SimpleRCS

18/05/21 23:42:11

Table of Contents

1 Description

SimpleRCS is a semi automated revision control system designed with a single developer in mind. Working on a project on a local machine it is flexible, easy to setup and use, with no requirement for network access.

SimpleRCS is a semi automated revision control system inspired by the GNU rcs and as the name suggests, it is simple to use and easy to configure. It works with text based source files and stores backups in a RCS folder within the selected directory.

As a single developer, I needed a system that is reliable, easy to setup and use with a few clicks. It needed to store log and revision messages in the controlled files for reference.

So the general approach is a relatively manual system in comparison to the enterprise systems but convenient enough to make revision control painless in both configuration and use. The system puts entries in your files for quick reference and minimises the need to go through the painful process of having to use history and diffs if you want to revert back a couple of revisions.

I hope that you as a single windows developer finds this software useful.

What this is not, is a collaborative system that caters for teams working out of central servers. There are very good tools out there for that specific purpose. Another consideration is that unlike GNUrcs disk space is not a consideration so there is no compression of backups. An external compression software can be used without any effect to the SimpleRCS operation.

If you are new to revision control systems, I would suggest that you pursue some reading material to get a feel of how they work and the concepts used. There is a big variety with differing methodologies.

2 Backup Operation

2.1 Entering Keywords

Prior to backing up you will need to decide what keywords if any to place in your files (please see keywords). Normally you can place them in comments but it is possible to declare them as strings or char * variables and use them in print statements etc. Avoid using the LOG keyword for that purpose as it grows with each backup see General Practise Example.

2.2 Select Directory

Normal operation starts with selecting the directory or folder that contains the file/s that you would like to back up. Once the directory is selected, a file list appears in the list box where you can select the file/s that you would like to process. If an RCS folder doesn’t exist, you will be prompted to create one see Figure 2: Selecting the project folder.

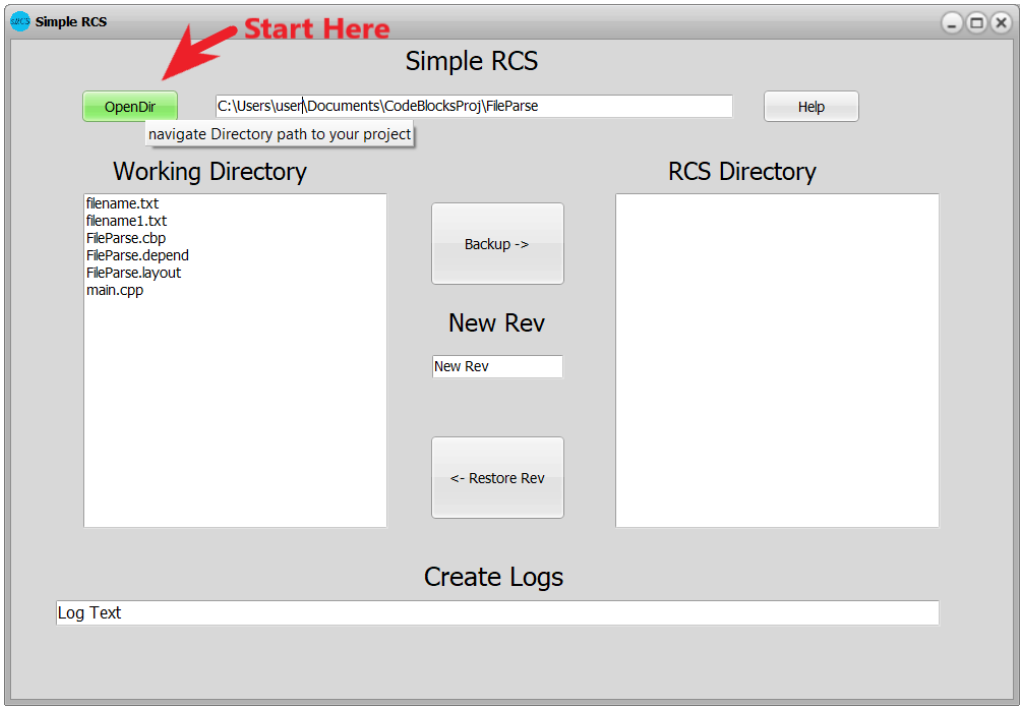


Figure 1: Start by selecting the project folder

2.3 Selecting Files

Multiple selections are made by holding the Control key and clicking on the files. Also, the Shift key can be used to select sequential files as you might use in a file manager see Figure 5: Selecting the HelloWorld.cpp file for revision control.

2.4 Entering fields

Enter or populate the REV and LOG fields with the information of your choosing. Examples below.

Entering these fields is not necessarily strict as to when you enter them as long as it is before the backup button is pressed.

2.5 Press Backup

Congrats, It’s done.

You can inspect the backup and the working files for the correct entries.

3 Keywords

LOG - plain text information that is entered on the log edit. This keyword grows with each revision to enable the display of a history. The newest entry is at the top.

REV - revision number that the user enters. Note that in this version there is no mechanism to auto increment but maybe in later versions with the right motivation. This keyword is suitable for print statements

$PATH$ - absolute path of the file that is revision controlled. This keyword is suitable for print statements but your path could be quite long.

The key words can be used for scripts etc if needed.

3.1 Revision Numbering

Revision numbering is left to the user to control manually. For a single developer, the numbering is is simply a matter of checking the log messages and determining what the next revision should be. Any set of ASCII characters will be valid.

Examples

new rev - valid

0.1 - valid

0.1a - valid

1.0 - valid

1.0test - valid

Jane'sCopy – valid

If a revsion number has been used, a warning will appear and the user is asked to confirm an overwrite. Use cancel to abort the operation and enter a new revision.

3.2 Logging

- All key identifiers are uppercase. Eg LOG
- LOG provides a logging mechanism for future reference and includes the log entry, Date Time and the revision number. For every revision, a new entry is added above the previous all of which are commented.
- Each log entry is prefixed with cpp/c++ line comment characters //. At this point there is no other mechanism for other languages. Later versions of the software will incorporate other comment characters depending on the interest.

4 General Practise Example

The intended use is to back up your working files and create logs logs within the working versions and the backups.

The first step is to enter the keywords into your working file/s where ever you deem is appropriate.

An example of initial condition:

```
// Hello World C++ Program
//$LOG$
//$REV$
//$PATH$
#include <iostream>
int main() {
   std::cout << "Hello World!";
   return 0;
}
```

The Rev in both working and backup files:

```
// Hello World C++ Program
//$LOG:
// 1.0 23/05/2021 5:07:10 AM initial revision for hello world$
//$REV: 1.0$
//$PATH: D:\Documents\HelloWorld$

#include <iostream>
int main() {
   std::cout << "Hello World!";
   return 0;
}
```

Rev 1.1 in both working and backup files:

```
// Hello World C++ Program
//$LOG:
// 1.1 23/05/2021 5:09:29 AM Updated hello world to hello there
// 1.0 23/05/2021 5:08:38 AM initial revision for hello world$
//$REV: 1.1$
//$PATH: D:\Documents\HelloWorld$

#include <iostream>
int main() {
   std::cout << "Hello There!";
   return 0;
}
```

The backups are stored in an RCS directory within the project folder and the user is prompted to create the folder if it doesn't exist.

If compressing your backups is important. Simply navigate to the RCS directory and compress/zip the files using your 3rd party software.

Before you use the system for production work, you should test your installation and process to get a good feel of what to expect.

5 Restoring/Revert

Before restoring I suggest you make a backup first and mark it with an appropriate revision No.

To Restore or revert to a previous version, select the file you wish to restore to the working directory and press the restore button.

6 Privacy and Security

SimpleRCS has no requirement for network access and does not have any networking capability so you can be assured that no usage logs or personal data is kept in any form.

Ideally, you should maintain an md5 checksum and verify that the software comes as an original executable package. You should refer to the many resources available for verification procedures.

7 Licence stuff

SimpleRCS use is without warranty or guarantees of any kind and you should test your system prior to production use.

Source code is closed and copyright remains with the developer: Pudo Engineering Pty Ltd.

Any suggestions or bug reports are welcome SimpleRCS@pudoengineering.com.au

8 Screen Shots

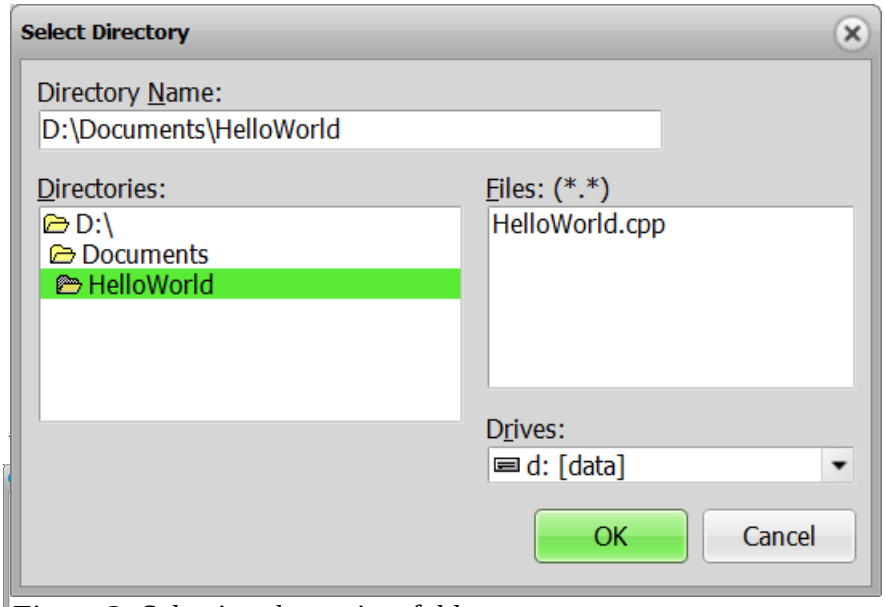
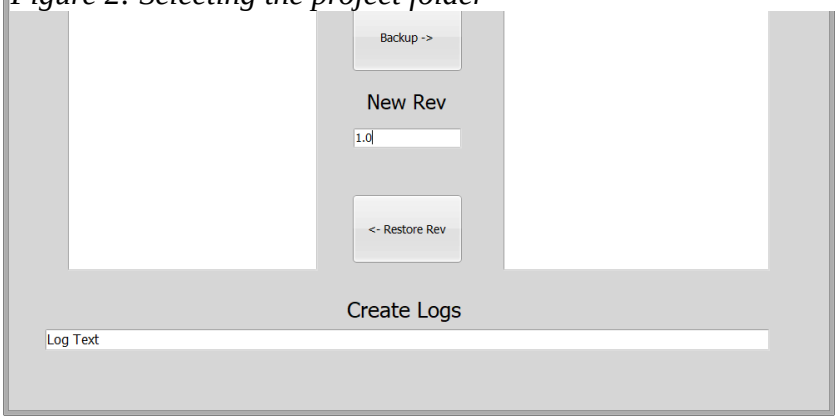


Figure 2: Selecting the project folder



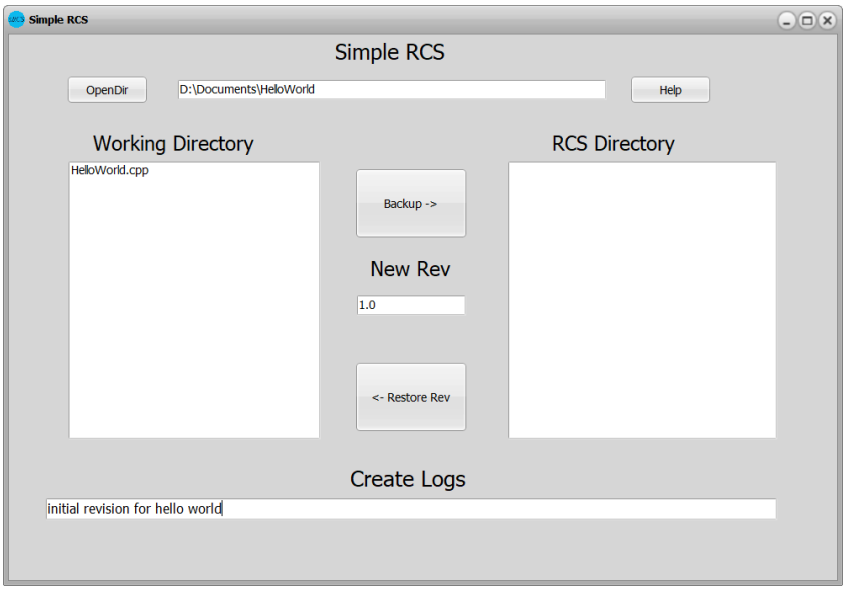


Figure 4: Entering log text (Create Logs)

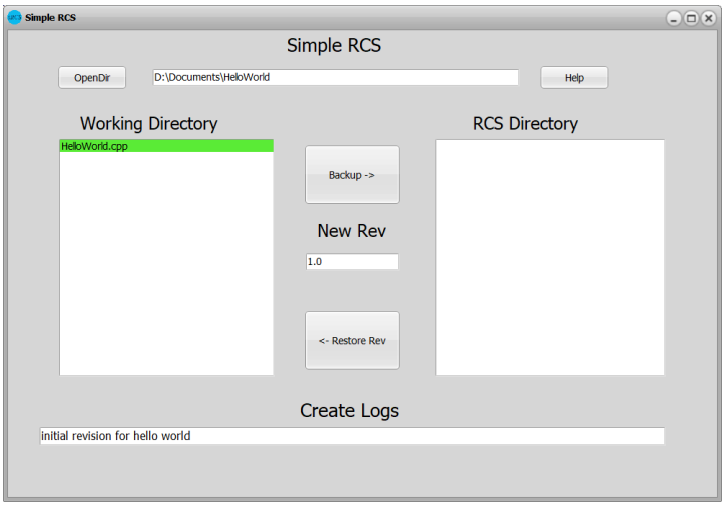


Figure 5: Selecting the HelloWorld.cpp file for revision control

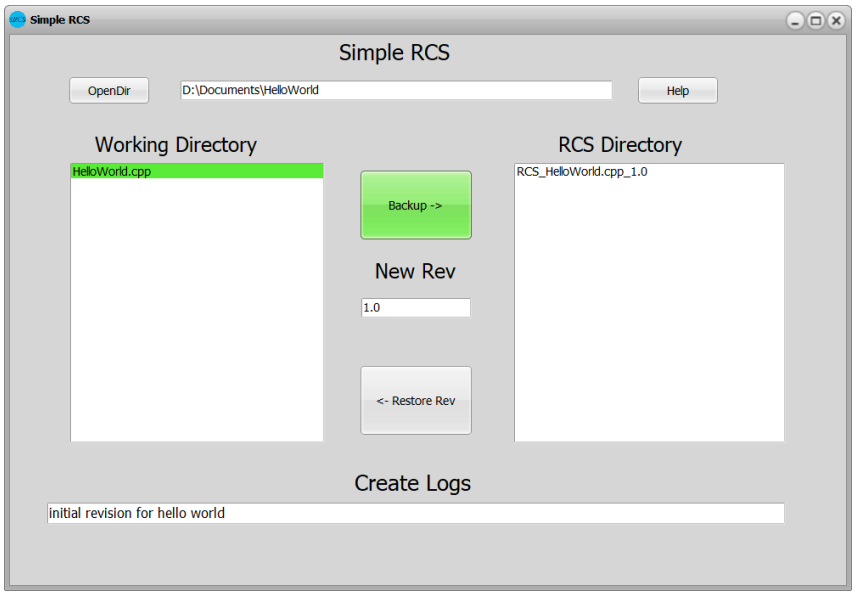


Figure 6: HelloWorld.cpp backed up to RCS folder. Note filename

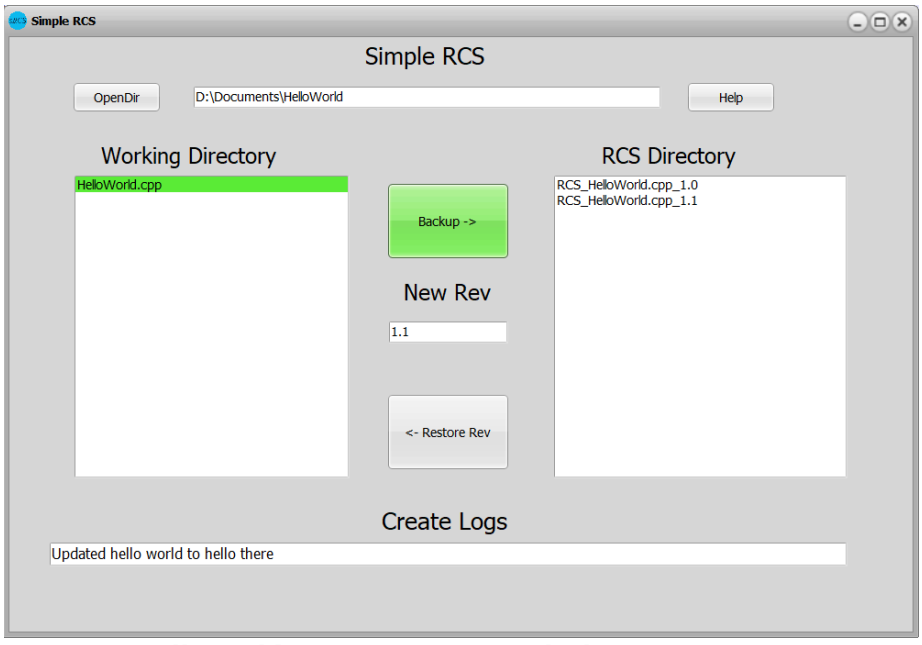


Figure 7: HelloWorld revision 1.1 - Note the log entry

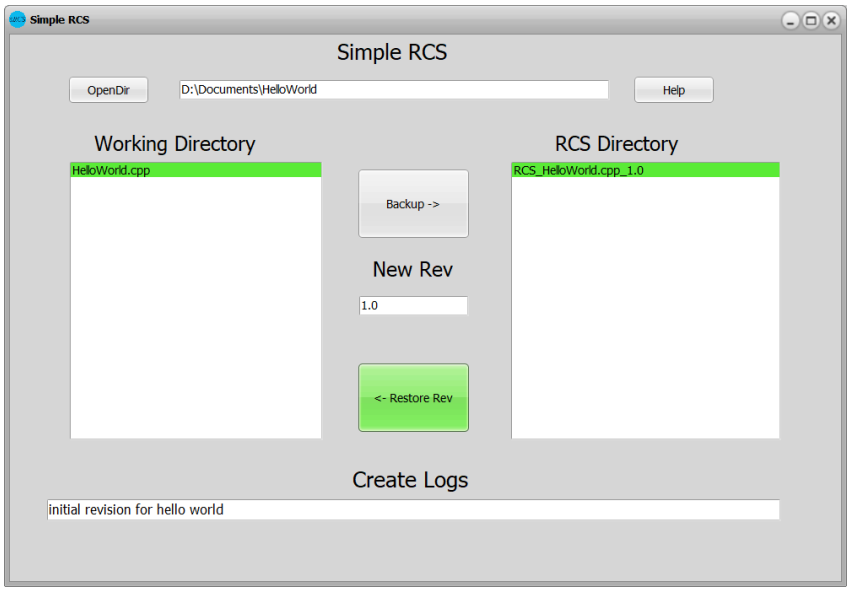


Figure 8: initiating a restore/revert from previous backup

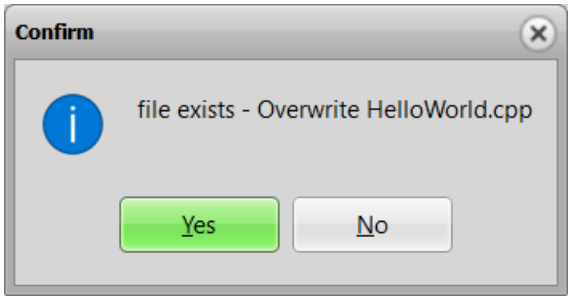


Figure 9: Message for reverting a file from previous version

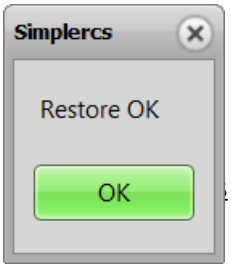


Figure 10