

## Tema: VI. Bibliotecas: creación y uso

### Herramientas Avanzadas para el Desarrollo de Aplicaciones

Departamento de Lenguajes y Sistemas Informáticos  
Universidad de Alicante

Curso 2014-2015, Copyleft © 2011-2015.  
Reproducción permitida bajo los términos de la licencia de  
documentación libre GNU.



1 / 26

## Contenido

- ① ¿Qué es una biblioteca?
- ② ¿Porqué distribuir algo en formato binario?
- ③ Ejemplo sencillo
- ④ Código ejemplo monolítico
- ⑤ Creación de una biblioteca
- ⑥ ¿Cómo generamos una biblioteca de enlace estático?
- ⑦ ¿Cómo enlazamos con una biblioteca de enlace estático?
- ⑧ ¿Cómo generamos una biblioteca de enlace dinámico?
- ⑨ ¿Cómo enlazamos con una biblioteca de enlace dinámico?
- ⑩ Aplicaciones útiles para archivos '.o', '.a' y '.so'
- ⑪ Cómo crear y usar una biblioteca en Vala
- ⑫ El código de ejemplo en Vala



2 / 26

## ¿Qué es una biblioteca?

- De manera muy resumida podemos decir que una *biblioteca*<sup>1</sup> es un compendio de recursos: subprogramas, clases, datos, etc. . .
- Cuando distribuimos 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.

<sup>1</sup>A lo largo del tema emplearemos el término '*biblioteca*' en lugar de '*librería*' el primero más apropiado.



3 / 26

## ¿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.



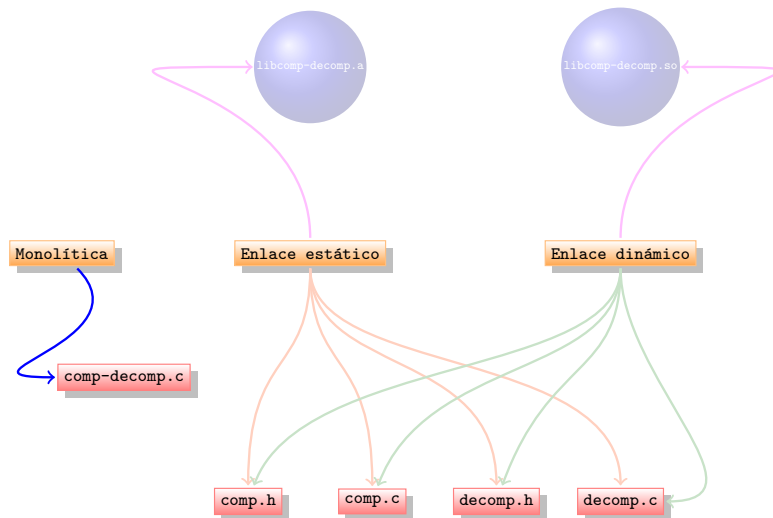
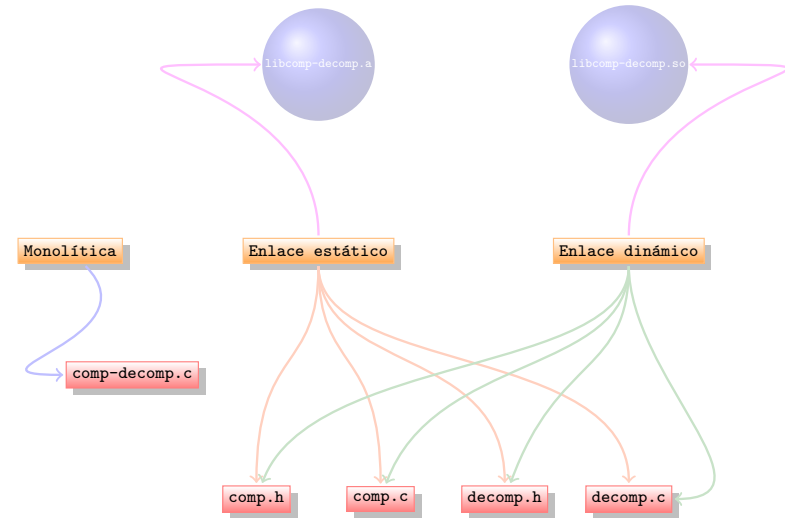
4 / 26

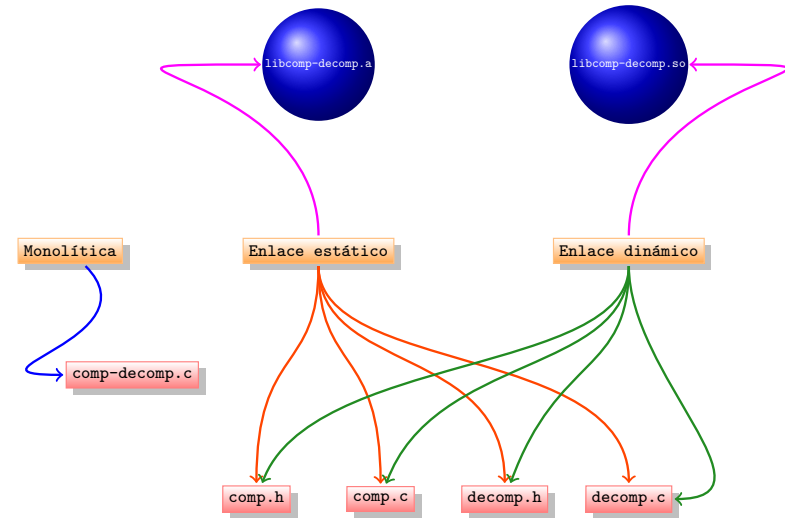
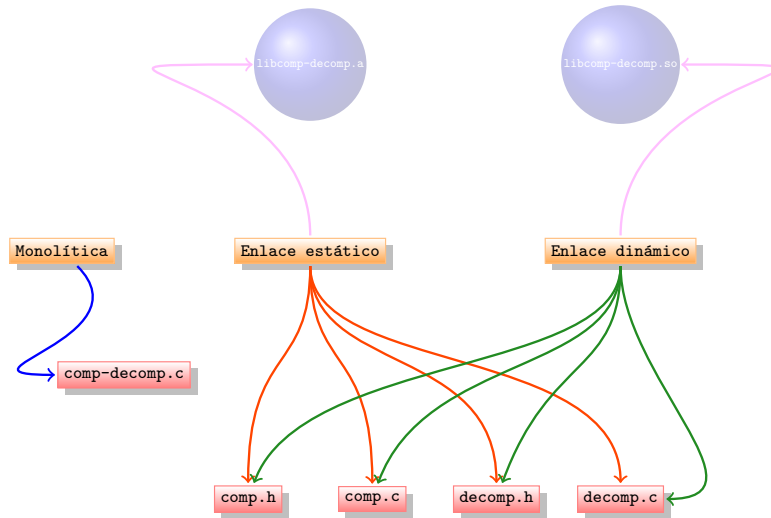
- Veamos en qué consiste una biblioteca con un ejemplo...
- Para ello partimos de un código *monolítico* donde el programa principal y las funciones que usa están en un único archivo.
- Se trata de una aplicación que implementa un sencillo algoritmo de compresión/descompresión sobre cadenas.
- Invocado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

```
comp-decomp -c ccccaassssssssaaaaaaa
```

 produce esta salida:  

```
Compresion de 'ccccaassssssssaaaaaaa'(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*.





## Código ejemplo monolítico I (comp-decomp.c)

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>

5  /* Devuelve el numero de caracteres iguales al primero de 's' */
6  static int caracteres_iguales (char* s) {
7      int l = 0;
8      int cont = 1;
9
10     if (s == NULL) return 0;
11
12     l = strlen (s);
13     if (l < 2)
14         return l;
15     else {
16         int i = 1;
17         while (s[0] == s[i++])
18             cont++;
19         return cont;
20     }
21 }

```

## Código ejemplo monolítico II (comp-decomp.c)

```

1  /* Devuelve en 'cs' la cadena comprimida de 's'. */
2  void comprime (char* s, char* cs) {
3      int l = cont = i = 0;
4      char num[5], *s2;
5
6      if (s == NULL) return;
7
8      l = strlen (s);
9      if (l < 3) strcat (cs, s);
10     else {
11         int csl = 0;
12         cont = caracteres_iguales (s);
13         if (cont > 2) {
14             sprintf (num, "%d", cont);
15             strcat (cs, num);
16             csl = strlen(cs);
17             cs[csl++] = s[0];
18             cs[csl] = '\0';
19         } else {
20             csl = strlen(cs);
21             for (i = 0; i < cont; i++)
22                 cs[csl++] = s[i];
23             cs[csl] = '\0';
24         }
25
26         s2 = &s[cont];
27         comprime (s2, cs);
28     }
29 }

```

```

1  /* Devuelve en 's' la cadena descomprimida de 'cs'. */
2  void descomprime (char* cs, char* s) {
3      /* por hacer */
4      strcat (s, cs);
5  }
6
7  int main(int argc, char *argv[])
8  {
9      char c[100];
10
11     if (argc != 3) {
12         printf("Uso: comp-decomp [-c|-d] cadena\n");
13         return 1;
14     } else {
15         if ( (strcmp(argv[1], "-c") != 0) && (strcmp(argv[1], "-d") != 0) ) {
16             printf("Uso: comp-decomp [-c|-d] cadena\n");
17             return 2;
18         }
19     }
20
21     if (strcmp(argv[1], "-c") == 0) {
22         c[0] = '\0';
23         comprime (argv[2], c);
24         printf ("Compresion de \"%s\"(%d) es \"%s\"(%d)\n", argv[2], strlen(argv[2]), c,
25             strlen(c));
26     }
27     if (strcmp(argv[1], "-d") == 0) {
28         c[0] = '\0';
29         descomprime (argv[2], c);
30         printf ("Descompresion de \"%s\"(%d) es \"%s\"(%d)\n", argv[2], strlen(argv[2]),
31             c, strlen(c));
32     }
33     return 0;
34 }

```

- 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 */
2
3  void comprime (char* s, char* cs);

```

```

1  /* comp.c */
2
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <string.h>
6
7  /* Es privada, la biblioteca no la exporta. (modificador static) */
8  static int caracteres_iguales (char* s) {
9      int l = 0, cont = 1;
10
11     if (s == NULL) return 0;
12     l = strlen (s);
13     if (l < 2) return 1;
14     else {
15         int i = 1;
16         while (s[0] == s[i++]) cont++;
17         return cont;
18     }
19 }

```

## comp.h y comp.c II

```

1  /* comp.c */
2
3  void comprime (char* s, char* cs) {
4      int l = cont = i = 0;
5      char num[5], *s2;
6
7      if (s == NULL) return;
8
9      l = strlen (s);
10     if (l < 3) strcat (cs, s);
11     else {
12         int csl = 0;
13         cont = caracteres_iguales (s);
14         if (cont > 2) {
15             sprintf (num, "%d", cont);
16
17             strcat (cs, num);
18             csl = strlen(cs);
19             cs[csl++] = s[0];
20             cs[csl] = '\0';
21         } else {
22             csl = strlen(cs);
23             for (i = 0; i < cont; i++) cs[csl++] = s[i];
24             cs[csl] = '\0';
25         }
26         s2 = &s[cont];
27         comprime (s2, cs);
28     }
29 }

```

```

1  /* decomp.h */

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

```

```

1  /* decomp.c */

3  #include <string.h>

5  /* Devuelve en 's' la cadena descomprimida de 'cs'.
   */
7  void descomprime (char* cs, char* s) {
8      /* por hacer */
9      strcat (s, cs);
10 }

```

```

1  /* comp-decomp-driver.c -- Es el programa principal. */
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <string.h>
5
6  #include "comp.h"
7  #include "decomp.h"
8
9  int main(int argc, char *argv[])
10 {
11     char c[100];
12
13     if (argc != 3) {
14         printf("Uso: comp-decomp [-c|-d] cadena\n");
15         return 1;
16     } else
17     if ( (strcmp(argv[1], "-c") != 0) &&
18         (strcmp(argv[1], "-d") != 0) )
19     {
20         printf("Uso: comp-decomp [-c|-d] cadena\n");
21         return 2;
22     }
23     if (strcmp(argv[1], "-c") == 0) {
24         c[0] = '\0';
25         comprime (argv[2], c);
26         printf ("Compresion de \"%s\"(%d) es \"%s\"(%d)\n", argv[2], strlen(argv[2]), c,
27             strlen(c));
28     }
29     if (strcmp(argv[1], "-d") == 0) {
30         c[0] = '\0';
31         descomprime (argv[2], c);
32         printf ("Descompresion de \"%s\"(%d) es \"%s\"(%d)\n", argv[2], strlen(argv[2]),
33             c, strlen(c));
34     }
35     return 0;
36 }

```

## ¿Cómo generamos una biblioteca de enlace estático?

## ¿Cómo enlazamos con 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.

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'.

```

#####
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
    $(CC) -c $(CFLAGS) comp-decomp-driver.c
8
10  libcomp-decomp.a: comp.o decomp.o
    $(AR) crs libcomp-decomp.a comp.o decomp.o
12
    comp.o: comp.c
    $(CC) -c $(CFLAGS) comp.c
14
16  decomp.o: decomp.c
    $(CC) -c $(CFLAGS) decomp.c

```

- 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**: **P**osition **I**ndependent **C**ode.
- 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?

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

```

1  #####
2  # Biblioteca de enlace dinamico #
3  #####
4  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
10  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

```

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.

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

```

- ¿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-decomp.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-decomp,
ls -l comp-decomp = 6008 bytes

```

- Después de conocer 'strip' qué crees que son los 'targets' [Debug](#) y [Release](#) de VisualStudio?

- 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)

```

- 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'.

```

1  /* comp.vala */
2  namespace CompDecomp {
3      private int caracteres_iguales (string s) {
4          int l = 0;
5          int cont = 1;
6
7          l = s.length;
8          if (l < 2) return l;
9          else {
10             int i = 1;
11             while (s[0] == s[i++]) cont++;
12             return cont;
13         }
14     }
15
16     public void comprime (string s, ref string cs) {
17         int l = 0, cont = 0;
18         string num, s2;
19
20         l = s.length;
21         if (l < 3) cs += s;
22         else {
23             cont = caracteres_iguales (s);
24             if (cont > 2) {
25                 num = cont.to_string();
26                 cs += num; cs += s[0].to_string();
27             } else { cs += string.nfill (cont, s[0]); }
28             s2 = s[cont:s.length];
29             comprime (s2, ref cs);
30         }
31     }
32 }

```

```

1  /* comp-decomp-driver.vala */
2  using CompDecomp;
3
4  int main(string[] args) {
5      if (args.length != 3) {
6          stdout.printf ("Uso: comp-decomp [-c|-d] cadena\n");
7          return 1;
8      } else if ( (args[1] != "-c") && args[1] != "-d" ) {
9          stdout.printf ("Uso: comp-decomp [-c|-d] cadena\n");
10         return 2;
11     }
12     if (args[1] == "-c") {
13         string c="";
14         comprime (args[2], ref c);
15         stdout.printf ("Compresion de \"%s\"(%d) es \"%s\"(%d)\n",
16             args[2], args[2].length,
17             c, c.length);
18     }
19     return 0;
20 }

```