled.

Indent by

The number of columns to automatically indent by.

Indent with tabs where possible

If the language supports it, use tabs to indent of this is switched on

Braces

Specifies where braces should be position on a line.

Spaces around operators

If on, spaces are placed around operators.

A space after commas
Place a space after commas if switched on.

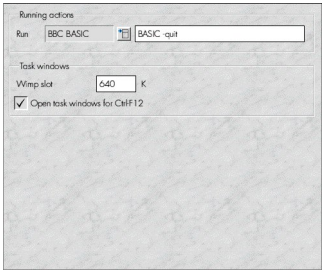
Keyword case
Whether to enforce a case on keywords upon formatting.

Blank lines in a row
The maximum number of blank lines to keep together.

Format Files of this languages when saving them
If one, when the files is saved it will be reformatted.

Running

Options that control the way files or command are run.



Task windows

Wimp slot sets the memory each task window's task gets (640K to start with), and **Open task windows for Ctrl-F12** sets whether **Ctrl-F12** and the Task Manager open a ***Grimoire*** task window.

Run actions

Run actions allow you to specify a command to be run to execute a file and can be set for each language type.

Using Grimoire with Sourcery

When you open a source file from ***Sourcery***, it sets **Grimoire$IncludePath** to the project's include paths, the same ones the compiler is given.

Completion, parameter hints and Ctrl-click then find the project's headers and those of the libraries it uses, and so do the headers you open from the file with Ctrl-click.

Other programs can do the same: set **Grimoire$IncludePath** to a list of paths and directories, separated by commas or spaces, such as **C:,OSLib:,ADFS::HD4.$.MyLib.Source**, just before asking ***Grimoire*** to open the file. ***Grimoire*** keeps a copy with the file and clears the variable, so the next file does not pick it up.