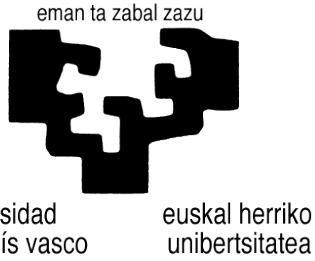
UPV / EHU



Programación Concurrente en Linux

Historia de un programa

OCW OpenCourseWare

UPV / EHU

Contenido

1. Compilación y montaje de un programa
2. Resolución de las direcciones del programa
3. Gestión de la carga de programas en memoria

OCW OpenCourseWare

UPV / EHU

1. Compilación y montaje de un programa

OCW OpenCourseWare

Del código fuente a la memoria

- Un compilador como *gcc* lleva a cabo varios pasos:
  1. Preproceso: resuelve definiciones y macros (*#define*) e incluye las bibliotecas fuente (*#include*).
  2. Compilación propiamente dicha: genera módulos objetos. Puede hacerse por separado.
  3. Montaje de los diferentes módulos: genera el fichero ejecutable.
- La carga del programa ejecutable en memoria se hace normalmente cuando se va a ejecutar (por ejemplo desde el *shell* en Linux).

Módulos fuente

mi_proyecto.c

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

struct mi_struct v;

main()
{
  int i;
  …
}
```

funciones.h

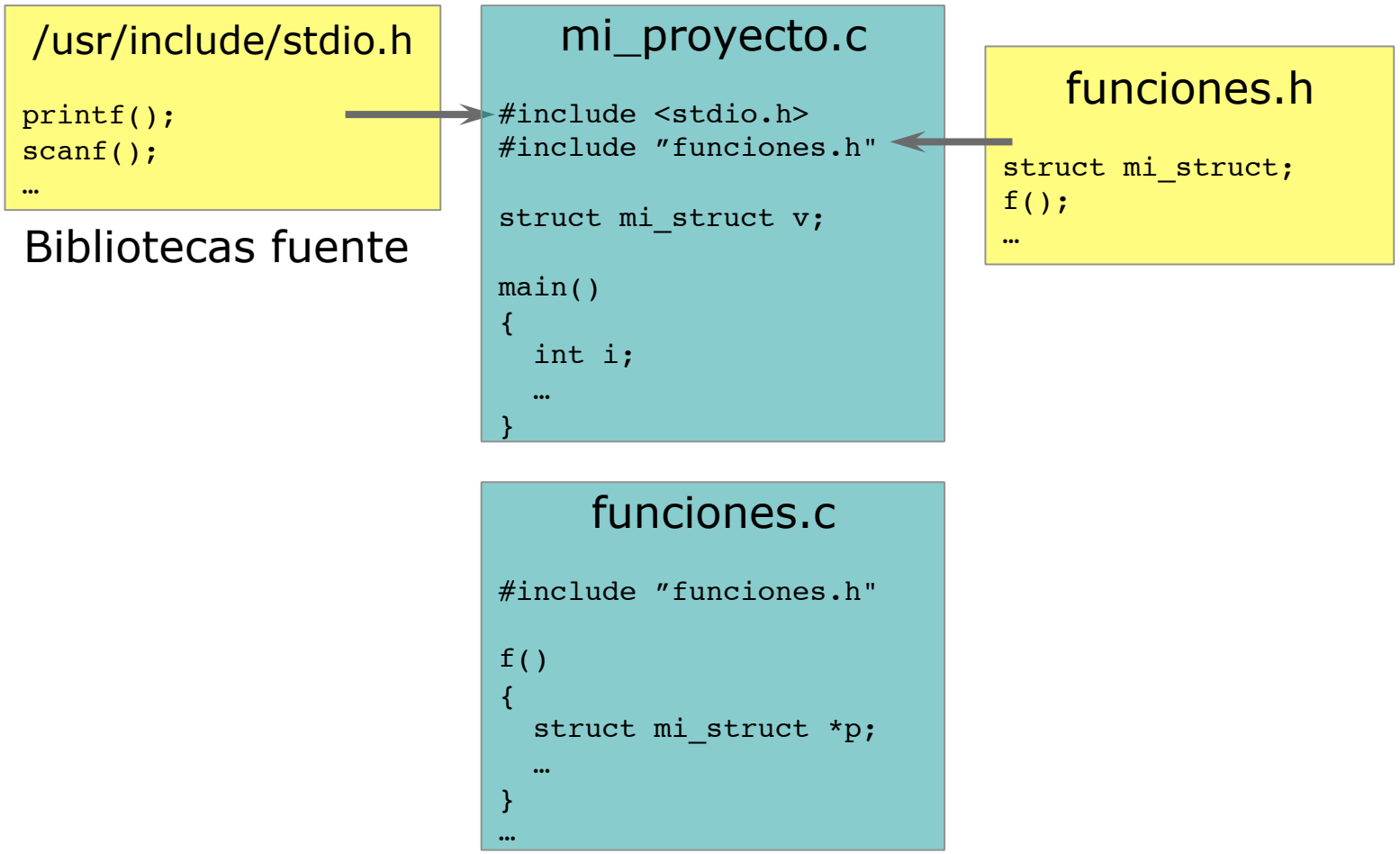
```
struct mi_struct;
f();
…
```

funciones.c

```
#include "funciones.h"

f()
{
  struct mi_struct *p;
  …
}
…
```

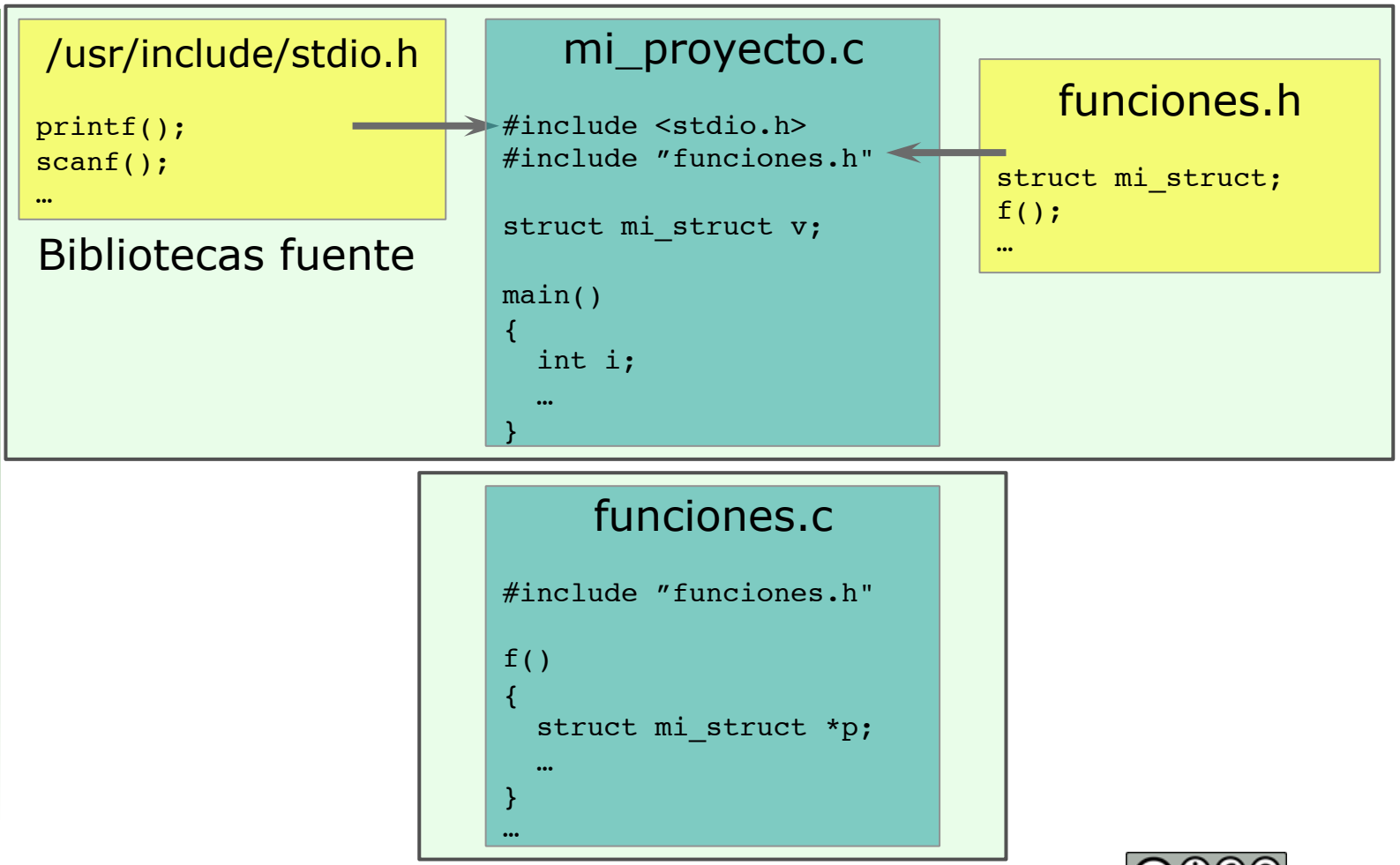
Módulos fuente



1. Preproceso

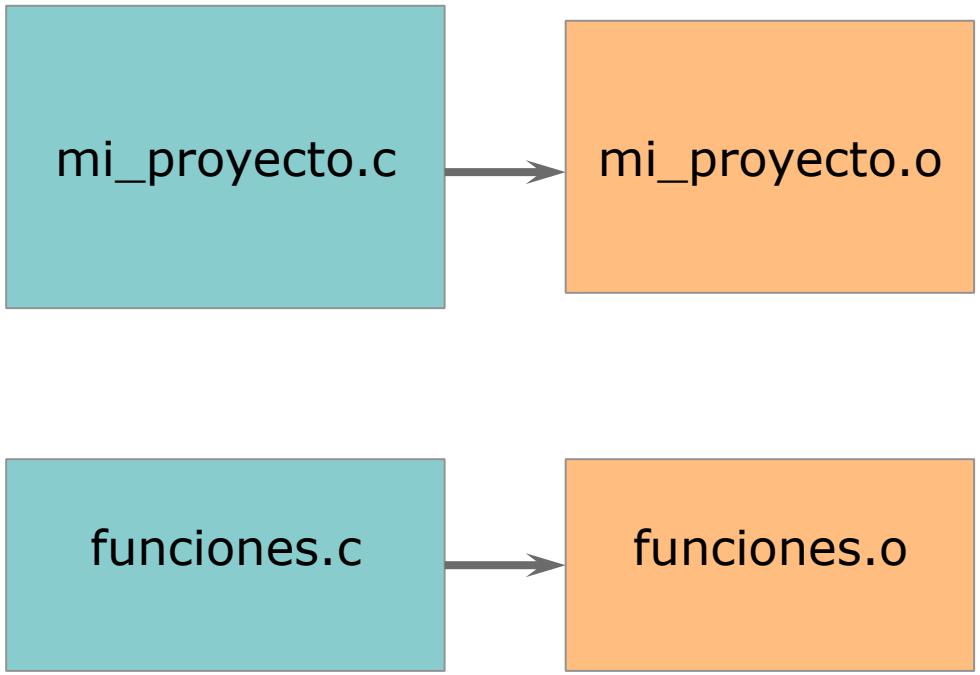
gcc -o mi_proyecto mi_proyecto.c funciones.c

(Paso 1: preproceso)

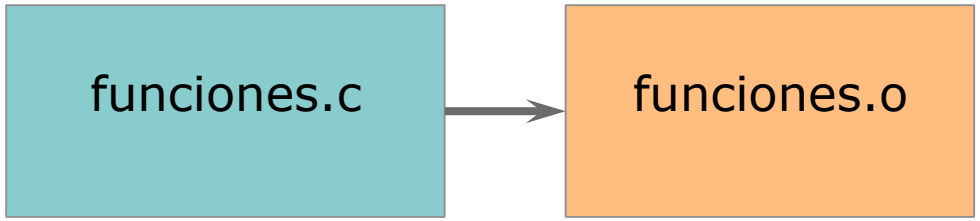


2. Compilación y módulos objeto

`gcc -o mi_proyecto mi_proyecto.c funciones.c`
(Paso 2: compilación)



Alternativa: compilación separada



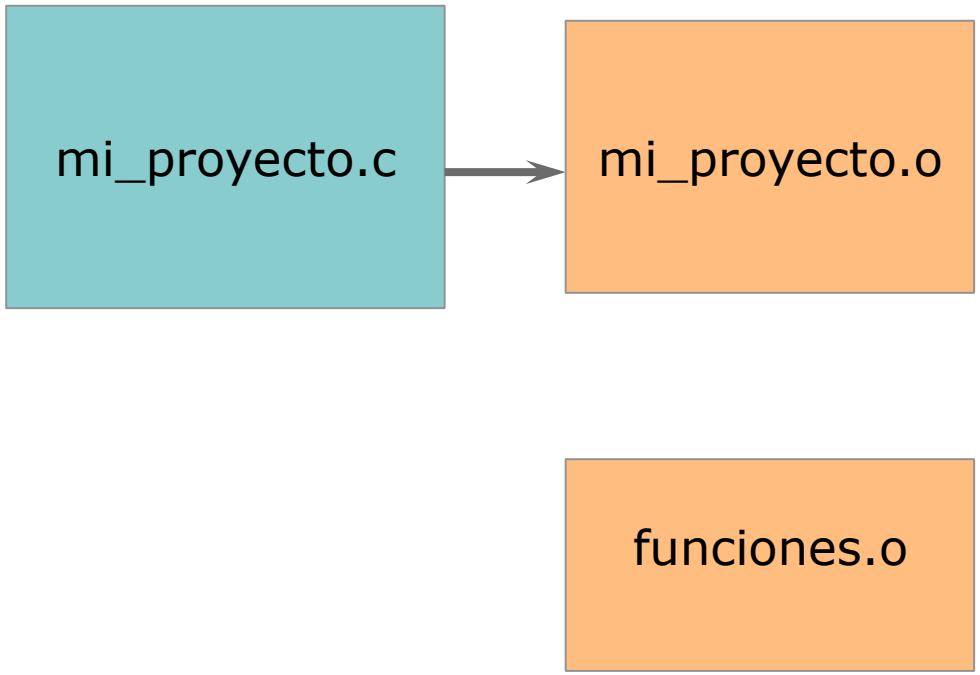
`gcc –c funciones.c`

(Paso 2a)

Alternativa: compilación separada

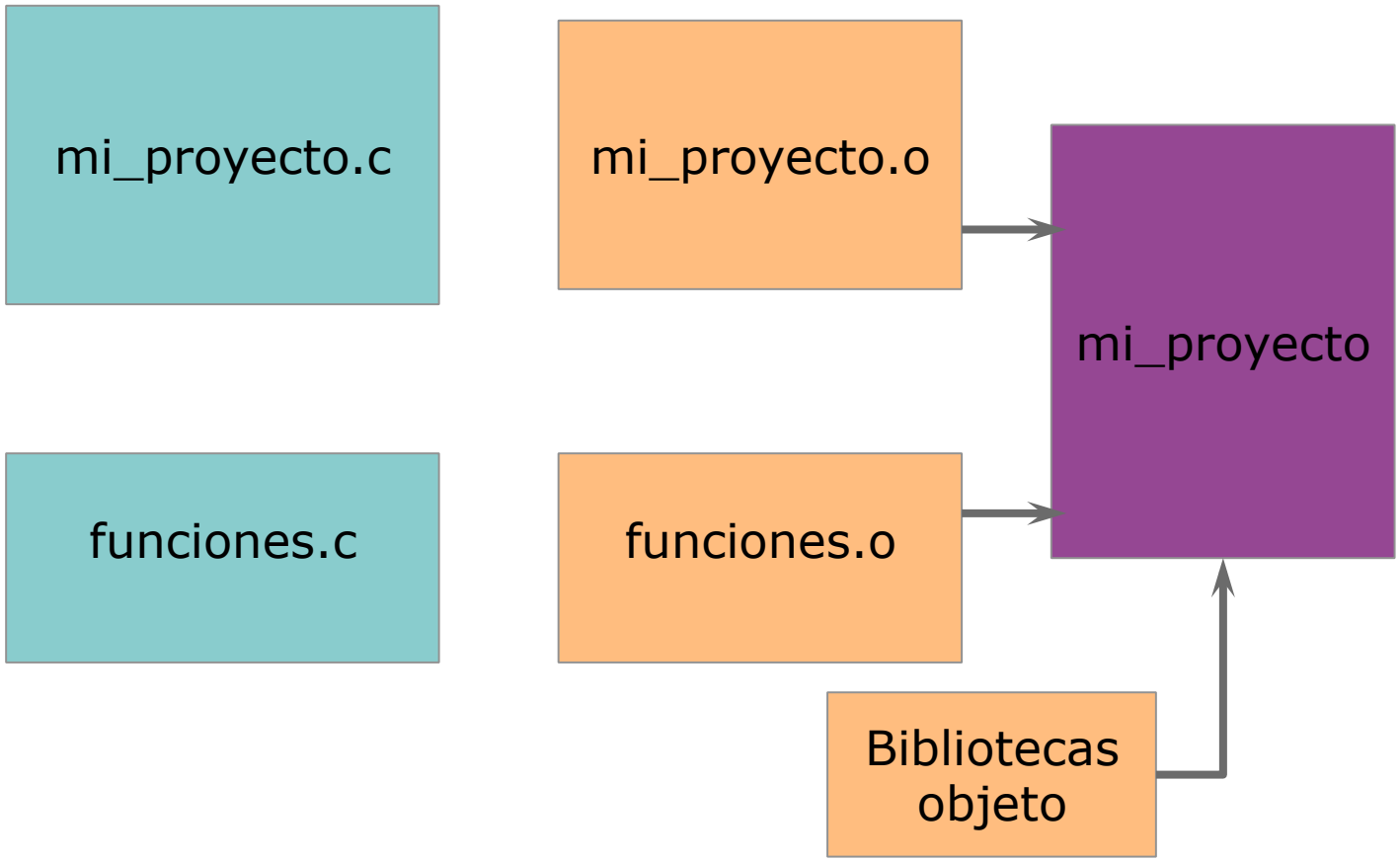
`gcc -o mi_proyecto mi_proyecto.c funciones.o`

(Paso 2b)



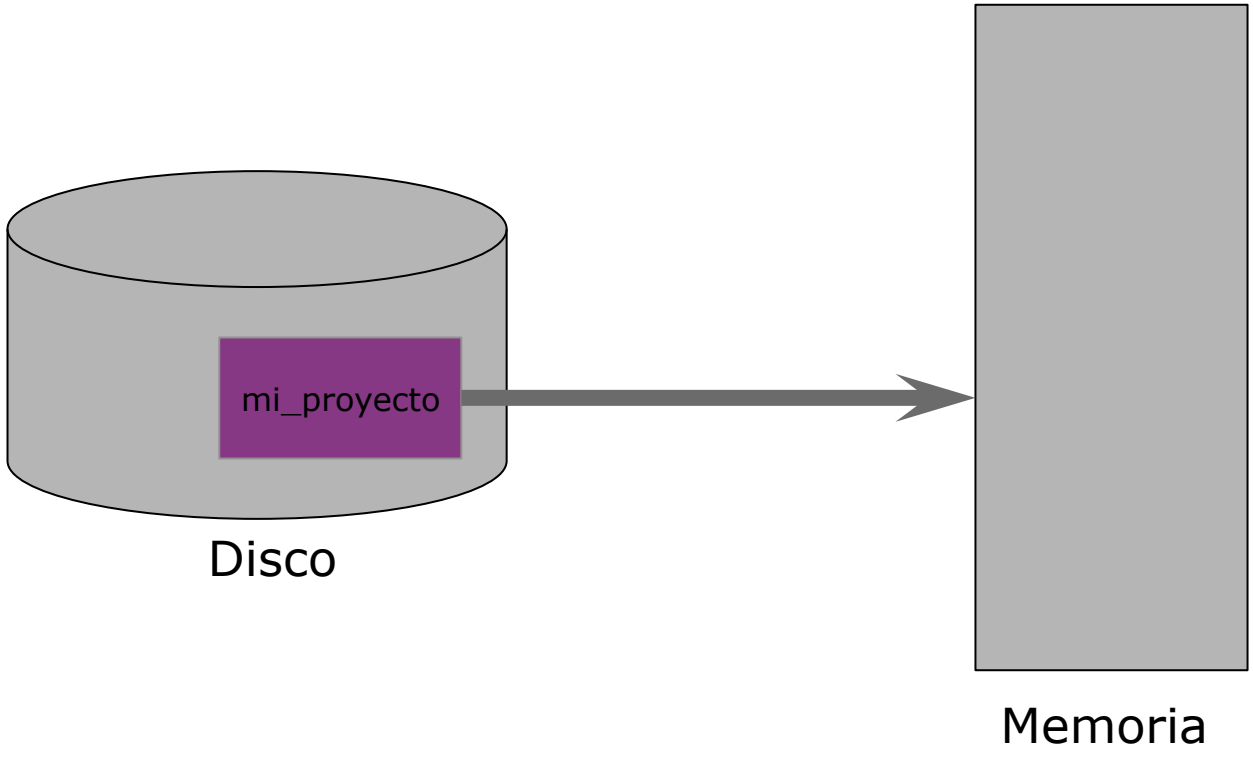
3. Montaje de los módulos en el ejecutable

`gcc -o mi_proyecto mi_proyecto.c funciones.c`
(Paso 3: montaje)



Ejecutable en el disco (listo para cargar)

`./mi_proyecto`



UPV / EHU

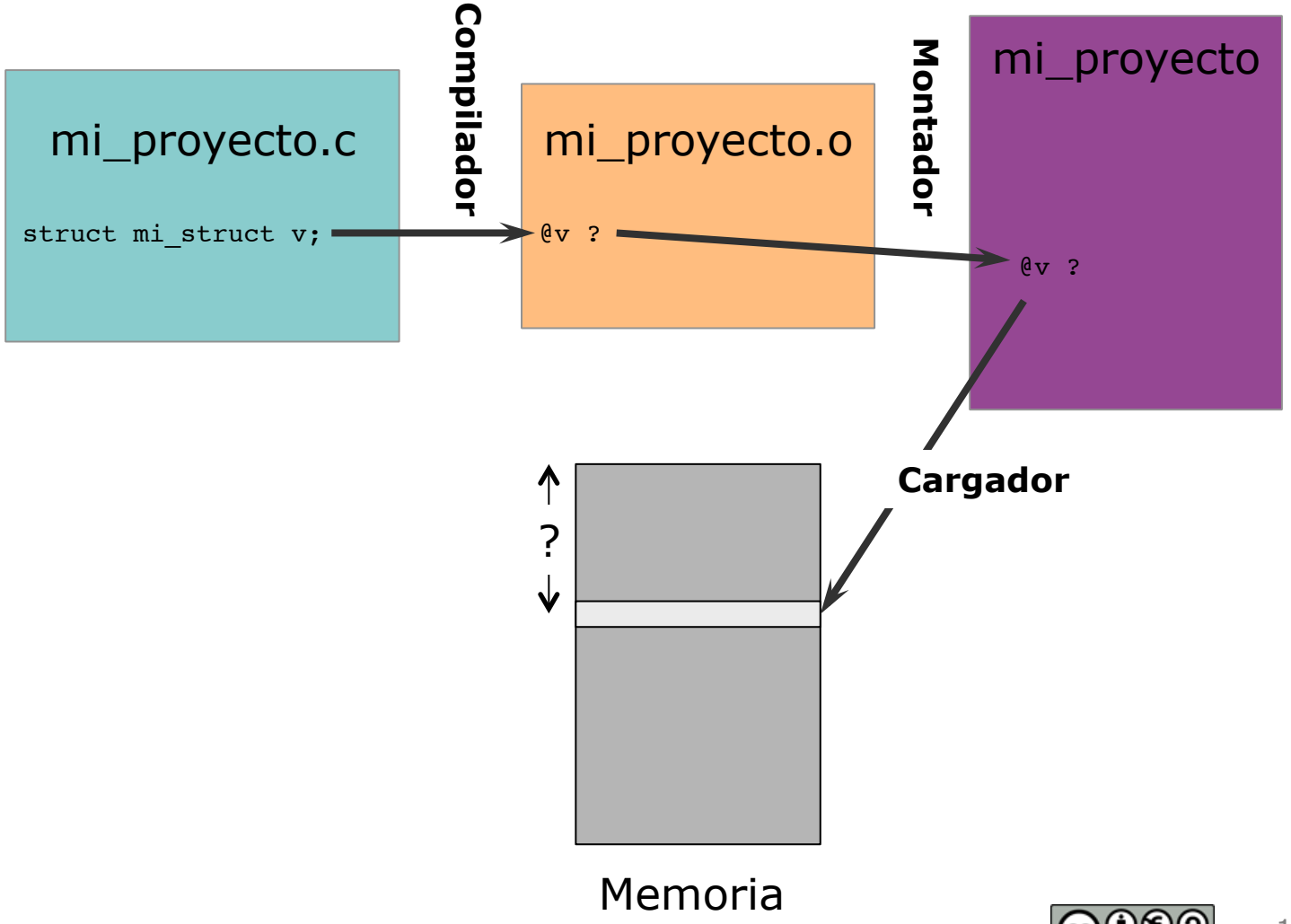
2. Resolución de las direcciones del programa

OCW OpenCourseWare

Resolución de direcciones

- ¿En qué direcciones de memoria se cargará el programa?
  - ¿Quién asigna (*resuelve*) esas direcciones?
    - ¿el compilador?
    - ¿el montador?
    - ¿el cargador?
    - ¿?

Quién resuelve las direcciones



Quién resuelve las direcciones

- Si el compilador…
  - x No puede haber compilación separada.
  - x El programa siempre se cargará en el mismo sitio: difícilmente puede haber más de un programa en memoria.
- Si el montador…
  - ✓ Puede haber compilación separada.
  - x El programa siempre se cargará en el mismo sitio: difícilmente puede haber más de un programa en memoria.
- Si el cargador…
  - ✓ Puede haber compilación separada.
  - ✓ El programa se carga en un sitio libre: puede haber más de un programa en memoria.
  - Se habla entonces de *reubicación dinámica*

Reubicación dinámica

- La resolución de direcciones ha de hacerse al cargar el programa (o después…).
- Todas las direcciones del programa serán relativas a un *Registro Base*.
- El cargador busca una dirección de carga y asigna un valor al Registro Base.
- El sistema operativo debe gestionar el estado de la memoria para decidir dónde carga el programa.

Resolución de direcciones Resumen

- Las direcciones que genera un programa son *direcciones lógicas* dentro del programa:
  - La primera dirección del programa es la cero.
- Cuando el programa está cargado en memoria ocupa *direcciones físicas*
  - Hay una correspondencia (@lógica, @física)

    Ejemplo: @física = R_{base} + @lógica
- En reubicación dinámica esta *traducción* se realiza en el momento de acceder a memoria.

UPV / EHU

3. Gestión de la carga de programas en memoria

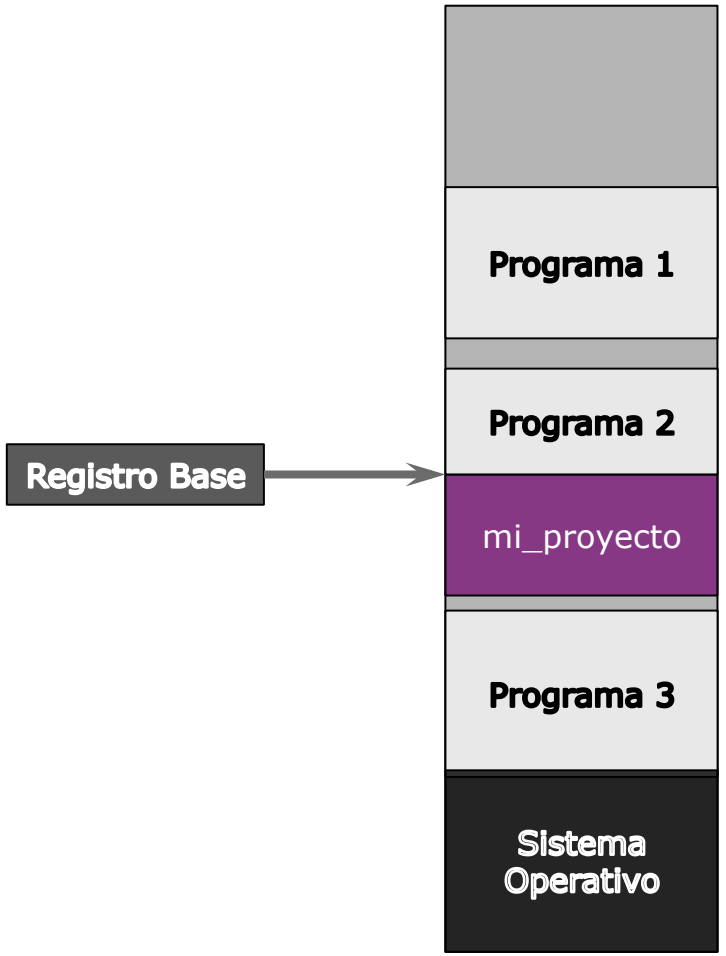
OCW OpenCourseWare

Carga del programa en memoria

- ¿Cómo se carga el programa en memoria?
  - ¿Permanece siempre en el mismo sitio o puede cambiar?
  - ¿En posiciones contiguas de memoria o a trozos?
  - ¿Tiene que estar cargado entero para ejecutarse?

Dónde cargar el programa

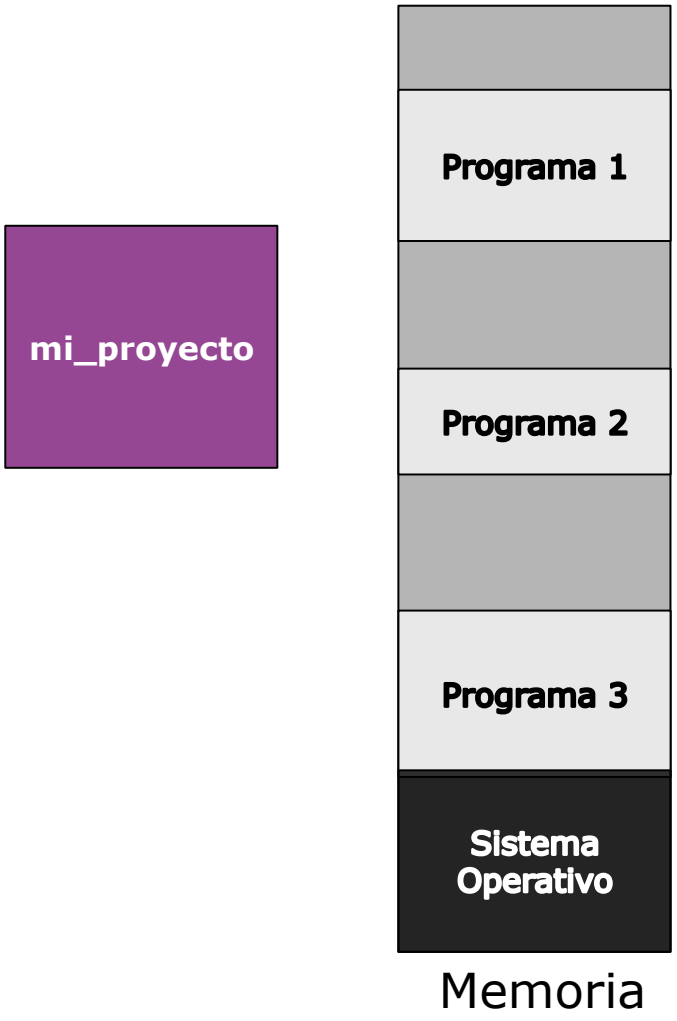
Dónde cargar el programa



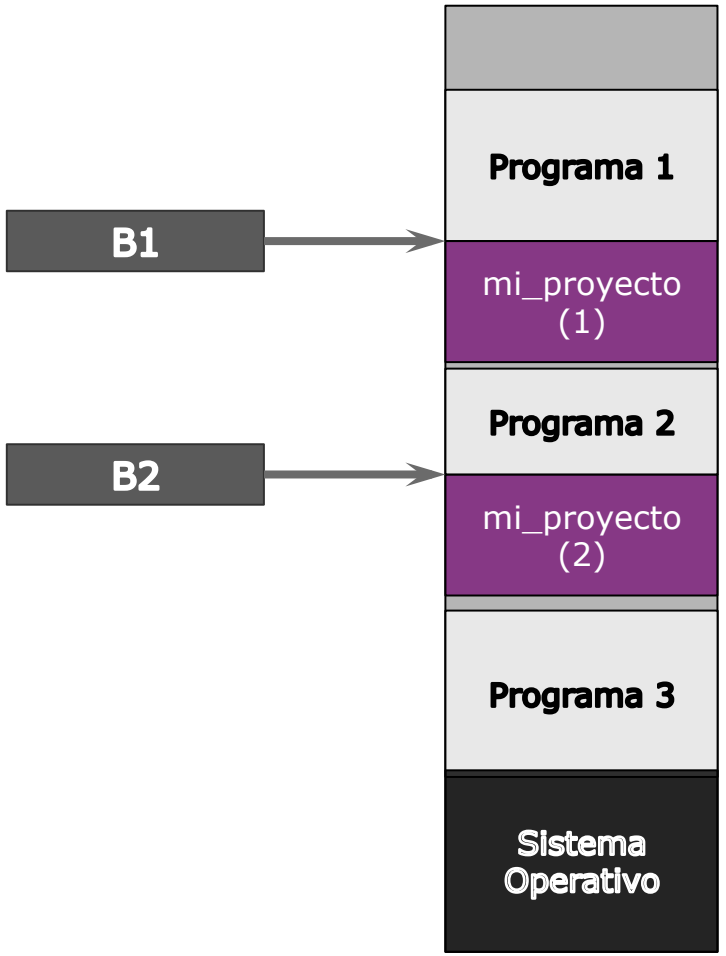
Gestión de la memoria

- ¿Qué pasa si no hay un hueco suficientemente grande en memoria? Alternativas:
  A. Se espera a que acabe otro.
  B. Se saca otro programa (menos prioritario) para ejecutar el nuestro (luego hay que volver a cargar el otro).
  C. Se carga no contiguo en los huecos disponibles.
  D. Se carga un trozo del programa para que vaya ejecutándose; luego se van cargando otros trozos según se requiera.

Carga no contigua



Carga no contigua

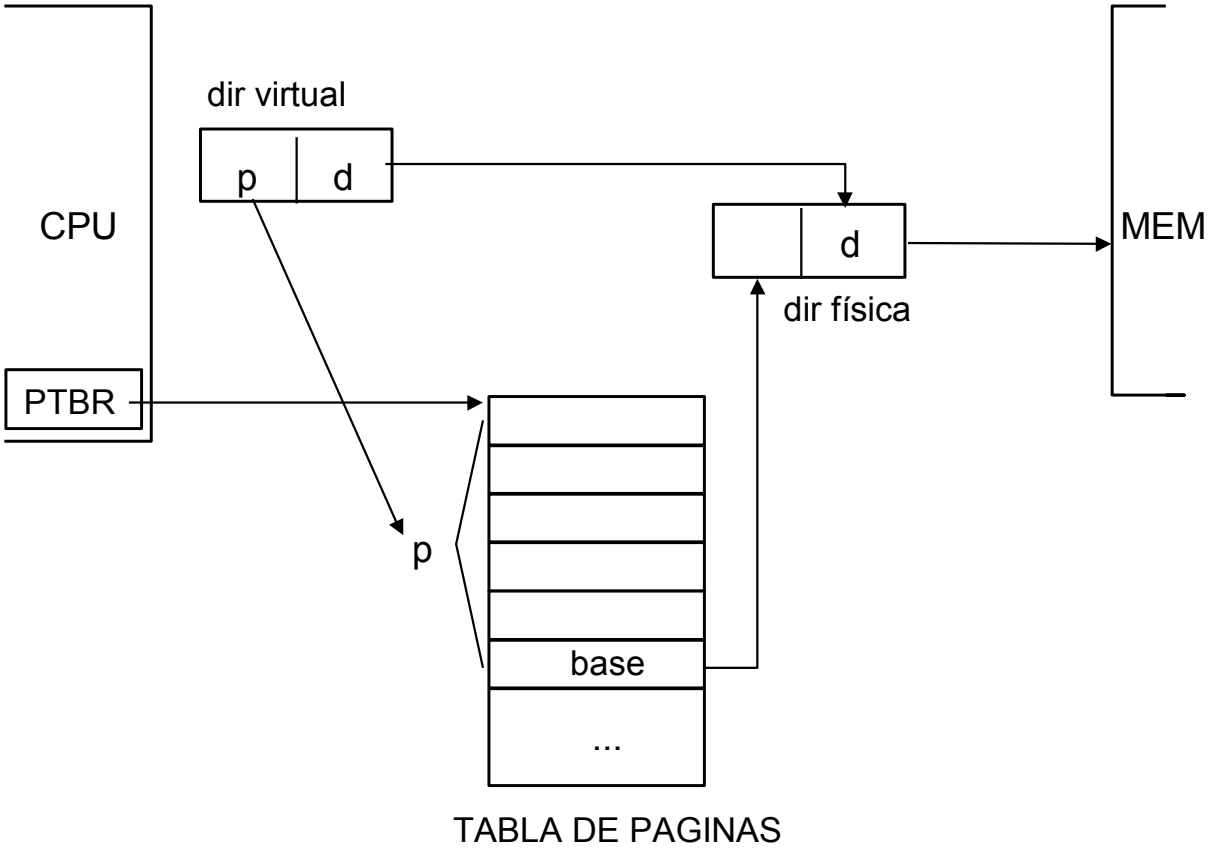


Memoria

Carga no contigua: Paginación

- El programa se divide en trozos del mismo tamaño: *páginas*
- La memoria física (*paginada*) deba dividirse en trozos de ese tamaño: *marcos de página*
- Las direcciones lógicas (*virtuales*) tienen una estructura:

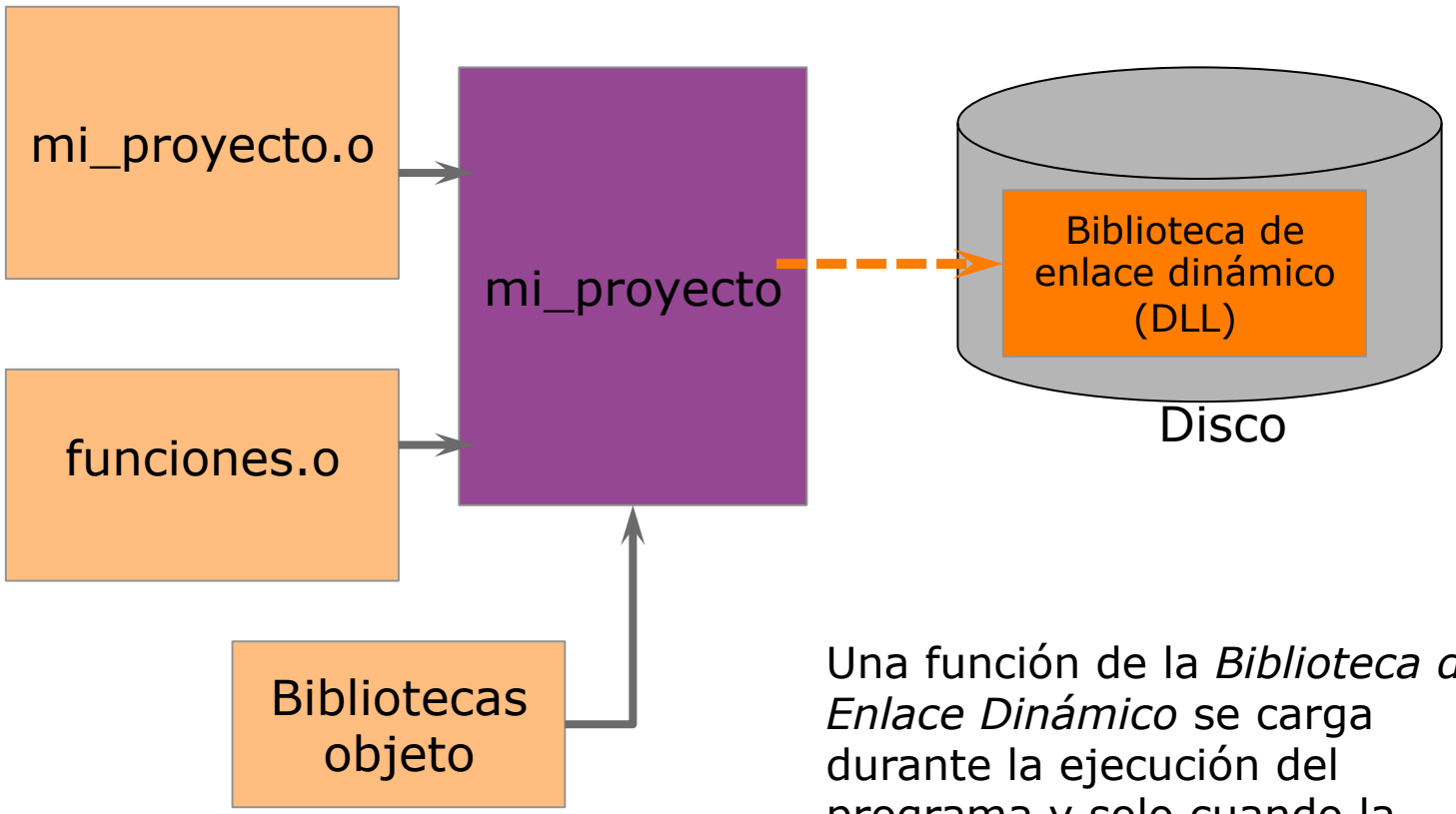
Carga no contigua: Paginación



Carga de programas no enteros: Memoria virtual

- Se basa en memoria paginada.
- Ahora una página del programa puede estar cargada en memoria o no.
- Si está cargada, se accede mediante el mecanismo de memoria paginada.
- Si no, se genera un trap (interrupción interna) para cargarla del disco.
- La Memoria Virtual es impensable para Tiempo Real: no es posible acotar el tiempo de acceder una posición de memoria.

Carga de programas no enteros: Bibliotecas de enlace dinámico



Una función de la *Biblioteca de Enlace Dinámico* se carga durante la ejecución del programa y solo cuando la necesita.