

Práctica 1. Instalación de un Entorno de Desarrollo, Compilación Cruzada y Simulación para el Procesador LEON 3

1. Introducción

En este documento se describe el proceso de instalación de un entorno de desarrollo y compilación cruzada para el procesador LEON3. La práctica explica además cómo integrar, en dicho entorno, el control de la ejecución de programas sobre un simulador de este procesador denominado *TSIM2*. LEON3 pertenece a la familia SPARC, y ha sido seleccionado como procesador de referencia por la Agencia Espacial Europea - European Space Agency (ESA) para sus misiones espaciales. Por este motivo forma parte, en numerosos satélites, del sistema empujado denominado *On-Board Data Handling* (OBDAH), encargado de gestionar en vuelo los datos de sus distintos subsistemas.

El entorno de desarrollo ha sido instalado sobre el sistema operativo Linux. La distribución concreta sobre la que se ha realizado la instalación ha sido la Ubuntu 18.04 (Bionic Beaver), si bien los pasos a dar pueden considerarse análogos en el caso de que la distribución de Linux elegida fuera otra. Una solución que permite evitar cualquier problemática asociada a la distribución es utilizar una máquina virtual (Con VMPlayer o VirtualBox) construida a partir de una imagen de Ubuntu 18.04 de 64 bits como la que se puede descargar de desde el siguiente enlace <https://releases.ubuntu.com/18.04>

Si se utiliza una instalación de Ubuntu de 64 bits la instalación de los paquetes que vienen por defecto deberá completarse de la siguiente forma **(Si estás en el puesto de la universidad el usuario es atcsol ya se han instalado estos paquetes)**:

```
sudo dpkg --add-architecture i386
sudo apt-get update
sudo apt-get install libc6:i386 libncurses5:i386 libstdc++6:i386
sudo apt-get install libnspr4-dev:i386 zlib1g:i386 libx11-6:i386
sudo apt-get install build-essential
```

Además, para cualquiera de las instalaciones (32 bits o 64 bits) de Ubuntu, es necesario tener instalado el run-time Java que permite ejecutar aplicaciones Java como el entorno de desarrollo Eclipse que se va a usar en las prácticas. Para instalarlo se utiliza el comando apt-get de la siguiente forma **(Si estás en el puesto de la universidad el usuario es atcsol ya se ha instalado este paquete)**:

```
sudo apt-get update
sudo apt-get install openjdk-8-jre
```

En los siguientes apartados se irán enumerado los pasos requeridos para completar la instalación de todos los elementos de la toolchain, así como algunas pruebas que permiten comprobar que ésta se ha completado correctamente.

2. Instalación del entorno Eclipse

Los proyectos C/C++ que se van a crear sobre el procesador LEON3 pueden gestionarse mejor desde un entorno integrado de desarrollo como es Eclipse. Este entorno se puede instalar fácilmente dentro de la distribución Ubuntu 18.04 de Linux siguiendo los siguientes pasos.

1. Descarga del archivo tar.gz que contiene los archivos del entorno Eclipse configurado para trabajar con proyectos C/C++. Este archivo, para la versión *kepler* de Eclipse, se puede descargar directamente del enlace:

32 bits:

<http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/kepler/SR2/eclipse-cpp-kepler-SR2-linux-gtk.tar.gz>

64 bits:

http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/kepler/SR2/eclipse-cpp-kepler-SR2-linux-gtk-x86_64.tar.gz

2. Mover el archivo al directorio /opt

```
sudo mv eclipse-cpp-kepler-SR2-linux-gtk-x86_64.tar.gz /opt
```

3. Descomprimir el archivo en ese mismo directorio

```
cd /opt/  
sudo tar xzvf eclipse-cpp-kepler-SR2-linux-gtk.tar.gz
```

4. Añadir (utilizando, por ejemplo, gedit o vim) **al final del archivo /home/user/.profile (si estás en el puesto de la universidad el usuario es atcsol, por lo que el archivo a editar será /home/atcsol/.profile,)** la siguiente línea de modificación del PATH que incorpora el directorio donde se almacena el ejecutable que lanza el entorno eclipse **(NO COPIAR DEL PDF ya que se pueden añadir caracteres de control no deseados. Introduce el texto manualmente)**

```
PATH="/opt/eclipse:$PATH"
```

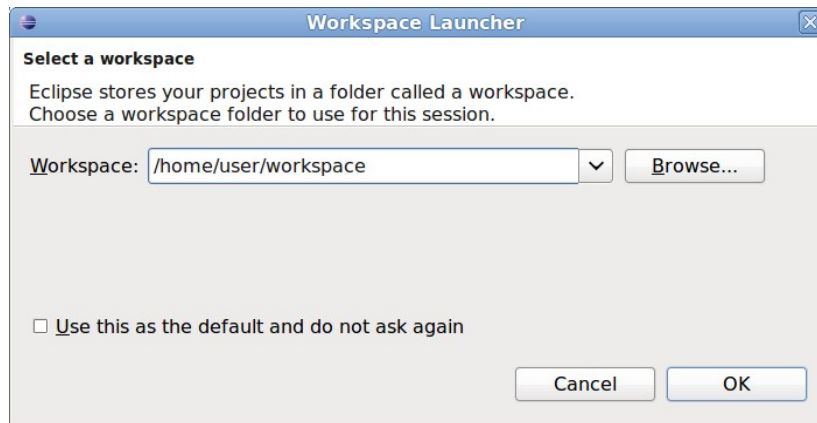
5. Forzar la ejecución del script ./profile con la orden:

```
source .profile
```

6. ejecutar eclipse y comprobar que se abre la aplicación:

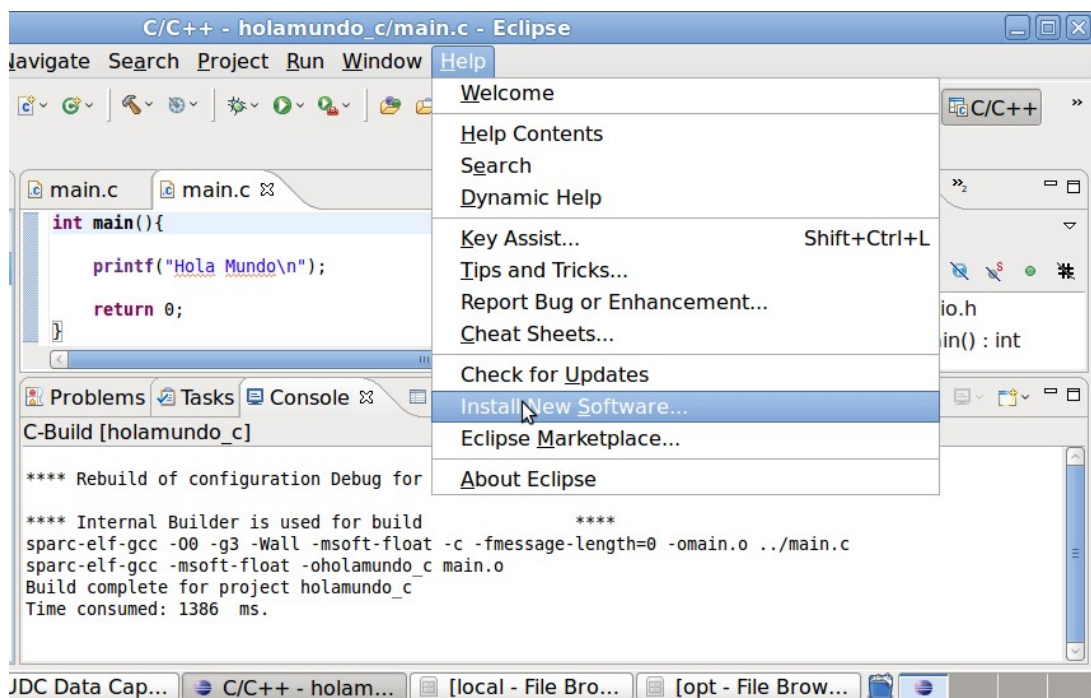
```
eclipse
```

7. Aceptar el nombre del workspace que propone la siguiente ventana inicial de eclipse:

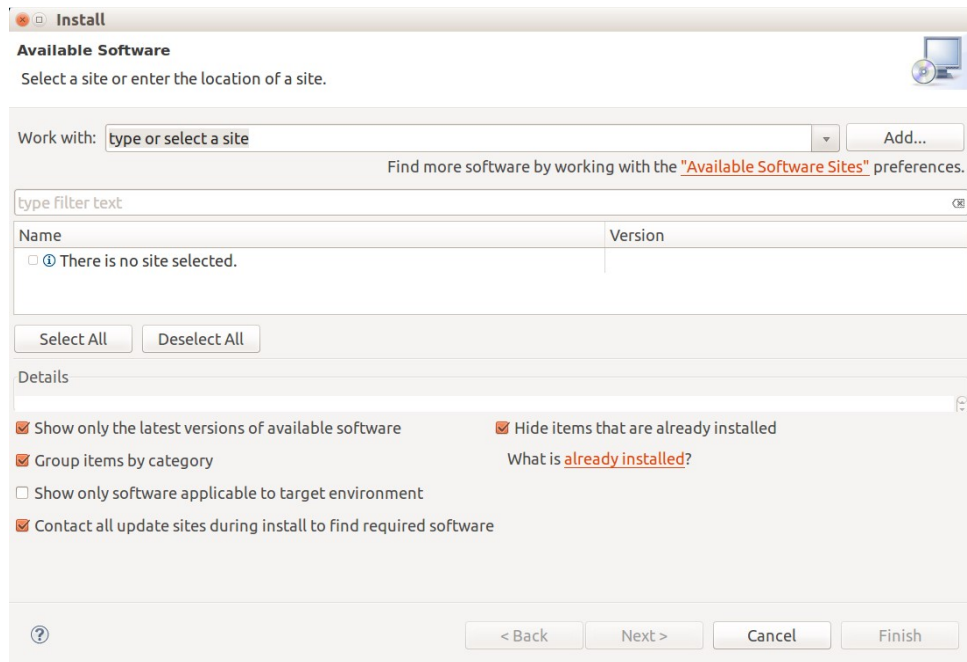


3. Plugin para el control de configuración con SVN desde Eclipse

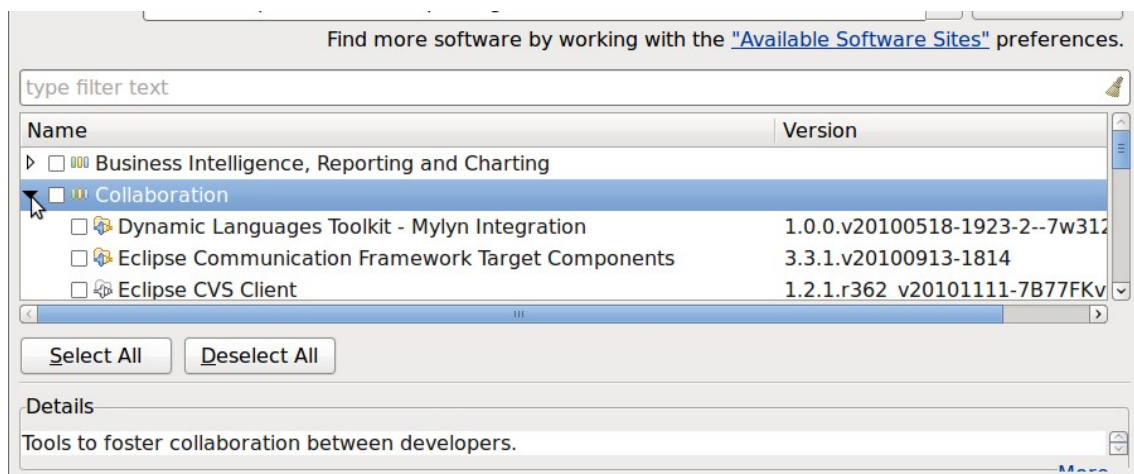
El control de configuración de proyectos permite a los ingenieros tener en todo momento la trazabilidad de los cambios que se han producido en el código fuente de sus proyectos, así como asociar etiquetas a la situación del proyecto en fases concretas del desarrollo. Para nuestra asignatura vamos a emplear como herramienta de control de configuración un cliente de Subversion integrado como plugin en el entorno Eclipse. Para su instalación, es necesario utilizar el menú *Install New Software* de Eclipse.



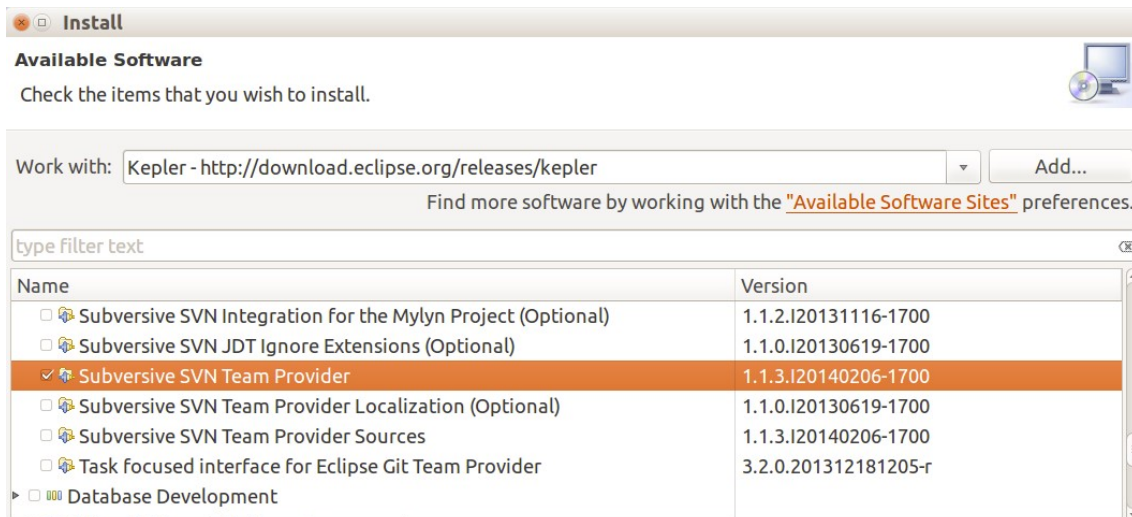
Seguidamente, seleccionar de la lista desplegable *Working With* el software site: Kepler <http://download.eclipse.org/releases/kepler>.



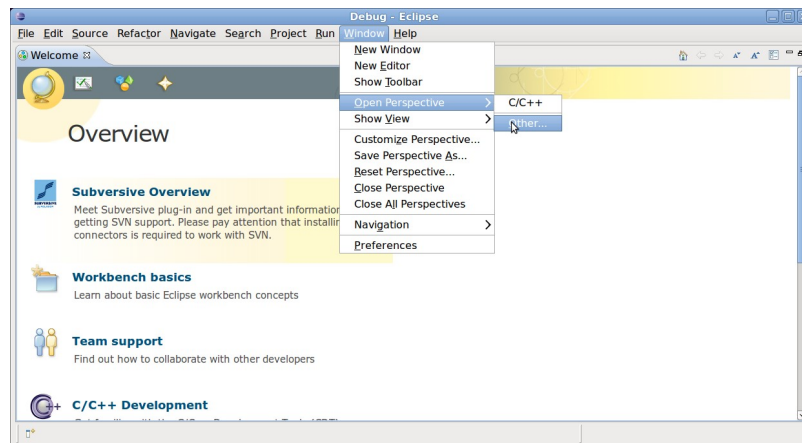
Desplegar la lista de Items que corresponde al grupo *Collaboration*



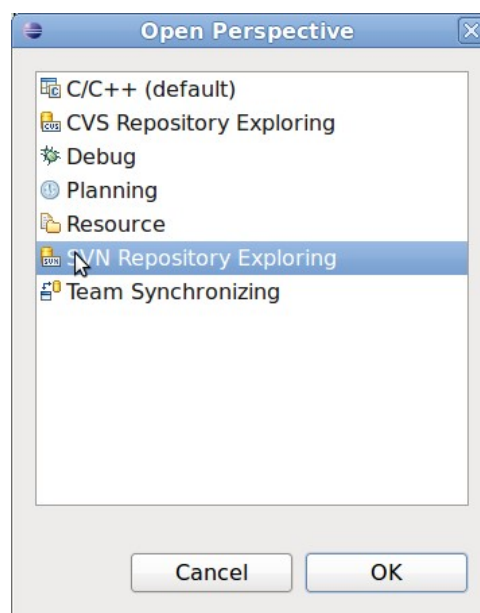
Seleccionar el último ítem de la lista *Subversive SVN Team Provider (Incubation)* y seleccionar *Next* para iniciar la instalación. Tras aceptar todos los pasos de la instalación, se solicitará el reinicio de Eclipse, que deberá aceptarse.



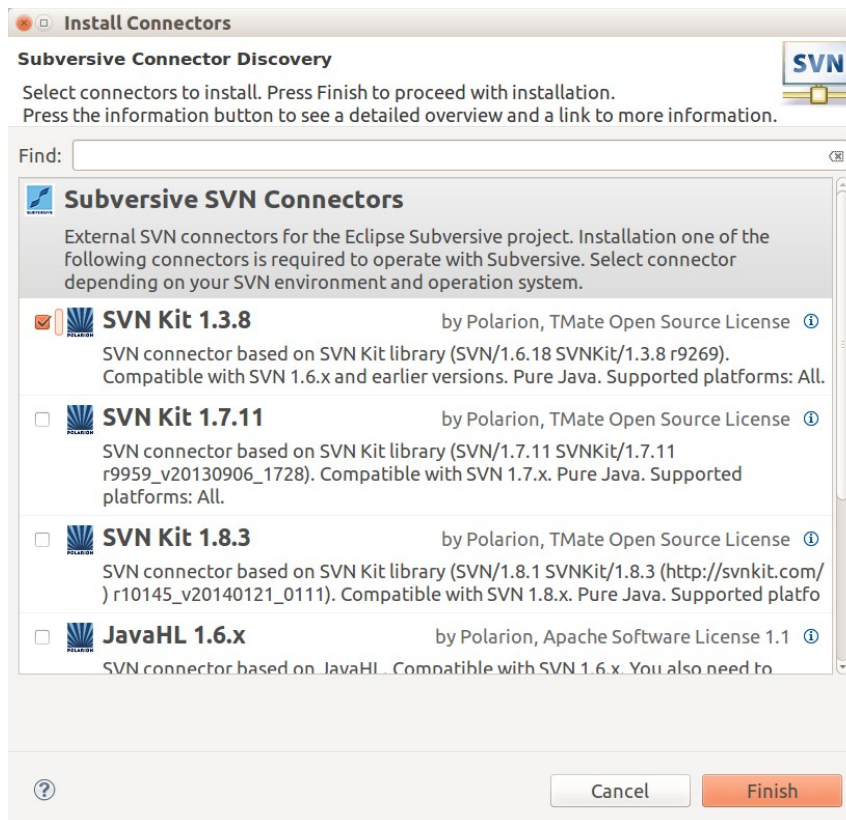
Tras el reinicio, se utilizará el menú *Open Perspective/Other* para comenzar con la utilización del cliente de *Subversion (SVN)*.



Seleccionar la perspectiva *SVN Repository Exploring*.



Seleccionar el conector *SVN Kit 1.3.8* para finalizar la instalación del cliente.



(si no aparece este menú es posible instalar el SVN Kit 1.3.8 utilizando Install New Software, y seleccionar manualmente el *site* <http://community.polarion.com/projects/subversive/download/eclipse/3.0/kepler-site/>)

Cerrar eclipse para proseguir con la instalación del resto de herramientas.

4. Instalación del compilador cruzado BCC

El compilador cruzado BCC (Bare-C Cross Compiler) es un compilador cruzado para los procesadores LEON2 y LEON3. La distribución del compilador incluye además la nueva biblioteca de C *Newlib standalone* y rutinas de bajo nivel para entrada y salida sobre LEON2 y LEON3. Finalmente, la distribución también proporciona la utilidad Mkprom que permite crear, a partir de un sistema software ejecutable en RAM, un nuevo software ejecutable desde una memoria no volátil (PROM o EEPROM) y cuyo objetivo es únicamente desplegar en RAM el sistema original y darle paso.

La instalación del compilador BCC comprende los siguientes pasos (Abrir otro terminal para poder ejecutar los comandos):

1. Descargarse la distribución de BCC `sparc-elf-4.4.2-1.0.50.tar.bz2` utilizando el siguiente enlace:
<http://www.gaisler.com/anonftp/bcc/bin/linux/sparc-elf-4.4.2-1.0.50.tar.bz2>

2. Descomprimirla en el directorio /opt

```
sudo tar xvfj sparc-elf-4.4.2-1.0.50.tar.bz2 -C /opt
```

3. Crear un enlace blando con el nombre `sparc-elf`

```
cd /opt
sudo ln -s sparc-elf-4.4.2 sparc-elf
```

4. Añadir **al final del archivo** `/home/user/.profile` (o `/home/atcsol/.profile`, **si estás en el puesto de la universidad**) la siguiente línea de modificación del PATH que incorpora el directorio donde se almacenan los archivos binarios del compilador BCC

```
PATH="/opt/sparc-elf/bin:$PATH"
```

5. Instalación del simulador TSIM2

El simulador TSIM2 permite simular el procesador LEON3. Su instalación comprende los siguientes pasos.

1. Debido a un cambio en la versión de evaluación del simulador del procesador LEON3 (ha cambiado a doble núcleo), en el laboratorio vamos a utilizar una versión antigua de dicho simulador, la TSIM2, que es una versión de evaluación de 2020, por lo que para que funcione este curso tendremos que proporcionarle a la aplicación una fecha de 2020. Para ello necesitaremos instalar `faketime` mediante el siguiente comando:

```
sudo apt-get install faketime
```

2. Después hay que descargar el simulador de LEON3 `tsim-eval-2.0.65.tar.gz` y moverlo al directorio /opt

<http://www.gaisler.com/anonftp/tsim/tsim-eval-2.0.65.tar.gz>

```
sudo mv tsim-eval-2.0.65.tar.gz /opt
```

3. Descomprimirla en el directorio /opt

```
cd /opt
sudo tar xvfz tsim-eval-2.0.65.tar.gz
```

4. Descargar de la web de la asignatura el lanzador del TSIM2 que utiliza el `faketime` [tsim-launch.tar.gz](http://www.gaisler.com/anonftp/tsim/tsim-launch.tar.gz), moverlo al directorio /opt y descomprimirlo allí

```
sudo mv tsim-launch.tar.gz /opt
cd /opt
sudo tar xvfz tsim-launch.tar.gz
```

5. Crear un enlace blando con el nombre `tsim` al lanzador

Para Ubuntu de 64 bits (**como el del puesto de la universidad**)

```
sudo ln -s tsim-launch/tsim/linux-x64 tsim
```

6. Añadir al final del archivo `/home/user/.profile` (o `/home/atcsol/.profile`, si estás en el puesto de la universidad) la siguiente línea de modificación del PATH que incorpora el directorio donde se almacenan el archivo del lanzador del simulador TSIM2.

```
PATH="/opt/tsim:$PATH"
```

7. Forzar la ejecución del script `./profile` con la orden:

```
source .profile
```

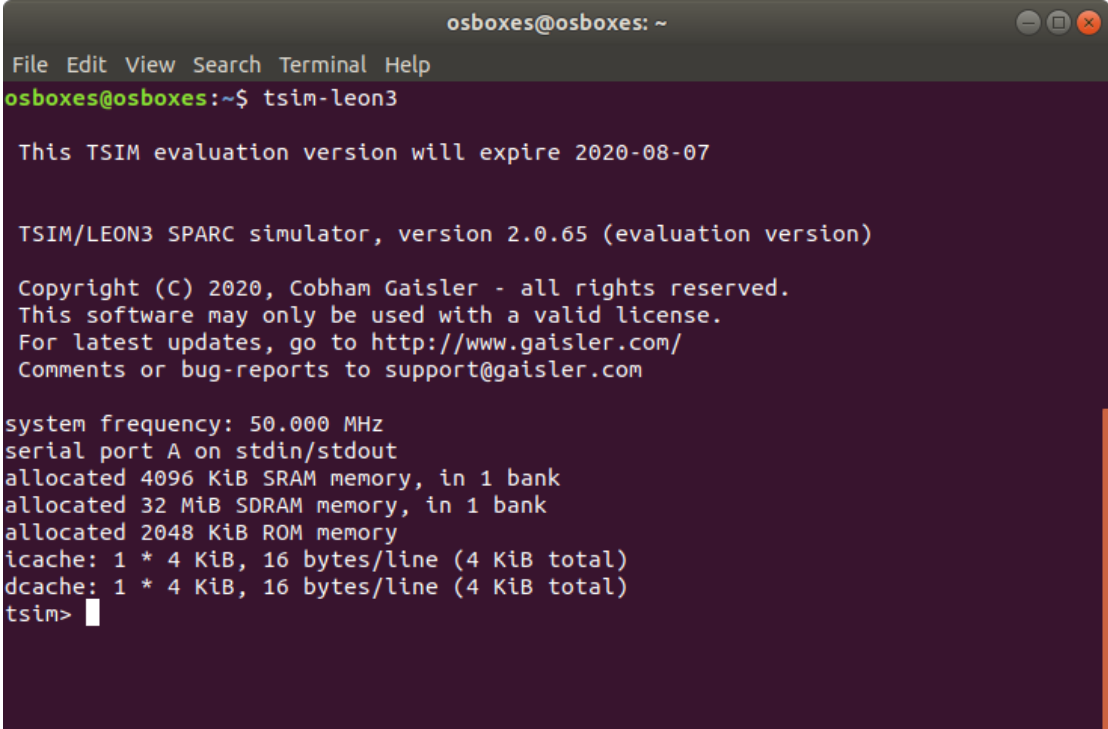
8. Mostrar el PATH actual y comprobar que aparecen los directorio `/opt/tsim` , `/opt/sparc-elf/bin` y `/opt/eclipse`.

```
echo $PATH
```

9. Una vez completada la instalación, vamos a lanzar el simulador TSIM2 para comprobar que funciona correctamente. Ejecutamos desde línea de ordenes:

```
tsim-leon3
```

La pantalla que debe mostrarse es la siguiente:



```
osboxes@osboxes: ~
File Edit View Search Terminal Help
osboxes@osboxes:~$ tsim-leon3

This TSIM evaluation version will expire 2020-08-07

TSIM/LEON3 SPARC simulator, version 2.0.65 (evaluation version)

Copyright (C) 2020, Cobham Gaisler - all rights reserved.
This software may only be used with a valid license.
For latest updates, go to http://www.gaisler.com/
Comments or bug-reports to support@gaisler.com

system frequency: 50.000 MHz
serial port A on stdin/stdout
allocated 4096 KiB SRAM memory, in 1 bank
allocated 32 MiB SDRAM memory, in 1 bank
allocated 2048 KiB ROM memory
icache: 1 * 4 KiB, 16 bytes/line (4 KiB total)
dcache: 1 * 4 KiB, 16 bytes/line (4 KiB total)
tsim> 
```

10. Cargar y lanzar el test `paranoia` dentro del simulador `tsim-leon3`

```
tsim > load /opt/tsim-eval/tests/paranoia
tsim > run
```


comprobar que el aparece el mensaje “Program exited normally” lo que indica que el simulador funciona correctamente

11. A continuación, vamos a utilizar eclipse para cargar y depurar programas sobre TSIM2. Para hacer posible esto debemos emplear la orden `gdb` dentro de TSIM2. Esto abrirá el puerto 1234, habilitándolo para que gdb se conecte a él, y controle el simulador. De esa forma podremos desde eclipse cargar y simular la ejecución de programas sobre TSIM2. La pantalla que se mostrará es la siguiente:

```
tsim> gdb
gdb interface: using port 1234
Starting GDB server. Use Ctrl-C to stop waiting for connection.
```

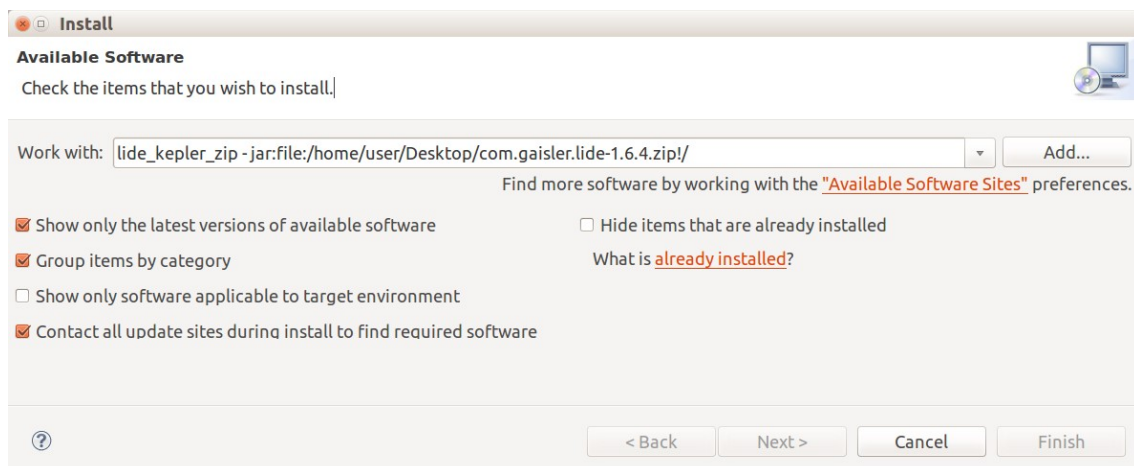
6. Instalación del plugin de Eclipse para la integración del compilador cruzado BCC y el simulador TSIM2

Abriremos un nuevo terminal, y desde él ejecutaremos eclipse tras haber cargado el PATH del profile

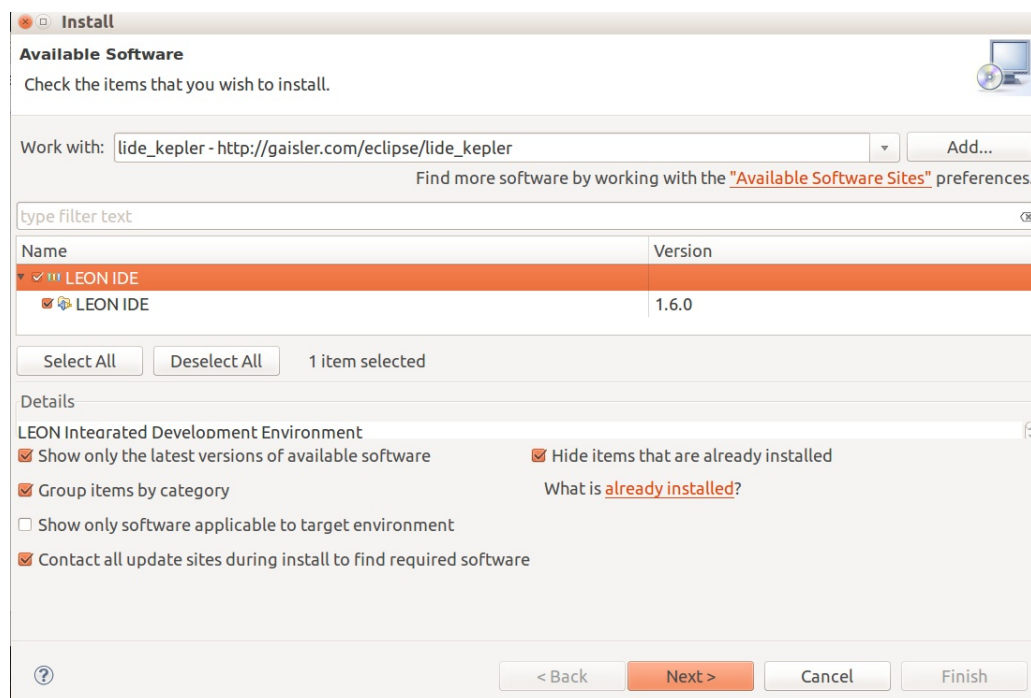
```
source .profile
eclipse
```

Para integrar el compilador y el simulador con el entorno de desarrollo Eclipse de forma sencilla utilizaremos el plugin de Gaisler denominado CDT Plugin 1.6.0 La instalación de este plugin comprende los siguientes pasos:

1. Seleccionar el menú de Eclipse **Help->Install New Software**
2. Añadir un nuevo repositorio con el boton de la parte derecha **Add...**, e incluir **lide_kepler**, utilizando el archivo .zip (opción **Archive**) disponible en la URL <https://www.gaisler.com/eclipse/com.gaisler.lide-1.6.4.zip>

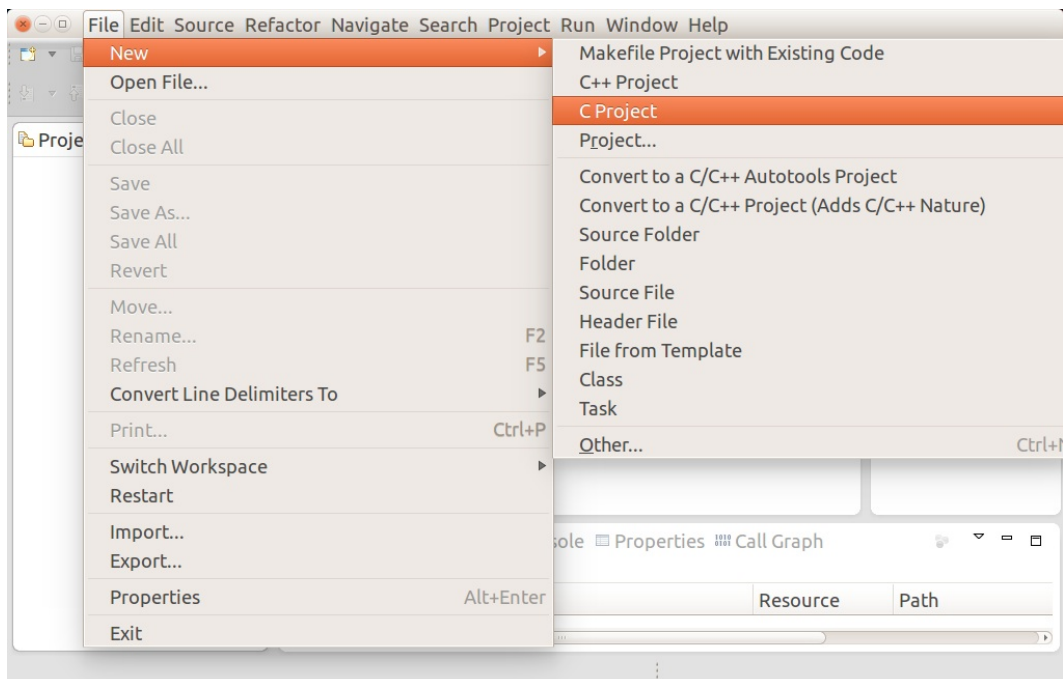


3. Seleccionar LEON IDE, como aparece en la figura siguiente, y aceptar el resto de menus, incluido el que finalmente fuerza el reinicio de eclipse.



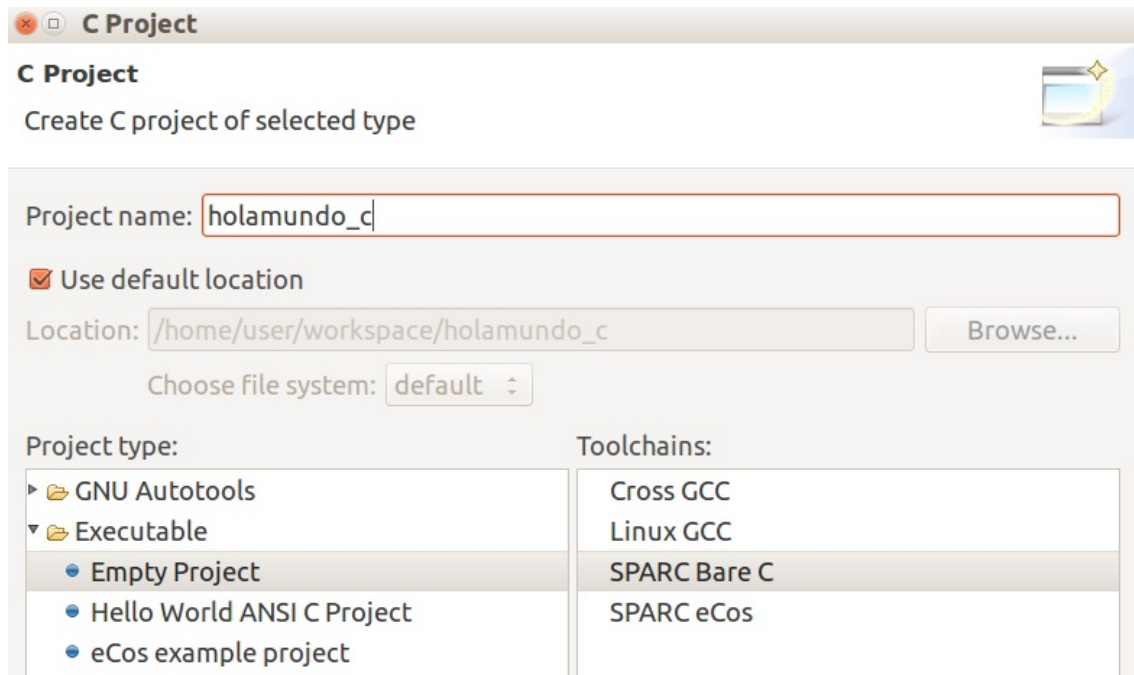
6. Creación un proyecto BCC

Para crear un proyecto C para el compilador cruzado BCC en el que el Makefile se genere automáticamente hay que seleccionar el menu **File->New-> C Project** tal como se muestra en la figura.

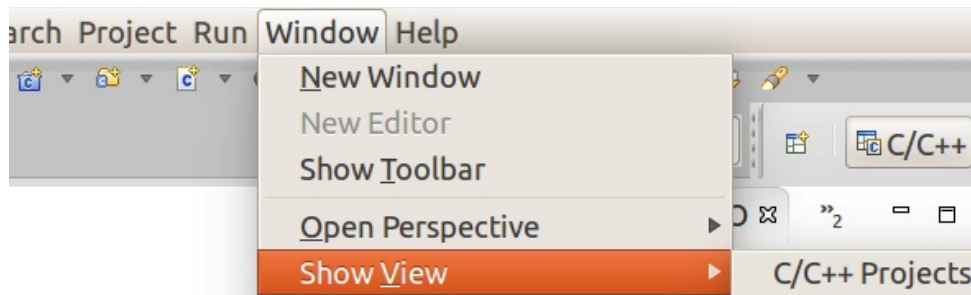


A continuación hay que asignarle un nombre al proyecto (holamundo_c en este caso) y definir el tipo de proyecto como **Executable Empty Project** y como **Toolchain Sparc**

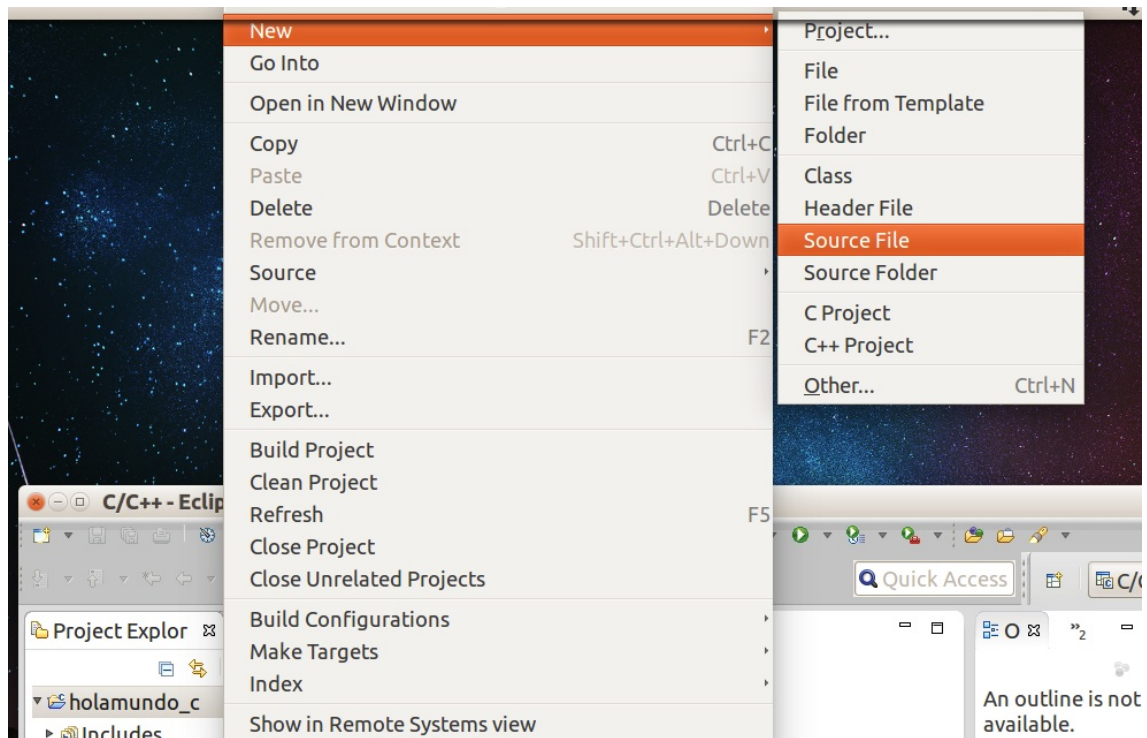
Bare C, para que se utilice el compilador cruzado de la arquitectura Sparc, a la que pertenecen los procesadores LEON.



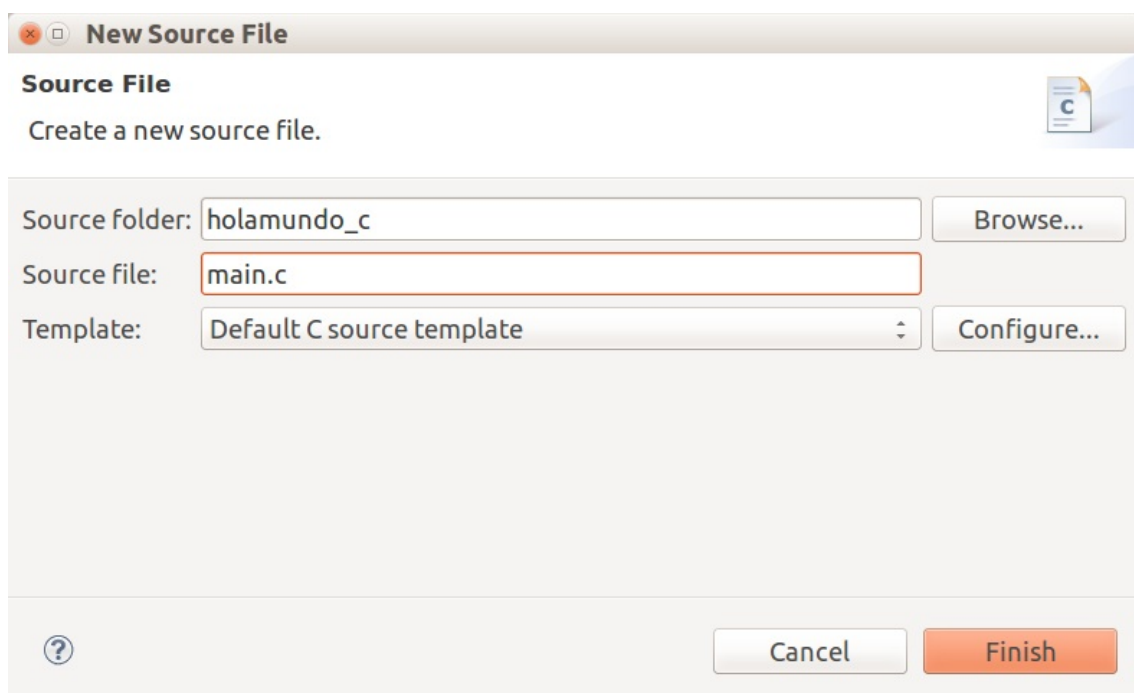
Una vez creado el proyecto, seleccionar la vista C/C++ utilizando el menu **Window->ShowView> C/C++ Projects**



y Añadimos un archivo fuente utilizando el botón derecho sobre el proyecto holamundo_c



, y le damos el nombre **main.c**



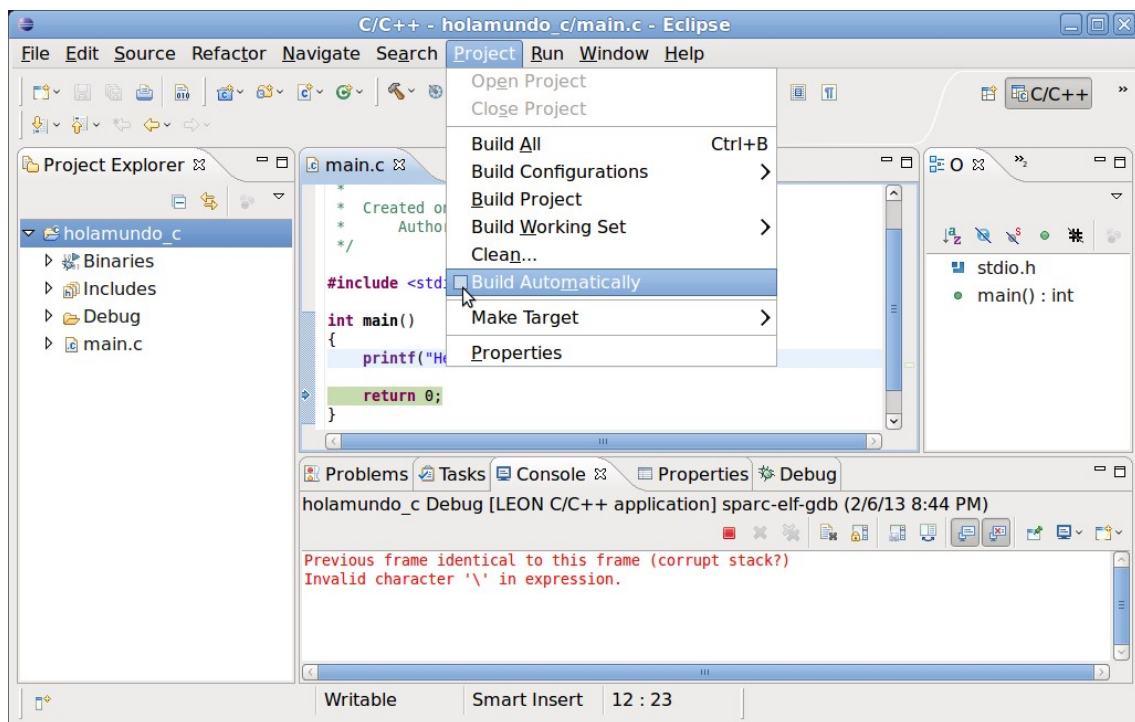
Haciendo doble clic en este archivo añadimos un código básico de hola mundo

```
#include <stdio.h>

int main()
{
    printf("Hello\n");

    return 0;
}
```

El ejecutable de este proyecto se crea automáticamente según la configuración inicial de Eclipse. Esta configuración se puede modificar desde el menú **Project** desmarcando la opción **Build Automatically**.



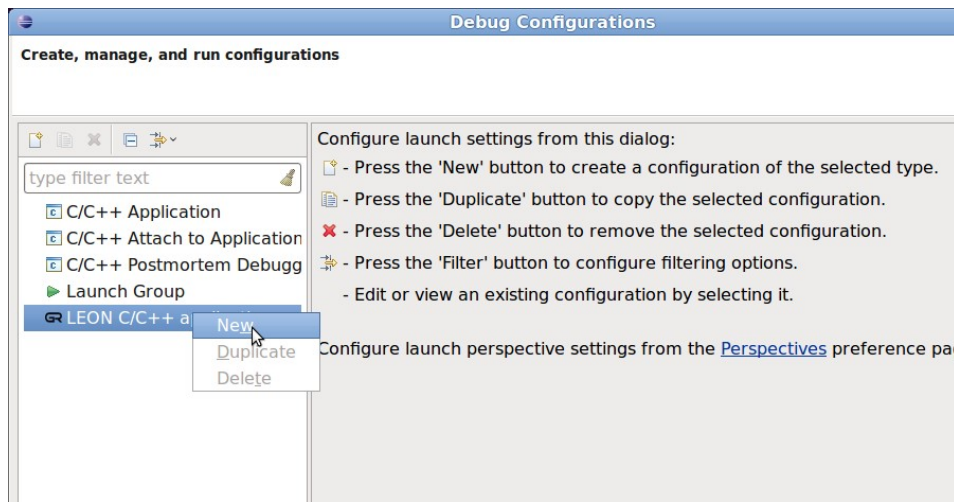
En el caso de que la configuración automática esté deshabilitada, se puede construir el ejecutable mediante el menú **Project->Build Project**

7. Ejecutar en Depuración el proyecto sobre el simulador TSIM2

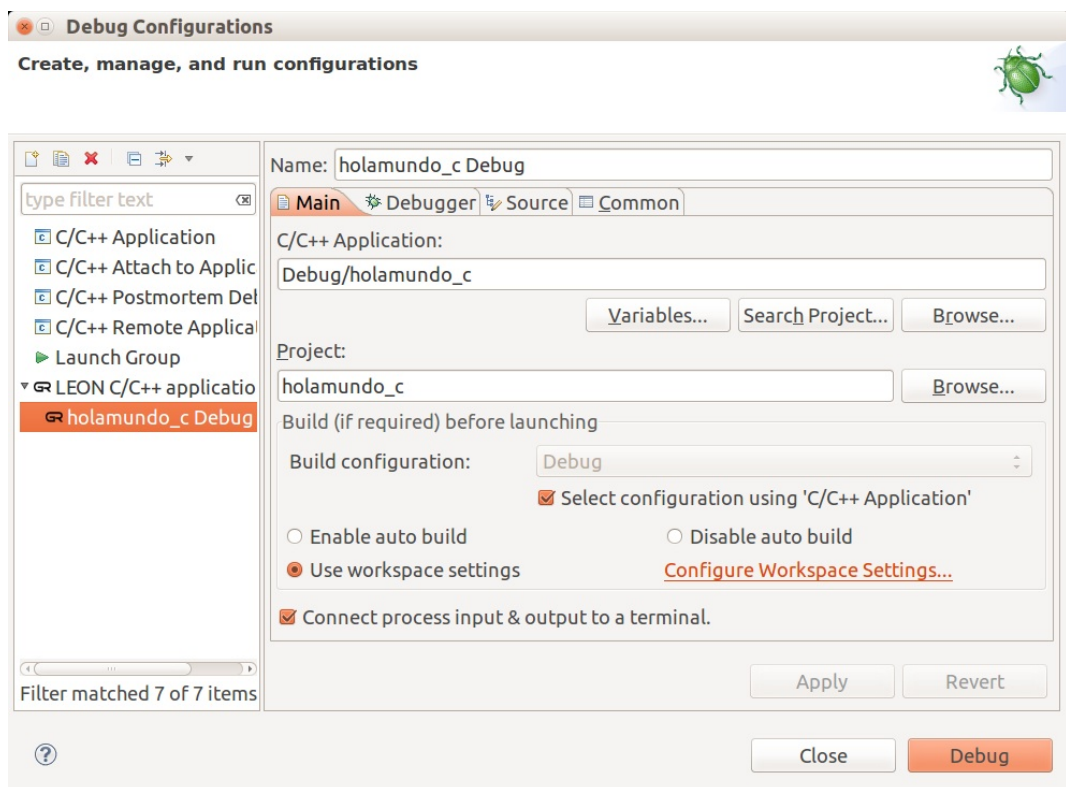
Para ejecutar el programa, es necesario crear un lanzador que se configure para trabajar con el simulador TSIM2. Para ello habrá que dar los siguientes pasos:

1. Acceder al menú de configuración de depuración mediante el menú **Run->Debug Configurations...**

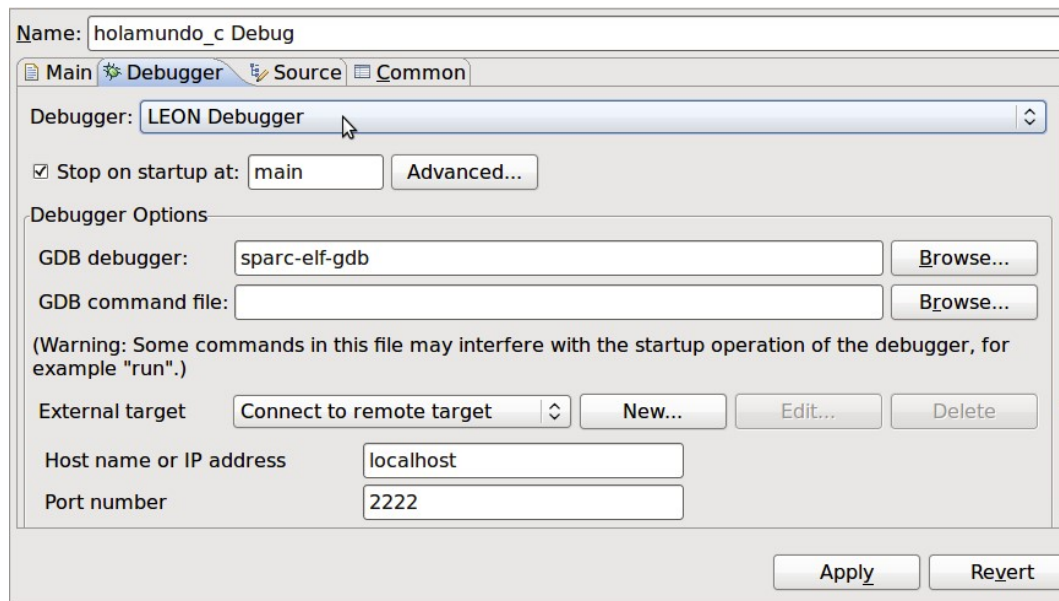
2. Crear un nuevo lanzador para una aplicación LEON C/C++



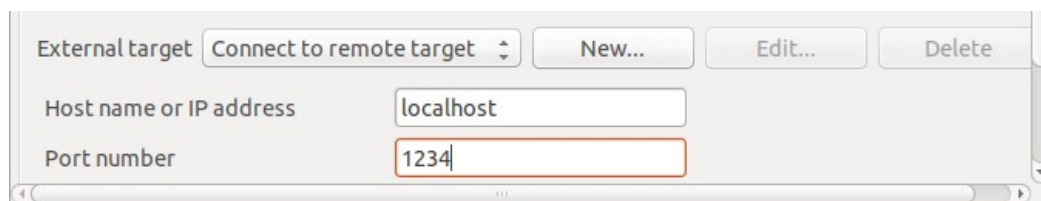
3. Comprobar que la configuración **Main** se corresponde con la siguiente:



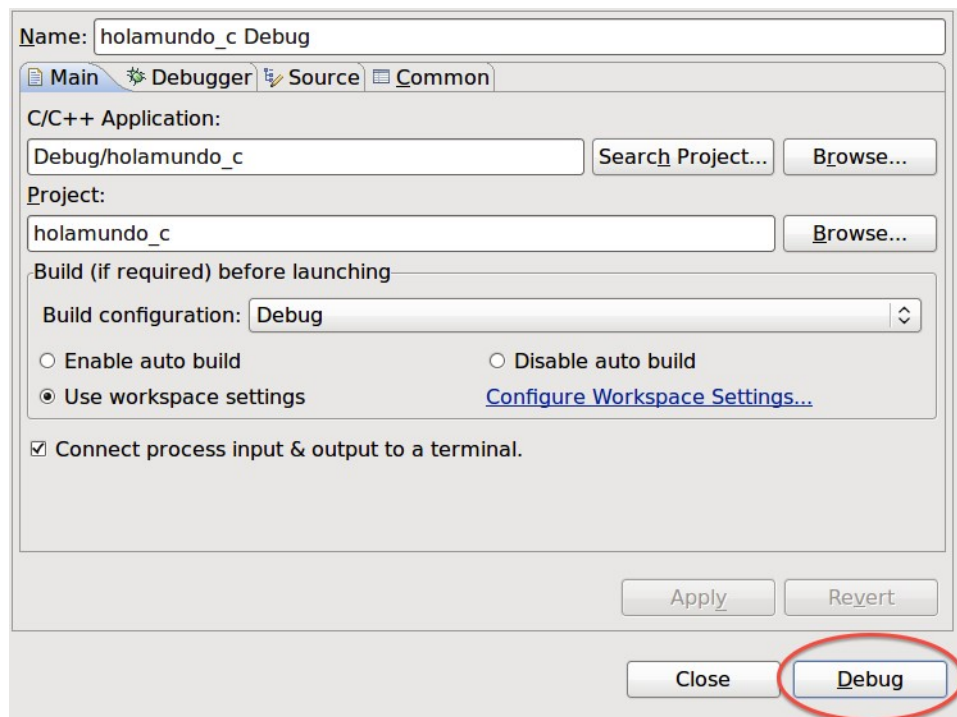
4. Seleccionar **LEON Debugger** como **Debugger** y **sparc-elf-gdb** en la opción de depuración **GDB debugger**:



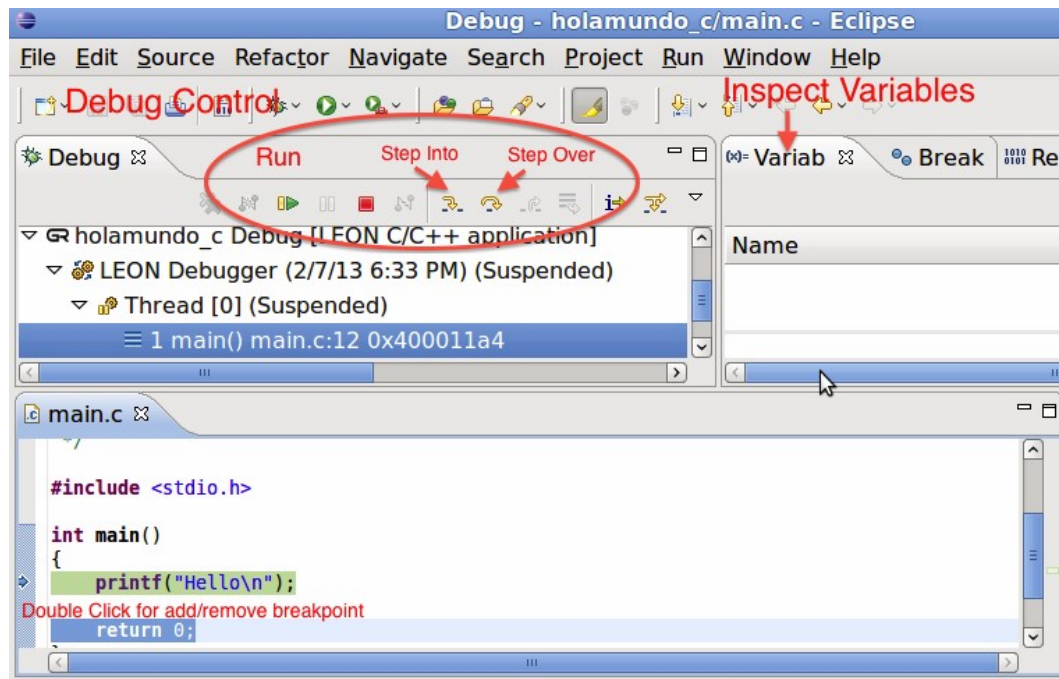
5. Cambiar el *Port number* a 1234 para que gdb se conecte a TSIM2



6. Por último lanzar la depuración utilizando el botón **Debug**



- Desde la vista **Debug** se puede controlar el lanzamiento de la ejecución continua (**Run**), la ejecución paso a paso (**Step Into** y **Step Over**), la edición de breakpoints (**Double Click for add/remove breakpoint**) y la inspección de variables (**Inspect Variables**).



- Ejecutar el programa seleccionando el control **Run** y comprobar en la consola de TSIM2 que aparece el mensaje "Hello".

```
system frequency: 50.000 MHz
serial port A on stdin/stdout
allocated 4096 KiB SRAM memory, in 1 bank
allocated 32 MiB SDRAM memory, in 1 bank
allocated 2048 KiB ROM memory
icache: 1 * 4 KiB, 16 bytes/line (4 KiB total)
dcache: 1 * 4 KiB, 16 bytes/line (4 KiB total)
tsim> gdb
gdb interface: using port 1234
Starting GDB server. Use Ctrl-C to stop waiting for connection.
connected
Hello
gdb: disconnected
gdb interface: using port 1234
Starting GDB server. Use Ctrl-C to stop waiting for connection.
```

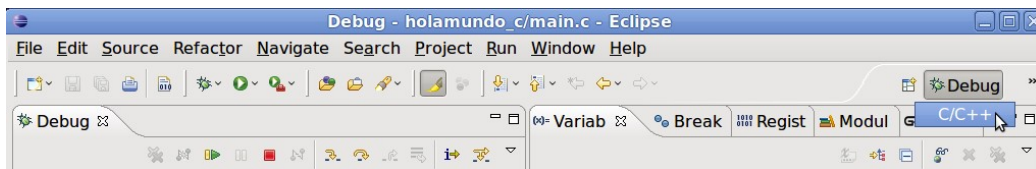
- Dado que el programa ha terminado, vamos a resetear el procesador desde TSIM2 para poder cargar un nuevo programa en la siguiente depuración. Para ello hay que primero utilizar **Ctrl+C** para volver a línea de comandos de TSIM2, y utilizar luego el comando **reset** que resetea el procesador. Finalmente, de nuevo usamos **gdb** para que TSIM2 esté listo para cargar un nuevo programa desde eclipse.

Tras estos comandos, la consola de TSIM2 debe de tener el siguiente status

```
Starting GDB server. Use Ctrl-C to stop waiting for connection.  
^C  
tsim> reset  
tsim> gdb  
gdb interface: using port 1234  
Starting GDB server. Use Ctrl-C to stop waiting for connection.
```

Antes de cada depuración, deberemos asegurarnos de que TSIM2 se encuentra en este estado (esperando a gdb tras un reset) para evitar que la carga o la ejecución fallen.

10. Volver a la Perspectiva C/C++

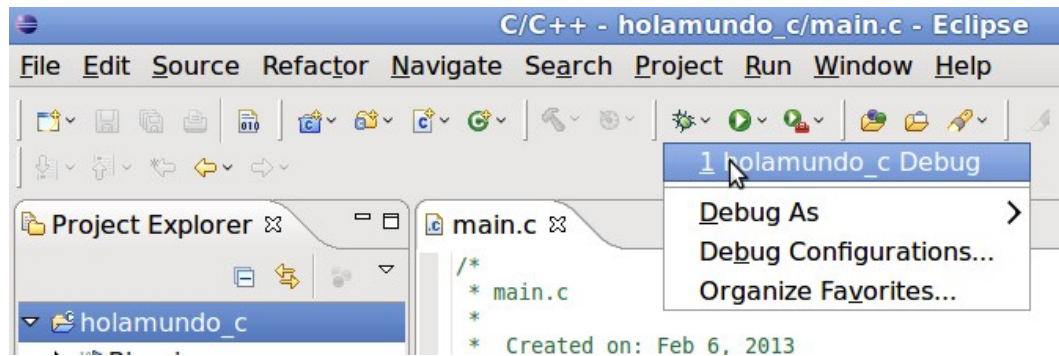


11. Editar el archivo fuente **main.c** y crear un bucle que muestre por pantalla 10 veces el mensaje *Hello*.

```
#include <stdio.h>  
  
int main()  
{  
    int i;  
    for(i=0; i< 10; i++)  
        printf("Hello\n");  
  
    return 0;  
}
```

12. Salvar el archivo e invocar **Build->Project** para construir el ejecutable.

13. Lanzar la depuración del nuevo ejecutable utilizando el siguiente menú desplegable.



14. Añadir un **breakpoint** sobre la línea printf y lanzar la ejecución (**Run**) comprobando como en el panel de inspección de variables varía el valor de i en cada iteración del bucle.

