Lab Assignment 2. Implementation of an UART basic driver for data transmission.

1. Objective

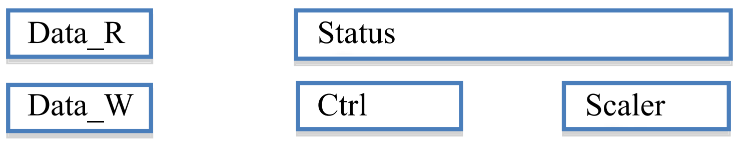
The purpose of the following practice is to implement a simple **driver** of an I/O interface. The selected interface is available in the configuration of the TSIM2 simulator of the LEON3 processor that is being used in this laboratory. Specifically, it is the **UART-A** (Universal Asynchronous Receiver-Transmitter-A), which allows asynchronous serial communication with other devices, both in input and output modes.

The implementation of a driver for such a device requires the knowledge of the **controller** behaviour at a certain level of detail. Each controller register has a specific function which must be analysed before to proceed the driver design. On the other hand, an interface can be configured to work in different ways, which will also determine the design approach of the controller. Some of the basic questions that condition this design are:

- Should the device be accessed in **block mode** or in **carater mode**?
- Should the driver be able to be used by more than one thread/process?
- The **driver** uses **polling** to check the controller state or interrupt controlled buffers are used to avoid the blocking of the polling method?

This practice will begin with the implementation of a simple driver in character mode for the UART-A of LEON3. The driver has a function that allows simply transmitting a character through this device using **polling**, so that the thread/process will remain in the function until the character has been queued into the transmission buffer. The practice also proposes an exercise related to the implementation of auxiliary routines used in functions of formatting and redirection of information to an output device. (like *printf*).

UART-A controller has 5 registers. Two of them are data registers, **Data_R** y **Data_W**, which are intended, respectively, to read the data received and write the data to be transmitted. The third one is the **Status** register, that provides information about the current status of the interface. The fourth is the **Ctrl** register, that controls the UART. Finally, the **Scaler** register, which is used to set the baudrate at which the UART will work. The following figure shows a schematic of the registers of the controller:

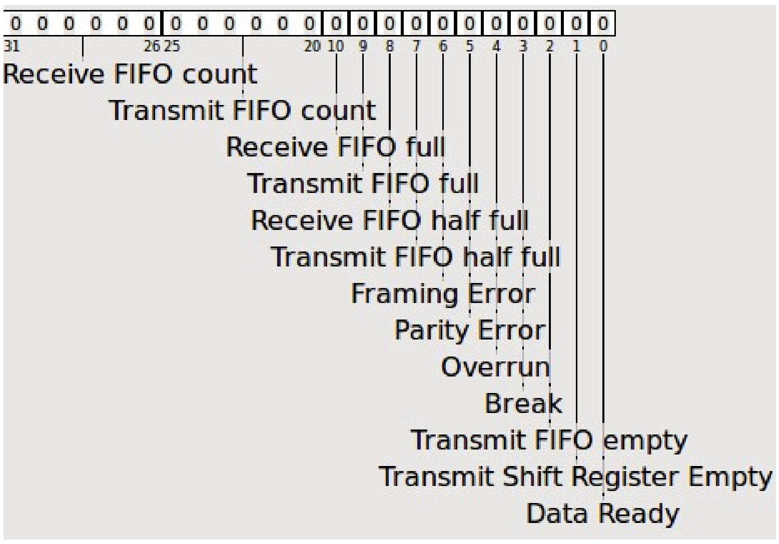


The simulator TSIM2 places the UART registers in the range of memory addresses 0x80000100-0x800010C.

Data_R and **Data_W** registers have the same address 0x80000100. The UART controller internally selects **Data_R** in the read operations and **Data_W** in the write operations. The fact that two registers share the same address is not unusual, since the controllers are designed trying to reduce the number of addresses used. However, it has an implication for debugging, since **it will not be possible to inspect the value that is written in Data_W.**

The **Status** register address is 0x80000104, while **Ctrl** and **Scaler** are located respectively at 0x80000108 and 0x8000010C.

The control of the UART used in this practice requires firstly knowing the behaviour of the **Transmission FIFO Full bit (TFF)** of the **Status** register (see figure below). This bit shows if the transmission fifo is full (value of **TFF** = 1). The algorithm to be used to transmit will simply consist of checking the value of the **TFF** bit before writing to the **Data_W** register. If it is 1, the fifo is full, and the driver will wait in a loop for a certain time for the bit to change its value. If after this time **TFF** remains at 1, it will be considered that there is an error and that it is not possible to transmit the data. In addition, in order to ensure that all the data in the FIFO has been transmitted, the ***Transmit FIFO Empty***(**TFE**) bit, which takes a value of 1 when the transmit FIFO is empty, will be monitored.



UART-A Status register description

2. LEON3 project creation

Create for this practice a new project (use the option *Empty Project*) called prac2 whose executable is for the Sparc Bare C platform. In that project create the subdirectories **include** and **src**. In the **include** directory create the file leon3_types.h with the following redefinition of the basic types for the architecture of the LEON3 processor

```
#ifndef LEON3_TYPES_H
#define LEON3_TYPES_H

typedef unsigned char         byte_t;
typedef unsigned short int    word16_t;
typedef unsigned int          word32_t;
typedef unsigned long int     word64_t;


typedef signed char           int8_t;
typedef signed short int      int16_t;
typedef signed int            int32_t;
typedef signed long int       int64_t;

typedef unsigned char         uint8_t;
typedef unsigned short int    uint16_t;
typedef unsigned int          uint32_t;
typedef unsigned long int     uint64_t;

#endif
```

3. *LEON3 UART-A driver basic functions for transmission.*

Create in the **include** directory the file *leon3_uart.h* with the following content:

```
#ifndef LEON3_UART_H_
#define LEON3_UART_H_

#include "leon3_types.h"

int8_t leon3_putchar(char c);

int8_t leon3_uart_tx_fifo_is_empty();

#endif /* LEON3_UART_H_*/
```

Create in the directory **src** the file *leon3_uart.c* with the following content in which is declared the structure UART_regs that facilitates the access to LEON3 UART-A registers
.

```
#include "leon3_uart.h"

//Data structure that facilitates the access to LEON3 UART-A
//registers

struct UART_regs
{
      /** \brief UART  Data Register */
      volatile uint32_t Data;       /* 0x80000100 */
      /** \brief UART  Status Register */
      volatile uint32_t Status;     /* 0x80000104 */
      /** \brief UART  Control Register */
      volatile uint32_t Ctrl;       /* 0x80000108 */
      /** \brief UART  Scaler Register */
      volatile uint32_t Scaler;     /* 0x8000010C */
};
```

The structure uses the **volatile** attribute that forces the compiler to not optimize the conditional execution code based on the value those fields can take. This allows these fields can be modified via hardware, and the software always checks the value they have, regardless of whether it has just been assigned to a constant value. The following two examples show the difference between declaring a variable as **volatile** or not.

```
uint8_t aux;
uint8_t i;

aux=0;

if(aux>0){

      // The optimizer does not include this code, nor the code
      // if(aux > 0), due to aux has been set to 0

}else{


}
```

```
volatile uint8_t aux;
uint8_t i;

aux=0;

if(aux>0){

      // The optimizer does include this code, and the code
      // if(aux > 0), because aux is volatile

}else{


}
```

Add to the file *leon3_uart.c* the definition `LEON3_UART_TFF` that is going to act as a mask of the ***Transmit FIFO Full*** bit of the Status register. Remember that for a character can be inserted in the transmission fifo queue it is necessary that the bit masked by `LEON3_UART_TFF` must **NOT** be 1.

```
                              (Complete MACROS ↓)
//!  LEON3 UART A Transmit FIFO is FULL
#define LEON3_UART_TFF

//!  LEON3 UART A Transmit FIFO is EMPTY
#define LEON3_UART_TFE
```

Add to file *leon3_uart.c* the declaration of the pointer `pLEON3_UART_REGS`. Complete the code so that this pointer points to the address 0x80000100, which is the address where the UART-A registers are placed

```
const struct   UART_regs * pLEON3_UART_REGS=   //<-COMPLETE
```

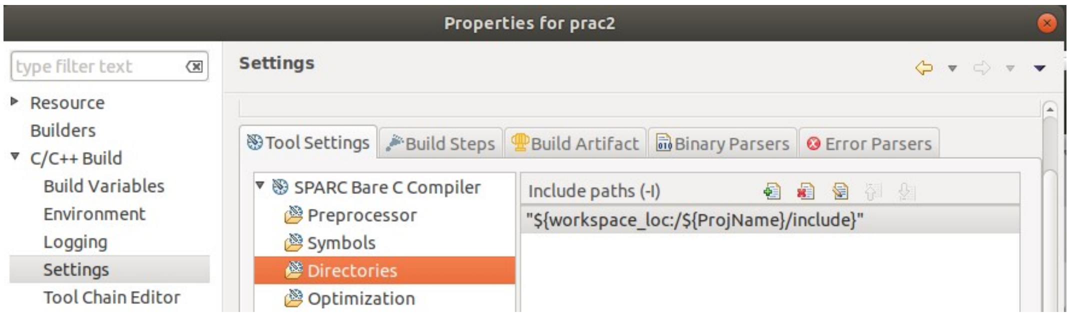
Complete the macro `leon3_UART_TF_IS_FULL` and the functions `leon3_putchar`, `leon3_uart_tx_fifo_is_empty` (all of them located at *leon3_uart.c* file *)*.

- `leon3_UART_TF_IS_FULL` is a macro that must return 0 only if the TFF bit of the Status register is set to 0, indicating that the transmission fifo queue is not full, and a character can be inserted in the transmission fifo.
- `leon3_putchar` on the other hand, after the TFF check, the function write the character c passed by parameter in the UART ***Data*** register. If after an interval (controlled by the write_timeout variable) the fifo is still full, the character will not be written in the ***Data*** register and the function will return a value other than 0.
- `leon3_uart_tx_fifo_is_empty` must return true if the TFE bit in the Status register is set to 1, indicating that the transmit FIFO queue is empty.

```

                  //COMPLETE MACRO ↓
#define leon3_UART_TF_IS_FULL() (   )


int8_t leon3_putchar(char c)
{

      uint32_t write_timeout=0;

      while(leon3_UART_TF_IS_FULL()&&(write_timeout < 0xAAAAA))
      {
                  write_timeout++;

      } //wait while Tx FIFO is full


      if(write_timeout <  0xAAAAA){
            //COMPLETE. Write the param c in the Data register

      }
      return (write_timeout ==  0xAAAAA);
}

int8_t leon3_uart_tx_fifo_is_empty (){
            //COMPLETE.

}
```

Before compiling, configure the project header file search *path* to search in the *include* directory of the project. To do this, select the **prac2** project, right-click on it and open the *Properties* panel, and following the menu sequence *C/C++|Build-Settings|Sparc Bare C Compiler|Directories*, add in *Include paths* the path to the ***include*** directory of the project, using a path relative to the ***workspace***, as shown in the following image.

Using a path relative to the workspace will avoid problems when taking your project to other computers.



"${workspace_loc:/${ProjName}/include}"

4. First version of the main program.

Create the *main.c* file with the following test code of the basic driver and check that the output is:

p2

```
#include "leon3_uart.h"

int main()
{

      leon3_putchar('p');
      leon3_putchar('2');
      leon3_putchar('\n');
      while(!leon3_uart_tx_fifo_is_empty())
        ;
      return 0;

}
```

What would be on the console if you used the following main program? And if the initial assignment was `char aux=0;`?

```
int main(){
      int i=0;
      char aux='0';
      for (i=0; i<10; i++){
            leon3_putchar(aux); leon3_putchar('\n');
            aux++;
      }
      while(!leon3_uart_tx_fifo_is_empty())
        ;
      return 0;
}
```

5. Routines for formatting and transmission through the UART.

Add the files *leon3_bprint.h* (to the include directory) and *leon3_bprint.c* (to the *src* directory) to the project. Implement in these files the following auxiliary routines for formatting information and transmit it through the UART.

```
//transmits through the UART the string passed as a parameter

int8_t leon3_print_string(char* str);  //see **

// transmits through the UART the integer formated as a string

int8_t leon3_print_uint8(uint8_t i);      // see ***
```

** (Note for the `leon3_print_string` implementation that the last character of a string is the '\0)
*** (Note for the `leon3_print_uint8` implementation that an 8-bit integer has a maximum value of 255, so for its transformation it will be enough to calculate 3 characters: that of the hundreds, that of the tens and that of the units)

Use internally in the implementation of both functions the function `leon3_uart_tx_fifo_is_empty`() to ensure that all characters have been transmitted before returning from the function.

Verify that the implementation is correct by means of the following main program that must give as output:

```
cadena
3
43
108
```

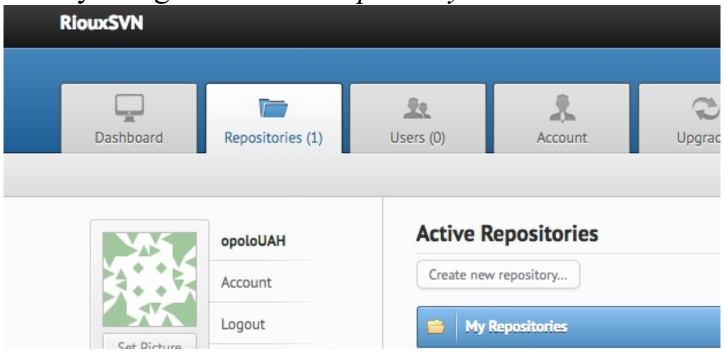
```
#include "leon3_uart.h"
#include "leon3_bprint.h"

int main()
{
      char * pchar="cadena\n";
      leon3_print_string(pchar);
      leon3_print_uint8(3); leon3_putchar('\n');
      leon3_print_uint8(43); leon3_putchar('\n');
      leon3_print_uint8(108); leon3_putchar('\n');
      return 0;
}
```

6. Configuration Control.

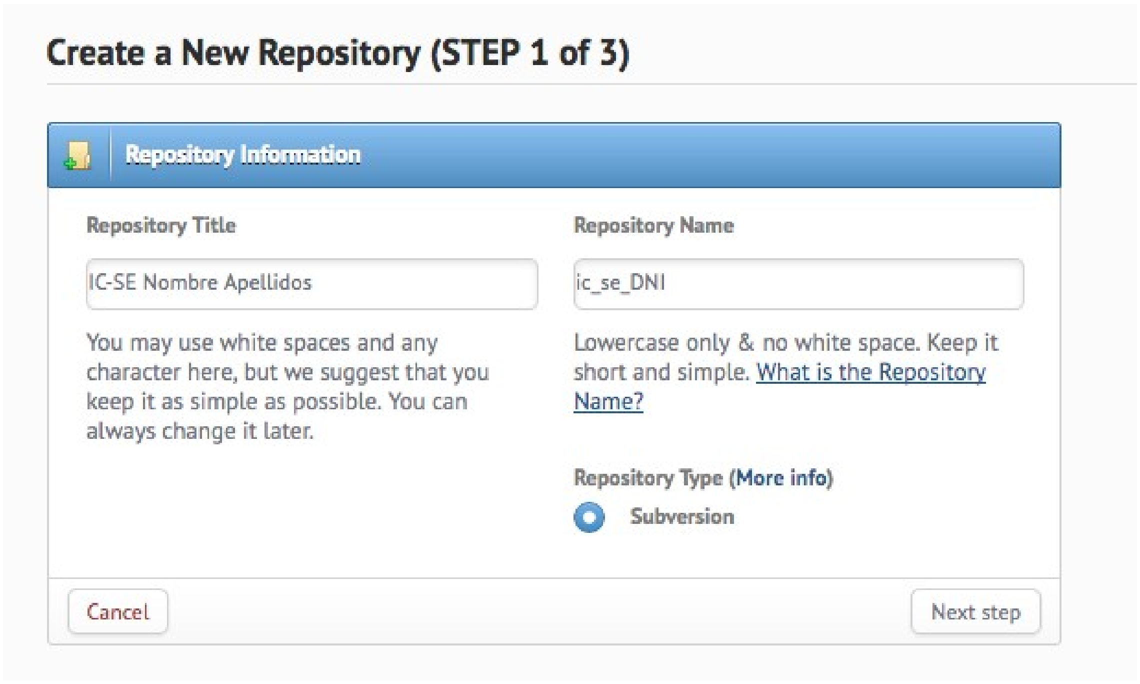
The software project configuration control allows engineers to have at all times the traceability of changes that have occurred in the source code of their projects, as well as to associate labels to the project status in specific phases of development. For this laboratory we are going to use as a configuration control tool a Subversion client (SVN) integrated as a plugin in the Eclipse environment (You can also use git if you wish, just share the project with the teachers). In addition, following the practice script you will learn how to create a SVN repository that you will share with your teachers.

How to create a SVN repository in riouxsvn.com.

Go to https://riouxsvn.com and create a new account to free access to SVN "hosting". Go to *Repositories,* as it is shown in the next, and use the section *Active Repositories* to create a new repository using *Create new repository*.



Se the **Repository Title** with your name and surname, adding IC-SE as a prefix, and use ic_se_DNI as the **Repository Name**, where DNI is the number of your national identity card, as shown in the following figure:


Create a New Repository (STEP 1 of 3)

Repository Information

Repository Title: IC-SE Nombre Apellidos

You may use white spaces and any character here, but we suggest that you keep it as simple as possible. You can always change it later.

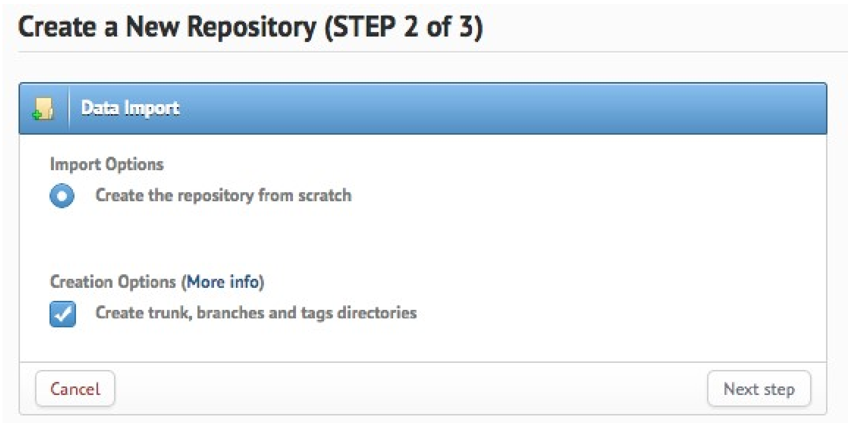
Repository Name: ic_se_DNI

Lowercase only & no white space. Keep it short and simple. What is the Repository Name?

Repository Type (More info)

● Subversion

In the **Creation Options** list, activate the option that create the *trunk, branches and tags* directories, as it is shown in the following figure:



Create a New Repository (STEP 2 of 3)

Data Import

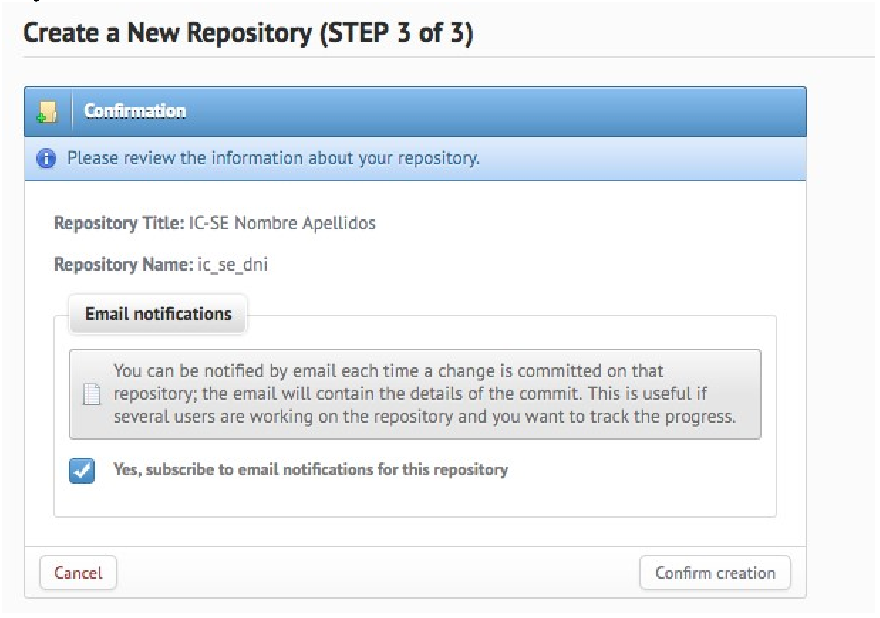
Import Options

[✓] Create the repository from scratch

Creation Options (More info)

[✓] Create trunk, branches and tags directories

In the step 3, activate the option that enable the email notification after modifications in the repository.



Create a New Repository (STEP 3 of 3)

Confirmation

Please review the information about your repository.

Repository Title: IC-SE Nombre Apellidos

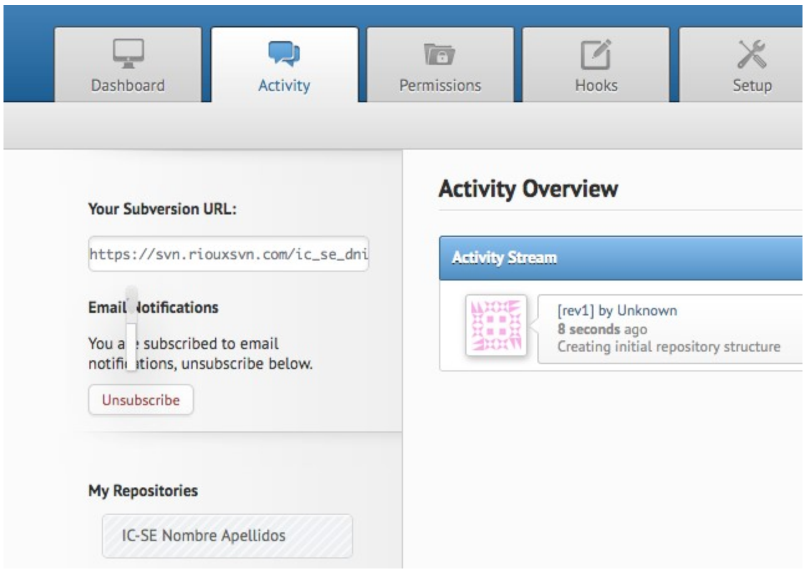
Repository Name: ic_se_dni

Email notifications

You can be notified by email each time a change is committed on that repository; the email will contain the details of the commit. This is useful if several users are working on the repository and you want to track the progress.

[✓] Yes, subscribe to email notifications for this repository

Finally, in the section *Checkout your repository*, copy and save in your personal notes the URL of your repository that appears in the *Your Subversion URL*:.



Your Subversion URL:

https://svn.riouxsvn.com/ic_se_dni

Email Notifications

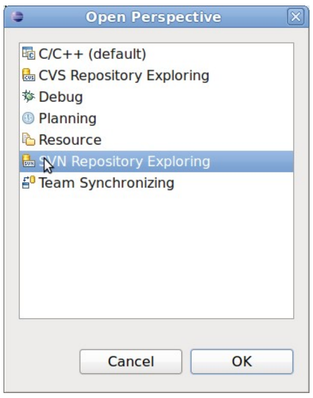
You a subscribed to email notifi tions, unsubscribe below.

IC-SE Nombre Apellidos

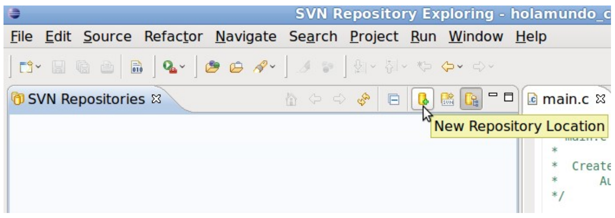
[rev1] by Unknown
8 seconds ago
Creating initial repository structure

How to add a project to your repository.

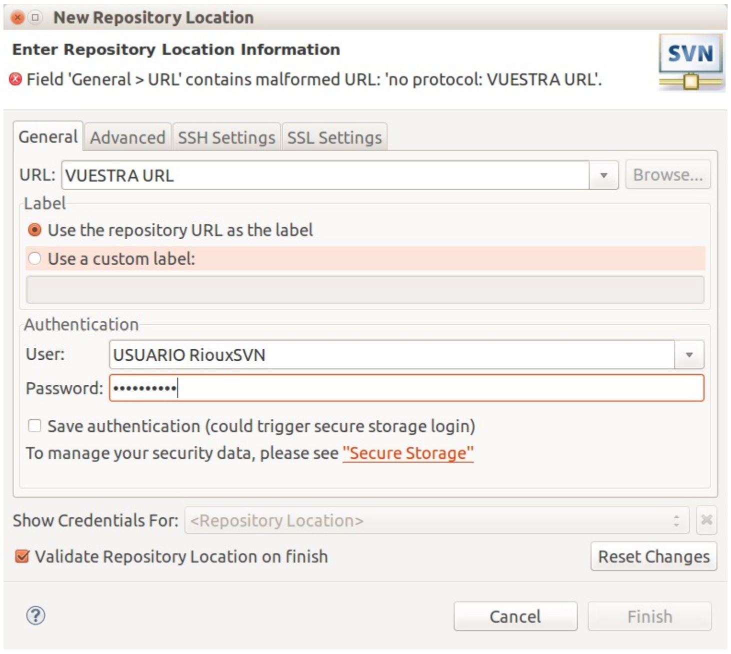
Open Eclipse, use the menu *Window/Open Perspective.* Select in it the perspective *SVN Repository Exploring* as it is shown in the following picture. This perspective allows the access to the SVN client.



Add a new repository location (*New Repository Location*).


SVN Repository Exploring - holamundo_c
File Edit Source Refactor Navigate Search Project Run Window Help
SVN Repositories
main.c
New Repository Location

Complete the repository location information using the URL of your RiouxSVN repository (check the RiouxSVN menu *Checkout your repository*), set *User* and *Password* with your RiouxSVN user and password.



New Repository Location

Enter Repository Location Information

Field 'General > URL' contains malformed URL: 'no protocol: VUESTRA URL'.

General | Advanced | SSH Settings | SSL Settings

URL: VUESTRA URL

Label

[✓ Use the repository URL as the label]

Use a custom label:

Authentication

User: USUARIO RiouxSVN

Password: ••••••••••

Save authentication (could trigger secure storage login)

To manage your security data, please see "Secure Storage"

Show Credentials For: <Repository Location>

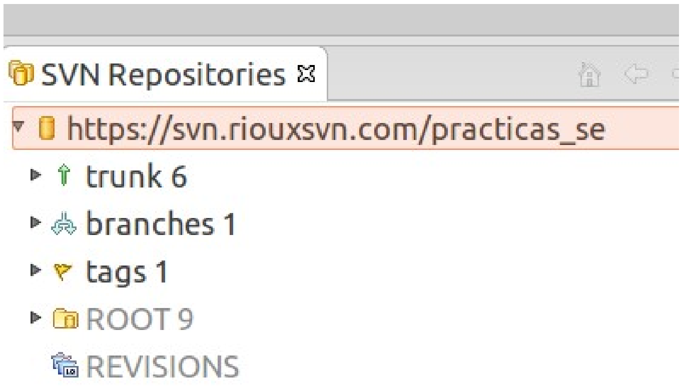
[✓ Validate Repository Location on finish]

SVN repository Organization.

Repositories are usually organized in three folders:

- trunk: it stores the last configured version of the project you are working on
- branches: it stores different alternatives of the project under study. If on of them is consolidated, it will be merged with the trunk.
- tags: it stores different intermediated versions of a project. Each tagged version allows to retrieve the existing version of the files at the time the tag was created. Each tag can be retrieved independently, can be compared with the trunk, of with other tags, and can be set as the current version of the trunk.

Check your repository has been created by Rioux with this organization:

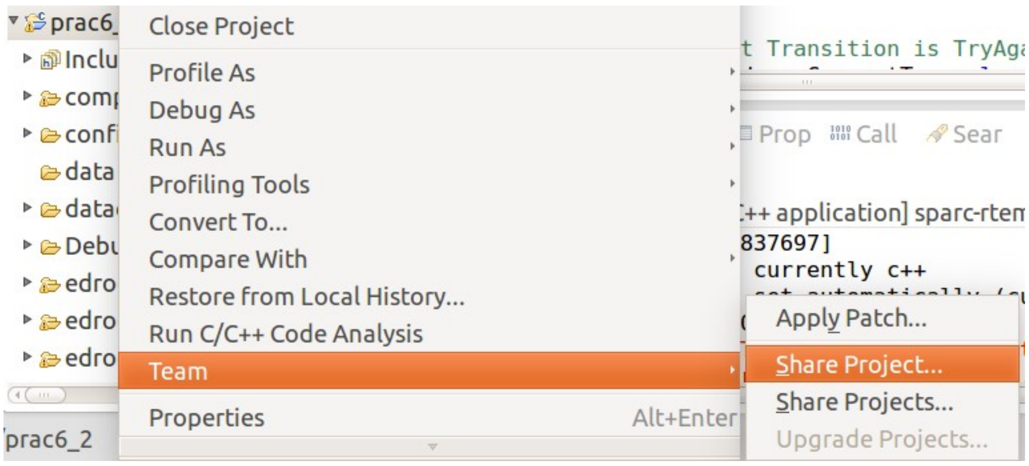


SVN Repositories

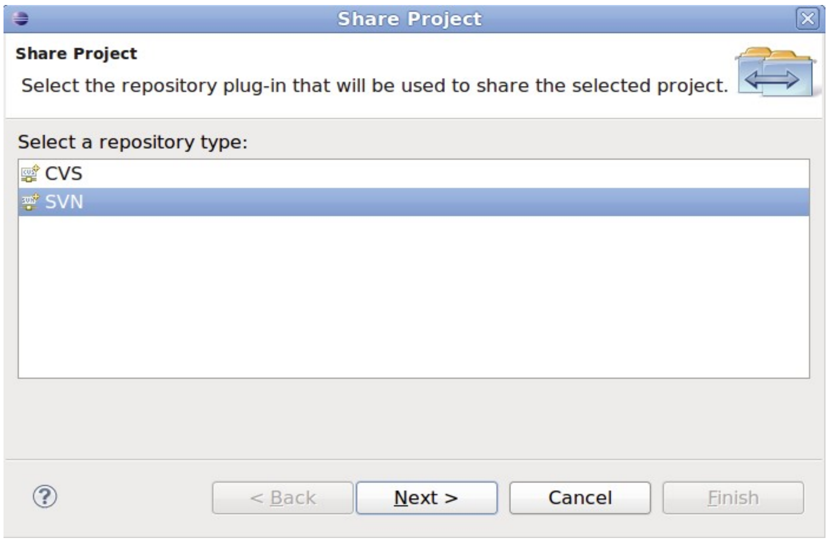
https://svn.riouxsvn.com/practicas_se
- trunk 6
- branches 1
- tags 1
- ROOT 9
- REVISIONS

To-do. Add your project in your SVN repository.

From *C++* view, and after the project ***prac2*** works properly, use the project ***prac2*** context menu *Team->Share Project* in order to add it to your repository.



Select *SVN* as repository type.



Share Project

Share Project

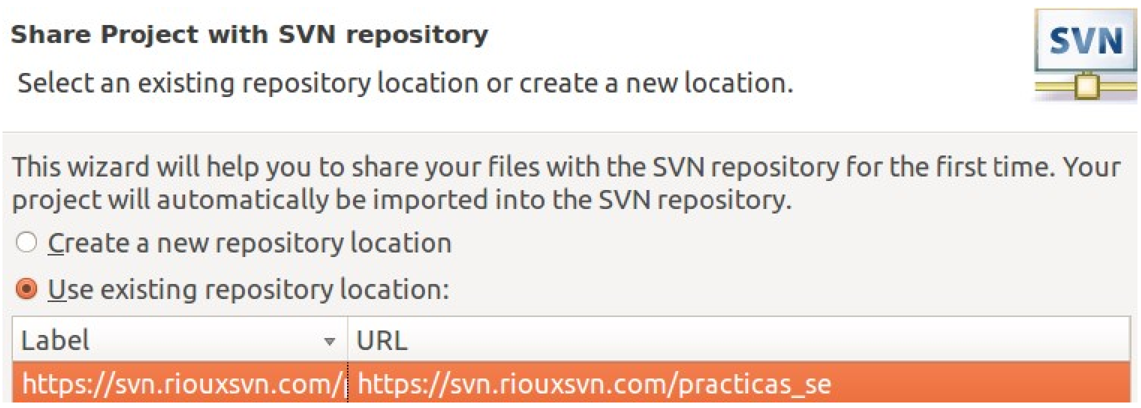
Select the repository plug-in that will be used to share the selected project.

Select a repository type:

CVS

SVN

Select ***Use existing repository location.***



Share Project with SVN repository

Select an existing repository location or create a new location.

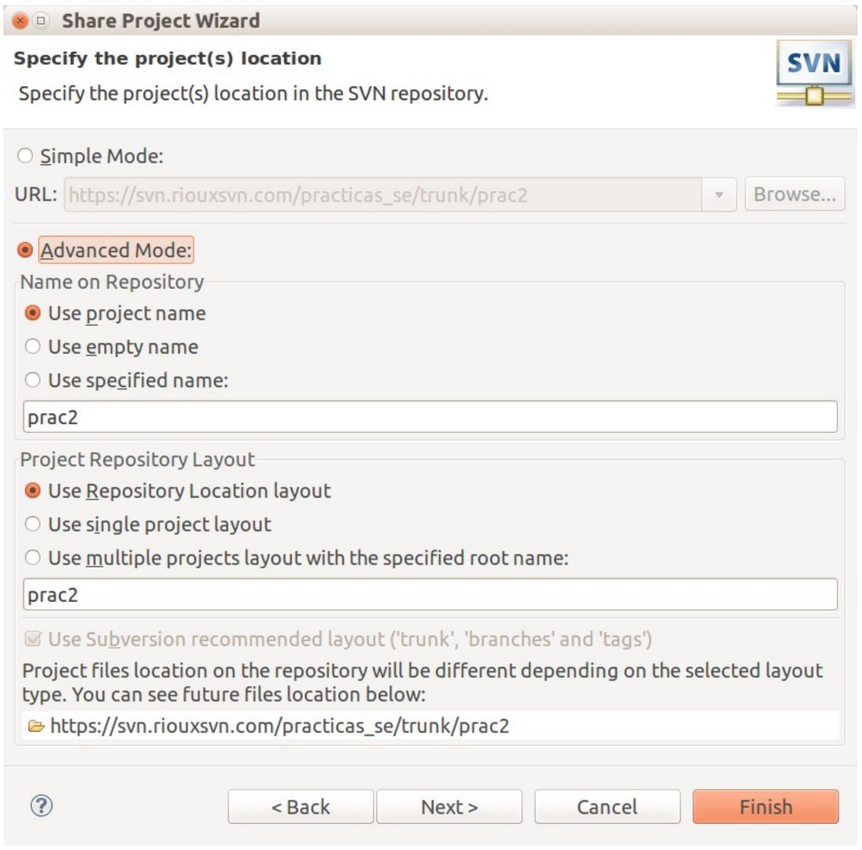
This wizard will help you to share your files with the SVN repository for the first time. Your project will automatically be imported into the SVN repository.

- [ ] Create a new repository location
- [✓] Use existing repository location:

Label	URL
https://svn.riouxsvn.com/	https://svn.riouxsvn.com/practicas_se

Use **doble click** on the URL and chose the URL of your repository.

In the following window, select **Advanced Mode,** and check the following options, checking the final URL is https://svn.riouxsvn.com/.../**trunk/prac2**.



Share Project Wizard

Specify the project(s) location

Specify the project(s) location in the SVN repository.

- [ ] Simple Mode:

URL: https://svn.riouxsvn.com/practicas_se/trunk/prac2

- [✓] Advanced Mode:

Name on Repository

- [✓] Use project name
- [ ] Use empty name
- [ ] Use specified name:

prac2

Project Repository Layout

- [✓] Use Repository Location layout
- [ ] Use single project layout
- [ ] Use multiple projects layout with the specified root name:

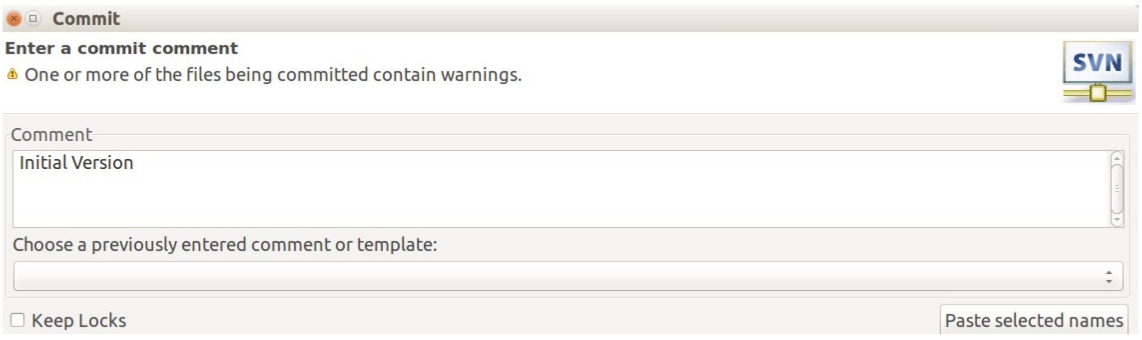
prac2

- [✓] Use Subversion recommended layout ('trunk', 'branches' and 'tags')

Project files location on the repository will be different depending on the selected layout type. You can see future files location below:

https://svn.riouxsvn.com/practicas_se/trunk/prac2

Finally, use *Finish* to add your project to your repository. The following window allows you to comment the version. Edit it with “Initial Version” and press OK.


Commit

Enter a commit comment

One or more of the files being committed contain warnings.

Comment

Initial Version

Choose a previously entered comment or template:

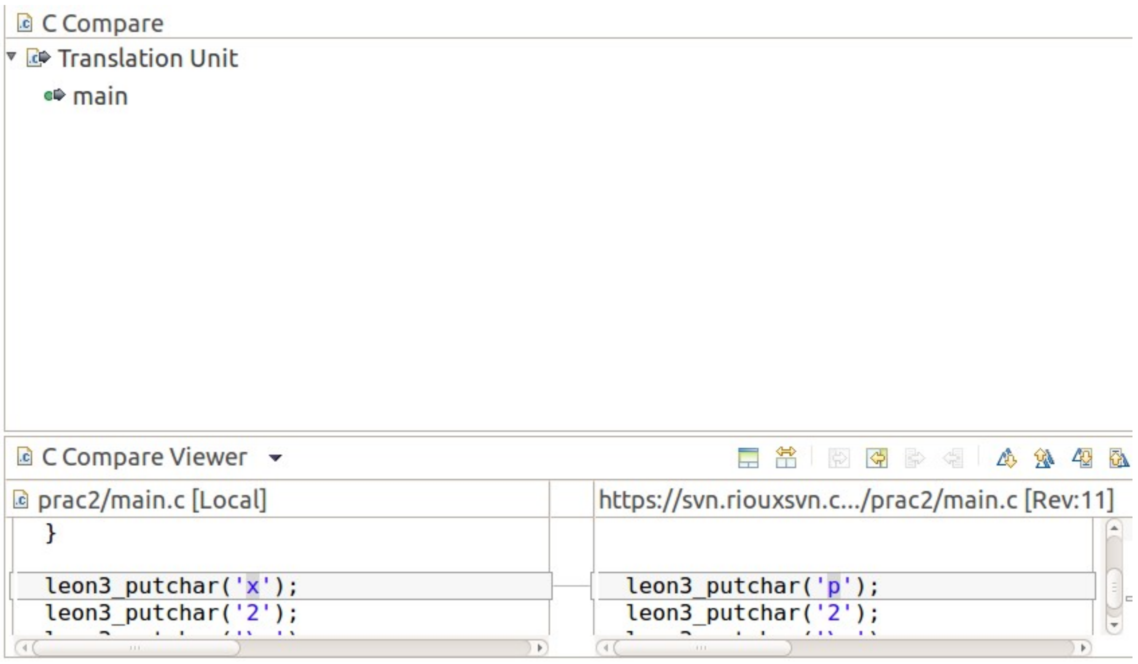
Keep Locks

Paste selected names

After the project has been configured, any change in a source file will be detected by the eclipse SVN plugin
The following figure shows an original line of the *main.c* file (with the sentence *leon3_putchar('p'))* that has been substituted by *leon3_putchar('x');*
Check the symbol > is showed near the file *main.c* to mark the modification.

Select the file and use the right button option *Compare With->Latest from Repository* to show a comparison with the last configured version of this file.

The following information:


C Compare

Translation Unit

main

C Compare Viewer

prac2/main.c [Local]

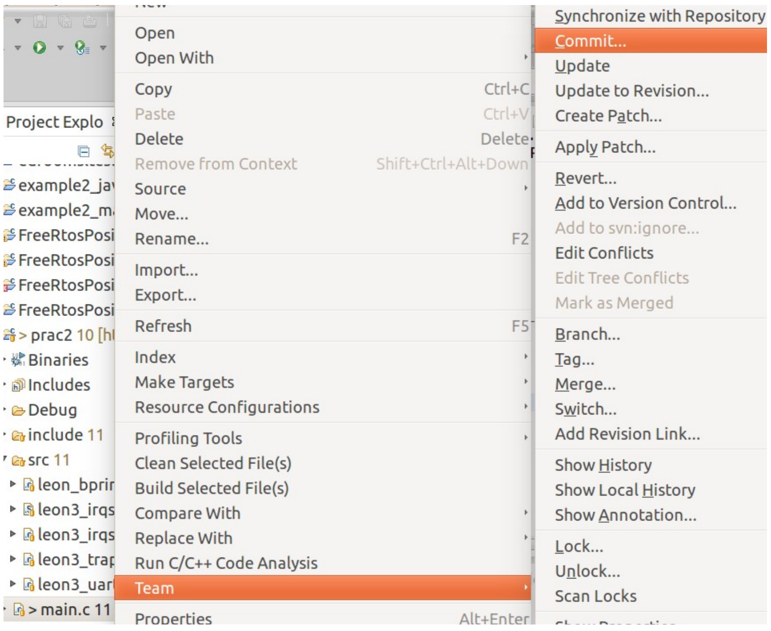
```
}

leon3_putchar('x');
leon3_putchar('2');
```

https://svn.riouxsvn.c.../prac2/main.c [Rev:11]

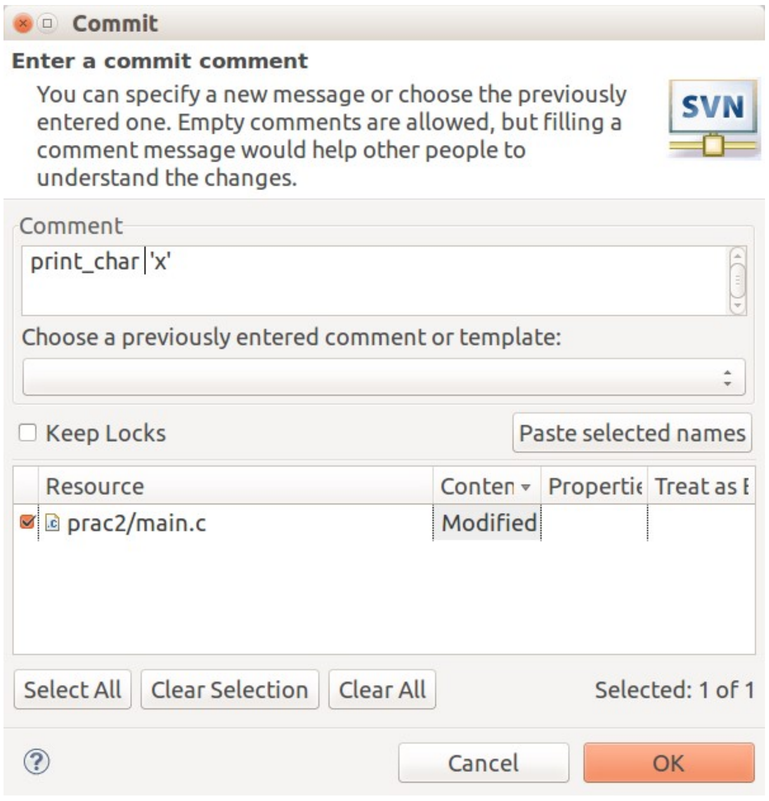
```
leon3_putchar('p');
leon3_putchar('2');
```

With this tool is possible to compare files, folders and complete projects. The files with differences will be appear marked and you can select each one to check de differences.

Finally, using the menu *Team->Commit* the new version of the file can be upload and configured into the repository.



A comment can be added to clarify the reasons of the modification.


Commit

Enter a commit comment

You can specify a new message or choose the previously entered one. Empty comments are allowed, but filling a comment message would help other people to understand the changes.

Comment

print_char 'x'

Choose a previously entered comment or template:

☐ Keep Locks

Resource	Conten	Propertie	Treat as
☑ prac2/main.c	Modified		

Selected: 1 of 1

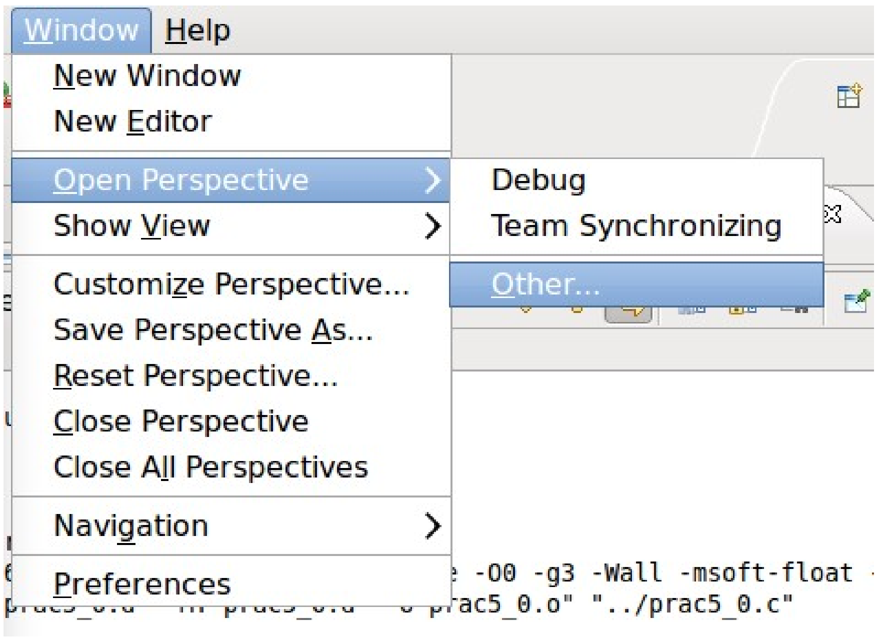
Other useful SVN commands are:

- Revert: Undoes changes made since the last configured version
- Update: Update your local project/folder/file with the last version configured in the repository.

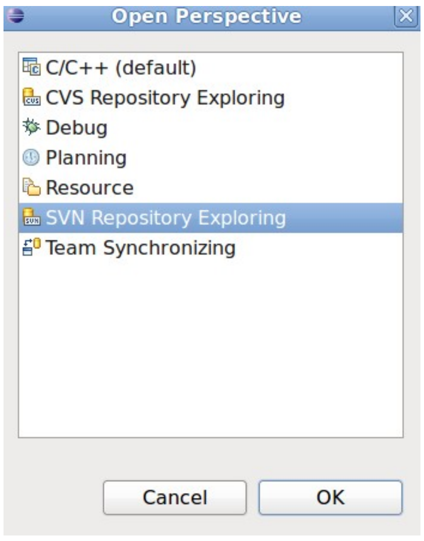
- Synchronize with Repository: This shows the source code modification respect to the configured versions.
- Branch: Create a branch using the working version.
- Tag: Create a tag using the working version.
- Switch: Change to other version configured in a Branch or in a Tag.
- Disconnect: Broke the link with the repository.
- Merge: Merge the current version with the version configured in the repository.

Project Check Out.

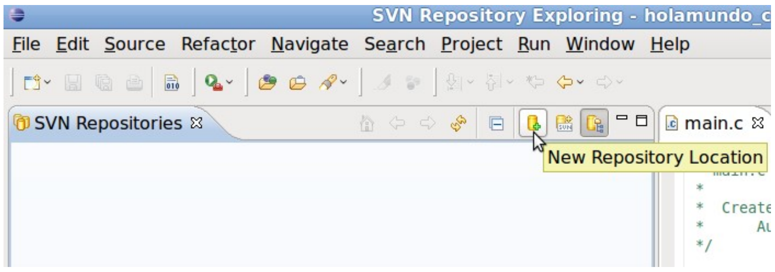
A project *Check Out* allows to download to any computer the configured project in order to work with it. This must be done when you use other computer, like your computer at home, so you can always work in any computer with the last committed version. To do a *Check Out* it is necessary to use the menu *Window->Open Perspective* and click the option *SVN Repository Exploring* to access to your SVN repositories.



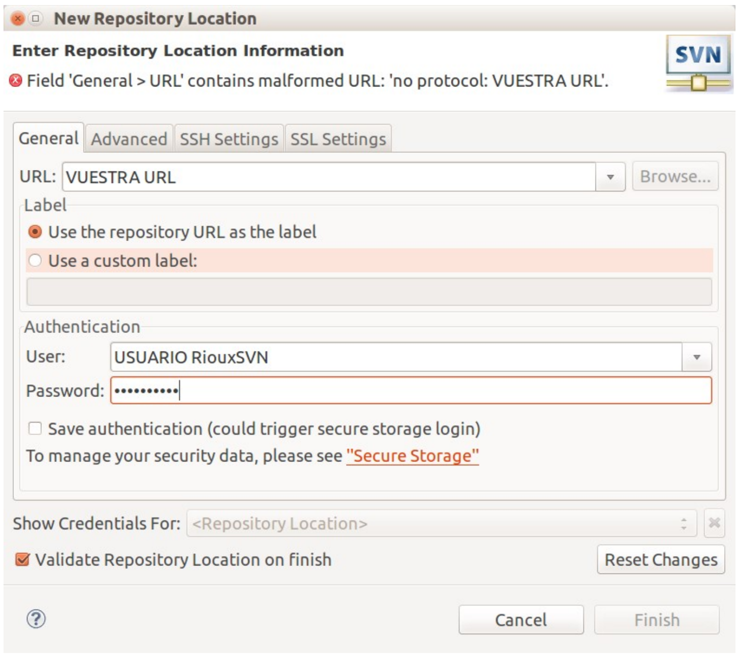
```
-O0 -g3 -Wall -msoft-float
rac5_0.o" "../prac5_0.c"
```



Once in this perspective, if you are in other computer you need to add the URL of your repository as a new location (*New Repository Location*),


SVN Repository Exploring - holamundo_c
File Edit Source Refactor Navigate Search Project Run Window Help
SVN Repositories
main.c
New Repository Location

the information about your repository includes the RiouxSVN URL, and the *User* and *Password* of your RiouxSVN account.


New Repository Location

Enter Repository Location Information

Field 'General > URL' contains malformed URL: 'no protocol: VUESTRA URL'.

General | Advanced | SSH Settings | SSL Settings

URL: VUESTRA URL

Label
- [✓] Use the repository URL as the label
- Use a custom label:

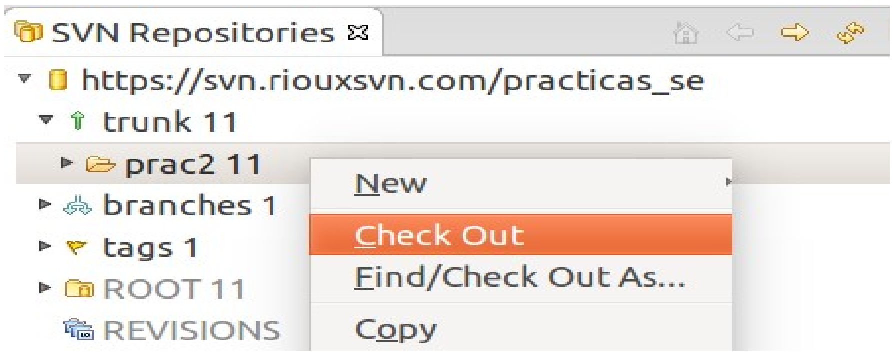
Authentication
User: USUARIO RiouxSVN
Password: ••••••••••

Save authentication (could trigger secure storage login)
To manage your security data, please see "Secure Storage"

Show Credentials For: <Repository Location>

[✓] Validate Repository Location on finish

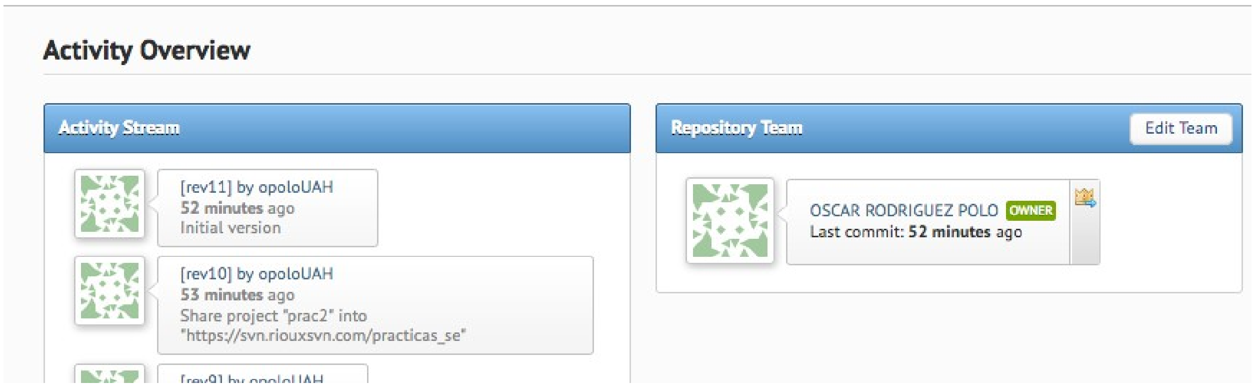
The *trunk* directory allows to perform a Check Out



After Check Out, the C++ perspective shows the project ready to work with it.

To-do. Share the project with the professor.

RiouxSVN allows to share the project with the professor. Do *clic* on the project name to entry in the menu **Activity,** where it is possible to share the repository with other users using **Edit Team**.



Activity Overview

Activity Stream

[rev11] by opoloUAH
52 minutes ago
Initial version

[rev10] by opoloUAH
53 minutes ago
Share project "prac2" into "https://svn.riouxsvn.com/practicas_se"

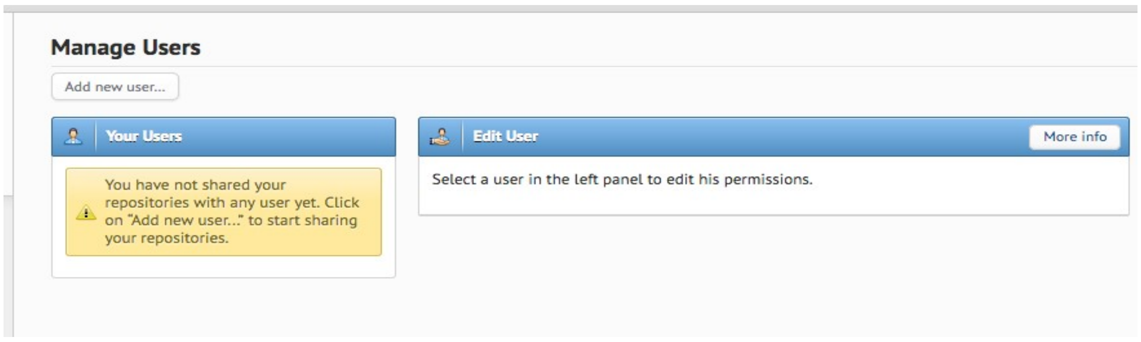
[rev9] by opoloUAH

Repository Team

Edit Team

OSCAR RODRIGUEZ POLO OWNER
Last commit: 52 minutes ago

Use the option **Add New User** to share your project with the user **edeldiaz.** That will enable the lab professor to check the practice that has been uploaded to the repository.



Manage Users

Add new user...

Your Users

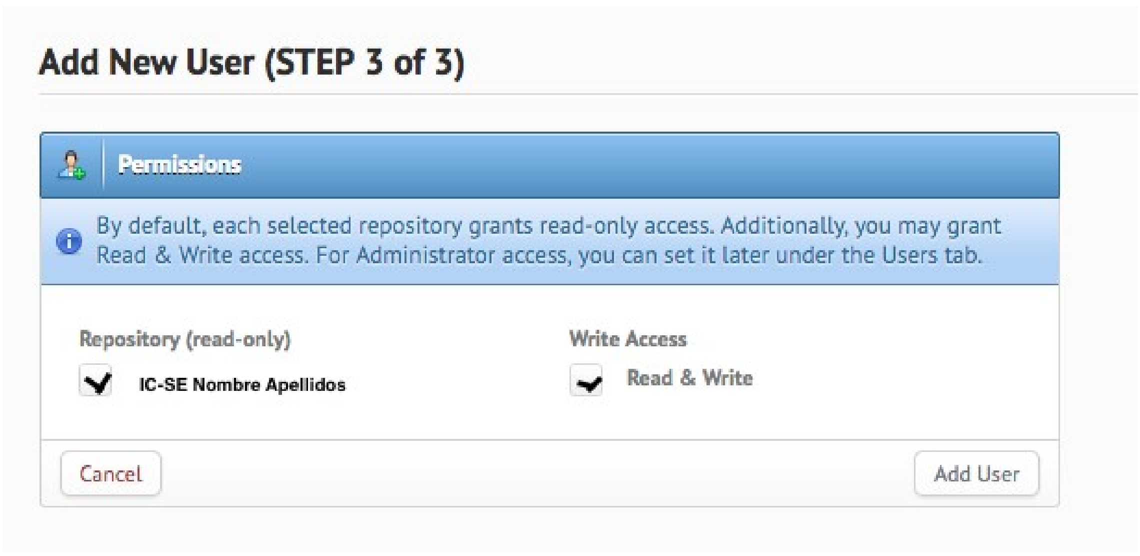
You have not shared your repositories with any user yet. Click on "Add new user..." to start sharing your repositories.

Edit User

More info

Select a user in the left panel to edit his permissions.

In the menus **Add New User** activate Read & Write Access, as it is shown in the following figure, in order the professor can review your practice.



Add New User (STEP 3 of 3)

Permissions

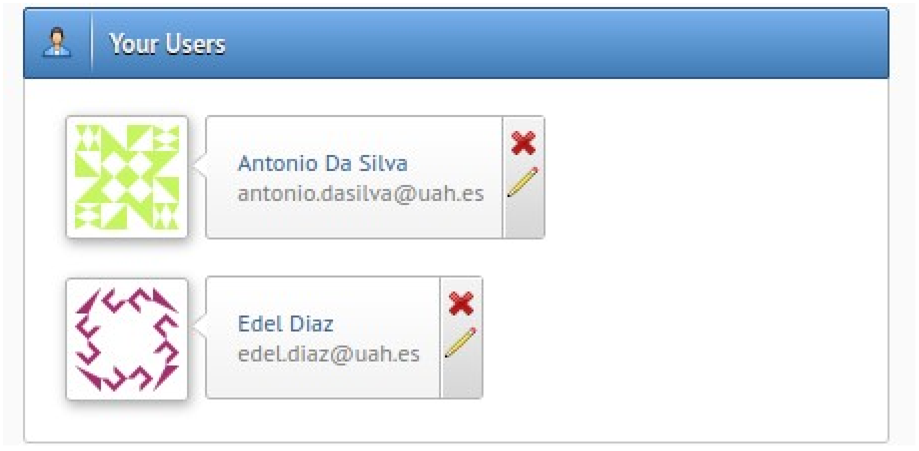
By default, each selected repository grants read-only access. Additionally, you may grant Read & Write access. For Administrator access, you can set it later under the Users tab.

Repository (read-only)	Write Access
✓ IC-SE Nombre Apellidos	✓ Read & Write

Cancel

Add User

Use again the menu **Add New** to add the other lab professor **dasiUAH**. The final configuration will be the following:



Antonio Da Silva
antonio.dasilva@uah.es

Edel Diaz
edel.diaz@uah.es

Once the projects have been shared, any comments made by the professor on your projects will be notified to you via email, and you will be able to analyze, from eclipse, the detail of the professor's comments with the tool *Compare With->Latest from Repository***.**