Tema: VI.
Bibliotecas: creación y uso

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Departamento de Lenguajes y Sistemas Informáticos
Universidad de Alicante

lsi Department of Software and Computing Systems

Contenido

¿Qué es una biblioteca?

- De manera muy resumida podemos decir que una *biblioteca*[1] es un compendio de recursos: subprogramas, clases, datos, etc...
- Cuando distribuímos estos recursos dentro de una biblioteca estamos favoreciendo su uso y reutilización.
- ¿Motivo?: En el caso de *código fuente* no es necesario recompilar ya que éste se distribuye dentro de la biblioteca en forma binaria, ya compilado; hasta ahora solo sabíamos redistribuirlo en forma de código fuente.
- Para emplear una biblioteca hemos de *enlazar* nuestro código con dicha biblioteca, de esta forma tenemos acceso a su contenido.

[1] A lo largo del tema emplearemos el término '*biblioteca*' en lugar de '*librería*' por ser el primero más apropiado.

¿Porqué distribuir algo en formato binario?

Por varios motivos:

- Si para usarlo debe estar en formato binario le evitamos al usuario del mismo tener que compilarlo.
- En ocasiones el proceso de compilación y obtención de una biblioteca es *costoso* y puede que no sea sencillo.
- En el caso de las bibliotecas de enlace dinámico tenemos la ventaja de poder cambiarlas para solucionar problemas sin necesidad de recompilar.

Ejemplo sencillo I

- Veamos en qué consiste una biblioteca con un ejemplo...
- Para ello partimos de un código *monolítico* donde el programa principal y las funciones que usa están en un único archivo.
- Se trata de una aplicación que implementa un sencillo algoritmo de compresión/descompresión sobre cadenas.
- Invocado de esta forma:

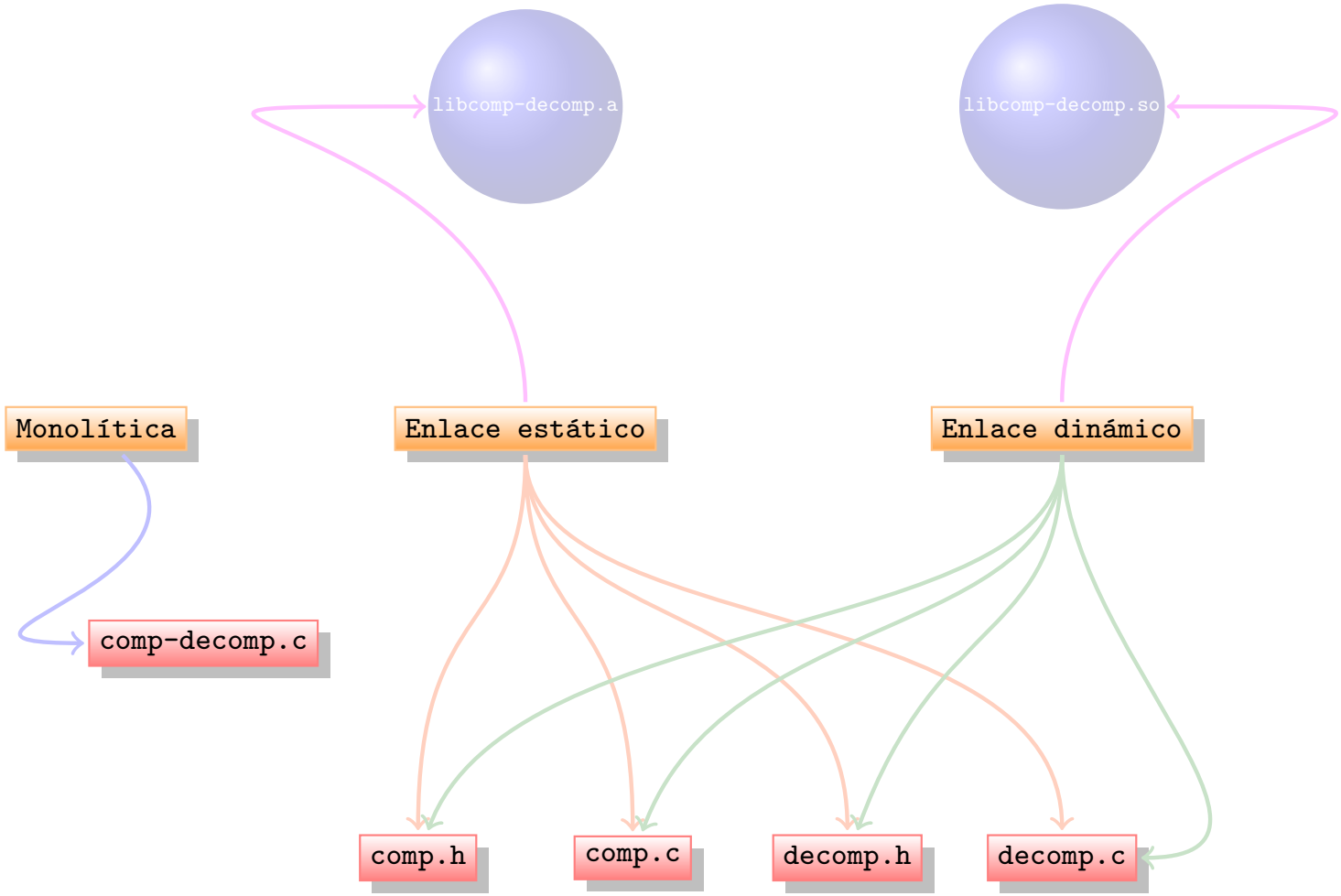
```
comp-decomp -c ccccaasssssssssaaaaaaa
```

produce esta salida:

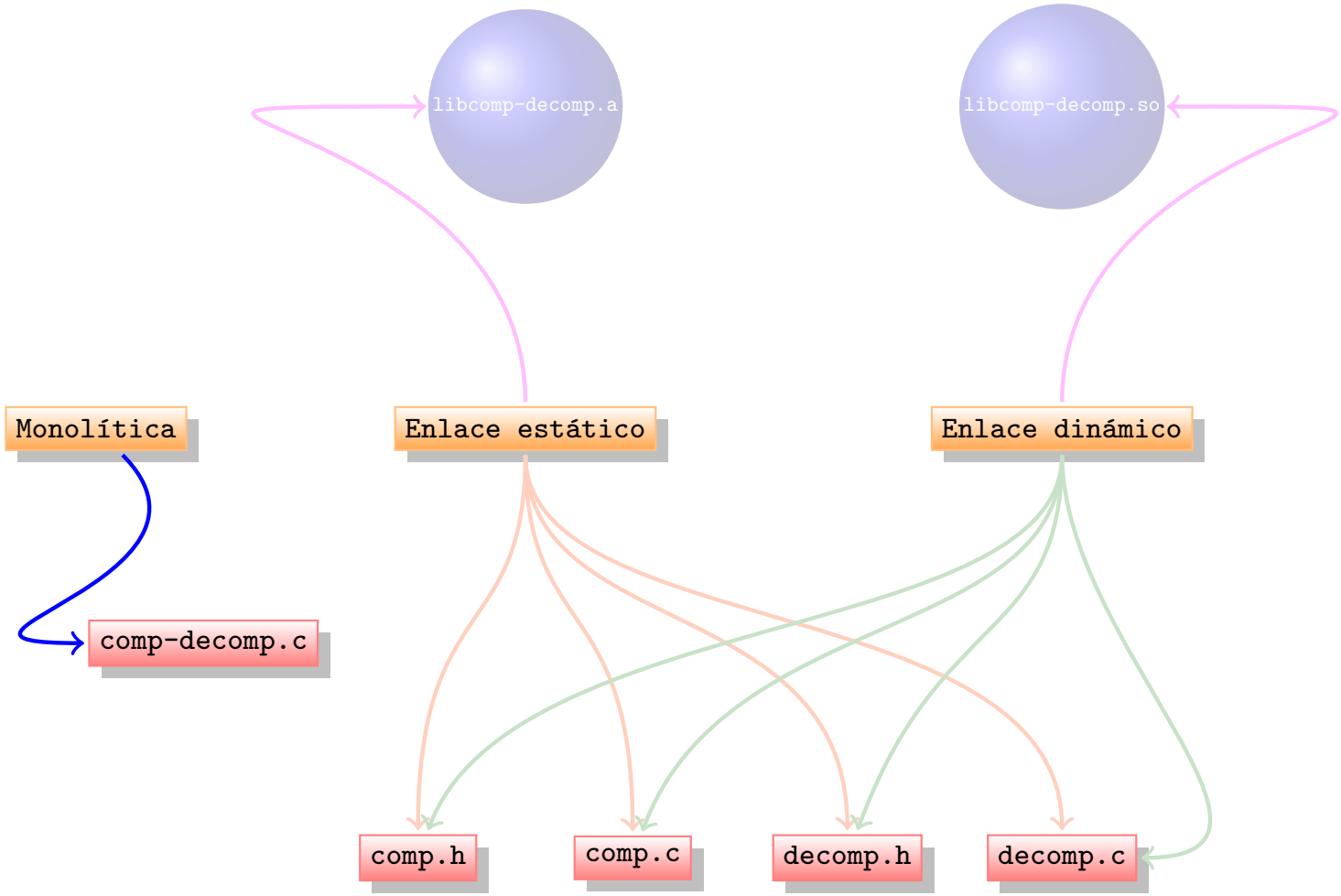
```
Compresion de ‘‘ccccaasssssssssaaaaaaa’’(21) es ‘‘4caa8s7a’’(8)
```

- Partiendo del mismo código vamos a crear tres versiones de la aplicación: *monolítica*, enlazada con una *con una biblioteca estática* y enlazada *con una biblioteca dinámica*.

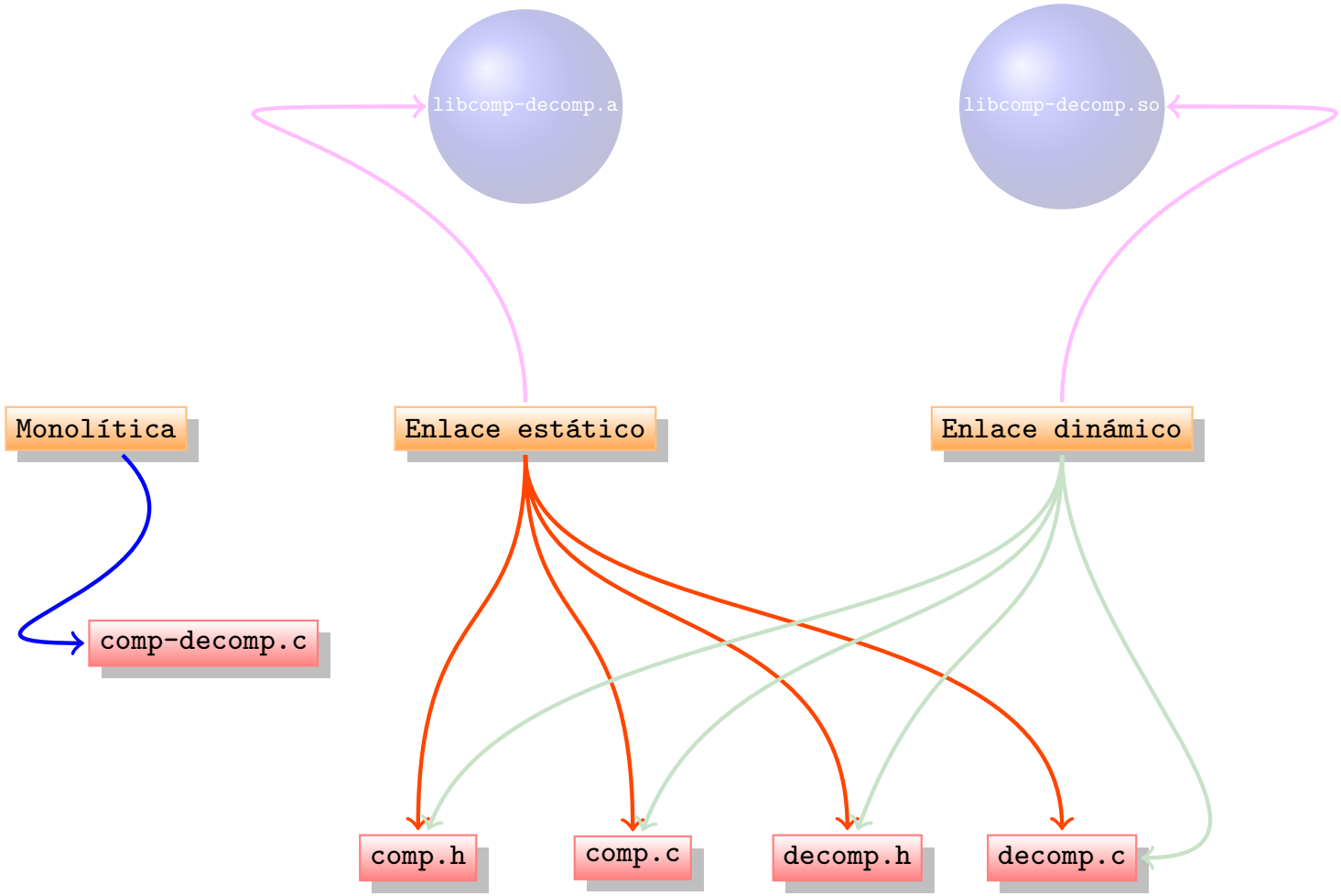
Ejemplo sencillo II



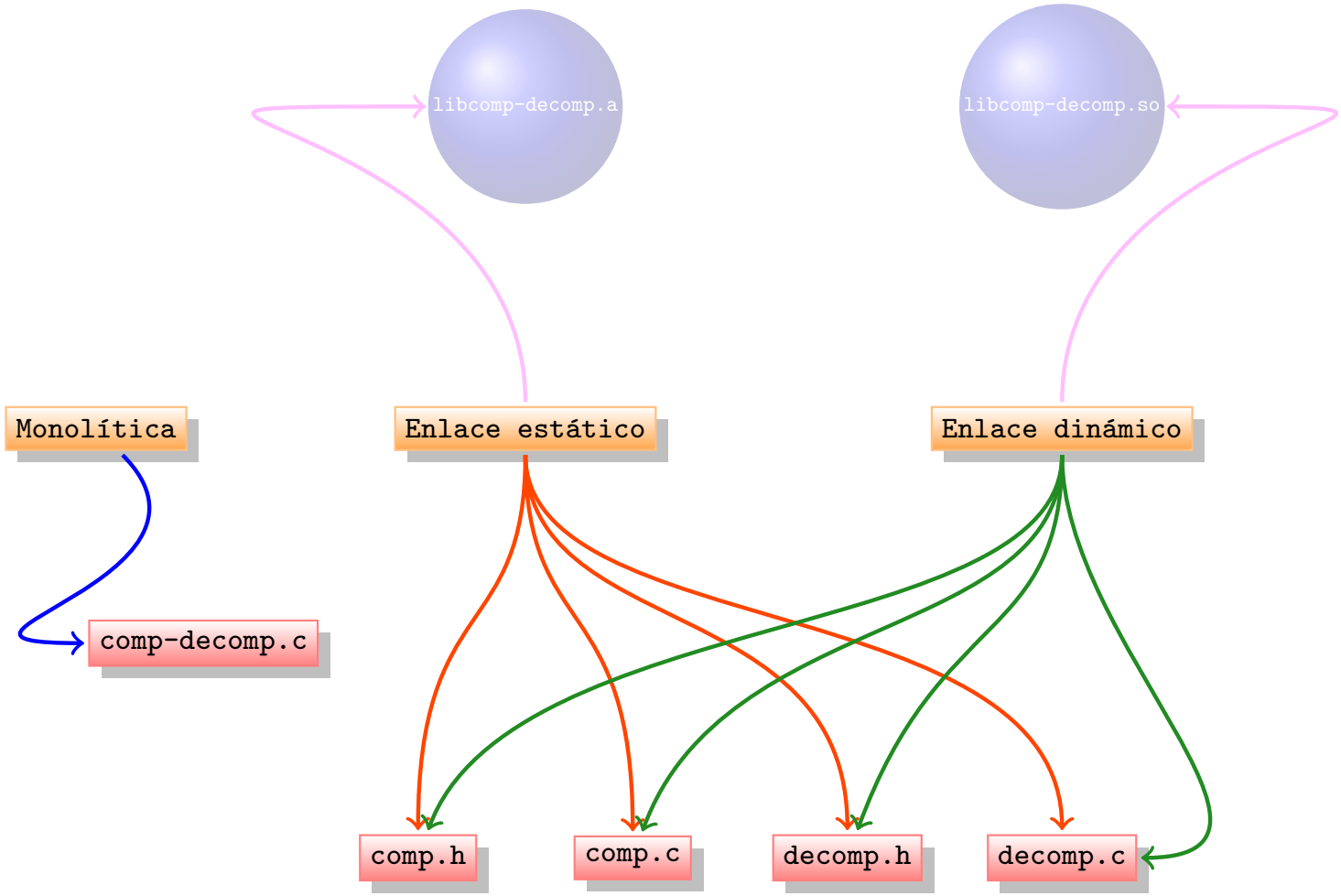
Ejemplo sencillo II



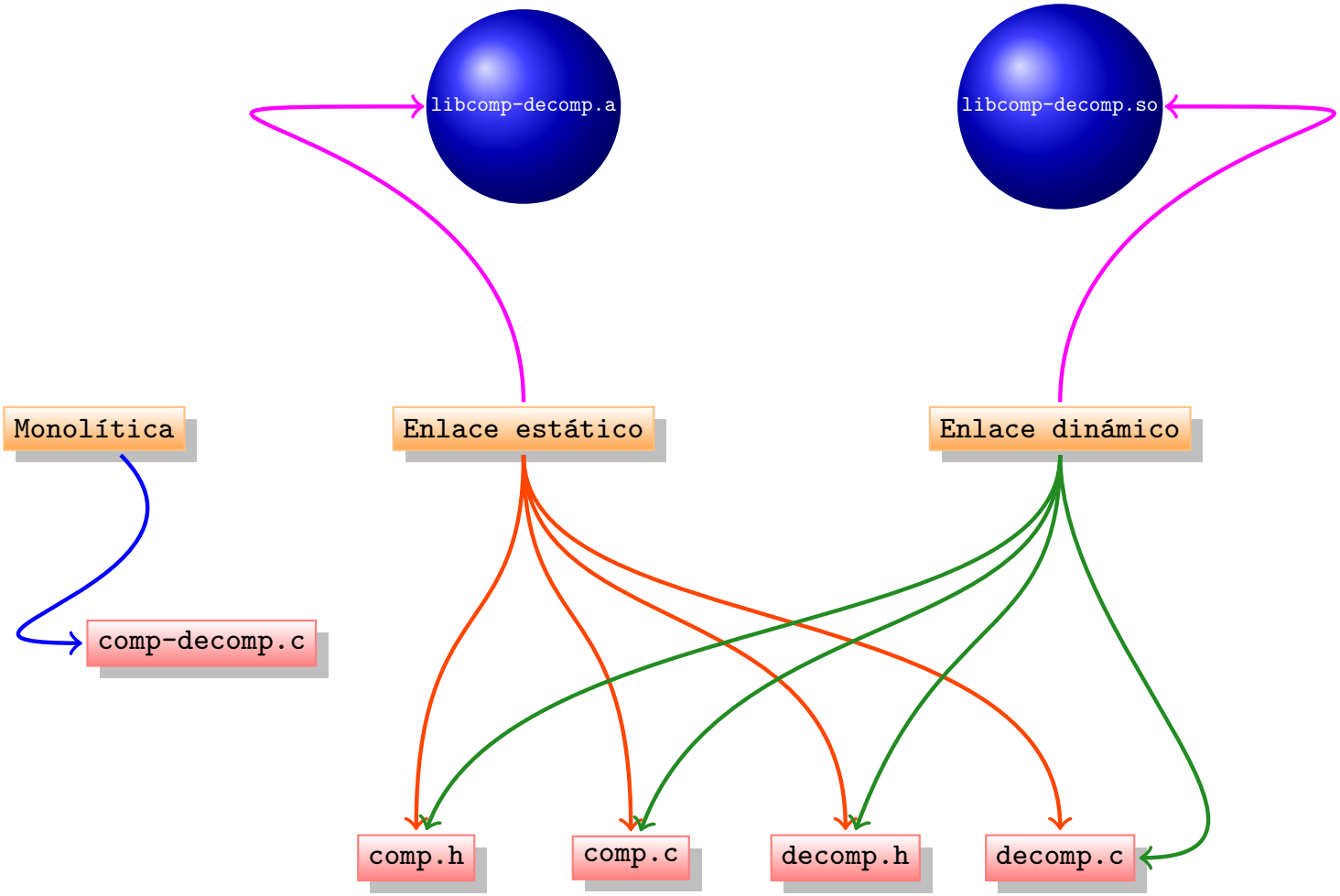
Ejemplo sencillo II



Ejemplo sencillo II



Ejemplo sencillo II



Código ejemplo monolítico I (comp-decomp.c)

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Devuelve el numero de caracteres iguales al primero de 's' */
static int caracteres_iguales (char* s) {
  int l = 0;
  int cont = 1;

  if (s == NULL) return 0;

  l = strlen (s);
  if (l < 2)
    return l;
  else {
    int i = 1;
    while (s[0] == s[i++])
      cont++;
    return cont;
  }
}
```

Código ejemplo monolítico II (comp-decomp.c)

```
1  /* Devuelve en 'cs' la cadena comprimida de 's'. */
   void comprime (char* s, char* cs) {
3    int l = cont = i = 0;
     char num[5], *s2;
5
     if (s == NULL)  return;
7
     l = strlen (s);
9    if (l < 3) strcat (cs, s);
     else {
11     int csl = 0;
       cont = caracteres_iguales (s);
13     if (cont > 2) {
         sprintf (num, "%d", cont);
15       strcat (cs, num);
         csl = strlen(cs);
17       cs[csl++] = s[0];
         cs[csl] = '\0';
19     } else {
         csl = strlen(cs);
21       for (i = 0; i < cont; i++)
           cs[csl++] = s[0];
23       cs[csl] = '\0';
       }
25
       s2 = &s[cont];
27     comprime (s2, cs);
     }
29 }
```

Código ejemplo monolítico III (comp-decomp.c)

```
1  /* Devuelve en 's' la cadena descomprimida de 'cs'. */
   void descomprime (char* cs, char* s) {
3    /* por hacer */
     strcat (s, cs);
5  }

7  int main(int argc, char *argv[])
   {
9    char c[100];

11   if (argc != 3) {
       printf("Uso: comp-decomp [-c|-d] cadena\n");
13     return 1;
     } else {
15     if ( (strcmp(argv[1],"-c") != 0) && (strcmp(argv[1],"-d") != 0) ) {
         printf("Uso: comp-decomp [-c|-d] cadena\n");
17       return 2;
       }
19   }
     if (strcmp(argv[1],"-c") == 0) {
21     c[0] = '\0';
       comprime (argv[2], c);
23     printf ("Compresion de \"%s\"(%d) es \"%s\"(%d)\n", argv[2], strlen(argv[2]), c,
           strlen(c));
     }
25   if (strcmp(argv[1],"-d") == 0) {
       c[0] = '\0';
27     descomprime (argv[2], c);
       printf ("Descompresion de \"%s\"(%d) es \"%s\"(%d)\n", argv[2], strlen(argv[2]),
           c, strlen(c));
29   }
     return 0;
31 }
```

Creación de una biblioteca

- Para crear una biblioteca de enlace dinámico o estático primero necesitamos dividir el código anterior.
- Colocaremos en uno o varios archivos las funciones y/o estructuras de datos que proporcionan la funcionalidad principal.
- Dejaremos fuera de estos archivos el código que hacía de programa principal, el cual estará en otro archivo que se limitará a hacer de 'consumidor' del código de la biblioteca.
- En nuestro caso la división en archivos nos proporciona los siguientes:
  - `comp.h` y `comp.c` Contienen el código relacionado con la función de compresión.
  - `decomp.h` y `decomp.c` Contienen el código relacionado con la función de descompresión.
  - `comp-decomp-driver.c` Es el programa principal en el cual se analizan los argumentos con el que lo hemos llamado y se llama a la función correspondiente.

comp.h y comp.c I

```
1   /* comp.h */

3   void comprime (char* s, char* cs);
```

```
1   /* comp.c */

3   #include <stdio.h>
    #include <stdlib.h>
5   #include <string.h>

7   /* Es privada, la biblioteca no la exporta. (modificador static) */
    static int caracteres_iguales (char* s) {
9     int l = 0, cont = 1;

11    if (s == NULL) return 0;
      l = strlen (s);
13    if (l < 2) return l;
      else {
15      int i = 1;
        while (s[0] == s[i++]) cont++;
17      return cont;
      }
19  }
```

comp.h y comp.c II

```
1  /* comp.c */

3  void comprime (char* s, char* cs) {
     int l = cont = i = 0;
5    char num[5], *s2;

7    if (s == NULL) return;

9    l = strlen (s);
     if (l < 3) strcat (cs, s);
11   else {
       int csl = 0;
13     cont = caracteres_iguales (s);
       if (cont > 2) {
15       sprintf (num, "%d", cont);

17       strcat (cs, num);
         csl = strlen(cs);
19       cs[csl++] = s[0];
         cs[csl] = '\0';
21     } else {
         csl = strlen(cs);
23       for (i = 0; i < cont; i++) cs[csl++] = s[0];
         cs[csl] = '\0';
25     }
       s2 = &s[cont];
27     comprime (s2, cs);
     }
29 }
```

decomp.h y decomp.c

```c
/* decomp.h */

void descomprime (char* s, char* cs);
```

```c
/* decomp.c */

#include <string.h>

/* Devuelve en 's' la cadena descomprimida de 'cs'.
   */
void descomprime (char* cs, char* s) {
  /* por hacer */
  strcat (s, cs);
}
```

comp-decomp-driver.c

```c
/* comp-decomp-driver.c -- Es el programa princiapal. */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "comp.h"
#include "decomp.h"

int main(int argc, char *argv[])
{
  char c[100];

  if (argc != 3) {
    printf("Uso: comp-decomp [-c|-d] cadena\n");
    return 1;
  } else
  if ( (strcmp(argv[1],"-c") != 0) &&
       (strcmp(argv[1],"-d") != 0) )
       {
         printf("Uso: comp-decomp [-c|-d] cadena\n");
         return 2;
       }
  if (strcmp(argv[1],"-c") == 0) {
    c[0] = '\0';
    comprime (argv[2], c);
    printf ("Compresion de \"%s\"(%d) es \"%s\"(%d)\n", argv[2], strlen(argv[2]), c,
          strlen(c));
  }
  if (strcmp(argv[1],"-d") == 0) {
    c[0] = '\0';
    descomprime (argv[2], c);
    printf ("Descompresion de \"%s\"(%d) es \"%s\"(%d)\n", argv[2], strlen(argv[2]),
          c, strlen(c));
  }
  return 0;
}
```

¿Cómo generamos una biblioteca de enlace estático?

- Compilamos los archivos que la componen por separado. De cada uno obtenemos su '`.o`'.
- Con la aplicación '`ar`' -similar a `tar`- creamos el archivo 'biblioteca' con extensión '`.a`' -**a** de *archivo*- así:

```
ar crs libcomp-decomp.a comp.o decomp.o
```

- Donde la cadena '`crs`' representa las opciones con las que lo llamamos:
  - `c` Crea el archivo con nombre: `libcomp-decomp.a`
  - `r` Reemplaza -si ya existiera- en el archivo '`.a`' cada uno de los archivos que se especifican a continuación.
  - `s` Crea un índice en el archivo '`.a`' para que sea más rápido acceder a los archivos que contiene.

¿Cómo enlazamos con una biblioteca de enlace estático? I

Podemos hacerlo de varias maneras:

- Especificando su ruta completa, igual que si fuera un archivo '`.o`': '`/ruta/hasta/fichero.a`'.
- Adicionalmente, si su nombre sigue este convenio: libnombre-bib.a, el enlazador lo reconoce y se puede poner de este modo en la línea de enlace: `-lnombre-bib`.
- El *enlazador* es una aplicación distinta al compilador, es independiente del lenguaje de programación empleado. Tradicionalmente en s.o. de la familia UNIX este programa se llama '`ld`'.

¿Cómo enlazamos con una biblioteca de enlace estático? II

```
    ########################
 2  # Biblioteca estatica  #
    ########################################################
 4  comp-decomp-estatico: comp-decomp-driver.o libcomp-decomp.a
        $(CC) -static comp-decomp-driver.o -L . -lcomp-decomp -o comp-decomp-estatico
 6
    comp-decomp-driver.o: comp-decomp-driver.c
 8      $(CC) -c $(CFLAGS) comp-decomp-driver.c

10  libcomp-decomp.a: comp.o decomp.o
        $(AR) crs libcomp-decomp.a comp.o decomp.o
12
    comp.o: comp.c
14      $(CC) -c $(CFLAGS) comp.c

16  decomp.o: decomp.c
        $(CC) -c $(CFLAGS) decomp.c
```

¿Cómo generamos una biblioteca de enlace dinámico?

- Al igual que antes compilamos los archivos que la componen por separado. De cada uno obtenemos su '`.o`' pero hemos de usar la opción `-fpic`. `pic`: Position Independent Code.
- Las bibliotecas de enlace dinámico usan la extensión '`.so`' en s.o. de la familia UNIX. Son equivalentes a las '`DLL`' de Windows.
- Para generarlas no necesitamos de ningun programa nuevo como antes, basta con el propio enlazador suministrándole la opción '`-shared`':

```
gcc -shared -o libcomp-decomp.so comp-pic.o decomp-pic.o
```

¿Cómo enlazamos con una biblioteca de enlace dinámico?

```
1  ###################################
   # Bibliotecea de enlace dinamico #
3  ###################################################################
   comp-decomp-dinamico: comp-decomp-driverdin.o libcomp-decomp.so
5      $(CC) comp-decomp-driverdin.o -L . -lcomp-decomp -o comp-decomp-dinamico

7  comp-decomp-driverdin.o: comp-decomp-driverdin.c
       $(CC) -fpic -c $(CFLAGS) comp-decomp-driverdin.c
9
   libcomp-decomp.so: comp-pic.o decomp-pic.o
11     $(CC) -shared -o libcomp-decomp.so comp-pic.o decomp-pic.o

13 comp-pic.o: comp.c
       $(CC) -c -o comp-pic.o -fpic $(CFLAGS) comp.c
15
   decomp-pic.o: decomp.c
17     $(CC) -c -o decomp-pic.o -fpic $(CFLAGS) decomp.c
```

Aplicaciones útiles para archivos '.o', '.a' y '.so' I

Son aplicaciones que nos permiten extraer información de este tipo de archivos o incluso modificarlos de cierto manera:

- **nm** Lista los símbolos dentro de un archivo binario. Además nos da cierta información sobre cada uno de ellos.
- **ranlib** Crea el índice en una biblioteca hecha con '`ar`', es decir, tiene la misma función que la opción '`s`' de '`ar`'.
- **strip** Elimina determinados símbolos del archivo binario con el fin de reducir su tamaño.

Aplicaciones útiles para archivos '.o', '.a' y '.so' II

Veamos la salida que produce '`nm`' sobre la biblioteca de enlace estático que hemos creado previamente:

```
1   Ejecutamos libcomp-decomp.a

3   comp.o:
    0000000000000000 t caracteres_iguales
5   0000000000000078 T comprime
                     U sprintf
7                    U strcat
                     U strlen
9
    decomp.o:
11  0000000000000000 T descomprime
                     U strcat
```

Aplicaciones útiles para archivos '.o', '.a' y '.so' III

- ¿Qué efecto tiene '`strip`' en nuestros archivos binarios?
- Básicamente el tamaño final de los mismos, compara:

  antes `ls -l libcomp-decomp.a` $= 3838$ bytes
  después `strip libcomp-decom.a`,
  `ls -l libcomp-decomp.a` $= 2536$ bytes

- En general puedes emplear `strip` con cualquier archivo binario generado en el proceso de compilación/enlace, incluso con un ejecutable final:

  antes `ls -l comp-decomp` $= 8658$ bytes
  después `strip comp-decom`,
  `ls -l comp-decomp` $= 6008$ bytes

- Después de conocer '`strip`' qué crees que son los 'targets' `Debug` y `Release` de VisualStudio?

Aplicaciones útiles para archivos '.o', '.a' y '.so' IV

- En el caso de archivos ejecutables enlazados dinámicamente con alguna biblioteca, la orden '`ldd`' es especialmente útil.
- Nos informa de con qué bibliotecas está enlazada de forma dinámica nuestra aplicación y si falta alguna; además nos indica la ruta hasta donde está cada una de ellas en el sistema de ficheros:

```
    ldd comp-decomp
2       linux-vdso.so.1 =>  (0x00007fffa57ff000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f38c1b46000)
4       libcomp-decomp.so => not found
        /lib64/ld-linux-x86-64.so.2 (0x00007f38c1ef6000)
```

Cómo crear y usar una biblioteca en Vala

- Separamos el código siguiendo los mismos criterios que en el ejemplo en 'C'.
- Para generar la biblioteca empleamos una orden como esta:

```
valac --library=comp-decomp -H comp-decomp.h comp.vala -X -fPIC -X -shared -o libcomp-decomp.so
```

- Mientras que para enlazar con ella usaremos una orden como esta otra:

```
valac comp-decomp.vapi comp-decomp-driver.vala -X libcomp-decomp.so -X -I. -o comp-decomp-driver
```

- Los ficheros '`.vapi`' son para '`Vala`' similares a los '`.h`' para '`C`'.

El código de ejemplo en Vala I

```
1  /* comp.vala */
   namespace CompDecomp {
3    private int caracteres_iguales (string s) {
       int l = 0;
5      int cont = 1;

7      l = s.length;
       if (l < 2) return l;
9      else {
         int i = 1;
11       while (s[0] == s[i++]) cont++;
         return cont;
13     }
     }
15
     public void comprime (string s, ref string cs) {
17     int l = 0, cont = 0;
       string num, s2;
19
       l = s.length;
21     if (l < 3) cs += s;
       else {
23       cont = caracteres_iguales (s);
         if (cont > 2) {
25         num = cont.to_string();
           cs += num; cs += s[0].to_string();
27       } else { cs += string.nfill (cont, s[0]); }
         s2 = s[cont:s.length];
29       comprime (s2, ref cs);
       }
31   }
   }
```

El código de ejemplo en Vala II

```
/* comp-decomp-driver.vala */
using CompDecomp;

int main(string[] args) {
  if (args.length != 3) {
    stdout.printf ("Uso: comp-decomp [-c|-d] cadena\n");
    return 1;
  } else if ( (args[1] != "-c") && args[1] != "-d") ) {
    stdout.printf("Uso: comp-decomp [-c|-d] cadena\n");
    return 2;
  }
  if (args[1] == "-c") {
    string c="";
    comprime (args[2], ref c);
    stdout.printf ("Compresion de \"%s\"(%d) es \"%s\"(%d)\n",
                   args[2], args[2].length,
                   c,       c.length);
  }

  return 0;
}
```