

# RCS Browser



The RCS Browser is an easy-to-use tool for version control, using a graphical interface. It is based on the *Revision Control System* (RCS).

RCS Browser is copyright 2022 CompuPhase, and distributed as freeware.

## Why you need it?

There are two ways to interpret this question: why do you need version control, and why choose RCS over the various alternatives?

Why use version control at all? If you work with software, design files or documents, you are likely already doing it by hand. If you keep archives of your project in ZIP files, where each ZIP file represents a release, or prototype design, that's version control —albeit an ad-hoc, coarse-grained and manual method of version control. You should consider to step up, and use the appropriate tools for a more structured, finer-grained approach.

Why choose RCS? Unlike most of the alternatives, the Revision Control System runs on a local system as a default. It neither requires a server, nor setting up a project. This makes it the tool of choice when you are a single developer working on a single workstation.

The RCS utilities themselves are command-line tools, with sometimes counter-intuitive options. The RCS Browser allows you to perform the basic tasks for revision control, from a graphical user interface.

## What else do you need?

Apart from the RCS Browser itself, you need the RCS utilities. If you are running Linux, you can install them from the package manager. When running Windows, you can download the 'GNU RCS' binary package that has been ported to Windows. RCS Browser requires version 5.7 or later of RCS.

You will also need a file compare utility. One of the best for Windows, which is also free, is WinMerge. In Linux, you can use Meld.

## Configuration

On first launch of RCS Browser, you need to adjust the settings (the button 'Settings' is in the top right corner of the main interface). What you need to set are:

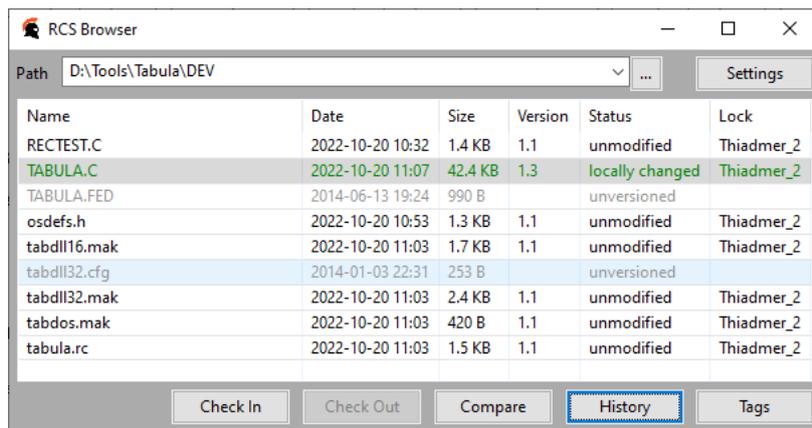
- The path to the RCS utilities.
- The filename and path to the file compare utility.

The other options are covered in section [Details on the user-interface](#) on page 3.

## Usage

RCS Browser shows you a list of files in the directory that you have selected. The top row of the user interface lets you type in a path, or browse for a path. Recent paths are saved in a drop-down list, so that you can easily switch to directories where your source files are.

If you have selected a directory, the list fills with all files that are present in that directory. Initially, these will all be marked as 'unversioned'.



## Adding Files to RCS

The first step is to add files to the system. In the terminology of RCS, a file is 'checked in'. When you select an unversioned file, 'Check In' is the only enabled button on the bottom row of RCS Browser.

A 'check in' is also done after you have made changes to a file, and you wish to preserve these in the history. However, the very first check-in has slightly different options:

- The 'description' field is a description for the file. It is optional, and it can be anything. You might say 'main application form' or 'bill of materials', or anything similar.
- Whether the file is a text file or a binary file. This is not important in Linux (in which there is no difference between text files and binary files), but essential in Windows. After the first check-in operation, you can no longer change the text/binary mode of the file.

## Committing changes

After you have edited a file, or changed it by other means, you can 'check in' those changes to RCS. It is very helpful, for future review, to add a brief, but specific description of the changes done to the file. Before doing the check-in operation, it is therefore recommended that you first review the changes by clicking on the 'Compare' button.

## The 'Lock File' and 'Keep locked' options

In RCS, only the person who has locked a file, is able to edit it. Unless you are working in a team, there is never a need to unlock a file. See section [Locking and unlocking files](#) on page 5 for more information, but as a rule-of-thumb, you can keep the 'lock file' and 'keep locked' at their defaults.

## Reverting changes

If you want to undo changes that you made since the last check-in operation, you can launch the file comparison tool with the 'Compare' button. In this tool, you can copy selected lines back from the head of the archive back to the work copy. You would typically use this function if you do not want to revert *all* changes.

Alternatively, you can go to the history of the file, and then use the 'Export' function to save the head, and overwrite the work copy.

You can use these functions as well to revert changes from earlier revisions than the head. Before overwriting a work copy file with an earlier revision, it is recommended to first commit any changes of the work copy. This way, you are sure to never lose work, because you can then also 'revert the revert'.

## Marking a release or milestone

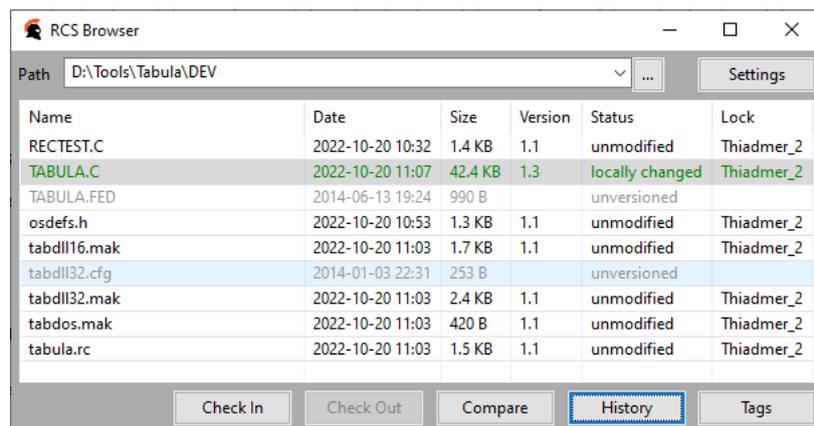
A project is usually made up of several files, and some of those files will be edited and updated more often than others. Thus, when a project arrives at a milestone, such as 'release candidate' or 'version 1.3', the revision numbers are widely spread.

With a *tag*, you assign a name to the set of files and their revisions. A revision number applies to a file; a tag applies to a *set* of files. In the 'Tags' dialog, you can assign a new tag to the files in a directory, but also export the files with a particular tag. When you export a tag, you export all files with the revision that they had when the tag was added. In other words, you restore the files to the earlier milestone.

Due to limitations of RCS, the tag name must be a single word —it may not contain space characters or a few or special characters (like a ':' or a '@').

## Details on the user-interface

The main interface has a selection on the top row for the directory, a file list and a row with buttons on the bottom.



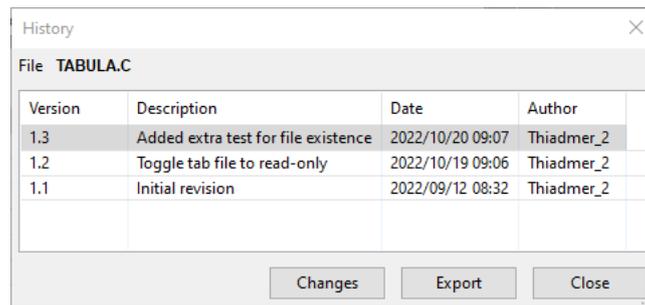
The file list shows all files in the directory, with the exception of the *archive files* (the files that RCS creates for storing the history of changes). The buttons on the bottom are enabled or disabled, depending on the status of the selected file.

**Check In:** When a file is 'unversioned', the only available action is to perform a 'Check In'. A check-in operation adds the file to version control. If the file was already under version control, a 'Check In' commits those changes.

**Check Out:** A 'Check Out' operation is only needed to lock a file for editing, if that file was not already locked by you. By default, RCS Browser keeps a file locked (to you), so if you use RCS Browser and you work alone (not in a team with multiple persons working on the same files), you won't need to ever use 'Check Out'.

**Compare:** This button opens a file compare utility to show the differences between the work file and the last revision of it that was committed to version control. In other words, it shows the changes you made to the file since the last check-in. See also section [The 'Lock File' and 'Keep locked' options](#) on page 2

**History:** The 'History' button opens a new window with the list of revisions of the selected file. What you see in this list is the date and time of each 'check in', and the brief description of the changes.

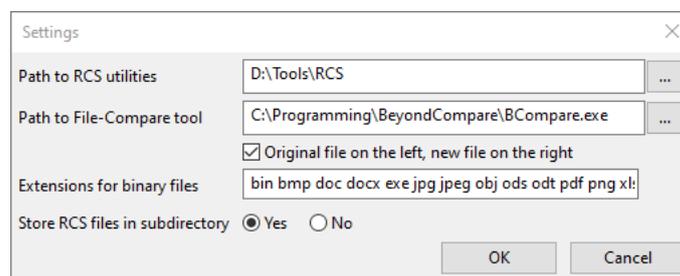


When you select a revision in the list, and then click on the button 'Changes', the changes of the selected revision relative to the previous, are shown in the file compare tool. For example, if you select revision 1.3 in the list, and click 'Changes', this will show the difference between revisions 1.3 and 1.2.

The button 'Export' stores the selected revision to a file. It pops up a 'File save' dialog to allow you to pick the directory and filename where to save the file.

**Tags:** See section [Marking a release or milestone](#) on 3.

**Settings:** In the Settings dialog, the path to the RCS utilities is mandatory. The path and filename for the File-Compare tool is technically optional, but it is highly recommended to install such a utility and configure it in the settings. The option below it, 'original file on the left' is a preference for which file you wish to see at the left and which at the right, in a side-by-side file compare.



When checking in a file for the first time, you can mark it as binary. This flag is automatically set for a file whose extension matches one of the extensions in the list that can be entered here. The field must contain the extensions separated by spaces (and without period).

When adding a file to version control, the RCS tools store the archive file (the file with the ',v' extension) in a subdirectory 'RCS' below the directory where the work copy is. This way, 'RCS files' are kept separated from the work files. The option to 'store RCS files in subdirectory' lets RCS Browser create this subdirectory on a first commit, if it does not exist already.

## Tips & Background information

### Breaking a lock

May happen if you move the entire set of work files to a new machine, on which you have a different username. What you will see then in RCS Browser is that it considers all files to have been locked by someone else (that is, your old username). And as discussed previously, RCS does not allow you to edit files that are locked by someone else.

Before breaking locks, make sure that you have a backup copy of all work files.

To break the lock and assign it to yourself, do a 'Check out' and place a check-mark in the option 'Lock file'. When confirming, RCS Browser will warn you that this operation will break the lock, and that the changes of the 'other user' may be lost. Click 'Yes' to proceed, and the lock is reassigned to you.

After breaking the lock, verify for each work file whether it is the same as the backup copy you made. You can also copy the backup files back to the work files, to be sure.

### Removing a file from RCS

To remove a file from RCS, delete its archive file (the file with ',v' appended at its end). This file is found either in the same directory as the work file, or in the 'RCS' subdirectory below the path where the work file is.

### Locking and unlocking files

One goal of RCS is to avoid two persons working on the same file at the same time, where both would trample over the other's changes each time they save their work.

RCS solves this by 'locking' the file for exclusive use by a single user. Only that user can change the file as long as he or she holds the lock. The basic workflow of RCS, when you work in a team, is that you check-out a file, work on it, and then check it in again. The check-out operation locks the file, the check-in operation unlocks it).

When you are the only one working on the files, there is no use in unlocking a file, because there is nobody but you who will lock it (and you need to lock the file to do further edits). Therefore, the RCS Browser has 'Lock file' or 'Keep lock' options in its 'check in' and 'check out' dialogs, and these options typically already have a 'tick mark'<sup>1</sup> in them. Thus, when you check in files after implementing a new feature or fixing a bug, the files stay locked so that you can continue to work on it. In practice, you will rarely use the 'check out' function.

As a side note: at the level of the operating system, file locking is implemented by making files read-write or read-only. The terminology may be confusing, though:

RCS status	File status (of the work copy)
locked	writable (you can edit the file)
unlocked or locked by somebody else	read-only

### Which files to store in RCS?

As a general rule, only store files that you edit (directly or indirectly) to RCS. There is no need to store *generated* files in RCS, as these can be easily generated again. When it comes to software development, you would store the source files, Makefiles and files containing project settings to RCS, but you do not add object files, executable files or 'hex' files.

RCS (and version control in general) is primarily aimed at text files. Although you can add binary files to RCS, if you have a lot of large binary files, RCS is not the best choice (for instance, RCS is unable to create 'deltas' of binary files and it stores the data in an uncompressed way).

### Learning RCS

This guide merely scratches the surface of version control and of RCS in particular. For example, it completely skips over concepts like 'branches' and access restrictions. The goal of RCS Browser is to focus on the common operations that you use every day. The advanced features of RCS are excluded in order to keep the user interface simple.

<sup>1</sup> "checkmark" in American English.

However, if you come to the point that you want to step up, I recommend the 'RCS Handbook' by Brian O'Donovan, a book that was never published, but available as a PDF file. The other resource is the GNU RCS manual, available on-line.

## History of RCS

The Revision Control System was developed by Walter Tichy at the Purdue University, and first released in 1982. At the time, there was only one other version control system for Unix. SCCS, and that was difficult to acquire. RCS overtook SCCS almost instantly as the most popular version control system, mainly because its general availability, but also because of a few improvements over SCCS. As of release 4.3 of RCS, in 1990, RCS is distributed as open-source software by the GNU organization.

## Version numbering

Version numbers in RCS have two parts: the *release* number and the *revision* number.

*Release●Revision*

RCS increments the revision number on every check-in operation. The release number is never automatically adjusted, but it can be set explicitly. However, the release number is only a single number; RCS does not have the concept of major and minor version numbers. Moreover, when an application is released, it has a single version number, but it is built from various source files. Each of these source files has its own revision history. The version of a product/project is unrelated to the revision numbering of the individual source files.

It has therefore become common practice to leave the release number at its default: 1. The revision number is monotonically updated: it counts the number of revisions made in the file from the first commit. Whenever the product hits a new release, a tag with the release name is added to the entire set of source files. The tag name is more flexible than a single number, but more importantly, it applies to the set of source files (rather than to a single source file).

## Glossary

Throughout its history, the jargon used by RCS (and other Version Control systems) has changed over the years. In the literature (including this guide), you may therefore see different terms referring to the same concept.

Archive file	The file that holds the settings, status and history of a work file. In RCS, this file has the same name as the work file, but with ',v' appended to the extension. It is therefore also called a 'comma-v' file. The RCS manual refers to the archive file as the 'RCS file' and the <i>RCS Handbook</i> calls it the 'library file'.
Check-in	Submit changes in a work file to the version control system. Other version control systems call this a 'commit'.
Check-out	Save the latest version of the file in version control (the 'head') to the work copy of the file. Other version control systems call this an 'update', though RCS also adds the concept of file locking to the check-out operation.
Comma-v file	See Archive file.
Commit	Called 'Check-in' in RCS.
Export	Extract a version of the file from version control and save it to a file. You can export the head of the file, but it is typically meant to get an earlier release of the file.

Freeze	'Freezing a file set' means adding a tag to it. The official distribution lacks a utility to assign a tag to a set of related files. The third-party PERL script that provided this function is called 'rcsfreeze'.
Head	The latest version of the file in version control.
Library file	See Archive file.
Lock	When a file is locked by a user, no one else can update that file until it is unlocked.
RCS file	When the RCS manual mentions the 'RCS file', it refers to the file that holds the settings, status and history of a work file. The RCS file has the same name as the work file, but with ',v' appended to the extension. Other version control systems call this an 'archive' or 'archive file'.
Repository	The database or data file where the settings, status and history of the files. In the case of RCS, there is one archive file for every local work file. The repository is the collection of all these archive files.
Revision	A revision is a state of the work copy of the file at the time that it was committed (checked-in). It is also called a 'version'.
Symbolic name	A 'tag' is referred to as a 'symbol' or 'symbolic name' in the RCS manual and handbook.
Tag	A name that marks the revisions of a set of files that together form a release or a milestone. A tag serves as a single 'version identifier' for the various files in a project (while each of the files may have its own revision number).
Update	Called 'Check-out' in RCS.
Work copy	The local copy of the file, which is the file that you work in.

## Index

<b>A</b>		History.....	6
Add files.....	2	<b>K</b>	
Archive file.....	3, 6	Keep Lock (option).....	5
<b>B</b>		Keep locked.....	2
Binary file.....	2, 4	<b>L</b>	
Branches.....	5	Lock file.....	2
Break lock.....	4	Break.....	4
<b>C</b>		for exclusive edit.....	5
Change description.....	2	option.....	5
check-in.....	2	versus read-only status.....	5
Check-out.....	5, 6	<b>M</b>	
Commit changes.....	2	Meld.....	1
Compare changes.....	2	Milestone.....	3
<b>D</b>		Modification description.....	2
Description.....		<b>R</b>	
of a file.....	2	RCS file.....	6, 7
of modifications.....	2	Read-only files.....	5
<b>E</b>		Release.....	3
Exclusive edit access.....	5	Repository.....	7
Export archived releases.....	2	Revert changes.....	2
<b>F</b>		<b>T</b>	
File compare utility.....	1, 3	Tag.....	3, 6
File description.....	2	<b>V</b>	
File locking.....	2	Version number.....	6
Freeze.....	7	<b>W</b>	
<b>H</b>		WinMerge.....	1
Head.....	7	Work copy.....	4, 7