

2º curso / 2º cuatr.

Grado en
Ing. Informática

Arquitectura de Computadores

Seminario 0. Entorno de programación: Plataforma Eclipse

Material elaborado por los profesores responsables de la asignatura:

Mancia Anguita – Julio Ortega

Licencia Creative Commons



ugr

Universidad
de Granada

ETSIIT

Escuela Técnica Superior
de Ingenierías Informática
y de Telecomunicación



ATC

Departamento de Arquitectura
y Tecnología de Computadores
UNIVERSIDAD DE GRANADA



Contenidos

- ¿Qué es Eclipse? ¿Qué es CDT?
- Usuarios de Eclipse
- Instalación
- Conceptos de Eclipse
- Ejemplo Hello
- Ejemplo Hello OpenMP

Contenidos

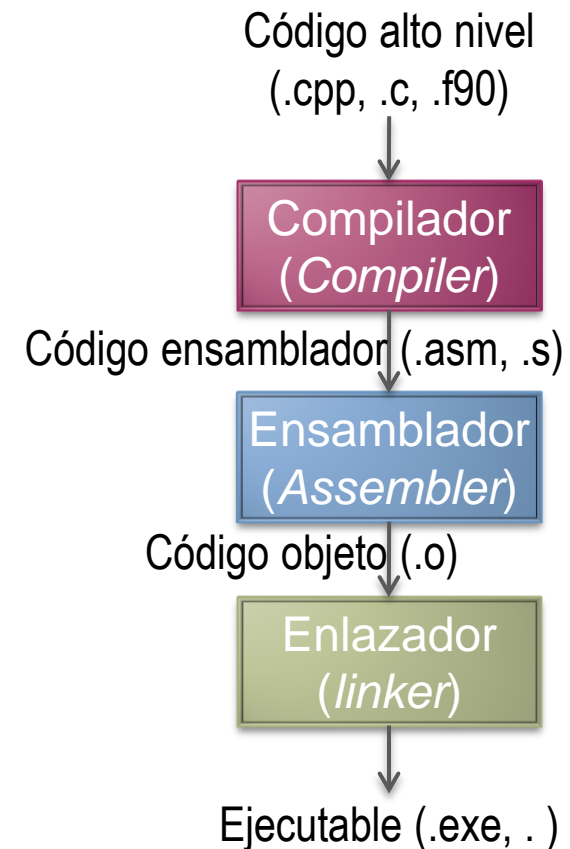
- ¿Qué es Eclipse? ¿Qué es CDT?
- Usuarios de Eclipse
- Instalación
- Conceptos de Eclipse
- Ejemplo Hello
- Ejemplo Hello OpenMP

¿Qué es Eclipse?

- Es un **entorno de desarrollo integrado** (IDE- *Integrated Development Environments*) más una serie de **complementos** (*plug-in*) que lo personalizan para distintos lenguajes de programación
 - ¿IDE? es una aplicación para el desarrollo de software. Incluye típicamente:
 - Editor de código fuente
 - Herramientas para la generación de código ejecutable: compilador, enlazador, ensamblador
 - Depurador
 - Con complementos (*plug-in*) se personaliza para varios lenguajes de programación (C/C++, Java, Python, etc.)
- Es un **software libre** de **código abierto** que se puede usar en Linux y Windows

¿Qué es CDT?

- Eclipse CDT (*C/C++ Development Tooling*) es el nombre que recibe el IDE Eclipse de C/C++ (IDE con el *plug-in* de C/C++)
- Permite editar/generar/depurar código C/C++
 - Proporciona un editor de código fuente
 - Usa generadores de código ejecutable de terceros:
 - Compilador, enlazador, ensamblador, etc. (de GNU, de Intel, MinGW, etc.)
 - Usa depuradores de código fuente de terceros (p. ej. gdb de GNU)



Contenidos

- ¿Qué es Eclipse? ¿Qué es CDT?
- Usuarios de Eclipse
- Instalación
- Conceptos de Eclipse
- Ejemplo Hello
- Ejemplo Hello OpenMP

Usuarios Eclipse

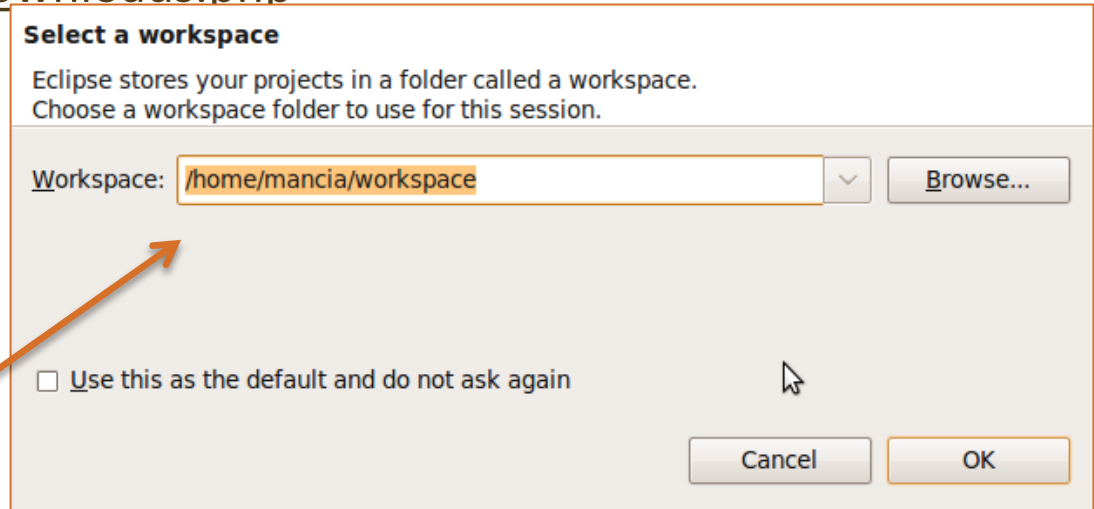
- Según un estudio de IDC (*International Data Corporation*) hay 2.27 millones de usuarios de Eclipse
- Ejemplo de vendedores que han adoptado Eclipse como IDE para, al menos, alguna herramienta de programación:
 - Intel (*C++, Fortran Compilers* para Linux)
 - Texas Instruments (*Code Composer Essentials*)
 - Altera (*NIOS II IDE*)
 - Xilinx (*Platform Studio SDK, Embedded Development Kit*)
 - ...

Contenidos

- ¿Qué es Eclipse? ¿Qué es CDT?
- Usuarios de Eclipse
- **Instalación**
- Conceptos de Eclipse
- Ejemplo Hello
- Ejemplo Hello OpenMP
- ¿Qué es TORQUE?
- Cluster de prácticas
- Ejecución del ejemplo Hello OpenMP en el cluster

Instalación

- Descargar la versión a instalar (**Linux64**, Linux32, Windows64 o Windows32) de <http://www.eclipse.org/cdt/downloads.php>
 - Indigo (2011)
 - Helios (2010)
 - Galileo (2009)
 - Ganymede (2008)
 - ...
- Instalar/descomprimir
- Ejecutar
 - Introducir el camino al espacio de trabajo (*workspace*)
 - *Workspace*: almacenamiento por defecto
 - Puede que al ejecutarlo la primera vez aparezca la **perspectiva** de Java en lugar de la de C/C++
 - En cuanto se cree un proyecto C o C++ (con *File->New->Project*) la IDE preguntará si pasa a la perspectiva C/C++
 - Se puede cambiar de perspectiva usando *Window->Open Perspective*
 - Para Indigo: en *Windows->Preferences->C/C++->Code Analysis* quitar algunos errores (en particular, "Symbol is not resolved")



Ventana de bienvenida



Se puede generar en cualquier momento con *Help->Welcome*

Contenidos

- ¿Qué es Eclipse? ¿Qué es CDT?
- Usuarios de Eclipse
- Instalación
- Conceptos de Eclipse
- Ejemplo Hello
- Ejemplo Hello OpenMP
- ¿Qué es TORQUE?
- Cluster de prácticas
- Ejecución del ejemplo Hello OpenMP en el cluster

Conceptos de Eclipse – *workbench, perspective*

- Mesa de trabajo (*workbench*)
 - Entorno de desarrollo integrado (**IDE**). Agrupa tareas en actividades de alto nivel (**perspectivas**).
 - Consta de una o varias perspectivas
 - Ej. mesas de trabajo: CDT (IDE C/C++), JDT (IDE Java), etc.
 - Abrir otra mesa de trabajo: *Window->New Window*
- Perspectiva (*perspective*)
 - Agrupa un conjunto de tareas para realizar una actividad de alto nivel. Visualmente es un contenedor con **editores** de contenidos y agrupaciones de **vistas**
 - Puede haber varias abiertas en un **workbench**, pero sólo una estará visible cada vez.
 - Ej. perspectivas: Desarrollo Java, Desarrollo C/C++, Depuración, etc.
 - Cambiar/visualizar perspectiva: *Window->Open Perspective*

Conceptos de Eclipse – *workbench, perspective*

Mesa de trabajo (*workbench*)

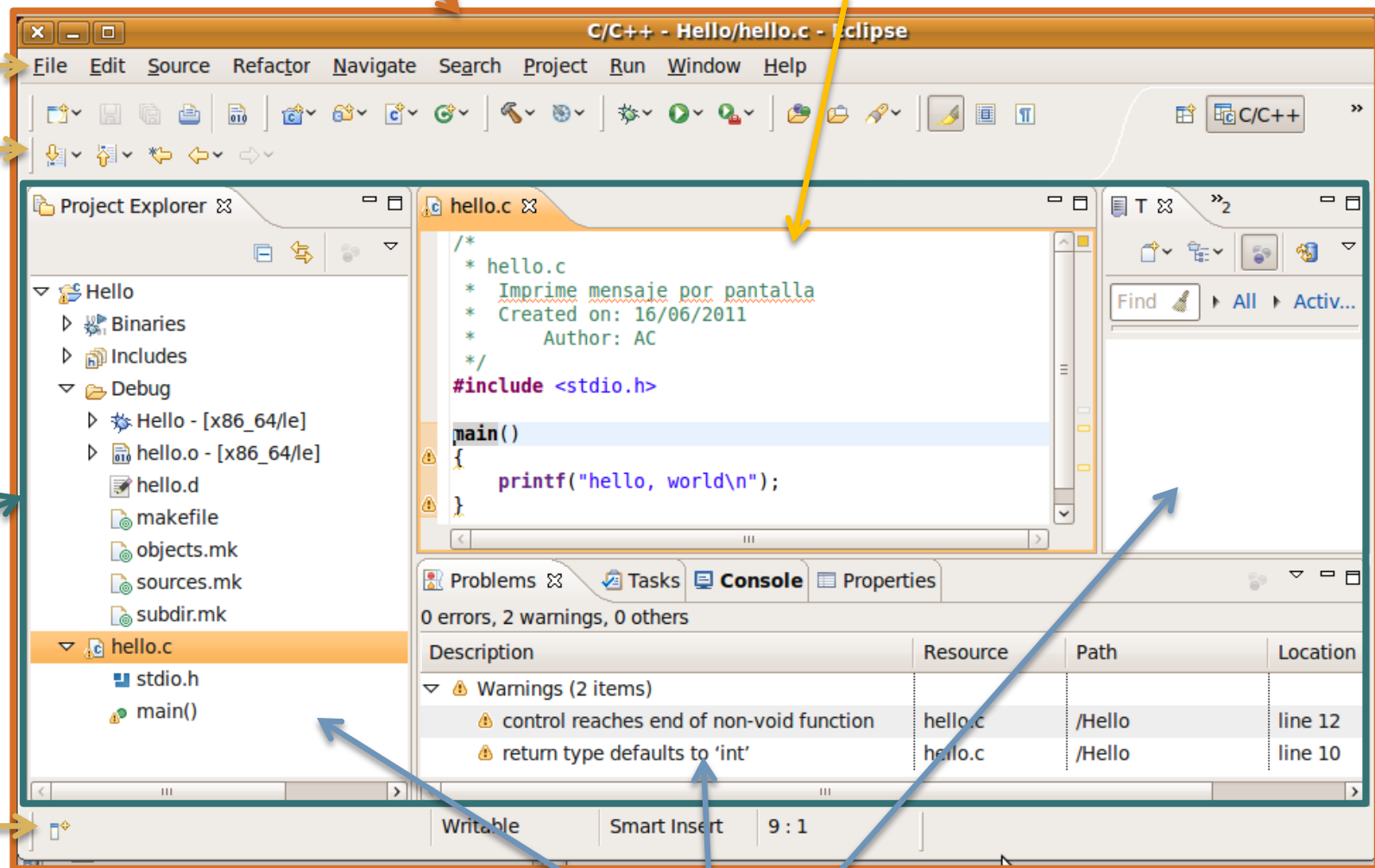
Editor

Barra de
menú

Barra de
herramientas

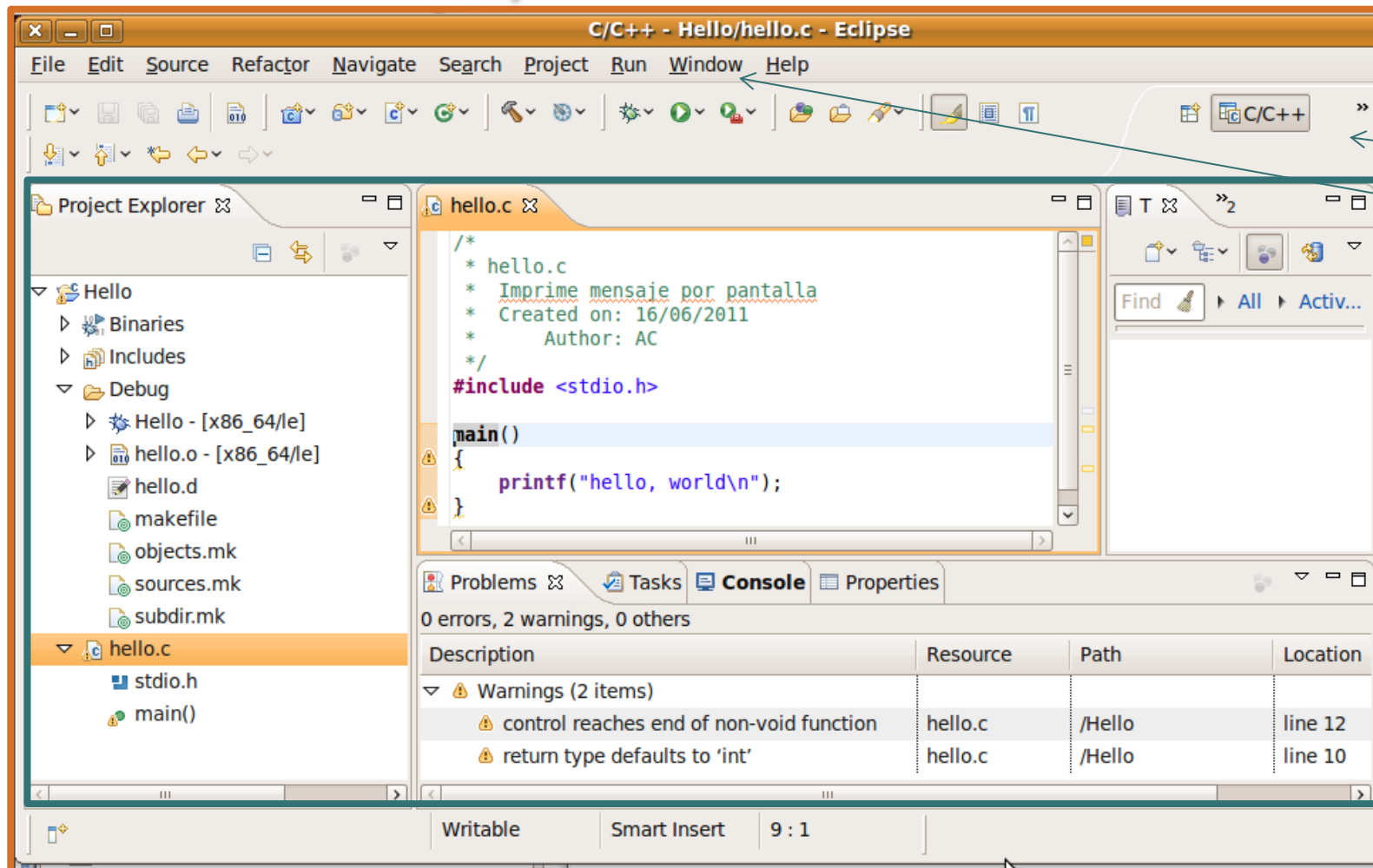
Perspectiva
(*perspective*)

Línea de
estado



Conceptos de Eclipse – *workbench, perspective*

Mesa de trabajo (*workbench*): barra de menú y barra de herramientas personalizada por la perspectiva activa, línea de estado personalizada por la vista activa, una o varias perspectivas (sólo una visible)



Para cambiar de perspectiva

Conceptos de Eclipse – *editor, view*

➤ Editor (*editor*)

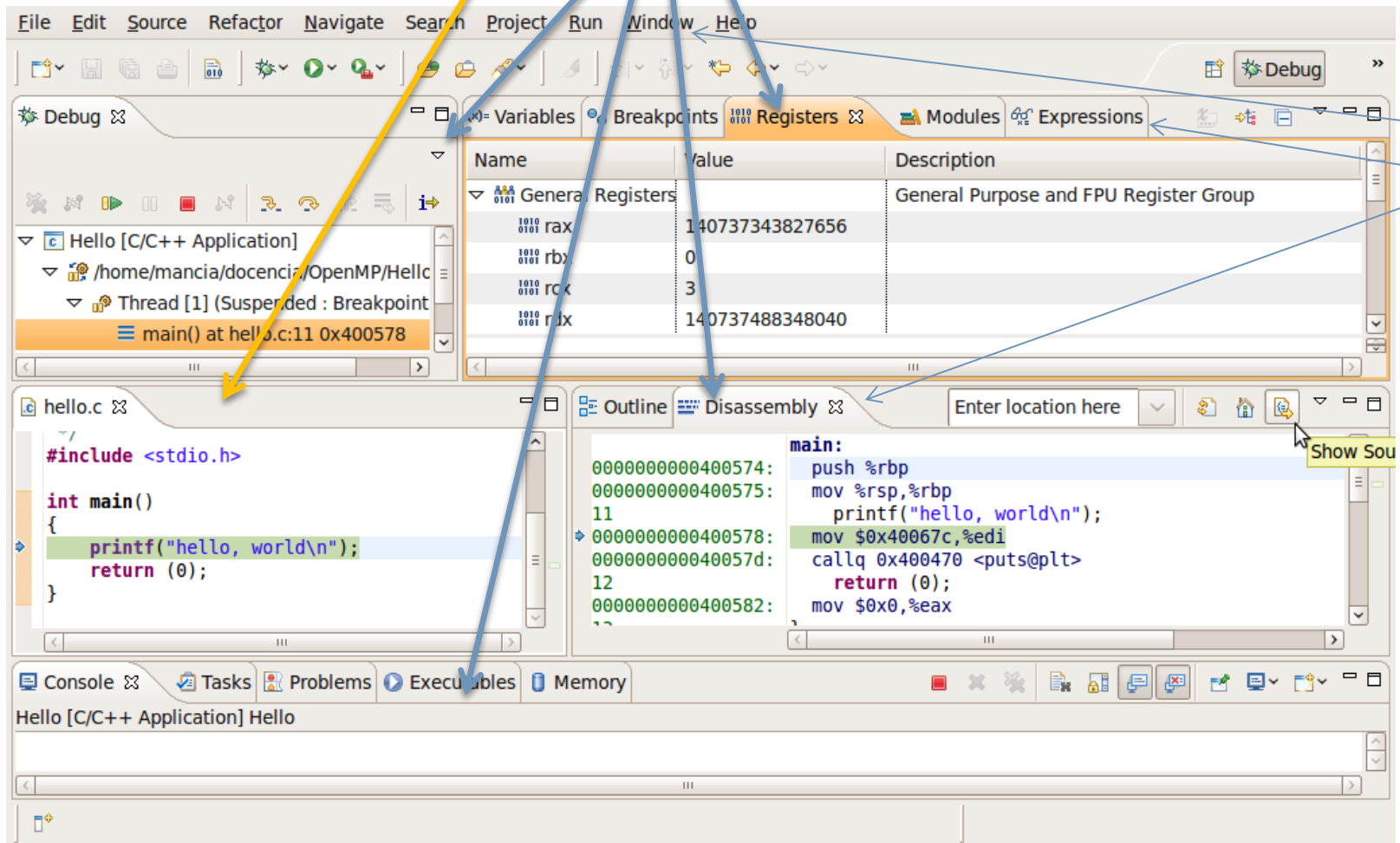
- Se comparte entre perspectivas de una mesa de trabajo (*workbench*).
- El editor que abre Eclipse depende de la extensión del fichero (.c, .cpp, .java, .txt, ...)
- Abrir nueva instancia editor: *Window->New Editor*

➤ Vista (*view*)

- Visualiza información para ayudar a realizar una tarea.
- No se comparte entre perspectivas
- Ej. vista: consola, problemas, registros, variables, explorador, ...
- Cambiar/visualizar vista: *Window->Show View*

Conceptos de Eclipse – *editor, view*

Perspectiva (perspective): editores y vistas (views)

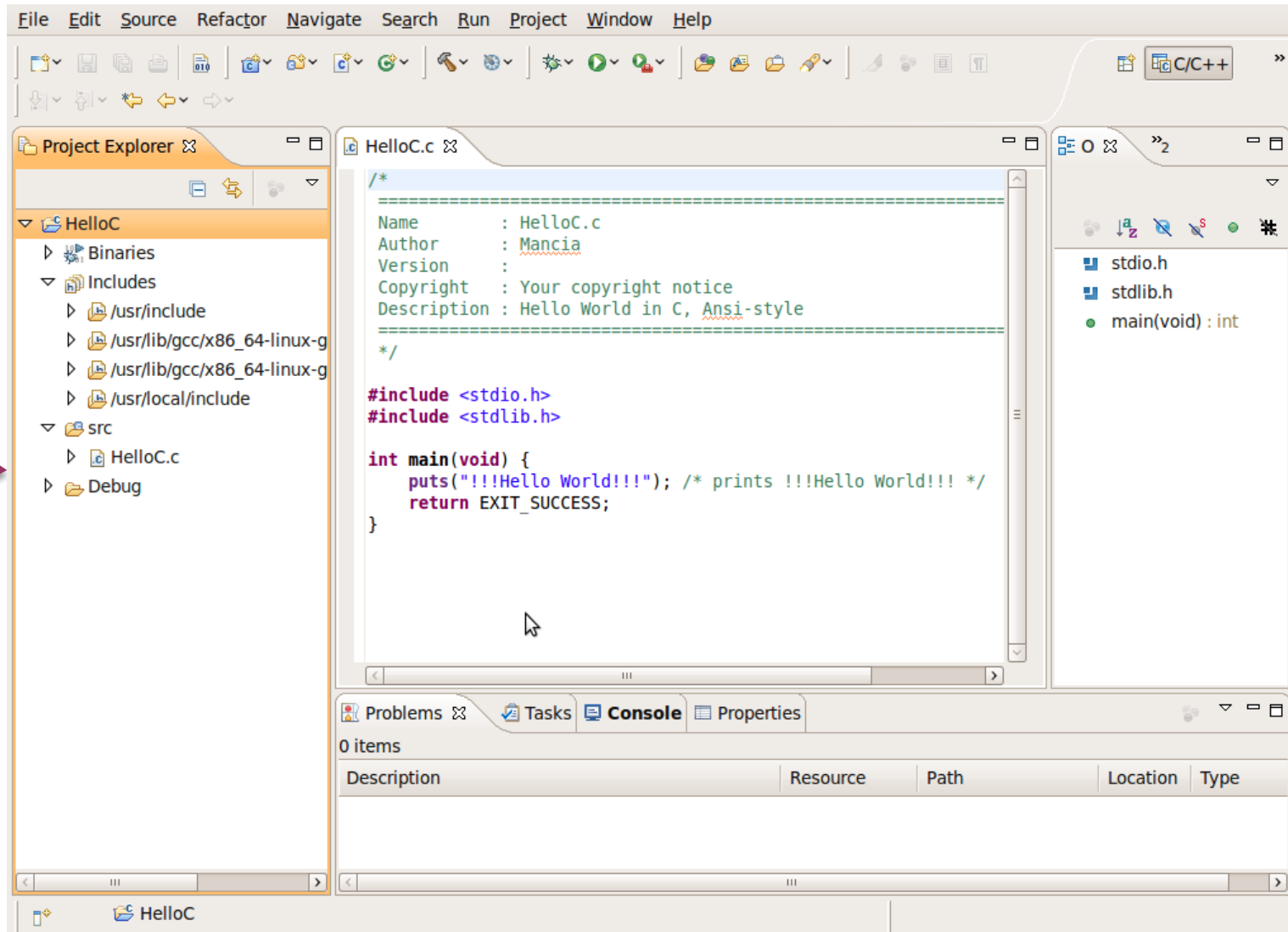


Conceptos de Eclipse – *workspace, project*

- Proyecto (*project*)
 - Agrupación de (*Window->Show View->Project Explorer*)
 - Carpetas (*src, Debug, Release, Includes, Binaries*, etc.), algunas de ellas son directorios del sistema de ficheros, y
 - Ficheros (*.h, .c, .cpp, .exe, ...*)
 - El más alto nivel de organización de los elementos de una aplicación
- Espacio de trabajo (*workspace*)
 - Almacén por defecto de (*Window->Show View->Navigator*)
 - Proyectos
 - Información de administración/control de eclipse (ocultos)
 - Se pueden tener varios (*File -> Switch Workspace*)
- Otros términos:
[http://wiki.eclipse.org/User Interface Guidelines#Glossary](http://wiki.eclipse.org/User_Interface_Guidelines#Glossary)

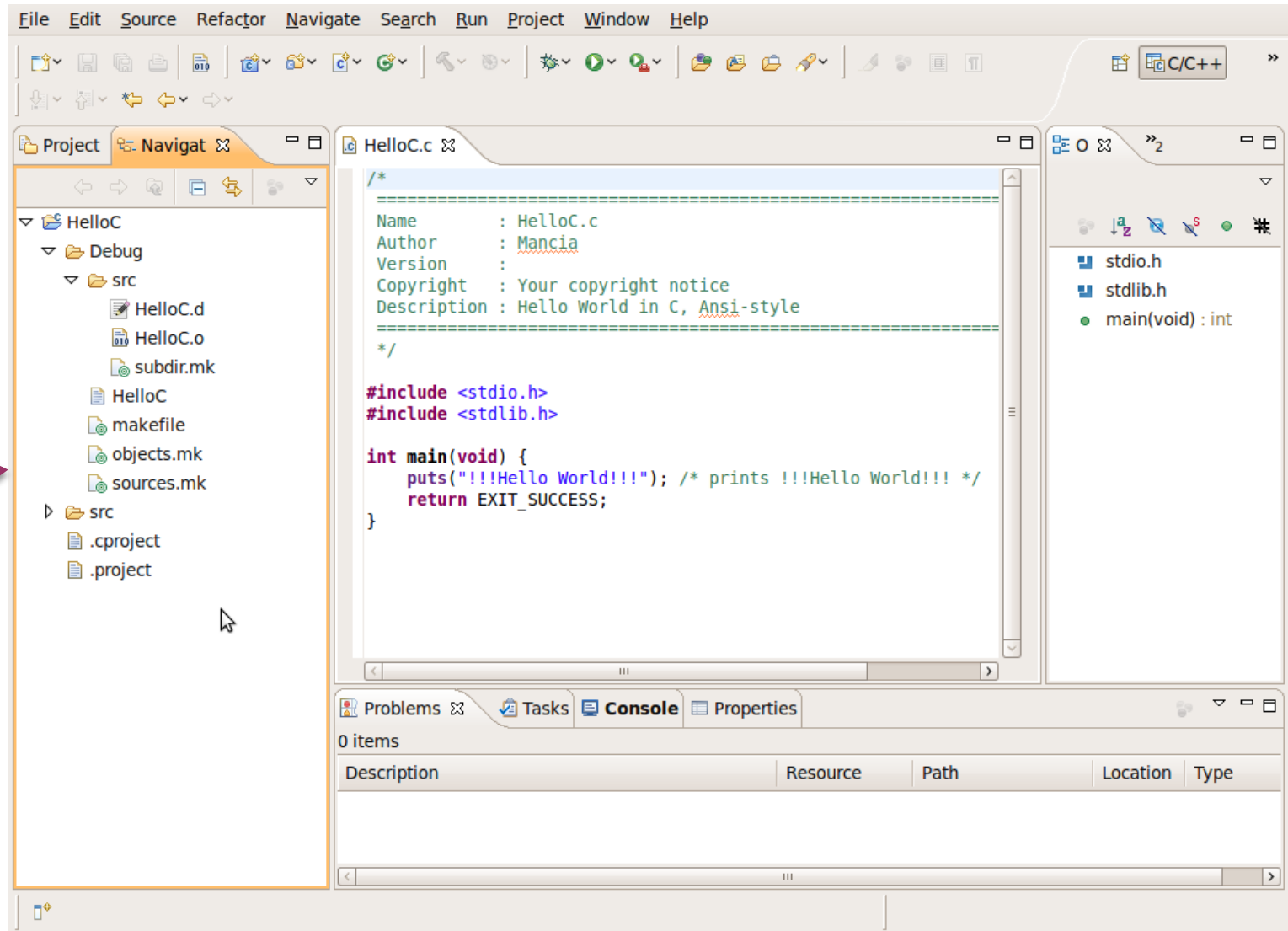
Conceptos de Eclipse – *project*

Para visualizar el proyecto:
Window->Show View->Project Explorer

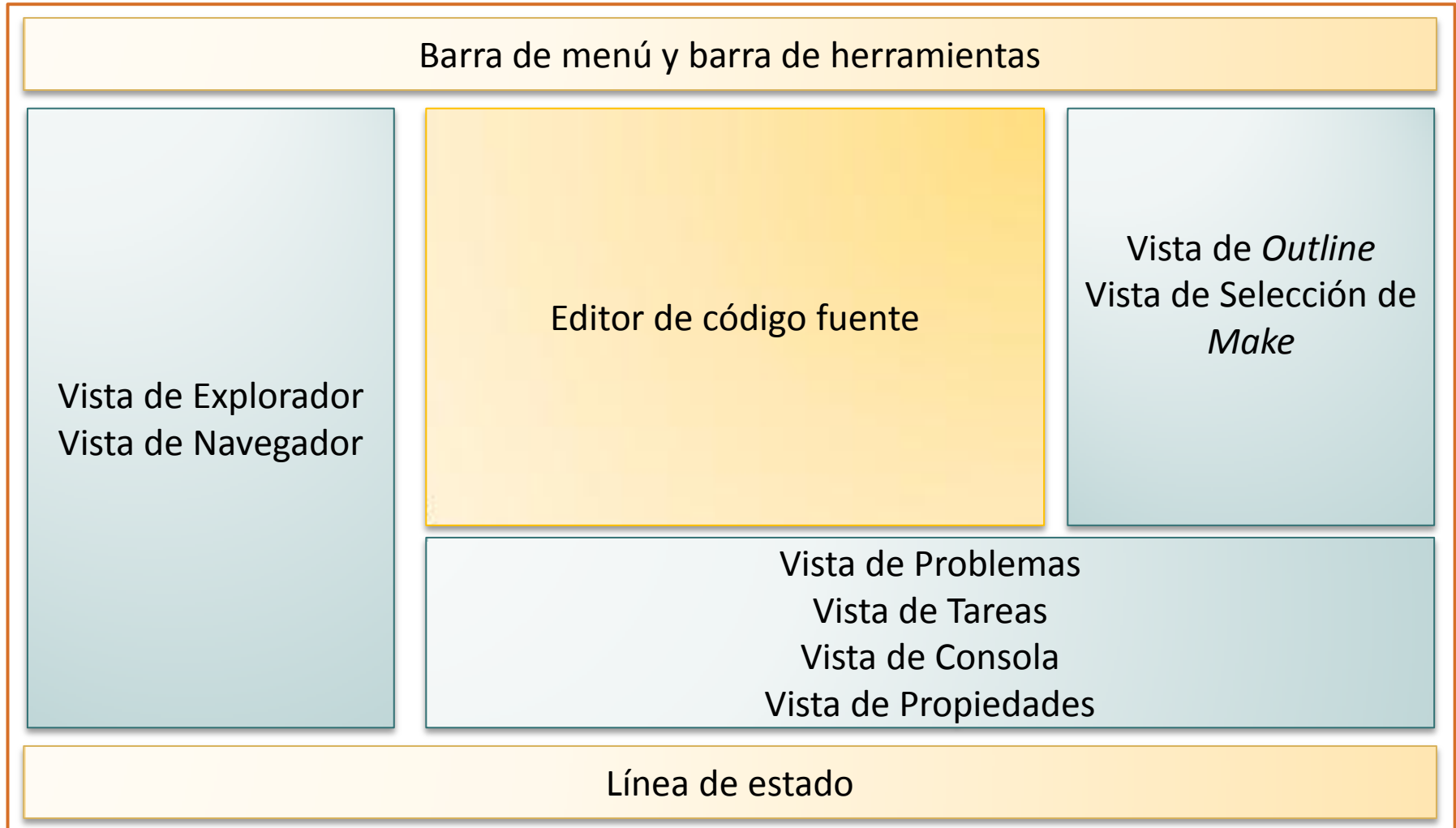


Conceptos de Eclipse – *workspace*

Para visualizar el espacio de trabajo
Window->Show View->Navigator



Organización de la mesa de trabajo de desarrollo (*workbench*)

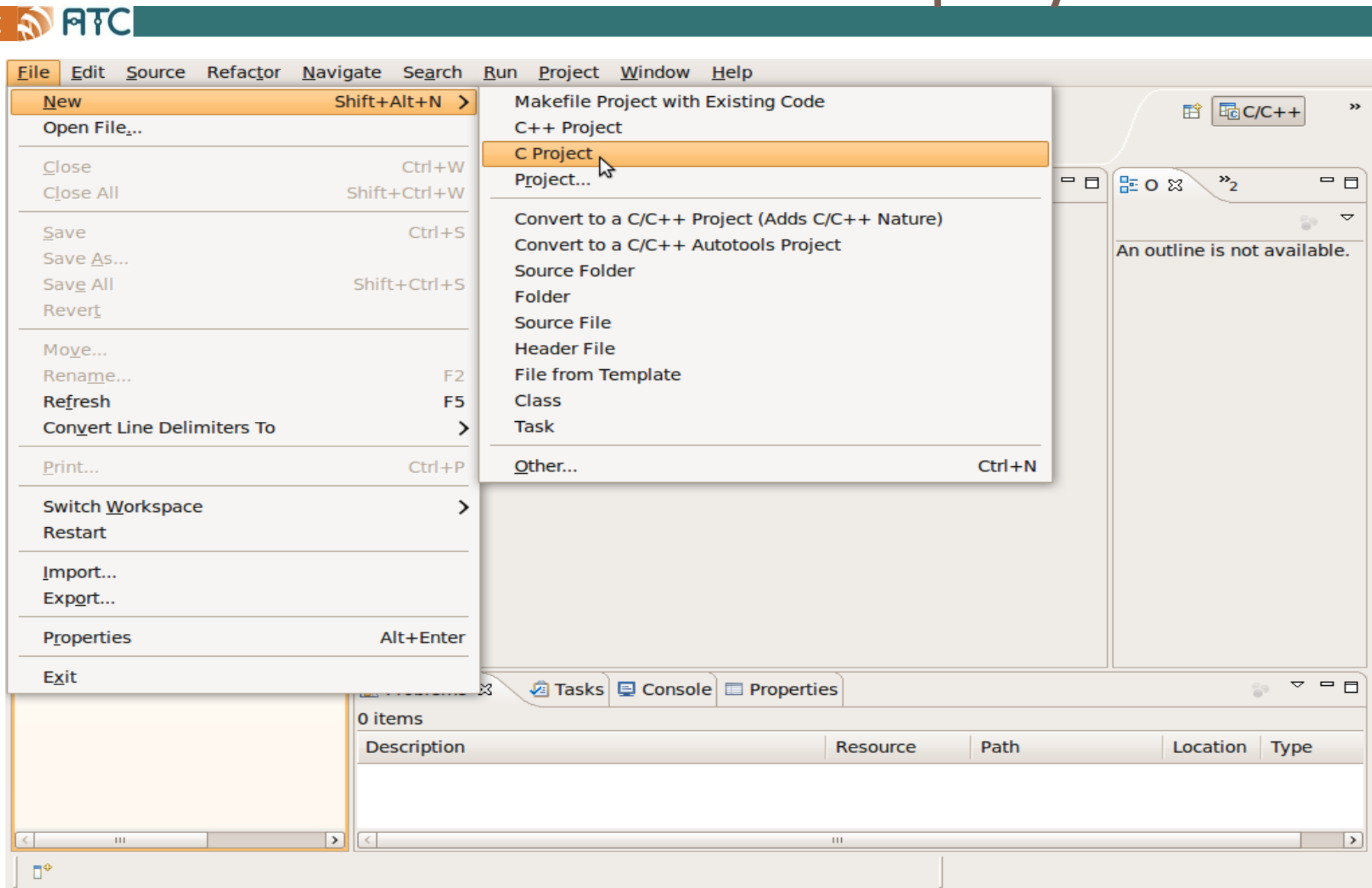


Contenidos

- ¿Qué es Eclipse? ¿Qué es CDT?
- Usuarios de Eclipse
- Instalación
- Conceptos de Eclipse
- **Ejemplo Hello**
 - Paso 1: Crear un nuevo proyecto (Hello)
 - Paso 2: Crear fichero fuente hello.c
 - Paso 3: Editar fuente hello.c
 - Paso 4: Generar ejecutable hello
 - Paso 5: Ejecución
 - Paso 6: Depuración
- Ejemplo Hello OpenMP

Ejemplo Hello

Paso 1: Crear un nuevo proyecto



Ejemplo Hello

Paso 1: Crear un nuevo proyecto

C++ Project

Create C++ project of selected type

Project name: Hello

☒ Use default location

Location: /home/mancia/docencia/OpenMP/Hello

Browse...

Project type:

- Executable
 - Empty Project
 - Hello World C++ Project
- Shared Library
- Static Library
- Makefile project

Toolchains:

Intel Toolchain for Executable on Intel(R) 64 (v11.1.0)
Linux GCC

☒ Show project types and toolchains only if they are supported on the platform



< Back

Next >

Cancel

Finish

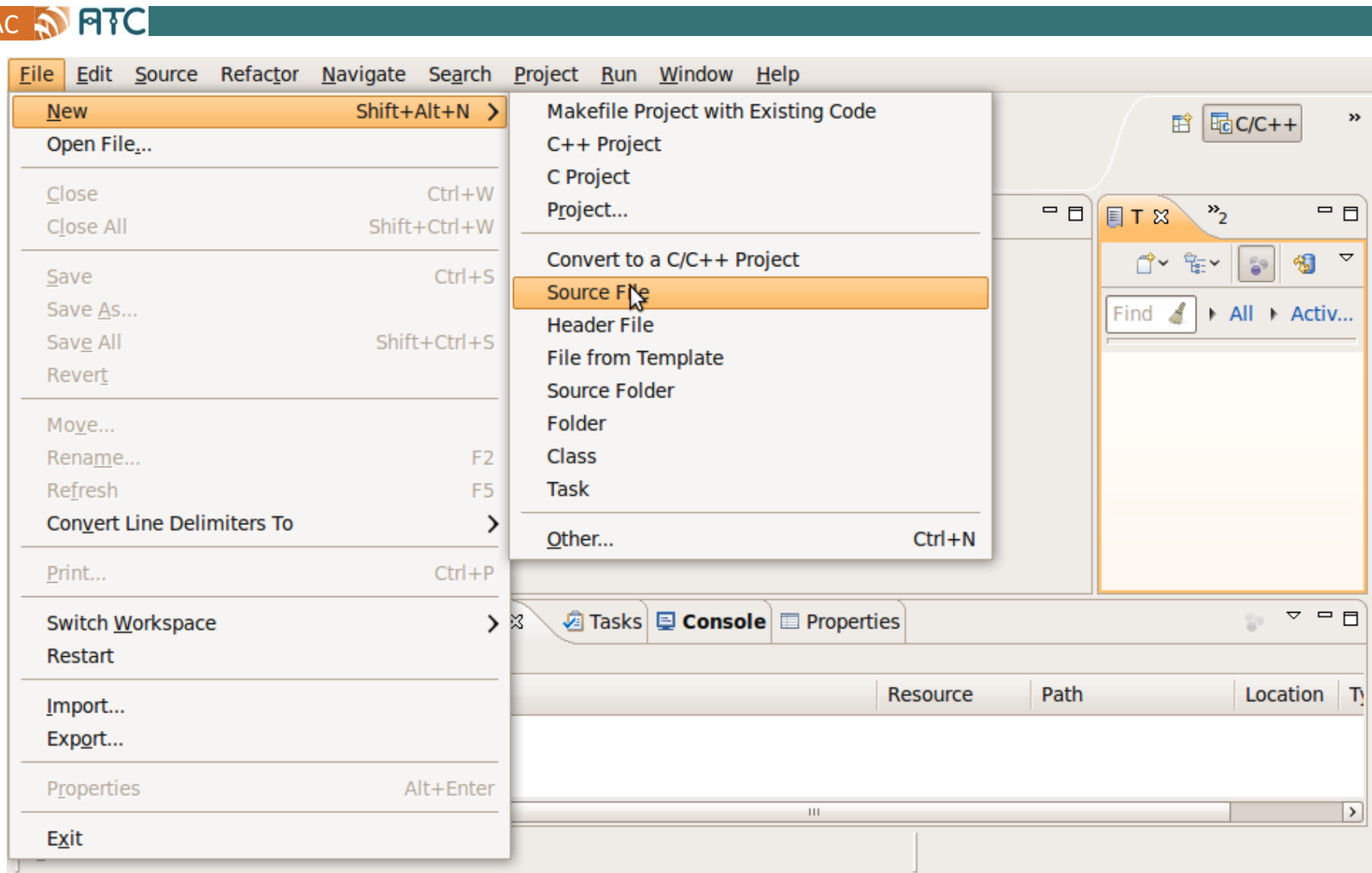
Tecleamos nombre proyecto

Seleccionamos proyecto vacío

Seleccionamos gcc como *toolchain* (compilador, enlazador, ensamblador) para generar ejecutable

Ejemplo Hello

Paso 2: Crear fichero fuente hello.c



Ejemplo Hello

Paso 2: Crear fichero fuente hello.c

Source File

Create a new source file.



Source folder: Hello

Browse...

Source file: hello.c

Template: Default C source template

Configure...



Cancel

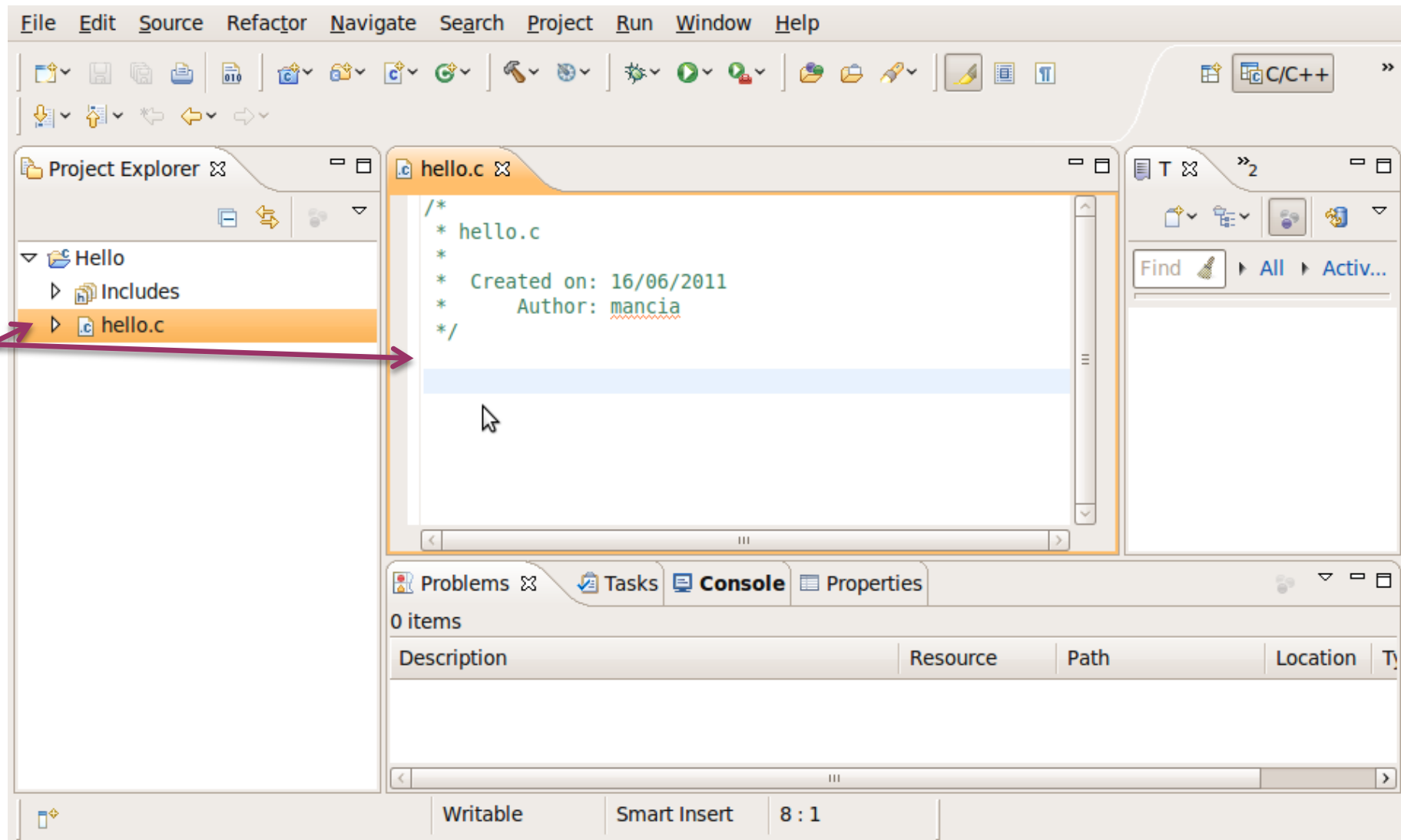
Finish

Escribimos nombre
del fuente (no
olvidar extensión
“.c”)

Seleccionamos
plantilla C

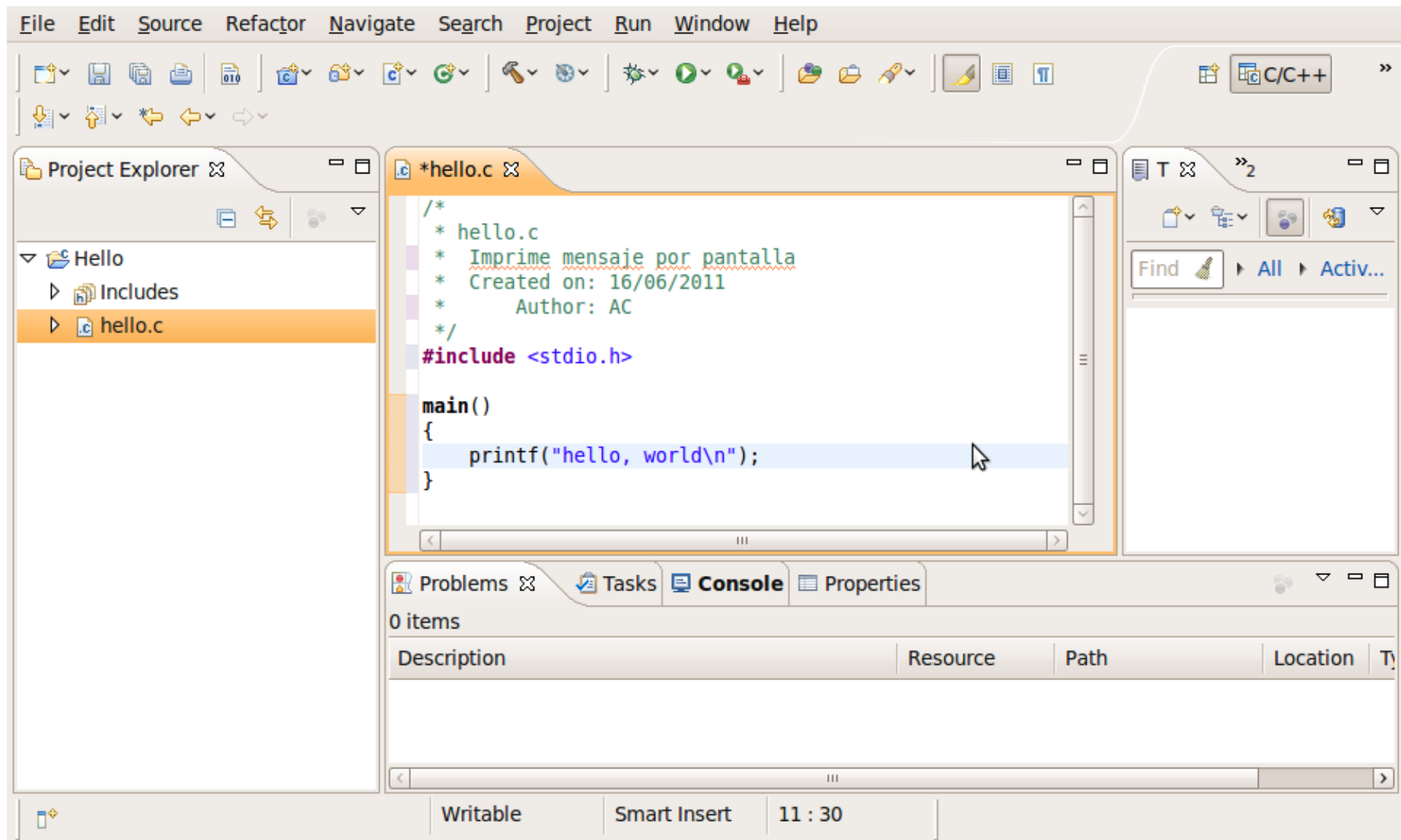
Ejemplo Hello

Paso 2: Fichero hello.c creado



Ejemplo Hello

Paso 3: Editar fuente hello.c



Ejemplo Hello

Paso 4: Generar ejecutable hello

AC ATC

File Edit Source Refactor Navigate Search **Project** Run Window Help

Open Project
Close Project
Build All Ctrl+B
Build Configurations >
Build Project
Build Working Set >
Clean...
☒ Build Automatically
Make Target >
Properties

Project Explorer

▼ Hello

- Includes
 - /usr/include
 - /usr/include/c++/4.4
 - /usr/include/c++/4.4/backward
 - /usr/include/c++/4.4/x86_64-l
 - /usr/lib/gcc/x86_64-linux-gnu/
 - /usr/lib/gcc/x86_64-linux-gnu/
 - /usr/local/include
- hello.c**

/*
* he
* I
* C
*/
#incl

main(
{
 printf("hello, world\n");
}

OJO: "*" significa que el fichero se ha modificado y no se ha guardado. Los cambios no guardados generalmente no se tendrán en cuenta al compilar

Problems Tasks Console Properties

0 items

Description	Resource	Path	Location
-------------	----------	------	----------

Writable Smart Insert 11 : 19

Ejemplo Hello

Paso 4: Generar ejecutable hello

AC ATC

Para generar ejecutable sólo del proyecto seleccionado

Project Explorer: Hello

- Includes
- hello.c

*hello.c

```
/* hello.c
 * Imprime m
 * Created o
 * Autho
 */
#include <std

main()
{
    printf("h
}
```

Project Menu:

- Open Project
- Close Project
- Build All (Ctrl+B)
- Build Configurations
- Build Project**
- Build Working Set
- Clean...
- ☒ Build Automatically
- Make Target
- Use Intel(R) C++ Compiler
- Properties

Problems: 0 errors, 2 warnings, 0 others

Description	Resource	Path	Location	Type
Warnings (2 items)				
control reaches end of non-void function	hello.c	/Hello	line 12	C/C++ I
return type defaults to 'int'	hello.c	/Hello	line 10	C/C++ I

Ejemplo Hello

Paso 4: Ejecutable hello generado

Ejecutable

The screenshot shows the Eclipse IDE interface. On the left, the Project Explorer displays the 'Hello' project structure, including 'Binaries', 'Includes', 'Debug', and 'hello.c'. The 'Hello - [x86_64/le]' executable file is highlighted with a red arrow. The main editor shows the 'hello.c' source code, which includes a comment in Spanish and a C program that prints 'hello, world'. The Problems view at the bottom shows two warnings: 'control reaches end of non-void function' and 'return type defaults to 'int''. A red arrow points to the Problems view.

```
/*
 * hello.c
 * Imprime mensaje por pantalla
 * Created on: 16/06/2011
 * Author: AC
 */
#include <stdio.h>

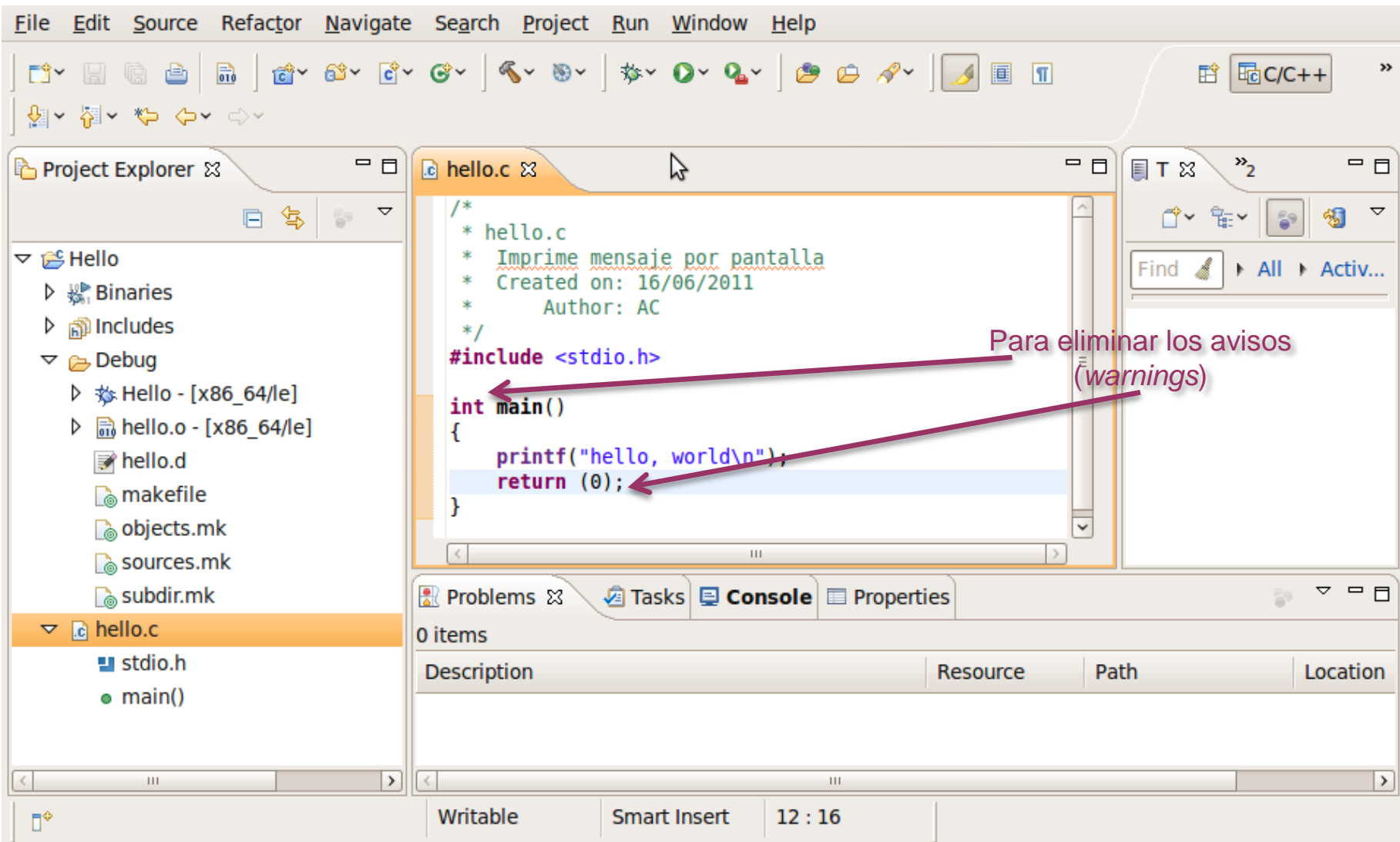
main()
{
    printf("hello, world\n");
}
```

Description	Resource	Path	Location	Type
Warnings (2 items)				
control reaches end of non-void function	hello.c	/Hello	line 13	C/C++ P
return type defaults to 'int'	hello.c	/Hello	line 10	C/C++ P

Vista de problemas (*problem view*): visualiza errores y avisos (*warnings*)

Ejemplo Hello

Paso 4: Eliminar *warnings*



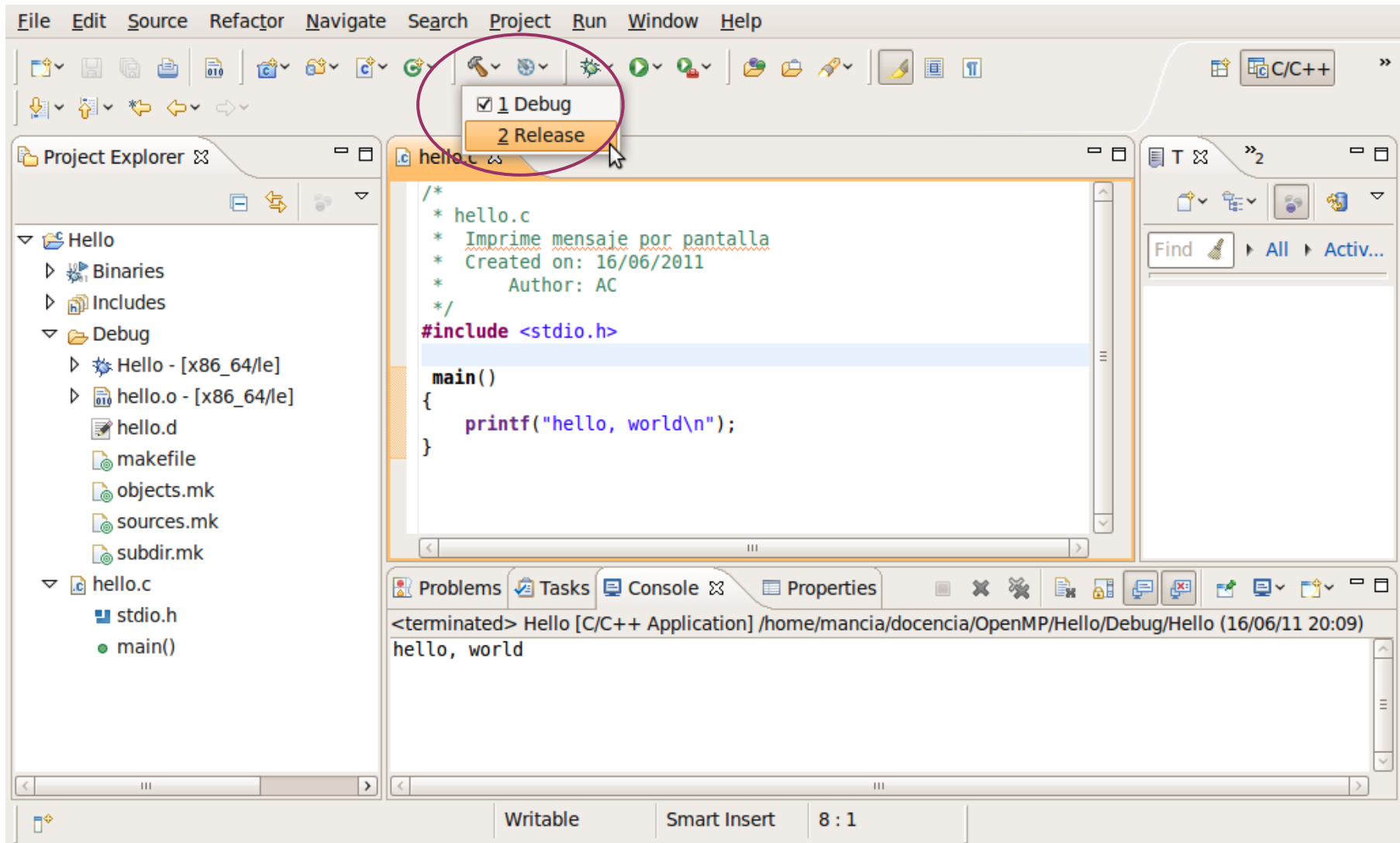
The screenshot shows the Eclipse IDE interface. The 'Project Explorer' on the left displays a project named 'Hello' with sub-projects 'Binaries', 'Includes', and 'Debug'. Under 'Debug', there are files 'Hello - [x86_64/le]', 'hello.o - [x86_64/le]', 'hello.d', 'makefile', 'objects.mk', 'sources.mk', and 'subdir.mk'. The 'hello.c' file is selected. The main editor window shows the code for 'hello.c' with the following content:

```
/*  
 * hello.c  
 * Imprime mensaje por pantalla  
 * Created on: 16/06/2011  
 * Author: AC  
 */  
#include <stdio.h>  
  
int main()  
{  
    printf("hello, world\n");  
    return (0);  
}
```

Two purple arrows point to the code with the text 'Para eliminar los avisos (warnings)'. One arrow points to the `#include <stdio.h>` line, and the other points to the `return (0);` line. The 'Problems' tab at the bottom shows '0 items'. The status bar at the bottom indicates 'Writable', 'Smart Insert', and '12 : 16'.

Ejemplo Hello

Paso 4: Ejecutable hello *release*



Ejemplo Hello

Paso 4: Ejecutable hello *release*

Ejecutable Release (para hacer público)

The screenshot displays the Eclipse IDE interface. The **Project Explorer** on the left shows the project structure for 'Hello'. Under the 'Release' configuration, the executable 'Hello - [x86_64/le]' is highlighted. The main editor shows the source file 'hello.c' with the following code:

```
/*  
 * hello.c  
 * Imprime mensaje por pantalla  
 * Created on: 16/06/2011  
 * Author: AC  
 */  
#include <stdio.h>  
  
main()  
{  
    printf("hello, world\n");  
}
```

The bottom console shows 0 errors and 2 warnings:

Description	Resource	Path	Location	Type
Warnings (2 items)				
control reaches end of non-void function	hello.c	/Hello	line 12	C/C++ P
return type defaults to 'int'	hello.c	/Hello	line 10	C/C++ P

Ejemplo Hello

Paso 5: Ejecución

File Edit Source Refactor Navigate Search Project Run Window Help

Para generar la perspectiva de depuración

Para ejecutar

Se ejecuta ejecutable Debug. Usar Run->Run Configuration->Main->Search Projects para cambiar entre ejecutable Debug y Release

Vista de consola (console view) : para ver lo que se imprime por pantalla

```
/*
 * hello.c
 * Imprime mensaje por pantalla
 * Created on: 16/06/2011
 * Author: AC
 */
#include <stdio.h>

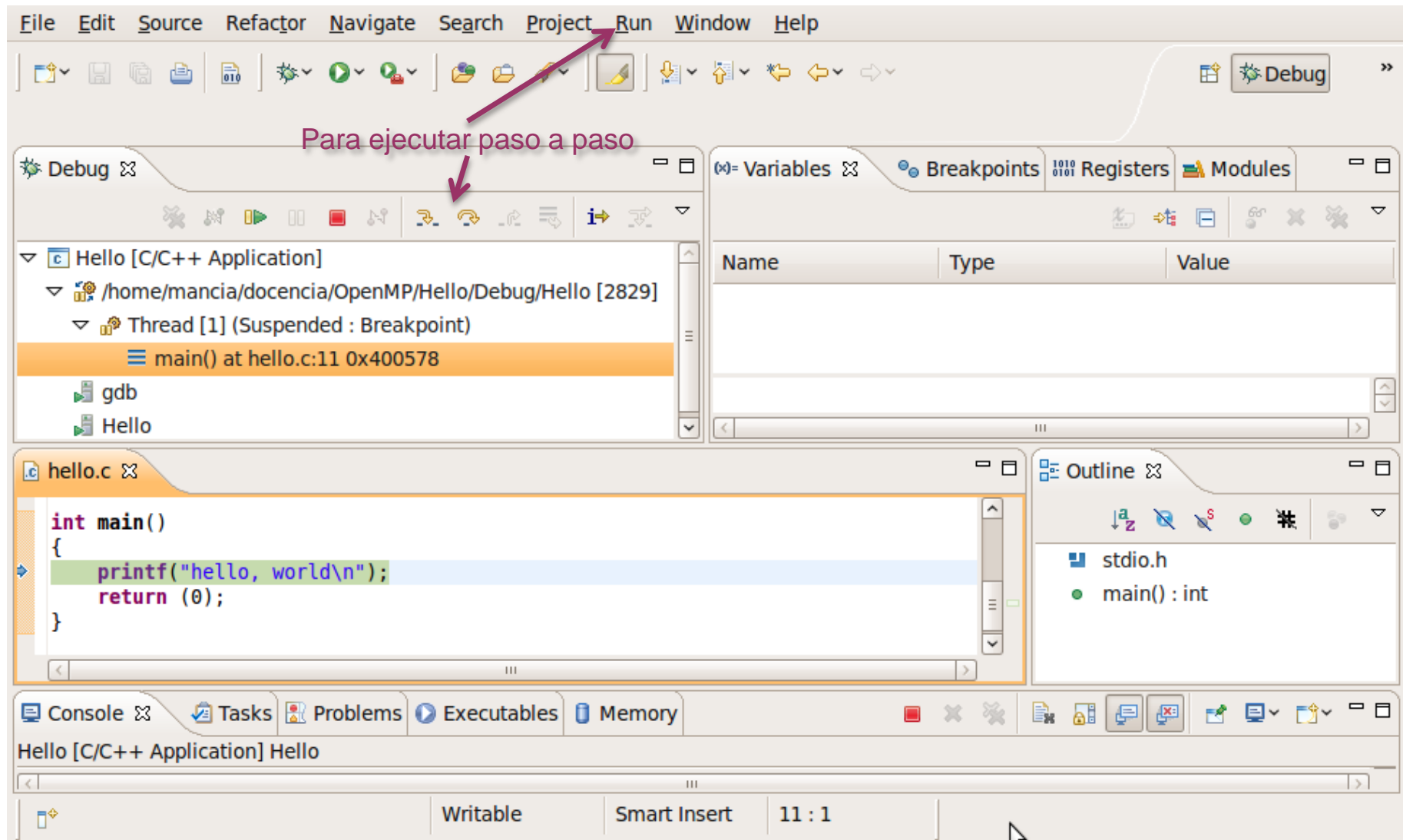
main()
{
    printf("hello, world\n");
}
```

<terminated> Hello [C/C++ Application] /home/mancia/docencia/OpenMP/Hello/Debug/Hello (16/06/11 19:26)
hello, world

Writable Smart Insert 9 : 1

Ejemplo Hello

Paso 6: Depuración



File Edit Source Refactor Navigate Search Project **Run** Window Help

Para ejecutar paso a paso

Debug

Variables Breakpoints Registers Modules

Name	Type	Value
------	------	-------

hello.c

```
int main()
{
    printf("hello, world\n");
    return (0);
}
```

Outline

- stdio.h
- main() : int

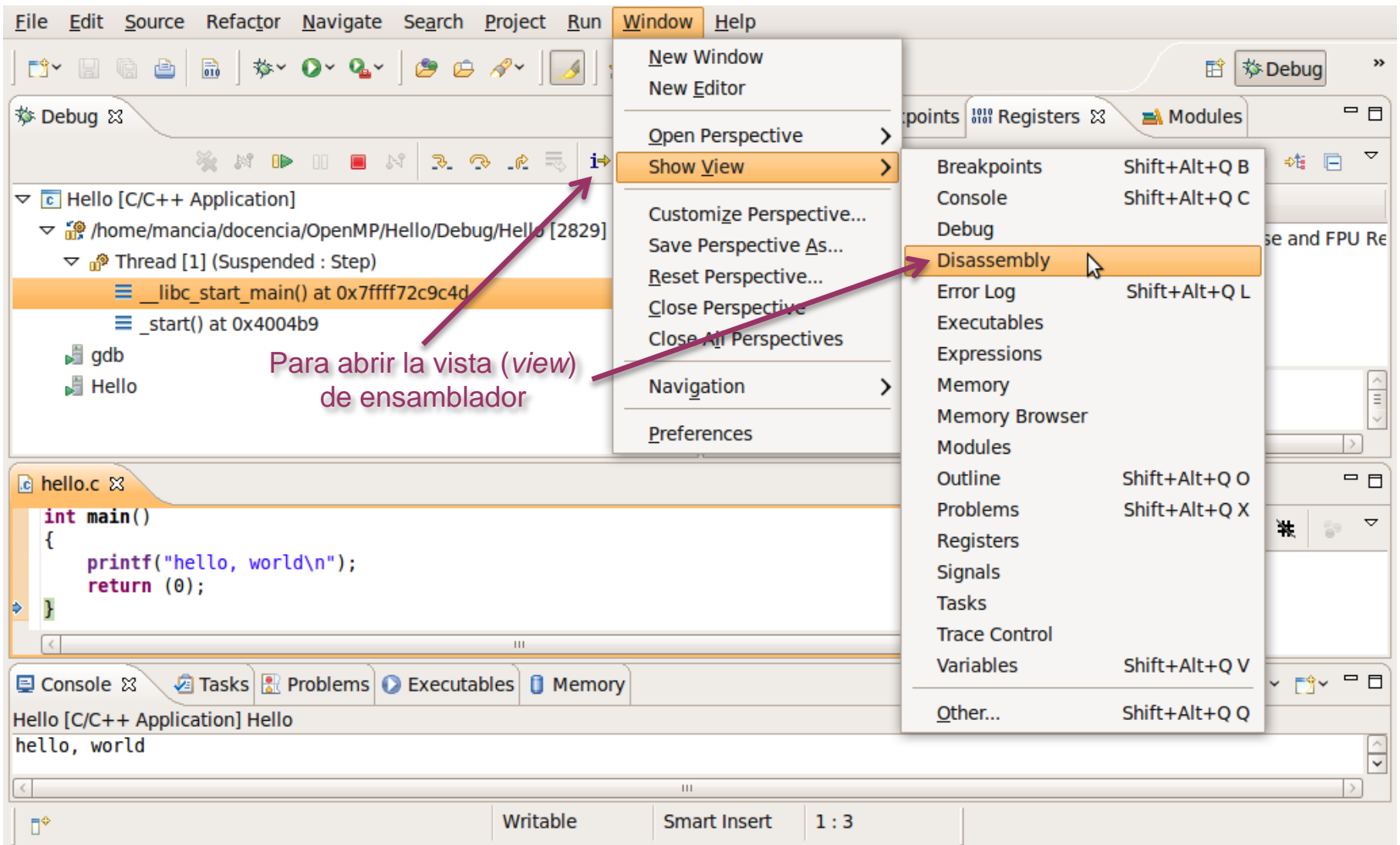
Console Tasks Problems Executables Memory

Hello [C/C++ Application] Hello

Writable Smart Insert 11 : 1

Ejemplo Hello

Paso 6: Depuración en ensamblador



Ejemplo Hello

Paso 6: Depuración en ensamblador



The screenshot shows the Eclipse IDE with the following components:

- Debug Console:** Shows the output "Hello [C/C++ Application] Hello".
- Registers Window:** Displays the values of general registers.

Name	Value	Description
General Registers		
rax	140737343827656	General Purpose and FPU Register Group
rbx	0	
rcx	3	
rdx	140737488348040	
- Disassembly Window:** Shows the assembly code for the 'main' function.

```
main:
0000000000400574: push %rbp
0000000000400575: mov %rsp,%rbp
11      printf("hello, world\n");
0000000000400578: mov $0x40067c,%edi
000000000040057d: callq 0x400470 <puts@plt>
12      return (0);
0000000000400582: mov $0x0,%eax
13
```
- Source Window:** Shows the C code for 'hello.c'. The line `printf("hello, world\n");` is highlighted.

A red arrow points to the 'Show Source' button in the disassembly window, with the text: "Para seleccionar ver código ensamblador+código fuente".

Contenidos

- ¿Qué es Eclipse? ¿Qué es CDT?
- Usuarios de Eclipse
- Instalación
- Conceptos de Eclipse
- Ejemplo Hello
- **Ejemplo Hello OpenMP**
 - Paso 1: Modificar propiedades del proyecto
 - Paso 2: Añadir la opción `-fopenmp`
 - Paso 3: Generar ejecutable
 - Paso 4: Ejecutar
 - Paso 5: Depurar

Ejemplo Hello OpenMP

Paso 1: Modificar propiedades proyecto



File Edit Source Refactor Navigate Search Run **Project** Window Help

Open Project
Close Project
Build All Ctrl+B
Build Configurations >
Build Project
Build Working Set >
Clean...
☒ Build Automatically
Make Target >
Properties

Project Explorer

- HelloC
 - Binaries
 - Includes
 - src

Code Editor: HelloC.c

```
/*  
===== Name  
===== Author  
=====*/  
#include <std  
#include <std  
#include <omp  
  
int main(void) {  
    omp_set_num_threads(2);  
    #pragma omp parallel  
    puts("!!!Hello World!!!"); /* prints !!!Hell  
    return(0);  
}
```

Properties: C/C++

- stdio.h
- stdlib.h
- omp.h
- main(void) : int

Problems: 0 items

Description	Resource	Path	Locatio
-------------	----------	------	---------

Se incluye la librería de funciones OpenMP: omp.h

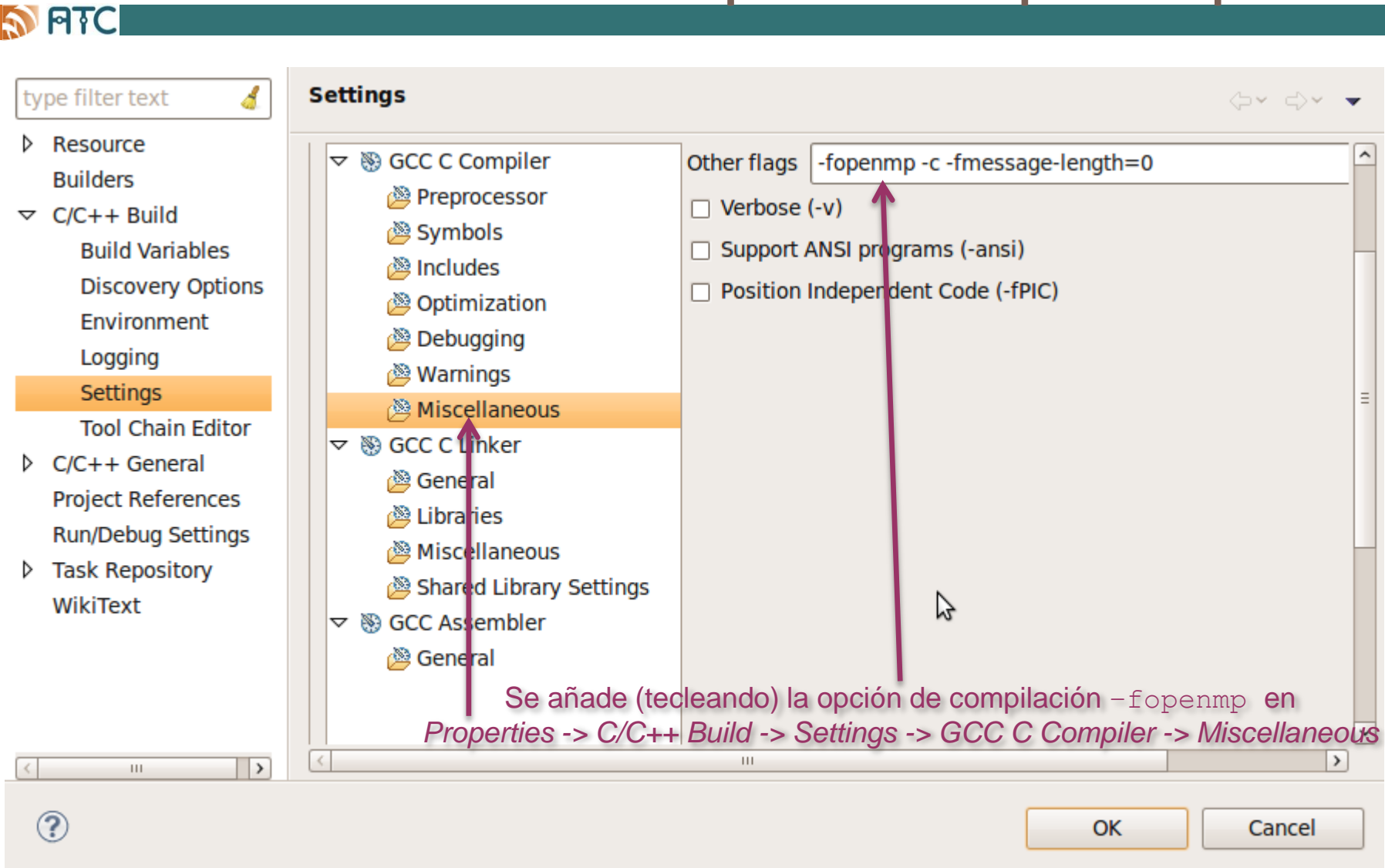
Función OpenMP para fijar el nº de hebras: omp_set_num_threads()

Directiva OpenMP para ejecución de código en paralelo: parallel

Ejemplo Hello OpenMP

Paso 2: Añadir la opción -fopenmp

AC ATC



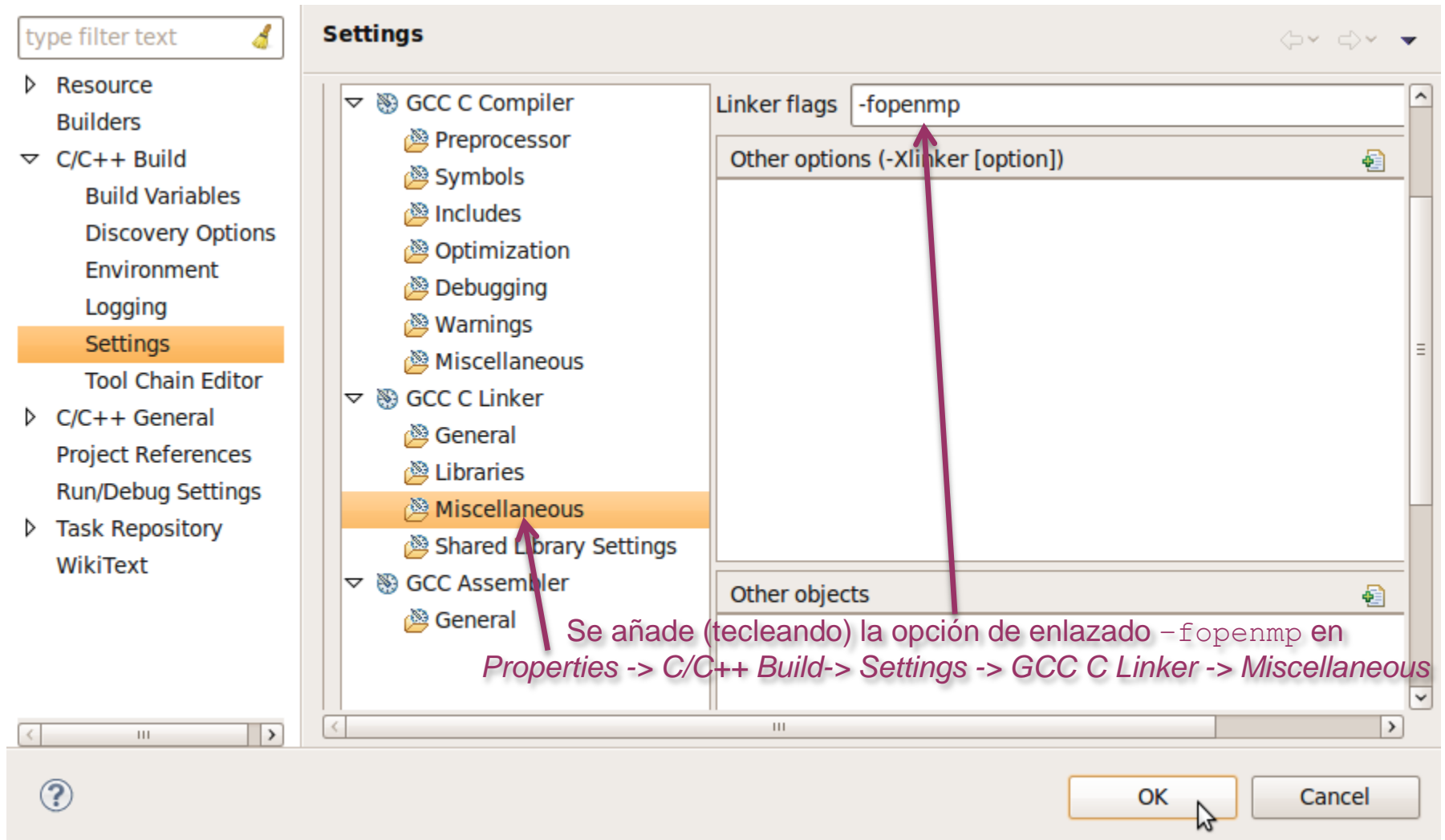
The screenshot shows the Eclipse IDE's 'Settings' dialog for the 'GCC C Compiler'. The left sidebar shows the project structure with 'Settings' selected. The main panel shows the 'Miscellaneous' tab for the 'GCC C Compiler'. The 'Other flags' field contains the text '-fopenmp -c -fmessage-length=0'. A purple arrow points from the text below to the '-fopenmp' flag in the field.

Se añade (tecleando) la opción de compilación `-fopenmp` en
Properties -> C/C++ Build -> Settings -> GCC C Compiler -> Miscellaneous

OK Cancel

Ejemplo Hello OpenMP

Paso 2: Añadir la opción -fopenmp



Ejemplo Hello OpenMP

Paso 3: generar ejecutable

File Edit Source Refactor Navigate Search Run Project Window Help

Project Explorer

- ▼ HelloC
 - Binaries
 - Includes
 - src
 - Debug
 - src
 - HelloC - [x86_64/le]
 - makefile
 - objects.mk
 - sources.mk
 - Release
 - prueba

HelloC.c

```
/*  
#include <stdio.h>  
#include <stdlib.h>  
#include <omp.h>  
  
int main(void) {  
    omp_set_num_threads(2);  
    #pragma omp parallel  
    puts("!!!Hello World!!!"); /* prints  
    return(0);  
}
```

CDT Build Console [HelloC]

Invoking: GCC C Compiler
gcc -O2 -g3 -Wall -fopenmp -c -fmessage-length=0 -MMD -MP -MF"src/
HelloC.d" -MT"src/HelloC.d" -o "src/HelloC.o" "../src/HelloC.c"
Finished building: ../src/HelloC.c

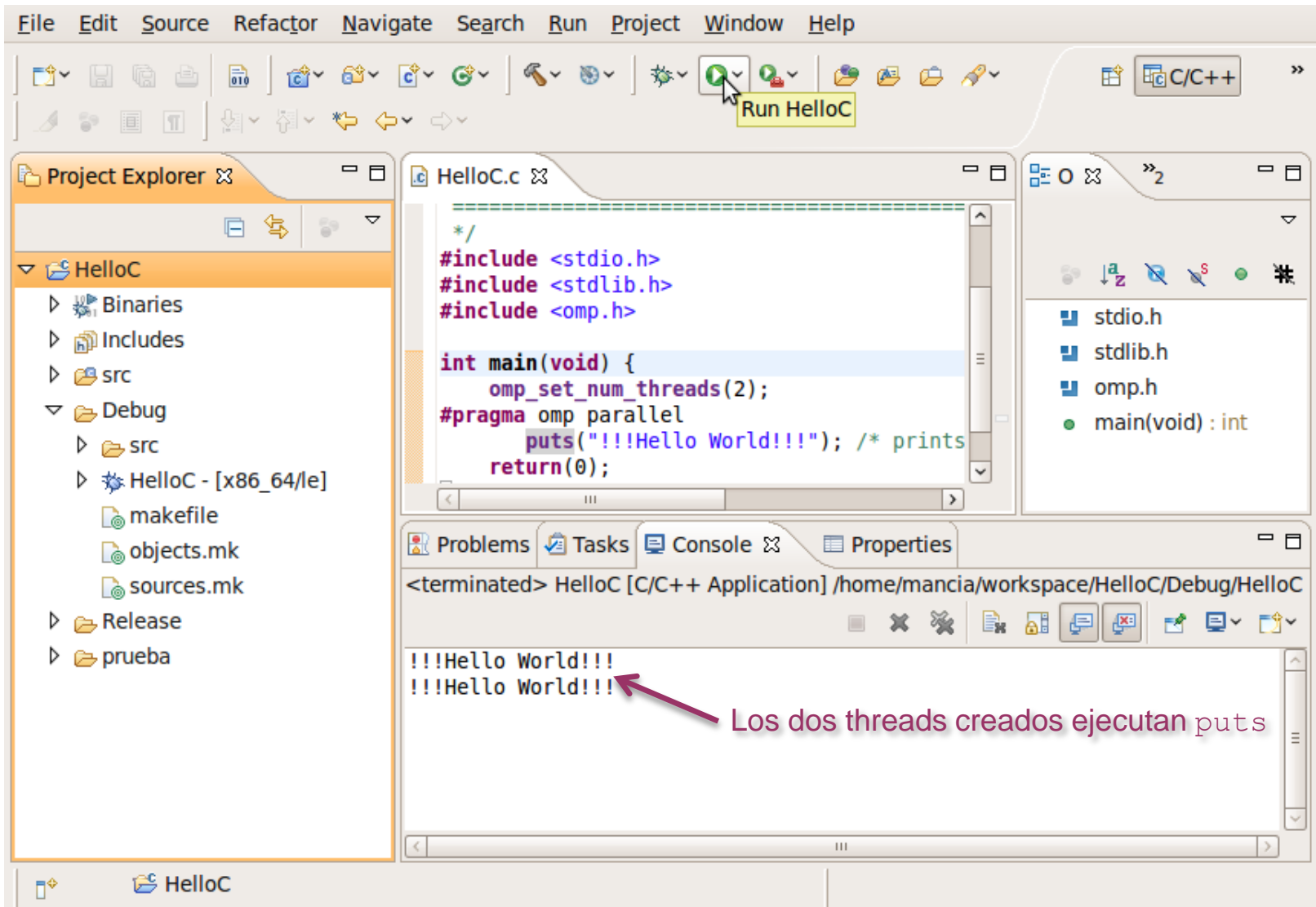
Building target: HelloC
Invoking: GCC C Linker
gcc -fopenmp -o "HelloC" ./src/HelloC.o
Finished building target: HelloC

Se usa -fopenmp en la generación de objetos por el compilador

Se usa -fopenmp en la generación del ejecutable por el enlazador

Ejemplo Hello OpenMP

Paso 4: ejecutar



Ejemplo Hello OpenMP

Paso 5: depurar

Las hebras se han ejecutado en paralelo en cores distintos (con Indigo aparece el identificador del core)

The screenshot shows the Eclipse IDE in debug mode. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Run, Project, Window, and Help. The toolbar contains icons for file operations, running, and debugging. The Debug console on the left shows two threads: Thread [2] 3457 [core: 1] (Suspended : Container) and Thread [1] 3453 [core: 0] (Suspended : Step). The main() function is highlighted in orange. The Source editor shows the code for main(), which uses OpenMP to parallelize the execution. The Disassembly view on the right shows the assembly code for the main() function, including instructions like callq, xor, add, and retq. The Console view at the bottom shows the output of the program: HelloC [C/C++ Application] HelloC, followed by two lines of "!!!Hello World!!!".

File Edit Source Refactor Navigate Search Run Project Window Help

Debug

Variable Breakpoint Register Modules

Name Type

Thread [2] 3457 [core: 1] (Suspended : Container)

Thread [1] 3453 [core: 0] (Suspended : Step)

main() at HelloC.c:16 0x4006f8

gdb

HelloC.c »1

```
int main(void) {
    omp_set_num_threads(2);
    #pragma omp parallel
    puts("!!!Hello World!!!");
    return(0);
}
```

Outline Disassembly

Enter location here

00000000004006f3: callq 0x4005a0 <GOMP_parallel_end@plt>

16 }

00000000004006f8: xor %eax,%eax

00000000004006fa: add \$0x8,%rsp

00000000004006fe: retq

Console Tasks Problems Executables Memory

HelloC [C/C++ Application] HelloC

!!!Hello World!!!

!!!Hello World!!!

Writable Smart Insert 16 : 1