Tema 8: Herramientas

Uso y construcción de bibliotecas con el programa *ar*

Introducción

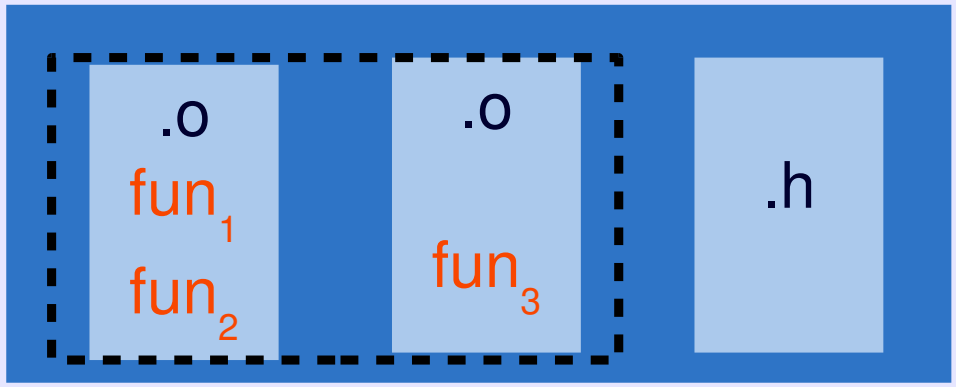
- Al programar, solemos crear funciones que son útiles en muchos programas, de igual o diferente índole.
  - Comprobar si existe un fichero, reservar memoria para un vector, ...
- La opción más común para reutilizar estas funciones es *copy & paste*.
  - El tamaño total del código fuente de los programas 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

Introducción

- Solución:
  - Agrupar las funciones usadas frecuentemente en **módulos biblioteca**.
- **Biblioteca**
  - Fichero que contiene el código objeto de un conjunto de funciones y que puede ser enlazado con el código objeto de un módulo que use una función de esa biblioteca.
    - Sólo existe una versión de cada función. Actualización sencilla
      - Recompilar el módulo donde está esa función y los módulos que la usan
    - La utilización de *makefiles* reduce el esfuerzo de mantenimiento

Estructura de una biblioteca

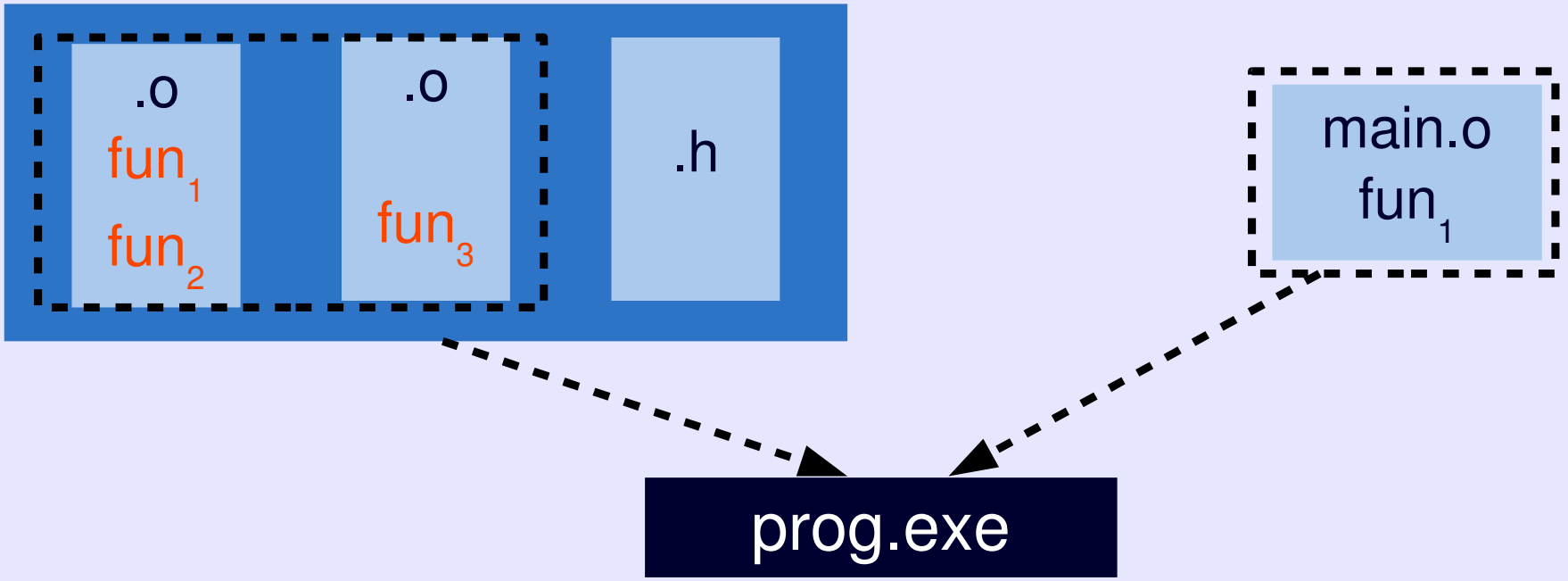
- Conjunto de módulos objeto (.o).
  - Cada uno es el resultado de la compilación de un fichero de código fuente (.c) que puede contener una o varias funciones.
  - La extensión por defecto de los ficheros de biblioteca es “.a” y su nombre suele empezar por “lib”.
- Cada biblioteca lleva asociado un 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 de ejecutables utilizando una 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

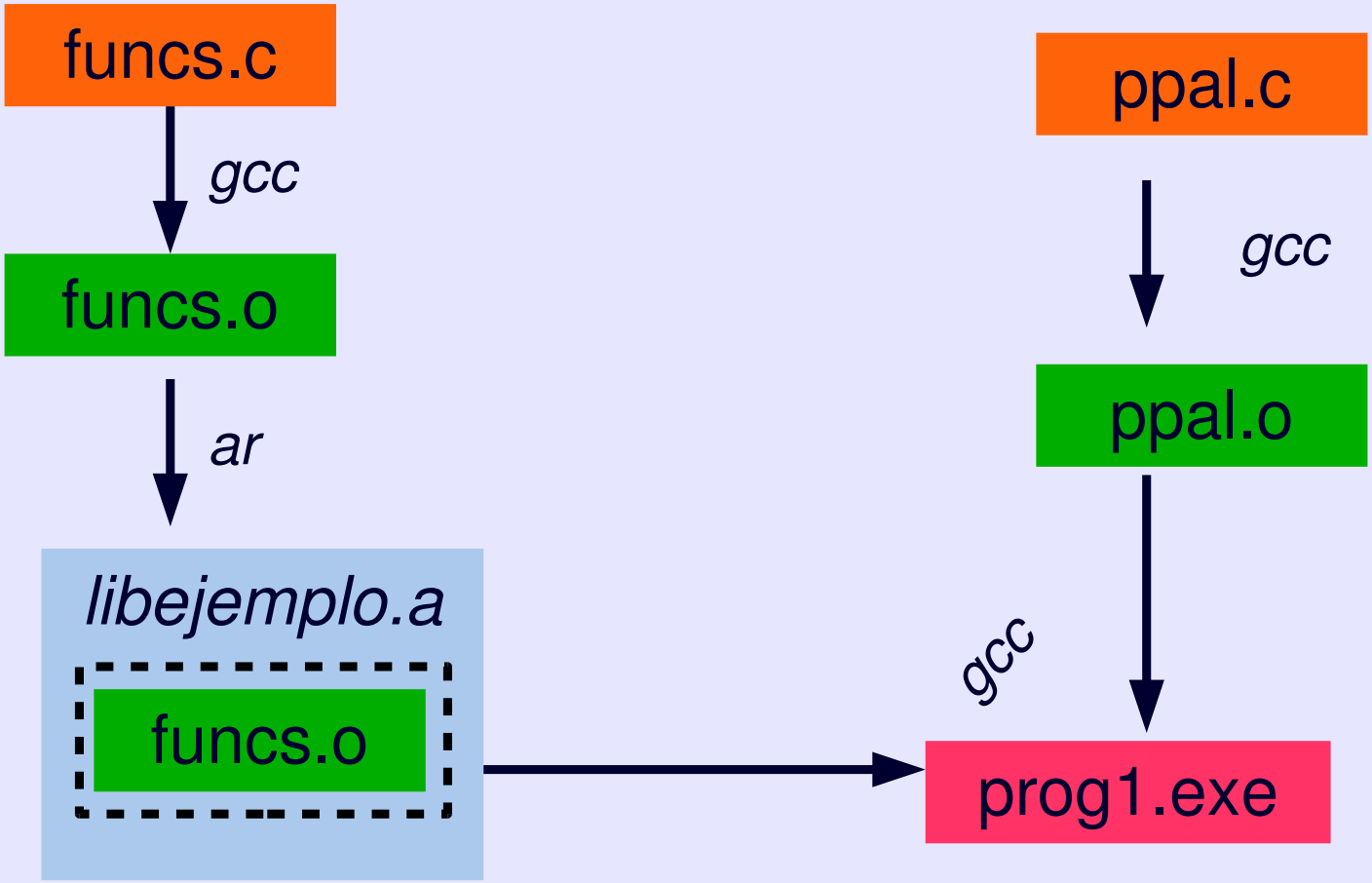
- Como diseñador de la biblioteca debemos tener en cuenta la estructura que vamos a adoptar:
  - Si las bibliotecas están formadas por módulos objeto con muchas funciones cada uno, los tamaños de los ejecutables serán muy grandes.
  - Si las bibliotecas están formadas por módulos objetos con pocas funciones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

Ejemplo

- La biblioteca *libejemplo.a* contiene 10 funciones.
- Los casos extremos de construcción son:
  - **Caso 1**: Un único fichero objeto, *funcs.o*, resultado de compilar *funcs.c*
    - *El módulo objeto "ppal.o" se enlaza con el módulo objeto "funcs.o" para generar el ejecutable "prog_1". El módulo "funcs.o" contiene el código objeto de todas las funciones, por lo que se enlaza mucho código que no se usa.*

Ejemplo

- ***Caso 1***

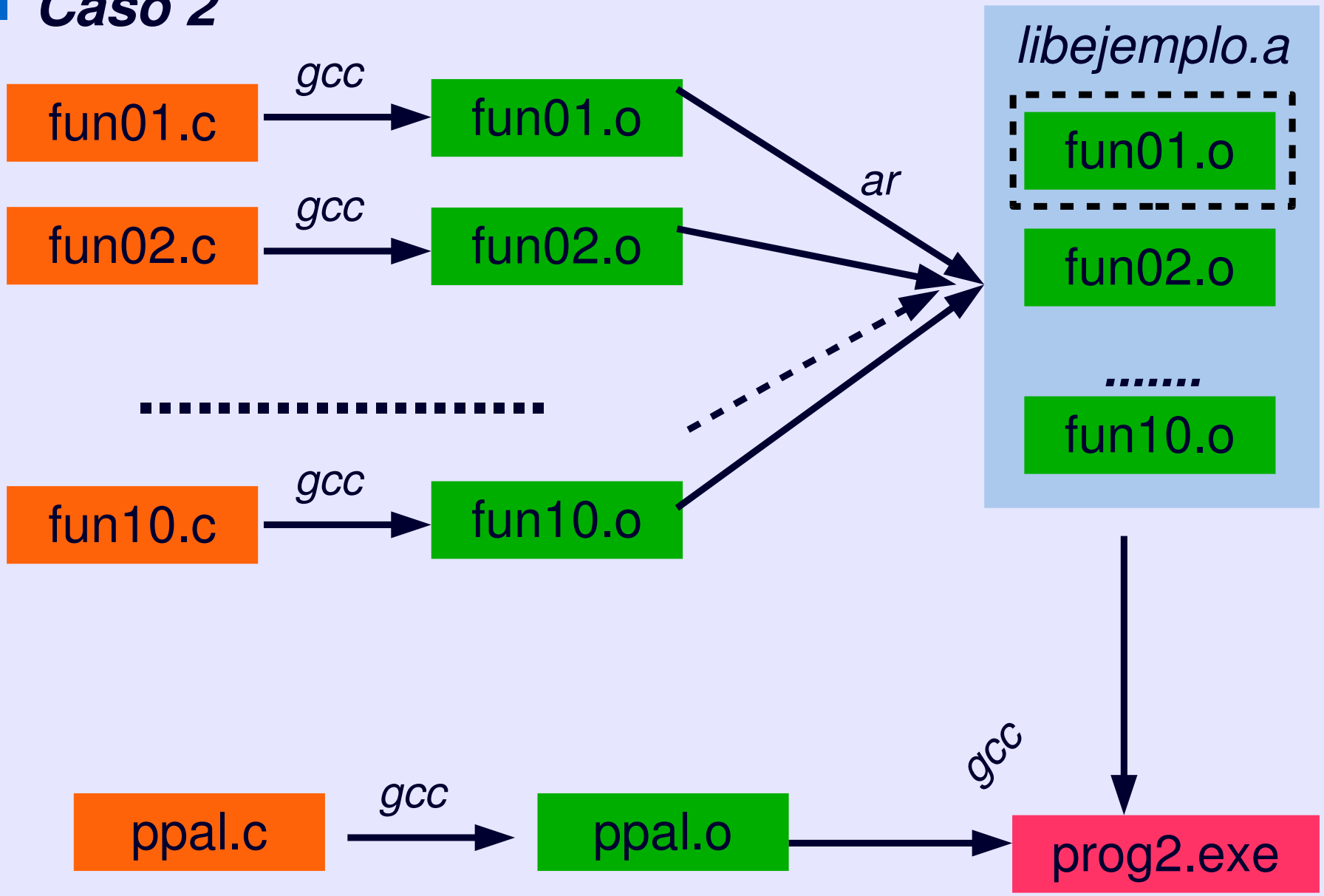


Ejemplo

- **Caso 2**: Diez ficheros objetos (*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.
  - *El módulo objeto "ppal.o" se enlaza con el módulo objeto "fun01.o" para generar el ejecutable "prog_2". El módulo "fun01.o" contiene únicamente el código objeto de la función que se usa, por lo que se enlaza el código estrictamente necesario.*

Ejemplo

- ***Caso 2***



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

ar [-]operacion[modificadores] biblioteca [módulos objeto]

- *Biblioteca*: nombre de la biblioteca
- *Módulos objeto*: lista de ficheros objeto

Gestión de bibliotecas con el programa *ar*

- *Operación*:
  - ***r***: adición o reemplazo (si el módulo ya está 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 todos los módulos objeto
- *Modificadores*
  - ***s***: indexación. Actualiza o crea (si no existía previamente) el índice de los módulos que componen la biblioteca
    - Es necesario para que se pueda enlazar
    - Puede emplearse acompañando a una operación o sólo
  - ***v***: verbose. Muestra información sobre la operación realizada

Gestión de bibliotecas con el programa *ar*

- Caso 1:
  - ar -r libejemplo.a funcs.o
- Caso 2:
  - ar -r libejemplo.a fun01.o fun02.0 ... fun10.o

Ejemplo

- Biblioteca con 5 funciones
  - suma, resta, producto, divisionEntera, factorial

operaciones.h

```
//Prototipos de las funciones
int suma(int a, int b);
int resta(int a, int b);
int producto(int a, int b);
int divisionEntera(int a, int b);
int factorial(int a);
```

main.c

```
#include "operaciones.h"

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

Ejemplo

```
/* ------------------------------------
   Funcion que resta dos numeros
   Se le pasa:a,b -> numeros a restar
   Devuelve:la resta de los dos numeros
   ------------------------------------*/
int resta(int a, int b){
  return (a+b);
}
```

```
/* ------------------------------------
   Funcion que suma dos numeros
   Se le pasa:a,b -> numeros a sumar
   Devuelve:la suma de los dos numeros
   ------------------------------------*/
int suma(int a, int b){
  return (a+b);
}
```

Ejemplo

```
/* ----------------------------------
   Funcion que multiplica dos numeros
   Se le pasa:a,b -> numeros a multiplicar
   Devuelve:el producto de los dos numeros
   ----------------------------------*/
int producto(int a, int b){
  return (a*b);
}
```

```
/* ----------------------------------
   Funcion que divide dos numeros
   Se le pasa:a,b -> numeros a dividir
   Devuelve:la división entera de los 2 numeros
   ----------------------------------*/
int divisionEntera(int a, int b){
  return (a/b);
}
```

Ejemplo

```
/*-------------------------------------
   Funcion que calcula el factorial de un número
   Se le pasa: a -> numero
   Devuelve: el factorial de a
  -------------------------------------*/
int factorial(int a){
  if (a == 0)
     return 1;
  else
     return(a*factorial(a-1));
}
```

Ejemplo

- ***Caso 1***
  - Todas las funciones en *operaciones.c*
  - Fichero de cabecera *operaciones.h*
  - Programa principal *main.c*
- Creación de los ficheros objeto
  - `gcc -c operaciones.c`
  - `gcc -c main.c`
- Construcción de la librería
  - `ar -rsv libOperaciones1.a operaciones.o`
- Creación del ejecutable
  - `gcc -o main1.exe main.o libOperaciones1.a`

Ejemplo

- ***Caso 2***
  - Las funciones en *resta.c, suma.c, producto.c, divisonEntera.c y factorial.c*
  - Fichero de cabecera *operaciones.h*
  - Programa principal *main.c*
- Creación de los ficheros objeto
  - `gcc -c suma.c`
  - `gcc -c resta.c`
  - `gcc -c producto.c`
  - `gcc -c divisionEntera.c`
  - `gcc -c factorial.c`
  - `gcc -c main.c`

Ejemplo

- Construcción de la librería
  - `ar -rsv libOperaciones2.a suma.o reta.o producto.o factorial.o divisionEntera.o`
- Creación del ejecutable
  - `gcc -o main2.exe main.o libOperaciones2.a`
- Tamaño final de los ejecutables
  - Caso 1:
    - *9.3 KB (9464 bytes)*
  - Caso 2:
    - *8.9 KB (9017 bytes)*

El nombre de la librería debe ir al final