

# Tema: VI.

## Biblioteques: creació i ús

### Herramientas Avanzadas para el Desarrollo de Aplicaciones

Departament de Llenguatges i Sistemes Informàtics  
Universitat d'Alacant

Curs 2014-2015 , Copyleft © 2011-2015 .

Reproducció permesa sota els termes de la llicència de documentació  
lliure GNU.

- 1 Què és una biblioteca?
- 2 Perquè distribuir alguna cosa en format binari?
- 3 Exemple senzill
- 4 Codi exemple monolític
- 5 Creació d'una biblioteca
- 6 Com generem una biblioteca d'enllaç estàtic?
- 7 Com enllacem amb una biblioteca d'enllaç estàtic?
- 8 Com generem una biblioteca d'enllaç dinàmic?
- 9 Com enllacem amb una biblioteca d'enllaç dinàmic?
- 10 Aplicacions útils per a arxius '.o', '.a' i '.so'
- 11 Com crear i usar una biblioteca en Vala
- 12 El codi d'exemple en Vala

# Què és una biblioteca?

- De manera molt resumida podem dir que una *biblioteca*<sup>1</sup> és un compendio de recursos: subprogrames, classes, dades, etc. . .
- Quan distribuïmos aquests recursos dins d'una biblioteca estem afavorint el seu ús i reutilització.
- Motiu?: En el cas de *codi font* no és necessari recompilar ja que aquest es distribueix dins de la biblioteca en forma binària, ja compilat; fins ara solament sabíem redistribuir-ho en forma de codi font.
- Per emprar una biblioteca hem de *enllaçar* nostre codi amb aquesta biblioteca, d'aquesta forma tenim accés al seu contingut.

---

<sup>1</sup>Al llarg del tema emprem el terme '*biblioteca*' en lloc de '*llibreria*' per ser el més apropiat.

# Perquè distribuir alguna cosa en format binari?

Per diversos motius:

- Si per usar-ho ha d'estar en format binari li evitem al usuari del mateix haver de compilar-ho.
- En ocasions el procés de compilació i obtenció d'una biblioteca és *costós* i pot ser que no sigui senzill.
- En el cas de les biblioteques d'enllaç dinàmic tenim la avantatge de poder canviar-les per solucionar problemes sense necessitat de recompilar.

# Exemple senzill I

- Vegem en què consisteix una biblioteca amb un exemple. . .
- Para això partim d'un codi *monolític* on el programa principal i les funcions que usa estan en un únic arxiu.
- Es tracta d'una aplicació que implementa un senzill algorisme de compressió/descompressió sobre cadenes.

- Invocat d'aquesta forma:

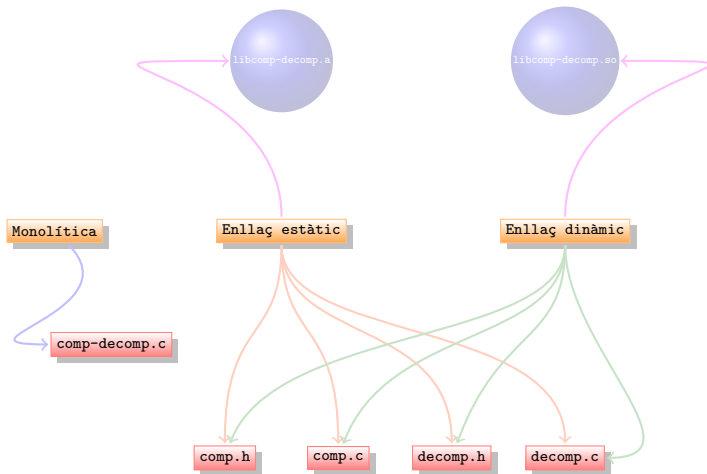
```
comp-decomp -c ccccaassssssssaaaaaaa
```

produeix aquesta sortida:

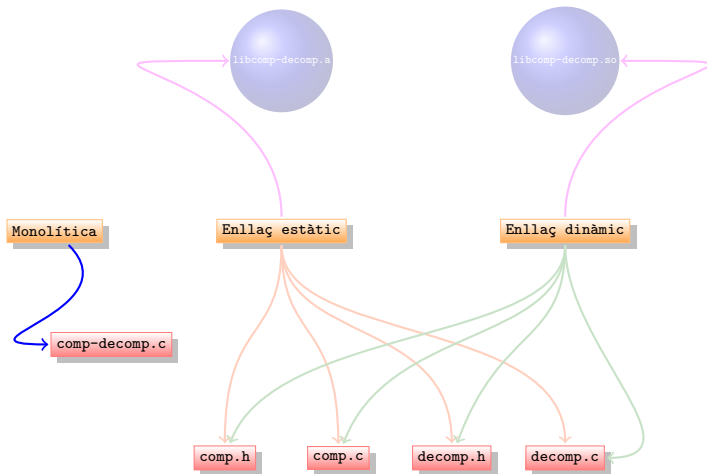
```
Compresion de 'ccccaassssssssaaaaaaa'(21) es '4caa8s7a'(8)
```

- Partint del mateix codi anem a crear tres versions de la aplicació: *monolítica*, enllaçada amb una *amb una biblioteca estàtica* i enllaçada *amb una biblioteca dinàmica*.

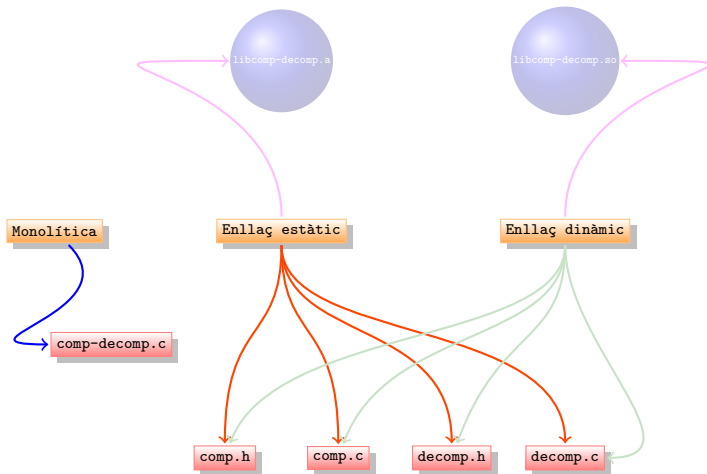
# Exemple senzill II



# Exemple senzill II

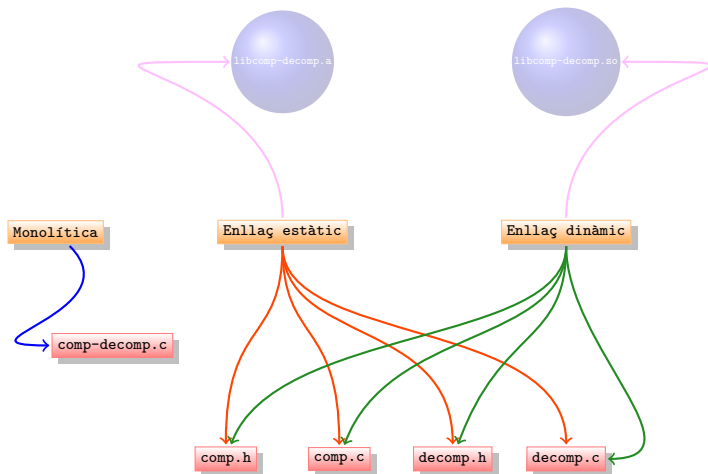


# Exemple senzill II

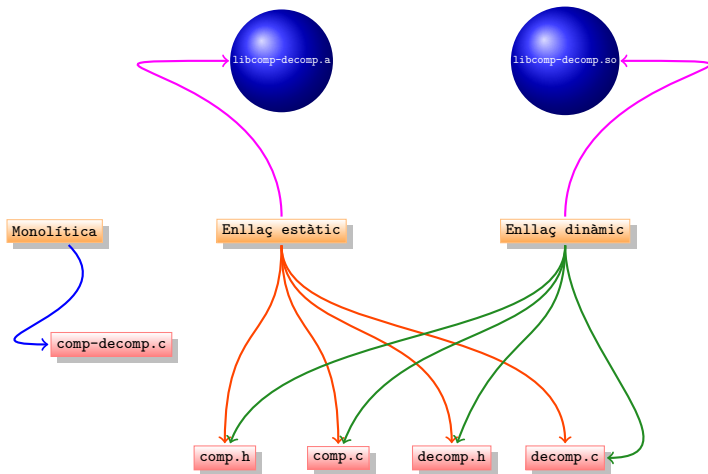




# Exemple senzill II



# Exemple senzill II



# Codi exemple monolític I (comp-decomp.c)

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

5  /* Retorna el numero de caracters iguals al primer de 's' /
   static int caracters-iguales (char* s) {
7      int l = 0;
       int cont = 1;

9      if (s == NULL) return 0;

11     l = strlen (s);
       if (l < 2)
13         return l;
       else {
15         int i = 1;
           while (s[0] == s[i++])
17             cont++;
19         return cont;
       }
21 }
```

# Codi exemple monolític II (comp-decomp.c)

```
1  /* Retorna 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[i];
23              cs[csl] = '\\0';
25          }
              s2 = &s[cont];
27              comprime (s2, cs);
29      }
```

# Codi exemple monolític III (comp-decomp.c)

```
1  /* Retorna 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) {
27            c[0] = '\0';
            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    }
```

# Creació d'una biblioteca

- Per crear una biblioteca d'enllaç dinàmic o estàtic primer necessitem dividir el codi anterior.
- Col·locarem en un o diversos arxius les funcions i/o estructures de dades que proporcionen la funcionalitat principal.
- Deixarem fora d'aquests arxius el codi que feia de programa principal, el qual estarà en un altre arxiu que es limitarà a fer de 'consumidor' del codi de la biblioteca.
- En el nostre cas la divisió en arxius ens proporciona els següents:
  - `comp.h` y `comp.c` Contenen el codi relacionat amb la funció de compressió.
  - `decomp.h` y `decomp.c` Contenen el codi relacionat amb la funció de descompressió.
  - `comp-decomp-driver.c` És el programa principal en el qual s'analitzen els arguments amb el que ho hem cridat i es diu a la funció corresponent.

```
1  /* comp.h */  
  
3  void comprime (char* s, char* cs);
```

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

```
1  /* comp.c */
3  void comprime (char* s, char* cs) {
4      int l = cont = i = 0;
5      char num[5], *s2;
7      if (s == NULL) return;
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);
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  /* Retorna en 's' la cadena descomprimida de 'cs'.  
   */  
void descomprime (char* cs, char* s) {  
7  /* por hacer */  
   strcat (s, cs);  
9  }
```

```
1  /* comp-decomp-driver.c -- Es el programa principal. */
   #include <stdio.h>
3  #include <stdlib.h>
   #include <string.h>
5
   #include "comp.h"
7  #include "decomp.h"

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

13     if (argc != 3) {
         printf("Uso: comp-decomp [-c|-d] cadena\n");
15         return 1;
     } else
17         if ( (strcmp(argv[1], "-c") != 0) &&
              (strcmp(argv[1], "-d") != 0) )
19             {
                 printf("Uso: comp-decomp [-c|-d] cadena\n");
21                 return 2;
             }
23         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,
25                     strlen(c));
27         }
29         if (strcmp(argv[1], "-d") == 0) {
             c[0] = '\0';
             descomprime (argv[2], c);
31             printf ("Descompresion de \"%s\"(%d) es \"%s\"(%d)\n", argv[2], strlen(argv[2]),
                     c, strlen(c));
33         }
         return 0;
     }
}
```

# Com generem una biblioteca d'enllaç estàtic?

- Compilem els arxius que la componen per separat. De cada un obtenim el seu '.o'.
- Amb l'aplicació 'ar' -similar a tar- creguem l'arxiu 'biblioteca' amb extensió '.a' -a de *arxivo*- així:

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

- On la cadena '**crs**' representa les opcions amb les quals ho cridem:
  - c** Crea l'arxiu amb nom: libcomp-decomp.a
  - r** Reemplaça -si ja existís- en l'arxiu '.a' cadascun dels arxius que s'especifiquen a continuació.
  - s** Crea un índex en l'arxiu '.a' perquè sigui més ràpid accedir als arxius que conté.

# Com enllacem amb una biblioteca d'enllaç estàtic? I

Podem fer-ho de diverses maneres:

- Especificant la seva ruta completa, igual que si fos un arxiu `‘.o’`: `‘/ruta/hasta/fichero.a’`.
- Addicionalment, si el seu nom segueix aquest conveni: `libnombre-bib.a`, el enlazador ho reconeix i es pot posar d'aquesta manera en la línia de enllac: `-lnombre-bib`.
- El *enlazador* és una aplicació diferent al compilador, és independent del llenguatge de programació emprat. Tradicionalment en s.o. de la família UNIX aquest programa es diu `‘ld’`.

# Com enllacem amb una biblioteca d'enllaç estàtic? 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
```

# Com generem una biblioteca d'enllaç dinàmic?

- Igual que abans compilem els arxius que la componen per separat. De cadascun obtenim el seu '.o' però hem de usar l'opció `-fpic`. **P**osition **I**ndependent **C**ode.
- Les biblioteques d'enllaç dinàmic usen l'extensió '.so' en s.o. de la família UNIX. Són equivalents a les 'DLL' de Windows.
- Per generar-les no necessitem de ningun programa nou com abans, n'hi ha prou amb el propi enlazador subministrant-li l'opció '-shared':

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

# Com enllacem amb una biblioteca d'enllaç dinàmic?

```
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
6
7 comp-decomp-driverdin.o: comp-decomp-driverdin.c
8     $(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
12
13 comp-pic.o: comp.c
14     $(CC) -c -o comp-pic.o -fpic $(CFLAGS) comp.c
15
16 decomp-pic.o: decomp.c
17     $(CC) -c -o decomp-pic.o -fpic $(CFLAGS) decomp.c
```

# Aplicacions útils per a arxius '.o', '.a' i '.so' I

Són aplicacions que ens permeten extreure informació d'aquest tipus d'arxius o fins i tot modificar-los de cert manera:

- nm** Llista els símbols dins d'un arxiu binari. A més ens dona certa informació sobre cadascun d'ells.
- ranlib** Crea l'índex en una biblioteca feta amb 'ar', és a dir, té la mateixa funció que l'opció 's' de 'ar'.
- strip** Elimina determinats símbols de l'arxiu binari amb la fi de reduir la seva grandària.



# Aplicacions útils per a arxius '.o', '.a' i '.so' II

Vegem la sortida que produeix 'nm' sobre la biblioteca de enllaç estàtic que hem creat prèviament:

```
1  Executem libcomp-decomp.a

3  comp.o:
    000000000000000000 t  caracteres_iguales
5  000000000000000078 T  comprime
                        U  sprintf
7                        U  strcat
                        U  strlen
9
   decomp.o:
11  000000000000000000 T  descomprime
                        U  strcat
```

# Aplicacions útils per a arxius '.o', '.a' i '.so' III

- Quin efecte té 'strip' en els nostres arxius binaris?
- Bàsicament la grandària final dels mateixos, compara:  
    antes `ls -l libcomp-decomp.a` = 3838 bytes  
    después `strip libcomp-decomp.a,`  
            `ls -l libcomp-decomp.a` = 2536 bytes
- En general pots emprar strip amb qualsevol arxiu binari generat en el procés de compilació/enllaci, fins i tot amb un executable final:  
    antes `ls -l comp-decomp` = 8658 bytes  
    después `strip comp-decomp,`  
            `ls -l comp-decomp` = 6008 bytes
- Després de conèixer 'strip' quin creïs que són els 'targets' **Debug** y **Release** de VisualStudio?

# Aplicacions útils per a arxius '.o', '.a' i '.so' IV

- En el cas d'arxius executables enllaçats dinàmicament amb alguna biblioteca, l'ordre 'ldd' és especialment útil.
- Ens informa d'amb quines biblioteques està enllaçada de forma dinàmica nostra aplicació i si falta alguna; a més ens indica la ruta fins a on està cadascuna d'elles en el sistema de fitxers:

```
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)
```

# Com crear i usar una biblioteca en Vala

- Separem el codi seguint els mateixos criteris que en el exemple en 'C'.
- Per a **generar** la biblioteca emprem una ordre com aquesta:

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

- Mentre que per a **enllaçar** amb ella usarem una ordre com aquesta altra:

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

- Els fitxers '.vapi' són per 'Vala' similars als '.h' per 'C'.

# El codi d'exemple en Vala I

```
1  /* comp.vala */
   namespace CompDecomp {
3     private int caracteres_iguales (string s) {
         int l = 0;
         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 codi d'exemple en Vala II

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

4  int main(string[] args) {
    if (args.length != 3) {
6      stdout.printf ("Us: comp-decomp [-c|-d] cadena\n");
      return 1;
    } else if ( (args[1] != "-c") && args[1] != "-d" ) {
8      stdout.printf("Us: comp-decomp [-c|-d] cadena\n");
10     return 2;
    }
12     if (args[1] == "-c") {
        string c="";
14         comprime (args[2], ref c);
        stdout.printf ("Compresion de \"%s\"(%d) es \"%s\"(%d)\n",
16                     args[2], args[2].length,
                        c,          c.length);
18     }
20     return 0;
}
```