Uso y construcción de bibliotecas con el programa *ar*

Eva Lucrecia Gibaja Galindo

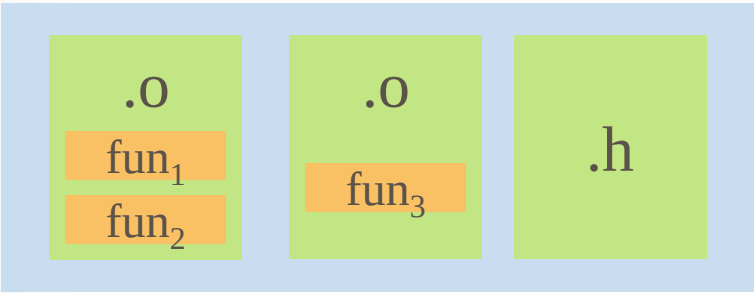
Dpto. Informática y Análisis Numérico

Introducción

- Habitualmente nos encontramos con conjuntos de funciones útiles para diferentes programas.
  - Opción *copy & paste*:
    - El tamaño total del código fuente de los programas se incrementa innecesariamente y es redundante.
    - Para actualizar el código de una función es preciso modificar TODOS los programas que usan esta función
      - Dificultad para controlar versiones.
      - Esfuerzo considerable para mantener la coherencia del software.
  - Solución: Construir una ***biblioteca***.
    - Fichero que contiene código objeto que se enlaza con el código objeto de un módulo que usa una función de esa biblioteca.
      - Sólo existe una versión de cada función. Actualización sencilla.
      - La utilización de *makefiles* reduce el esfuerzo de mantenimiento.

Estructura de una biblioteca

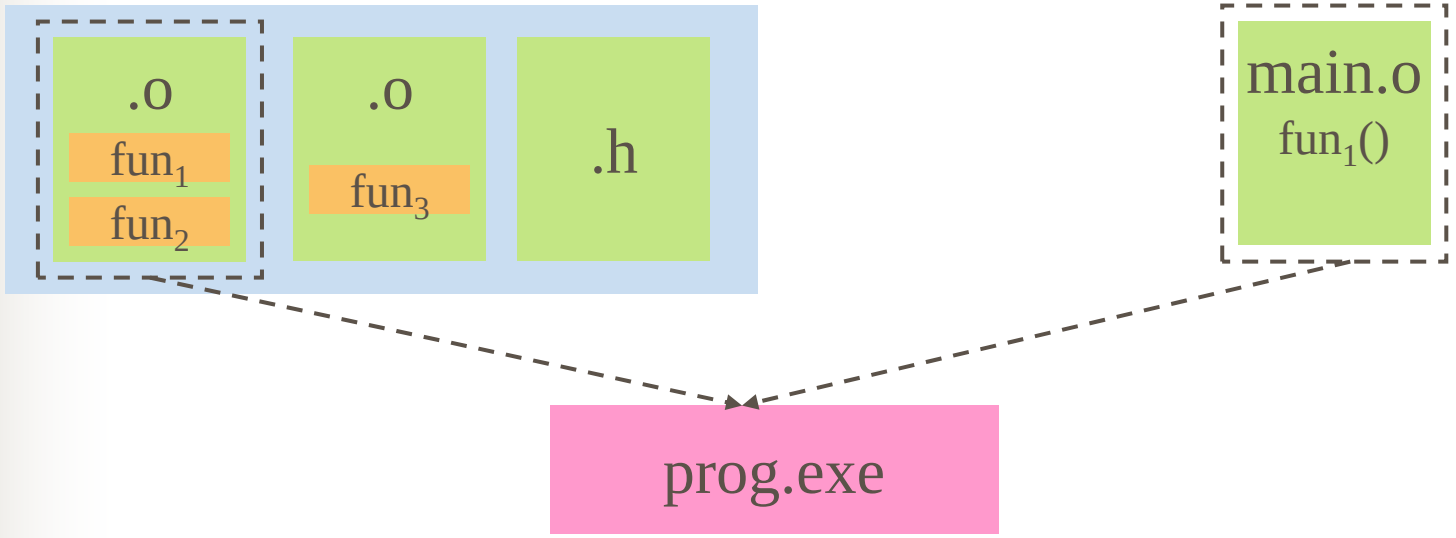
- Conjunto de módulos objeto (.o).
  - Cada uno es el resultado de la compilación de un fichero de código fuente (.c) que puede contener una o varias funciones.
  - La extensión por defecto de los ficheros de biblioteca es “.a” y se su nombre suele empezar por el prefijo “lib”.
- Cada biblioteca lleva asociado un fichero de cabecera con los prototipos de las funciones que se ofrecen (funciones públicas) que actúa de interfaz con los programas que la usan.



Esquema de una biblioteca

Construcción del ejecutable a partir de la biblioteca

- El ***enlazador*** enlaza el módulo objeto que contiene la función *main()* con el módulo objeto (**completo**) en que se encuentra la función de biblioteca usada.
- En el programa ejecutable sólo se incluyen los módulos objeto que contienen alguna función llamada por el programa.



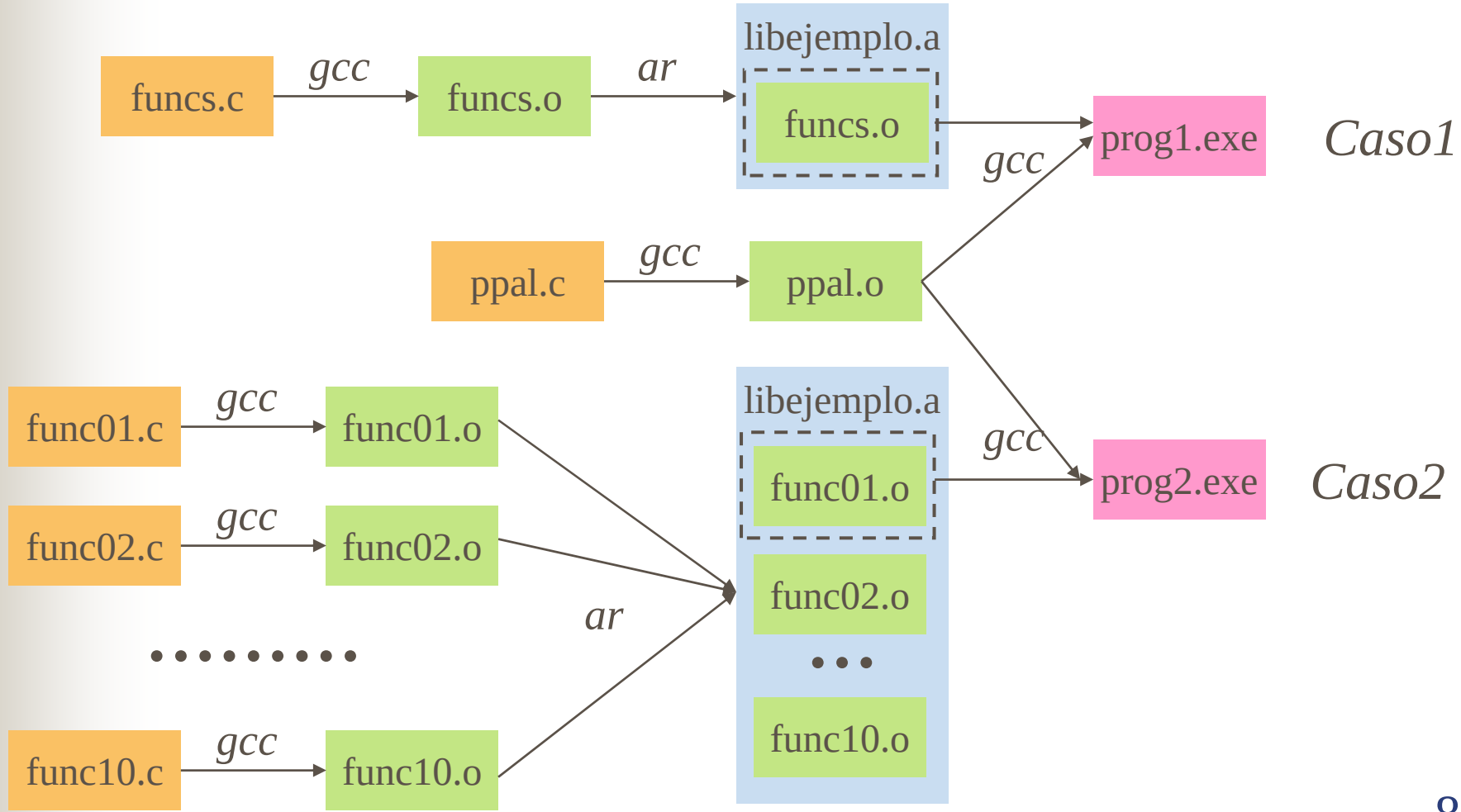
Cuestiones de diseño

- Como diseñador de la biblioteca debemos tener en cuenta la estructura que vamos a adoptar:
  - Si las bibliotecas están formadas por módulos objeto con muchas funciones cada una, los tamaños de los ejecutables serán muy grandes.
  - Si las bibliotecas están formadas por módulos objeto con pocas funciones cada una, los tamaños de los ejecutables serán más pequeños.
- Como usuario de la biblioteca se actúa de la misma manera en ambas situaciones.

Ejemplo

- La biblioteca *libejemplo.a* contiene 10 funciones.
- Los casos extremos de construcción son:
  - **Caso1**. Un único fichero objeto, *funcs.o*, resultado de compilar *funcs.c*.
  - **Caso2**. 10 ficheros objeto (*fun01.o*, *fun02.o*, ...,*fun10.o*) resultado de compilar 10 ficheros fuente (*fun01.c*, *fun02.c*, ...,*fun10.c*) que contienen, cada uno, la definición de una única función.

Ejemplo


funcs.c
gcc
funcs.o
ar
libejemplo.a
funcs.o
prog1.exe
Caso1
gcc
ppal.c
gcc
ppal.o
func01.c
gcc
func01.o
libejemplo.a
gcc
func01.o
prog2.exe
Caso2
func02.c
gcc
func02.o
func02.o
ar
func10.o
func10.c
gcc
func10.o

Gestión de bibliotecas con el programa *ar*

- Este programa permite:
  - Crear bibliotecas.
  - Modificar bibliotecas.
    - Añadir nuevos módulos objeto.
    - Eliminar módulos objeto.
    - Reemplazar módulos objeto por otros más recientes.

Gestión de bibliotecas con el programa *ar*

- *ar [-]operación [modificadores] biblioteca [módulos objeto]*
  - biblioteca: nombre de la biblioteca.
  - módulos objeto: lista de ficheros objeto.
  - operación:
    - ***r***. Adición o reemplazo (si el módulo ya se encuentra en la biblioteca).
    - ***d***. Borrado. Elimina un módulo de la biblioteca.
    - ***x***. Extracción. Copia en un fichero externo el contenido de un fichero objeto de la biblioteca (que queda inalterada).
    - ***t***. Listado de los módulos de la biblioteca.
  - modificadores:
    - "s". Indexación. Actualiza o crea (si no existía previamente) el índice de los módulos que componen la biblioteca.
      - Es necesario para que el enlazador sepa cómo enlazarla.
      - Puede emplearse acompañando a una operación o por si solo.
    - "v" .Verbose. Muestra información sobre la operación realizada.

Ejemplo

- Biblioteca con 5 funciones:
  - suma, resta, divide, muliplica, factorial
- main.c

```
#include <stdio.h>
#include "opers.h"


void main()
{
    printf("%d\n", suma(2,5));
}
```

Ejemplo

Caso 1.

- Todas las funciones en *opers.c*
- Fichero de cabecera, *opers.h*
- Programa principal, *main.c*
- Tamaño final del ejecutable: 78,6 KB (80.546 bytes)

1. Generamos los .o para el main y func.c:
```
gcc -c *.c
```
2. Generamos la biblioteca:
```
ar -rsv libopers.a opers.o
```
3. Generamos el ejecutable:
```
gcc -o main.exe main.o libopers.a
```

El nombre de la librería debe ir al final!!

Ejemplo

En este caso el tamaño del ejecutable es menor

Caso 2.

- Funciones en *suma.c*, *resta.c*, *divide.c*, *mult.c y fact.c*
- Fichero de cabecera, *opers.h*
- Programa principal, *main.c*
- Tamaño final del ejecutable: 78,5 KB (80.451 bytes)

1. Generamos los .o, *suma.c*, *resta.c*, *divide.c*, *mult.c* y *fact.c*:
   ```
   gcc -c *.c
   ```
2. Generamos la biblioteca:
   ```
   ar -rsv libopers.a suma.o resta.o mult.o fact.o divide.o
   ```
3. Generamos el ejecutable:
   ```
   gcc -o main.exe main.o libopers.a
   ```

El nombre de la librería debe ir al final!!