Arquitectura de sistemas

Abelardo Pardo

University of Sydney
School of Electrical and Information Engineering
NSW, 2006, Australia
Autor principal del curso de 2009 a 2012

Iria Estévez Ayres

Damaris Fuentes Lorenzo

Pablo Basanta Val

Pedro J. Muñoz Merino

Hugo A. Parada

Derick Leony

Universidad Carlos III de Madrid
Departamento de Ingeniería Telemática
Avenida Universidad 30, E28911 Leganés (Madrid), España

Tabla de contenidos

12.1. Actividades

12.1.1. Corrección de errores de compilación en C

Recursos

- Los errores y advertencias del compilador
- Fichero `errors1.c`.
- Ficheros `errors2a.c`, `errors2b.c` y `errors2.h`.

Plan de trabajo

1. Abre el documento " Los errores y advertencias del compilador " en el que se explican los mensajes de error más frecuentes con el compilador. Tenlo a mano para consultarlo mientras arreglas los errores de los siguientes programas.
2. Descarga el fichero `errors1.c` y copialo a tu carpeta de trabajo. Compila el programa de forma que se muestren ***todos*** los mensajes de error y advertencias posibles. Guarda los mensajes de error que aparecen en esta primera compilación.

   Busca la forma de arreglar los errores y apunta lo que has tenido que cambiar para cada uno de ellos. Comenta la solución con el profesor si lo necesitas.

   ¿Qué cambio has hecho para que el error "`errors1.c:16: error: 'M_PI' undeclared (first use in this function)`" haya desaparecido?
3. Descarga los ficheros `errors2a.c`, `errors2b.c` y `errors2.h` y copialos a otra carpeta de trabajo. El programa consta de dos ficheros de código y uno de definiciones. Compila el programa.

   Al igual que en el apartado anterior, anota todos los cambios que has hecho para eliminar los mensajes de error y advertencia. Puedes mover definiciones y/o declaraciones entre los tres ficheros. Recuerda también, que si una función no está definida es posible que sea parte de una biblioteca. Compruébalo con el comando `man`.

12.1.2. Preguntas sobre gcc

Plan de trabajo

1. Arranca la máquina virtual y abre una ventana de comandos.

2. Responde individualmente a estas preguntas y contrasta las respuestas con tu compañero de prácticas:
    - ¿Para qué sirve la opción **-o**?
    - Pon un ejemplo de advertencia que sólo se muestra con la opción **-Wall**.
    - ¿Cómo se puede saber el tiempo que tarda en ejecutarse un comando?
    - Si un programa se invoca desde el intérprete con el comando **programa a b c d**, ¿qué valor tiene el primer parámetro de la función `main`?

12.1.3. El preprocesador y las líneas que comienzan por "#"

Recursos

- La máquina virtual.
- Ficheros `main.c` y `main.h` que debes descargarte en tu directorio actual.

Plan de trabajo

- Asegúrate de que entiendes cual es el efecto de las directivas `#include`, `#define` y `#ifdef ... #endif`.
- Responde a las siguientes preguntas y comprueba las respuestas con tu compañero de prácticas antes de la sesión del laboratorio.
    1. ¿Qué diferencia hay entre las directivas `#include <fichero.h>` y `#include "fichero.h"`?
    2. Si el programa `main.c` incluye el fichero `main.h`, ¿es correcto el siguiente comando?

        ```
        $ gcc -o main main.c
        ```

    3. ¿Qué diferencia hay entre la directiva `#define SIMBOLO` y `#define SIMBOLO 10`?
    4. Supongamos que quieres tener dos versiones de un programa que has diseñado tú. La primera versión la utilizas para depurar errores con lo que el programa imprime por pantalla todo tipo de mensajes de comprobación. La segunda versión es para el cliente y esos mensajes no pueden aparecer en pantalla. ¿Cómo conseguirías esto utilizando la directiva `#ifdef`?
- Prepara la máquina virtual para trabajar con un terminal de comandos y el compilador.
- Edita los ficheros `main.c` y `main.h`. Además de la directiva `#ifdef` el programa utiliza la directiva `#ifndef` que simplemente es la negación de la anterior. Crea dos versiones del ejecutable una que imprima `msg1` y no `msg2`, y la otra que imprima los dos mensajes.