Tecnicatura Superior en Análisis, Desarrollo y Programación de Aplicaciones

Programación I

"Trabajando con Librerías Estáticas en el Entorno de Desarrollo Geany" Versión 1.0 Agosto 2011

Lic. Guillermo R. Cherencio

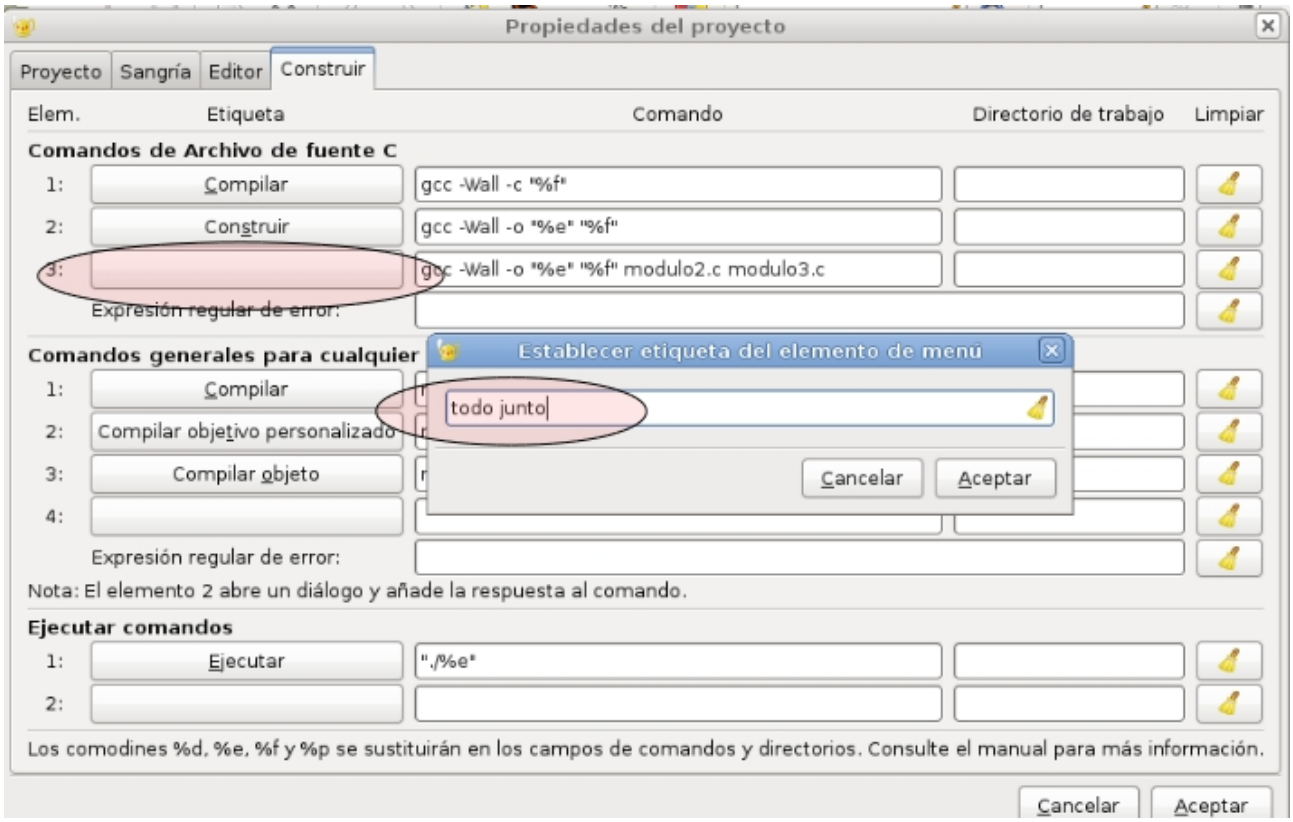
Índice

Introducción

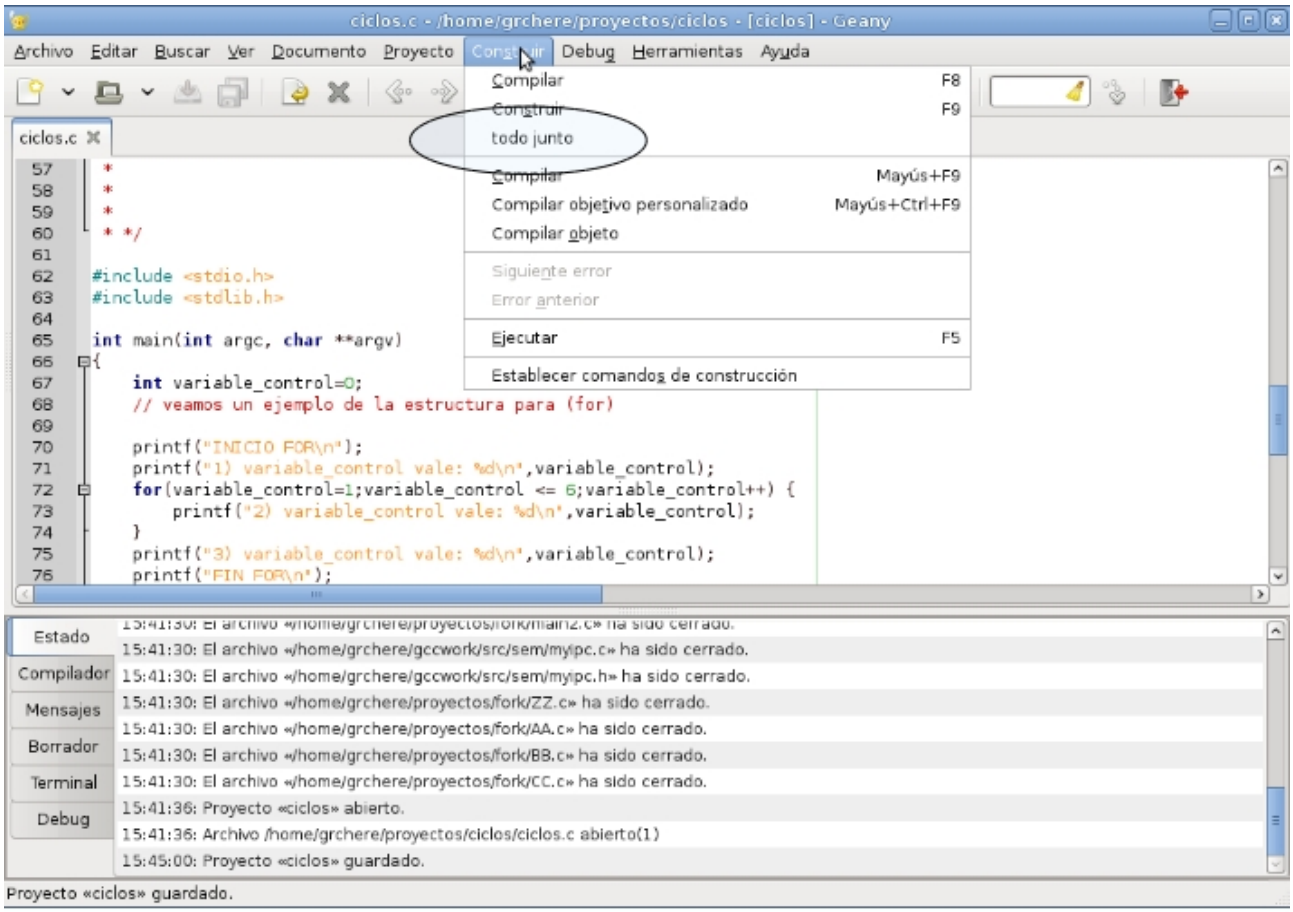
El entorno de desarrollo Geany (http://www.Geany.org) nos facilita crear programas en lenguaje C (entre otros lenguajes con los cuales puede usarse), para comenzar a trabajar, simplemente debemos crear un nuevo proyecto:

en este caso podemos ver que el trabajo se ordena dentro de la carpeta **/home/grchere/proyectos** ;vamos a crear un nuevo proyecto llamado **minuevoproyecto** ello implica crear una nueva carpeta dentro de la carpeta proyectos ; observe que Geany crea un archivo en donde guardará la configuración de este nuevo proyecto llamado **minuevoproyecto.Geany** *es importante que este archivo quede guardado dentro de la carpeta minuevoproyecto* ; de esta forma, copiando la carpeta minuevoproyecto podemos llevar todos los datos del proyecto de una máquina a otra sin olvidarnos nada.

Todo proyecto puede estar formado por uno o más archivos, en la medida en que agreguemos nuevos programas a nuestro proyecto, debemos ir modificando las propiedades del proyecto utilizando la opción **Proyecto / Propiedades** , especialmente en la lengüeta **Construir** ; allí podemos ver cómo Geany ejecuta al compilador GNU C/C++ y otras herramientas útiles en la construcción de programas. En la parte superior de la pantalla se encuentran los comandos de compilación y linkedición usados por Geany para este proyecto, estas opciones deben modificarse para que se adapten a nuestro proyecto. Otra opción es crear nuestra propia linea de comandos para la compilación o linkedición de nuestro proyecto y hasta incluso darle un nombre:



obsérvese que en este caso pretendemos crear nuestro propio comando para compilar y linkeditar nuestro programa (%f) que además esta formado por otros dos módulos más: modulo2.c y modulo3.c, todo ello se combina para formar un único ejecutable (%e). Una vez que presionamos Aceptar, este proyecto puede compilarse y linkearse directamente desde el menu Construir :



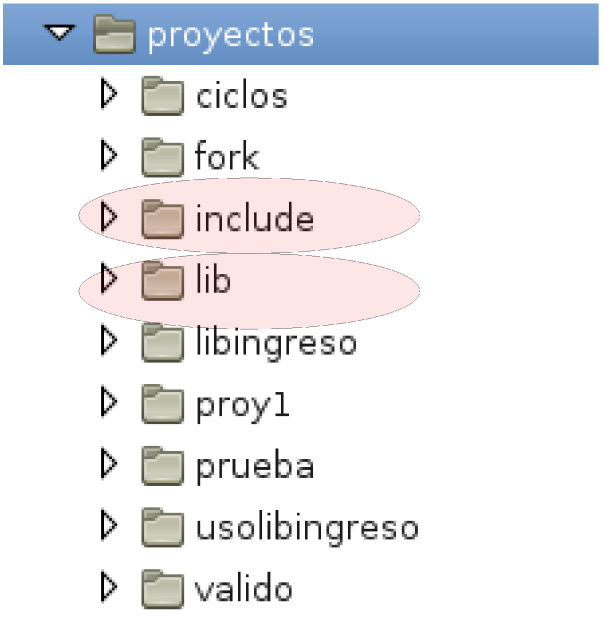
en donde podemos ver que se ha agregado una nueva opción “todo junto”.

Creación de Proyecto Librería

Uno de los problemas que nos vamos a encontrar a medida que nuestros programas crecen en complejidad es ¿cómo hacer para reusar código utilizando Geany? Una posiblidad es crear uno o más proyectos de tipo librería, generar una o más librerías estáticas y luego reusarlas en todo proyecto que necesite dichas librerías.

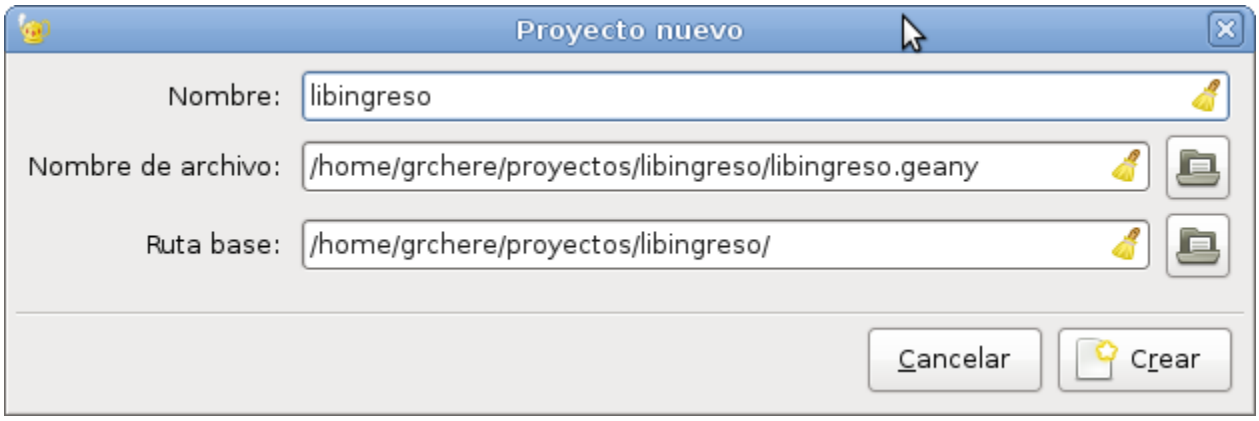
Carpetas Requeridas

Toda librería esta formada por -al menos- un archivo binario (ej: **libmilib.a**), un archivo de cabecera que contiene los prototipos de las funciones de la librería (ej: **libmilib.h**) y un archivo de documentación y/o programas de ejemplo para explicar el uso de dicha librería. Una forma posible de organización sería crear la carpeta **/include** en donde se guardarían todas las cabeceras y luego la carpeta **/lib** en donde se guardarían todos los archivos binarios. Por último también podría crearse una carpeta **/doc** en donde guardar la documentación:

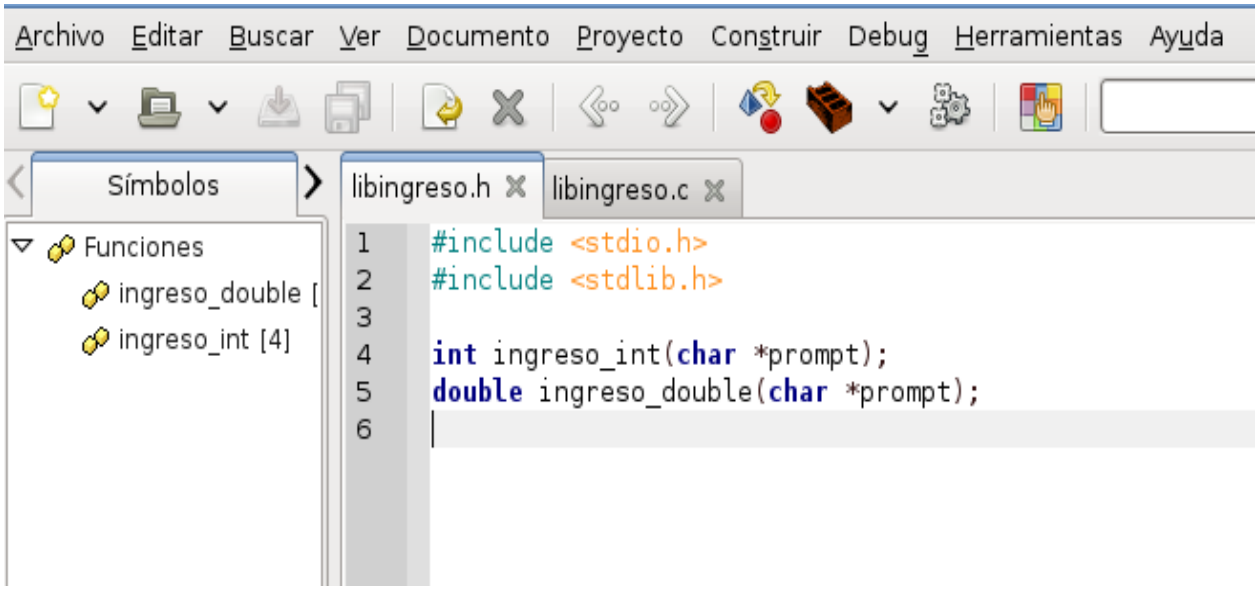


Creación de proyecto librería

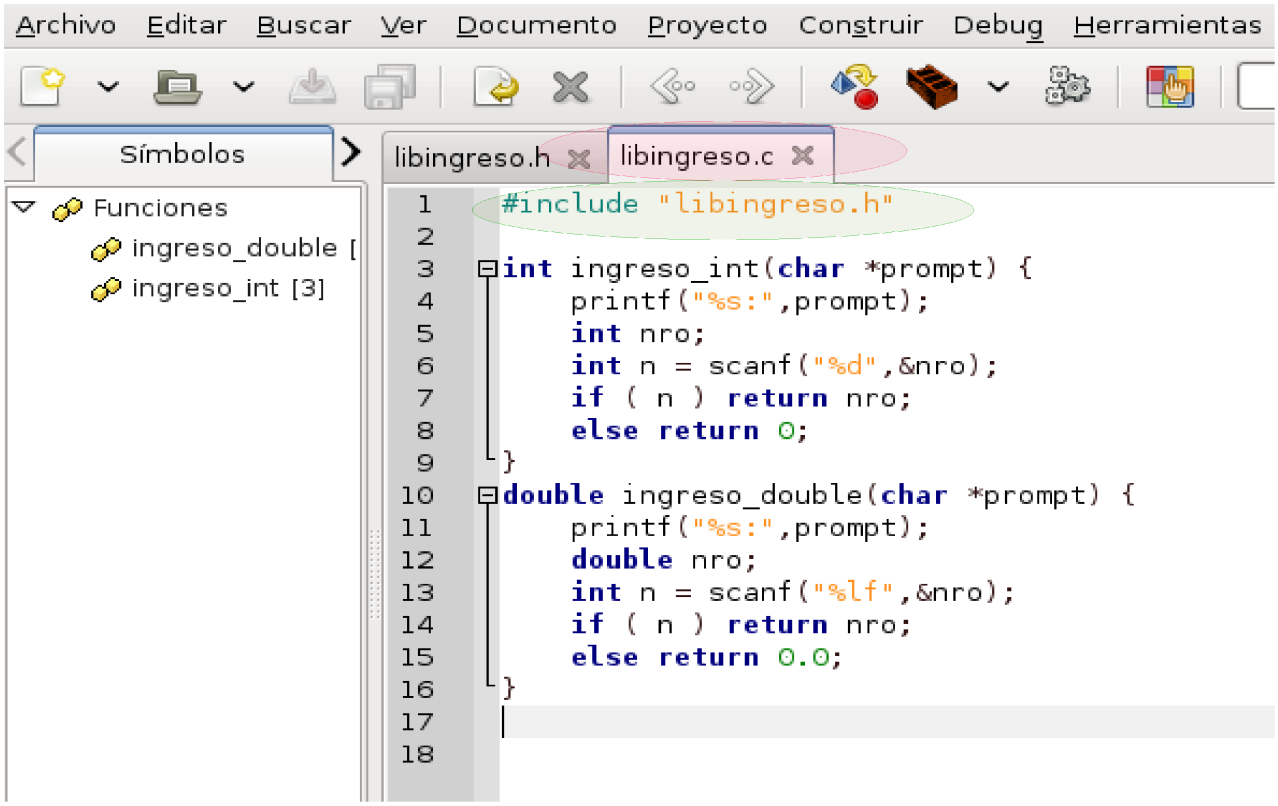
Dentro de **/proyecto** podemos crear un nuevo proyecto para nuestra librería a ser reusada en n proyectos Geany:



dentro del proyecto libingreso debemos crear -al menos- 2 archivos: **libingreso.c** (archivo de implementación de librería) y **libingreso.h** (archivo de cabecera) que contiene solo los prototipos y los #include's necesarios para esta librería:



el archivo de implementación debe hacer referencia al archivo de cabecera correspondiente:



El comando compilar deberá generar todos los archivos objeto que sean necesarios, en este caso, es sólo uno: **libingreso.o** y a partir de dicho archivo podemos construir la librería utilizando el comando **ar** :

$ **ar rcs libingreso.a libingreso.o**

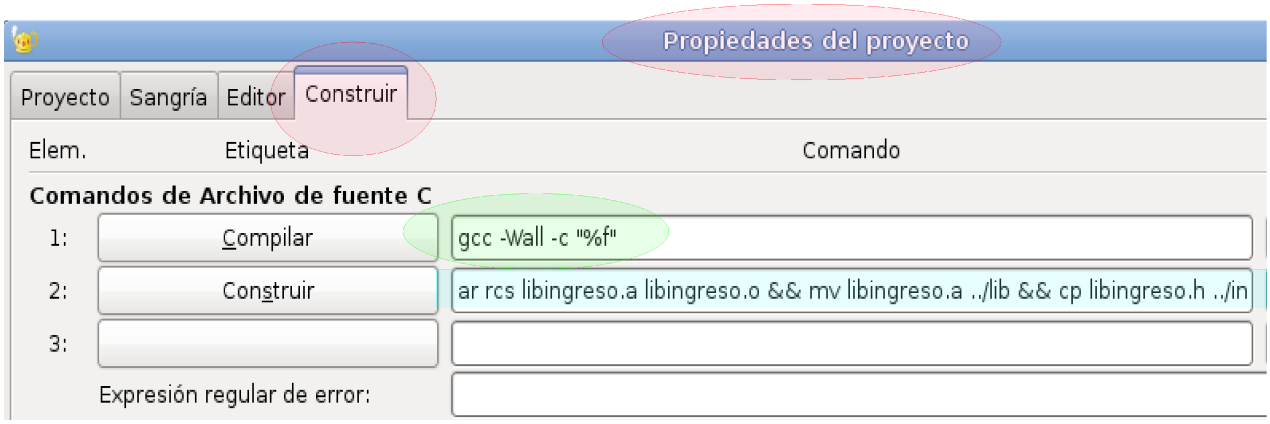
este comando creará la librería en el directorio actual del proyecto, debemos mover la librería hacia el directorio **/lib** que es el directorio desde donde vamos a reusarla en otros proyectos:

$ **mv libingreso.a ../lib**

por último, nos queda copiar el archivo de cabecera **libingreso.h** al directorio **/include** para que todos los proyectos tomen desde allí los prototipos de estas funciones:

$ **cp libingreso.h ../include**

la construcción y catalogación (colocar la librería en un lugar a partir del cual se pueda reusar) requiere de estos 3 comandos, debemos editar las propiedades del proyecto para que Geany cuando construya esta librería ejecute estos comandos. Para que 3 comandos puedan incluirse en una misma línea se deben conectar con el operador && , por lo tanto, la Construcción de este proyecto quedaría como:

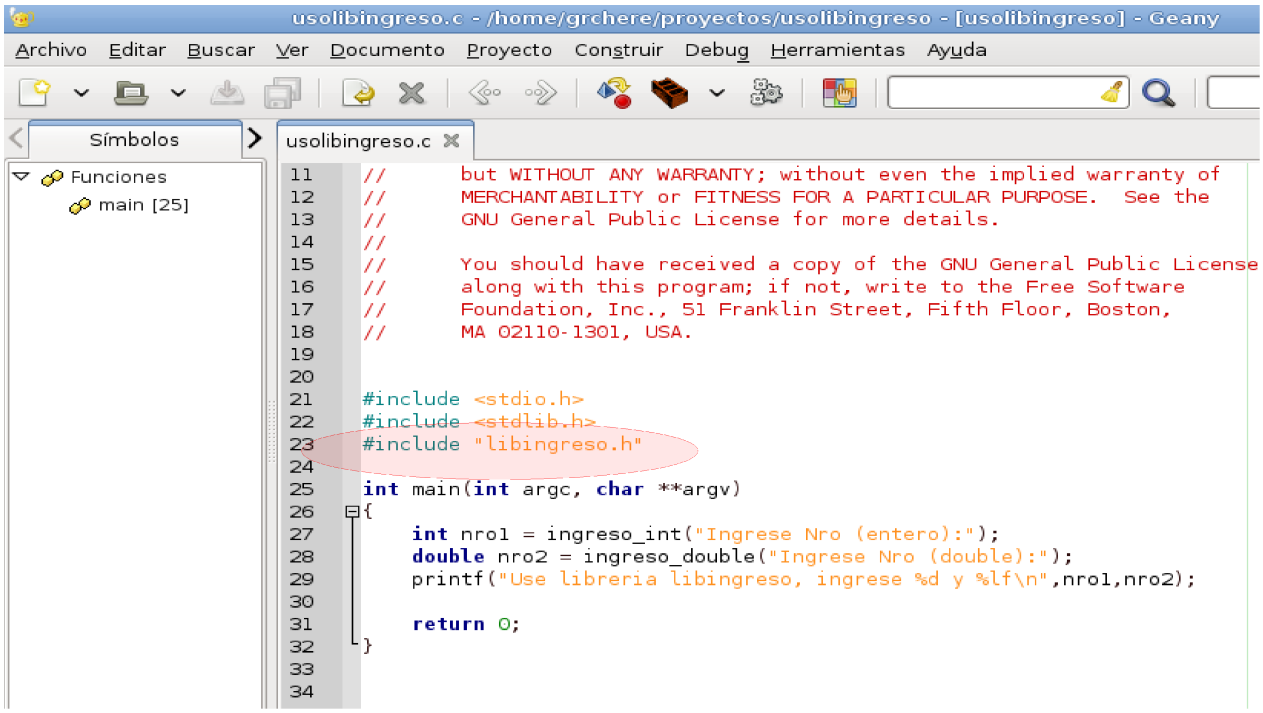


luego de compilar y construir, verifique los directorios **/lib** y **/include** para asegurarse de que contienen los archivos **libingreso.a** y **libingreso.h**, respectivamente.

Creación de Proyecto que utiliza librería

Creamos un nuevo proyecto para utilizar esta librería, en un nuevo directorio dentro del directorio **/proyectos** , en este caso, el proyecto se llama **usolibingreso** :

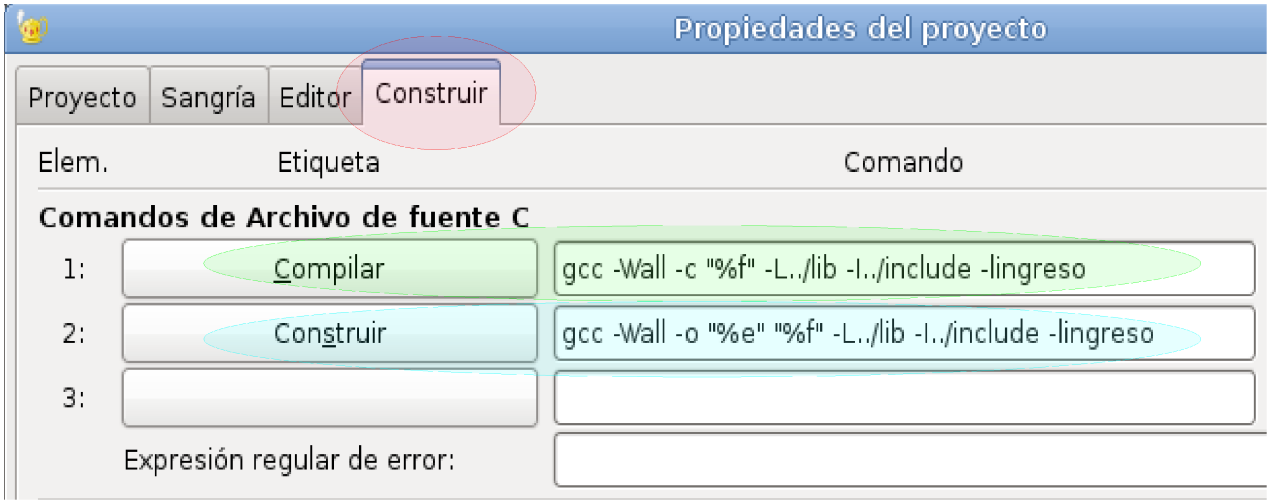
el proyecto deberá incluir los prototipos de las funciones de la librería (de la misma forma que sucede con cualquier otra librería, incluso las librerías standard de C), para ello deberá utilizar la cláusula **#include "libingreso.h"** :



el problema que se suscita aquí es que el compilador C va a buscar a **libingreso.h** en el directorio actual y no lo va a encontrar, puesto que el mismo se encuentra en **../include** ; lo mismo sucederá en la linkedición, cuando pretenda encontrar el código binario de la librería libingreso.a que se encuentra en **../lib** . El compilador GCC tiene una serie de argumentos (switches) que nos ayudarán en esta tarea:

Algunos Argumentos del Compilador GCC	
-L <directorio>	(ele mayúscula) Permite indicar directorio de librerías binarias
-l<nombre librería>	(ele minúscula) Permite indicar nombre librería, en donde si por ejemplo, usamos **-lserial** nos estamos refiriendo a la librería ***lib*serial.*a***
-I <directorio>	(i mayúscula) Permite indicar directorio de archivos de cabecera con prototipos de funciones
-i <cabecera>	(i minúscula) Permite indicar archivo de cabecera a utilizar
-c	Solo compila, no linkedita
-o <ejecutable>	Linkedita, indica nombre del archivo ejecutable a crear
-Wall	Activa todas las observaciones (warnings) del compilador, es decir, indicar a GCC que emita todo tipo de mensajes de advertencia

Debemos indicar estos argumentos tanto para la compilación como la linkedición, para que este proyecto pueda compilarse y linkearse utilizando la librería libingreso; esto debemos hacerlo en las propiedades del proyecto, lengüeta Construir :



entonces cada vez que presionamos el botón compilar y construir Geany le esta

indicando a GCC los argumentos apropiados para que encuentre la cabecera y el código binario de la librería libingreso.

¿Que sucede si un proyecto necesita utilizar más de una librería al mismo tiempo? Si tanto las cabeceras como las librerias se encuentran en **../include** y ..**/lib**, solo habrá que agregar otro argumento **-l<nombre librería>** . Si estuviese todo en directorios distintos, debemos agregar otros argumentos **-L** y **-l** para que GCC pueda encontrar la librería.

¿Qué sucede si un proyecto, ademas de usar varias librerías, también esta formado por varios programas C (supongamos modulo1.c modulo2.c) ? Simplemente a lo dicho anteriormente debemos agregarle los modulos a compilar, por ejemplo:

```
gcc -Wall -c "%f" modulo1.c modulo2.c -L../lib -L../otro.dir.para.lib -I../include
-I../otro.dir.para.include -lingreso -lotra.libreria
```

para la linkedición sería similar:

```
gcc -Wall -o "%e" "%f" modulo1.c modulo2.c -L../lib -L../otro.dir.para.lib -I../include
-I../otro.dir.para.include -lingreso -lotra.libreria
```

En cuanto a los argumentos GCC y Geany

En la sección anterior hablamos de los argumentos de GCC (tales como: **-Wall** , **-c**, **-o**, etc.) y ellos se mezclaron con argumentos que pertenecen a Geany (tales como **"%e"** y **"%f"**). No hay que confundirlos. Para facilitar la configuración de nuestro entorno de desarrollo, Geany reemplaza/sustituye los siguientes caracteres (al estilo de printf()):

Sustituciones Geany	
Caracteres	**Reemplazado por**
%d	Directorio completo (absoluto) del archivo actual
%e	Nombre del archivo actual (sin directorio y sin extensión), ejemplo: si el archivo actual es **/home/grchere/proyectos/p1/pepe.c** se sustituye por **pepe**
%f	Nombre del archivo actual, siguiendo el ejemplo anterior sería **pepe.c**
%p	Si hay un proyecto abierto, se sustituye por el directorio base del proyecto

cuando digo “archivo actual” me refiero al archivo que se está editando al momento de apretar el botón compilar o construir (linkear). Esta aclaración es importante cuando trabajamos en un proyecto formado por varios modulos, supongamos que tenemos un proyecto formado por los archivos **main.c**, **modulo1.c**, **modulo2.c** y **modulo3.c**; si en propiedades del proyecto, lengûeta Construir, compilar tipeamos:

```
gcc -Wall -c “%f” modulo1.c modulo2.c modulo3.c -L../lib -I../include -lingreso
```

esta configuración sólo funcionará cuando apretemos el botón compilar teniendo abierto (editando) al archivo main.c, puesto que Geany reemplazará a “%f” por main.c y el comando quedará:

```
gcc -Wall -c main.c modulo1.c modulo2.c modulo3.c -L../lib -I../include -lingreso
```

si, en cambio, presionamos el botón compilar mientras editamos el archivo modulo1.c, Geany reemplazará a “%f” por modulo1.c y el comando quedará:

```
gcc -Wall -c modulo1.c modulo1.c modulo2.c modulo3.c -L../lib -I../include -lingreso
```

y dicho comando es incorrecto, se está repitiendo modulo1.c y no estamos incluyendo a main.c. Por lo tanto, tenemos dos alternativas: no usar las sustituciones de Geany y compilar como:

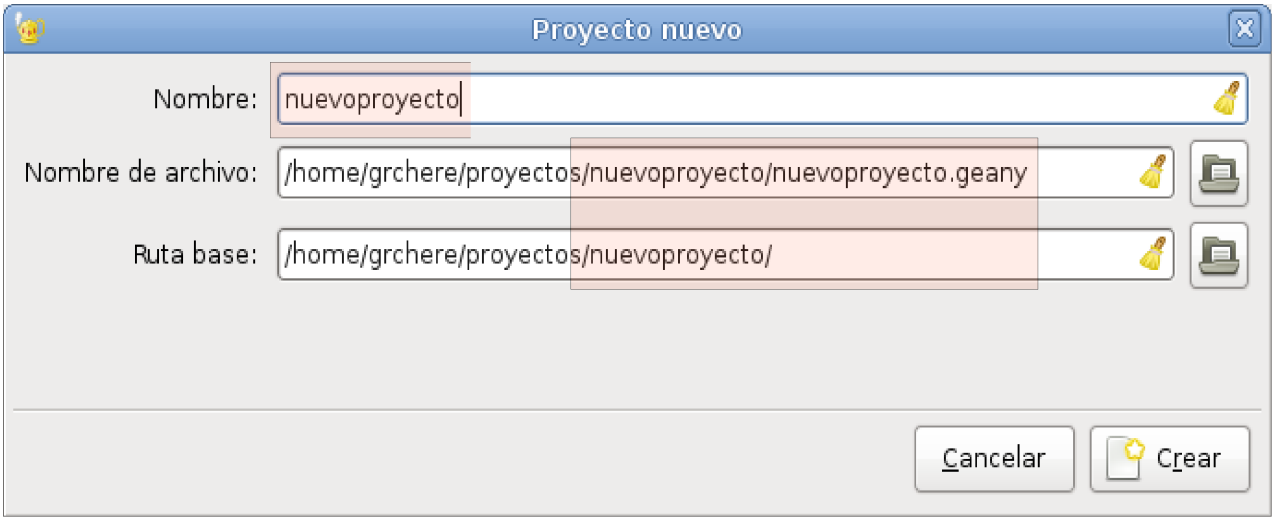
```
gcc -Wall -c main.c modulo1.c modulo2.c modulo3.c -L../lib -I../include -lingreso
```

de esta forma, no importa cual sea el “archivo actual”, la compilación funcionará siempre. O bien usar las sustituciones y compilar siempre teniendo abierto / editando el modulo principal del proyecto.

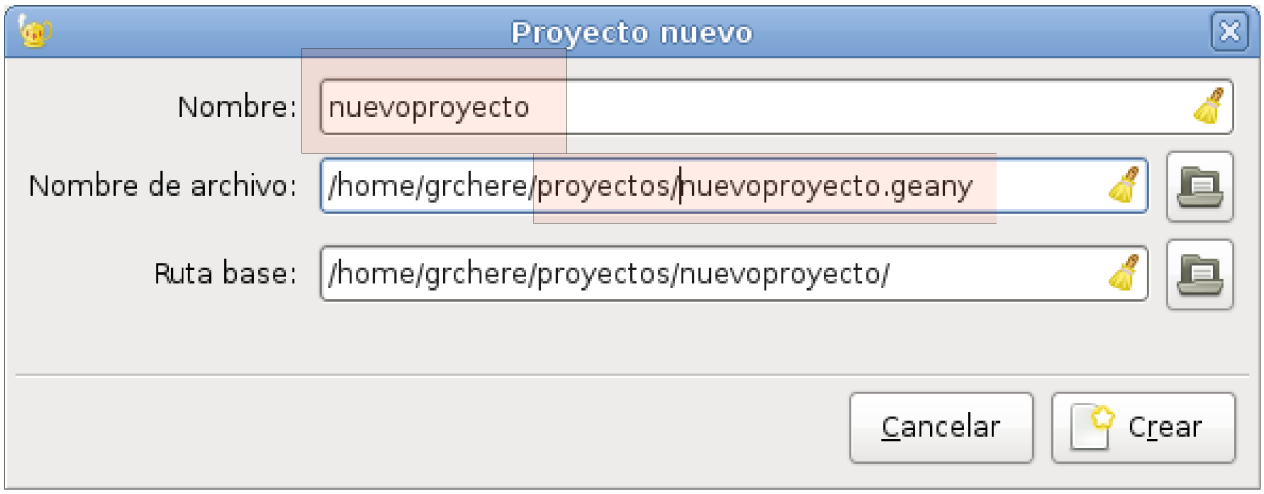
Consejos, Tips, Sugerencias

En cuanto a la creación de nuevos proyectos

Como se ha mencionado antes, cada vez que creamos un nuevo proyecto en Geany debemos estar atentos de que el archivo del proyecto (.Geany) sea creado dentro del directorio o carpeta que contiene al proyecto, de esta forma, copiando la carpeta del proyecto se copian todo y nada se pierde. Aquí podemos ver un ejemplo correcto de cómo crear un nuevo proyecto:



aquí podemos ver una forma incorrecta (para la forma de trabajo propuesta) de cómo crear un nuevo proyecto:



En cuanto a las propiedades de los proyectos

Una vez que cambiamos l**as propiedades de un proyecto**, por ejemplo utiilzando la opción Proyecto / Propiedades / Lengûeta Construir y agregamos allí, los argumentos -L para indicar directorio de librerías o bien -I para indicar directorios de include's; estas propiedades **quedarán siempre asociadas con dicho proyecto únicamente, a todo nuevo proyecto se le deberán indicar estos argumentos si fuera necesario**. Esto puede resultar algo engorroso en la etapa de aprendizaje si estamos creando un nuevo proyecto para cada programa, pero en realidad, en la práctica un proyecto abarca una serie de programas, con lo cual, se podrían realizar varios programas relacionados dentro de un mismo proyecto.

En cuanto a la ampliación de librerías

Si Ud. esta utilizando una librería (argumentos -L, -I, -l) en su proyecto y un programa del proyecto requiere el uso de una nueva función que debería estar en la librería, se aconseja que **primero implemente la función como si fuera propia del programa, la pruebe una y otra vez y una vez que esta seguro que la misma funciona correctamente, recién en ese momento, traslade el código de dicha función a la librería**. Puede abrir el proyecto de librería, cortar y pegar el código de la función en el archivo de implementación de la librería y cortar y pegar el código del prototipo en el archivo de cabecera; luego compilar y construir nuevamente la librería (recuerde que cada cambio que realiza en una librería requiere de su re-construcción); volver al proyecto y reusar la librería con la nueva función.

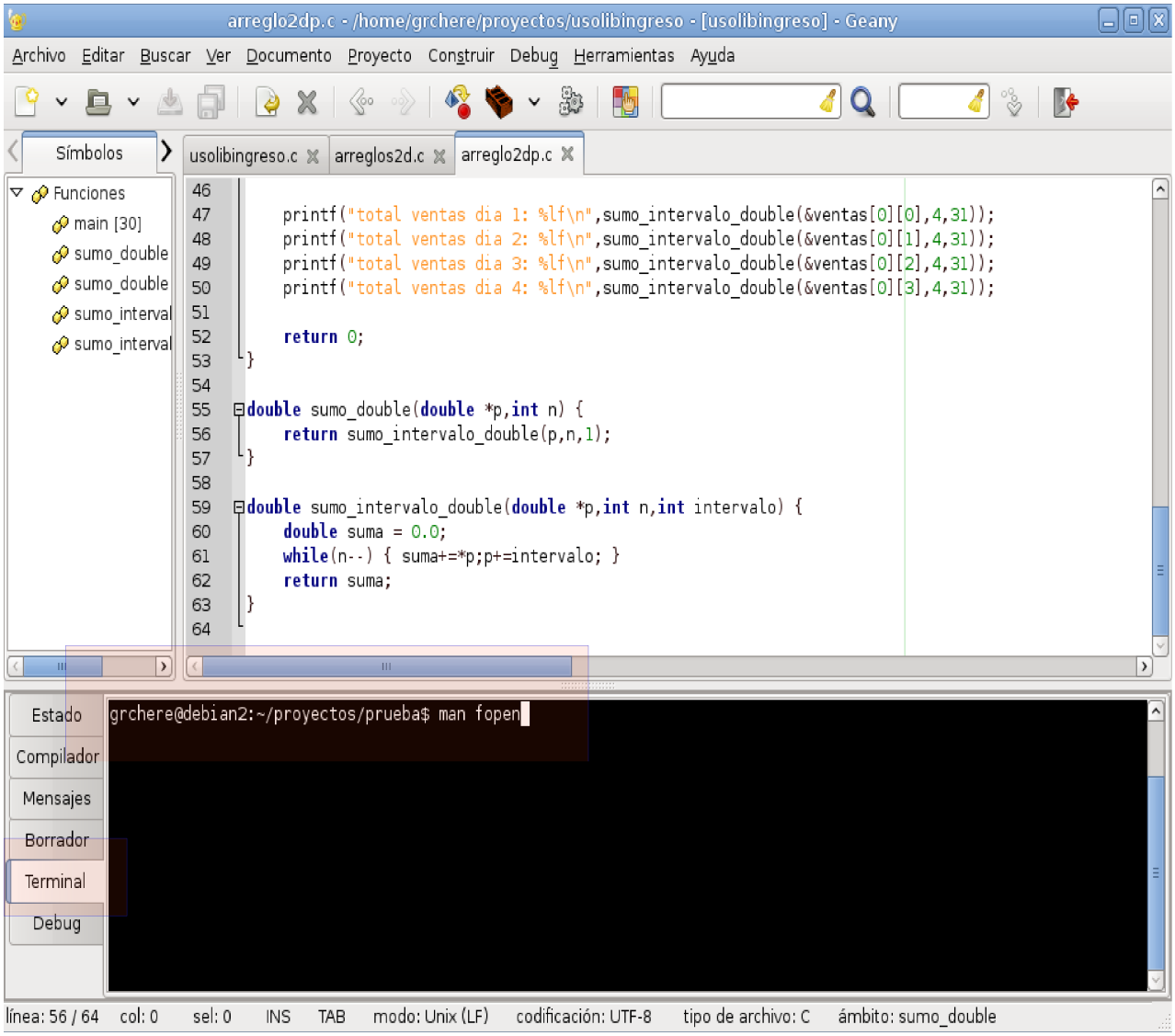
Se supone que **el código que contienen las librerías esta probado y libre de errores**, entonces, la falla sólo puede encontrarse en el código nuevo que incorporamos a nuestro proyecto, de esta forma, construimos nuestros programas sobre bases sólidas y es más fácil encontrar los errores.

En cuanto a cambios en librerías

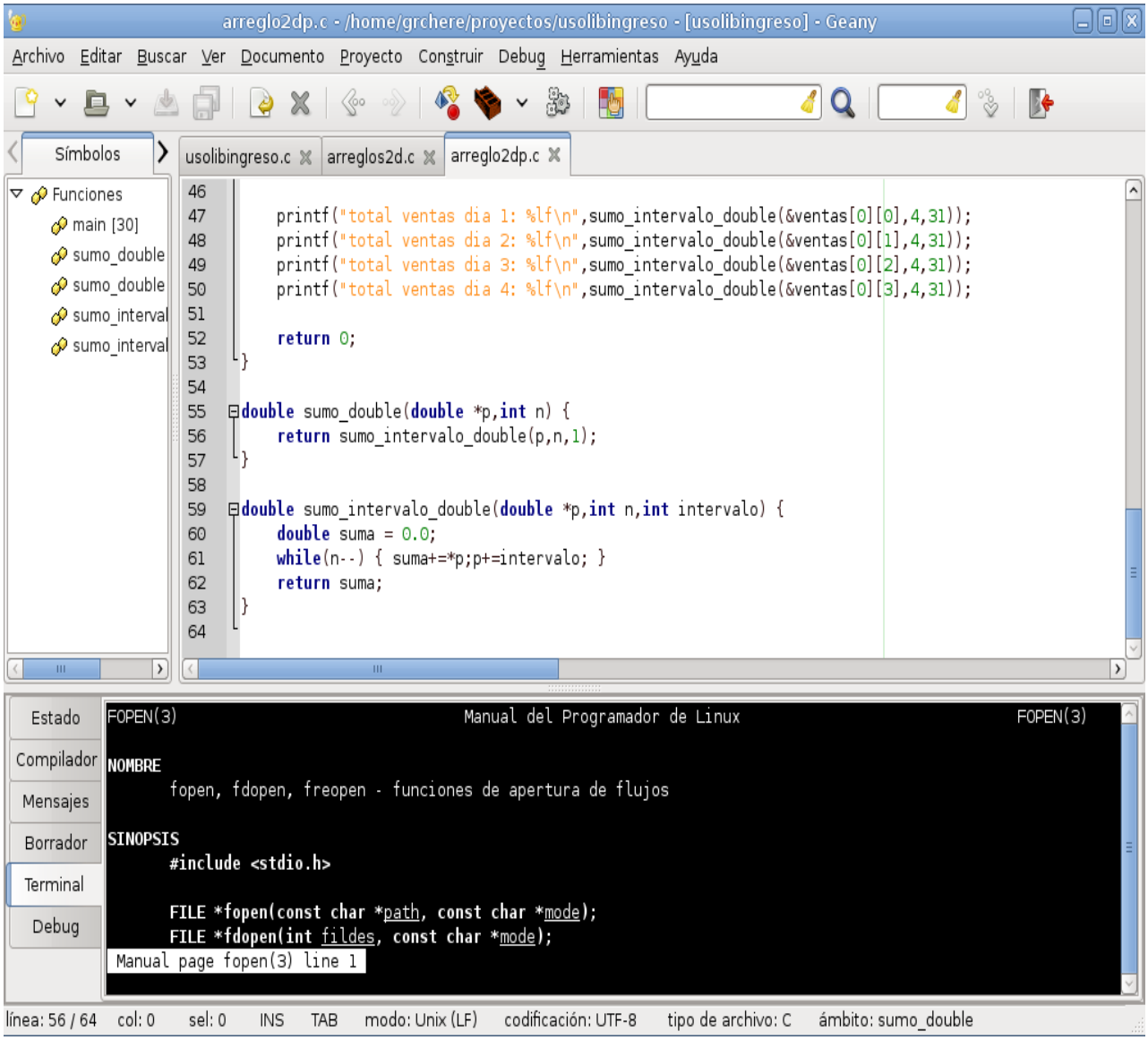
Si puede comprobar una falla en una función de la librería, sería conveniente crear una función local a su programa con otro nombre, probarla y luego trasladar el cambio al proyecto librería, compilar y construir nuevamente la librería y luego volver a su proyecto y probar que la función de librería funciona correctamente.

En cuanto a pedir ayuda sobre función de biblioteca standard

Geany por defecto no muestra ayuda en cuanto a funciones de la biblioteca standard de C, el editor nos muestra su prototipo cuando la utilizamos, pero no muestra una descripción de la misma, ni qué cabeceras requiere, etc. Una forma de obtener esta ayuda es utilizar las páginas del manual del linux (man pages), para hacerlo más fácil, se puede utilizar la consola o terminal incorporada dentro de Geany; por ejemplo, aquí podemos ver como solicitamos ayuda acerca de la función fopen() :



una vez que estamos ejecutando el comando **man** podemos utilizar las flechas, avanzar página, retroceder página para recorrer toda la información relacionada con la función, **para salir, presionar q**.



Atte. Guillermo Cherencio
Programación I
ISFT N° 189