Programación I

Estructura de un programa

Iván Cantador
Escuela Politécnica Superior
Universidad Autónoma de Madrid

Contenidos

- Archivos de cabecera y archivos fuente
- Compilación y enlazado de un proyecto: Makefile

Contenidos

- **Archivos de cabecera y archivos fuente**
- Compilación y enlazado de un proyecto: Makefile

Archivos de cabecera y archivos fuente (I)

- Un programa C consta de:
  - **Archivos de cabecera .h**
    - Macros: #define
    - Definición de tipos de datos: typedef
    - Declaración de funciones públicas
  - **Archivos fuente .c**
    - Definición de funciones públicas
    - Definición de funciones privadas

Archivos de cabecera y archivos fuente (II)


```
/* ========================================================================
   libro.h
   ======================================================================== */

#ifndef LIBRO_H
#define LIBRO_H

// Definicion de las estructuras de datos Autor y Libro
typedef struct {
    char *nombre;
    char *apellidos;
} Autor;

typedef struct {
    char *titulo;
    Autor *autor;
    int anio;
} Libro;

// Declaración de las funciones primitivas de Libro
Libro *libroCrear(char *tituloLibro, int anioLibro, char *nombreAutor, char *apellidosAutor);
void libroLiberar(Libro *pl);

#endif
```


Archivos de cabecera y archivos fuente (III)


```
/* ========================================================================
   libro.c
   ======================================================================== */

#include <stdio.h>
#include <stdlib.h>
#include "libro.h"

// Definicion de las funciones primitivas de Libro
Libro *libroCrear(char *tituloLibro, int anioLibro, char *nombreAutor, char *apellidosAutor) {
    Libro *pl = NULL;

    // Comprobamos argumentos de entrada
    if (!tituloLibro || anioLibro <= 0 || !nombreAutor || !apellidosAutor) {
        return NULL;
    }

    // Reservamos memoria para la estructura Libro
    pl = (Libro *) malloc(sizeof(Libro));
    if (!pl) {
        return NULL;
    }

    // Reservamos memoria para los campos de la estructura Libro
    pl->titulo = (char *) malloc((1 + strlen(tituloLibro)) * sizeof(char));
    if (!pl->titulo) {
        free(pl);
        return NULL;
    }
```


Archivos de cabecera y archivos fuente (III)


```
    pl->autor = (Autor *) malloc(sizeof(Autor));
    if (!pl->autor) {
        libroLiberar(pl);
        return NULL;
    }
    pl->autor->nombre = NULL;
    pl->autor->apellidos = NULL;

    pl->autor->nombre = (char *) malloc((1 + strlen(nombreAutor)) * sizeof(char));
    if (!pl->autor->nombre) {
        libroLiberar(pl);
        return NULL;
    }

    pl->autor->apellidos = (char *) malloc((1 + strlen(apellidosAutor)) * sizeof(char));
    if (!pl->autor->apellidos) {
        libroLiberar(pl);
        return NULL;
    }

    // Una vez reservada su memoria, incializamos todos los campos del libro
    strcpy(pl->titulo, tituloLibro);
    strcpy(pl->autor->nombre, nombreAutor);
    strcpy(pl->autor->apellidos, apellidosAutor);
    pl->anio = anioLibro;

    return pl;
}
```


Archivos de cabecera y archivos fuente (III)


```
void libroLiberar(Libro *pl) {
    if (pl) {
        if (pl->titulo) {
            free(pl->titulo);
        }
        if (pl->autor) {
            if (pl->autor->nombre) {
                free(pl->autor->nombre);
            }
            if (pl->autor->apellidos) {
                free(pl->autor->apellidos);
            }
            free(pl->autor); // Liberamos autor despues de liberar nombre y apellidos
        }
        free(pl);
        pl = NULL;
    }
}
```

Archivos de cabecera y archivos fuente (IV)

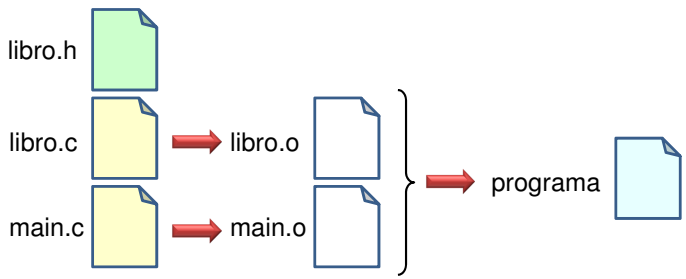

```
/* ============================================================================
   main.c
   ============================================================================ */

#include <stdio.h>
#include "libro.h"

void main() {
    Libro *libro = NULL;
    char nombreAutor[32], apellidosAutor[64], tituloLibro[64];
    int anioLibro;

    printf("Nombre del autor: ");
    gets(nombreAutor);
    printf("Apellidos del autor: ");
    gets(apellidosAutor);
    printf("Titulo del libro: ");
    gets(tituloLibro);
    printf("Anio de publicacion del libro: ");
    scanf("%d", &anioLibro);

    libro = libroCrear(tituloLibro, anioLibro, nombreAutor, apellidosAutor);

    if (libro) {
        printf("%s (%d), de %s %s.\n", libro->titulo, libro->anio, libro->autor->nombre,
                                                                   libro->autor->apellidos);
        libroLiberar(libro);
    }
}
```


Contenidos

- Archivos de cabecera y archivos fuente
- **Compilación y enlazado de un proyecto**

Proyectos con varios archivos

- Con C la construcción de un archivo ejecutable se realiza en 2 fases
  - Compilación de archivos fuente individualmente
    - archivos fuente .c → archivos objeto .o (.obj)
  - Enlazado de archivos objeto
    - archivos objeto .o → archivo ejecutable (.exe en Windows)



Compilación y enlazado de un proyecto (I)

- Con el compilador **gcc** la compilación y enlazado de los archivos de un proyecto se hace en 2 fases
  - Compilación de archivos fuente individualmente (generando los archivos objetos .o)

```
gcc -c libro.c
gcc -c main.c
```

  - Enlazado de archivos objeto (generando el archivo ejecutable)

```
gcc -o programa main.o libro.o
```

Compilación y enlazado de un proyecto (II)

- Para facilitar compilaciones y enlazados sucesivos se suele generar un fichero **Makefile**, que es interpretado por el programa **make**

Makefile

```
programa: main.o libro.o
   gcc -o programa main.o libro.o
libro.o: libro.c libro.h
   gcc -c libro.c
main.o: main.c libro.h
   gcc -c main.c
```