Tema: VI.
Biblioteques: creació i ús

Herramientas Avanzadas para el Desarrollo de Aplicaciones

Departament de Llenguatges i Sistemes Informàtics
Universitat d'Alacant

Contingut

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*[1] és un compendio de recursos: subprogrames, classes, dades, etc...
- Quan distribuímos 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.

[1] Al llarg del tema emprarem el terme '*biblioteca*' en lloc de '*llibreria*' per ser el primer més apropiat.

Perquè distribuir alguna cosa en format binari?

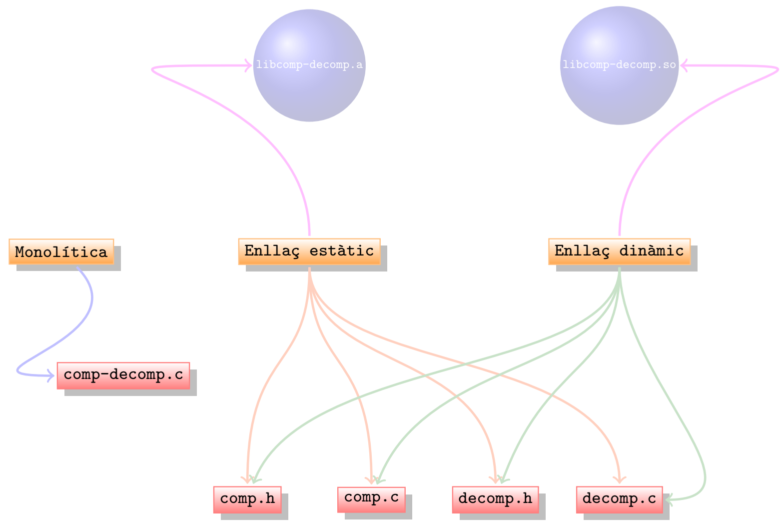
Per diversos motius:

- 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.

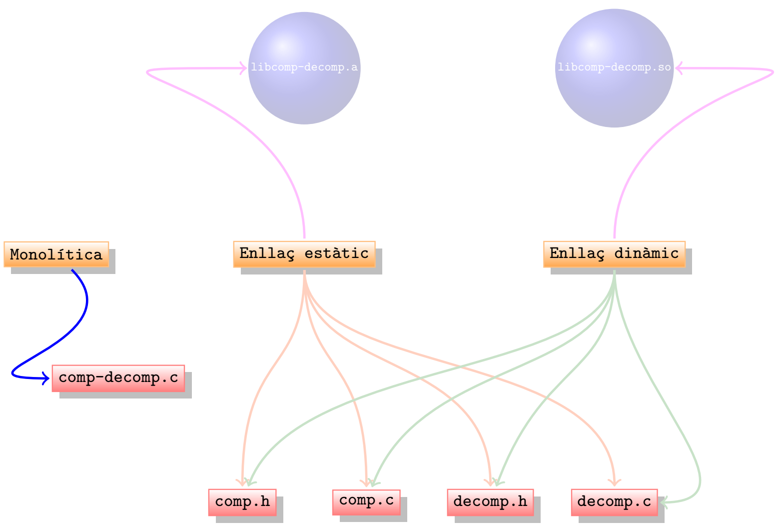
Exemple senzill I

- 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 ccccaasssssssssaaaaaaa`
  produeix aquesta sortida:
  `Compresion de ''ccccaasssssssssaaaaaaa''(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*.

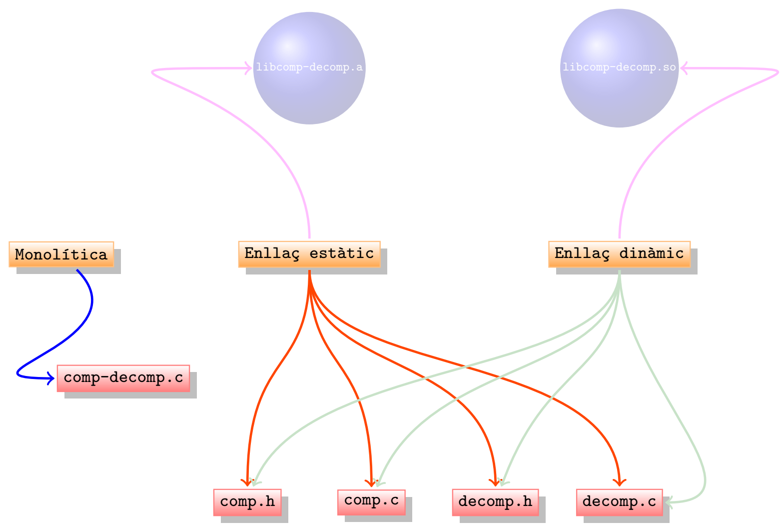
Exemple senzill II



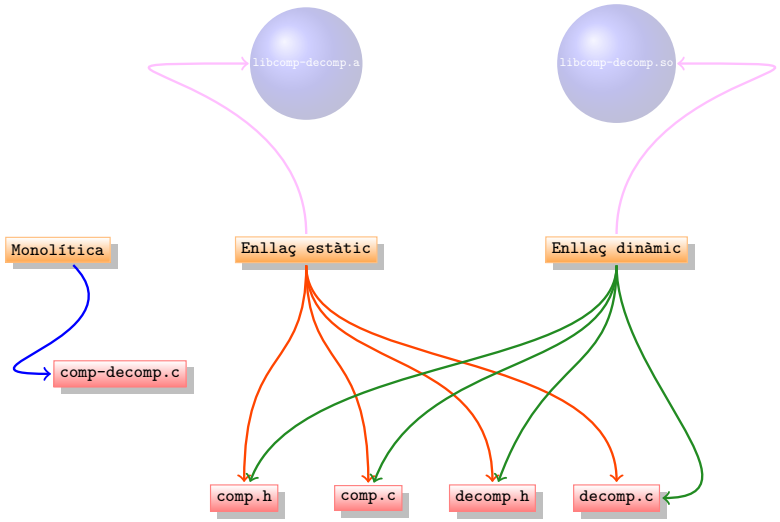
Exemple senzill II



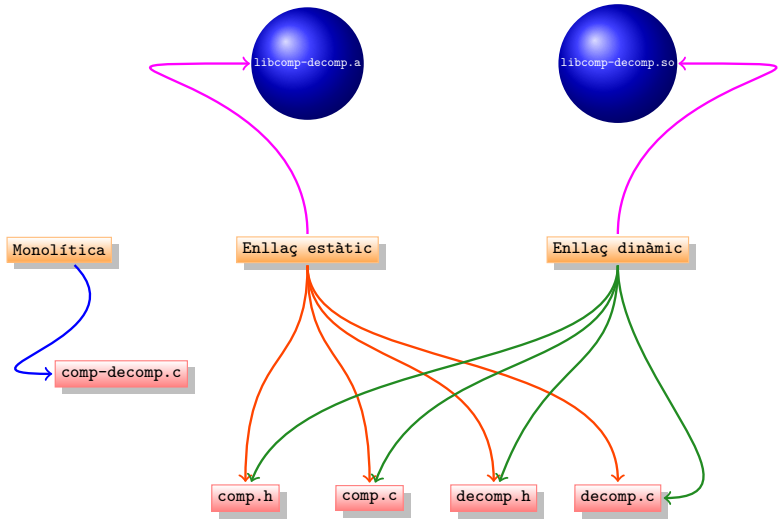
Exemple senzill II



Exemple senzill II



Exemple senzill II



Codi exemple monolític I (comp-decomp.c)

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

/* Retorna el numero de caracters iguals al primer 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;
  }
}
```


Codi exemple monolític II (comp-decomp.c)

```c
/* Retorna en 'cs' la cadena comprimida de 's'. */
void comprime (char* s, char* cs) {
  int l = cont = i = 0;
  char num[5], *s2;

  if (s == NULL)  return;

  l = strlen (s);
  if (l < 3) strcat (cs, s);
  else {
    int csl = 0;
    cont = caracteres_iguales (s);
    if (cont > 2) {
      sprintf (num, "%d", cont);
      strcat (cs, num);
      csl = strlen(cs);
      cs[csl++] = s[0];
      cs[csl] = '\0';
    } else {
      csl = strlen(cs);
      for (i = 0; i < cont; i++)
        cs[csl++] = s[0];
      cs[csl] = '\0';
    }

    s2 = &s[cont];
    comprime (s2, cs);
  }
}
```

Codi exemple monolític III (comp-decomp.c)

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

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;
}
```


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.

comp.h y comp.c I

```c
/* comp.h */

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

```c
/* comp.c */

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

/* Es privada, la biblioteca no la exporta. (modificador static) */
static int caracteres_iguales (char* s) {
  int l = 0, 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;
  }
}
```


comp.h y comp.c II

```c
/* comp.c */

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

  if (s == NULL) return;

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

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

decomp.h y decomp.c

```c
/* decomp.h */

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

```c
/* decomp.c */

#include <string.h>

/* Retorna 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;
}
```


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 enlace: `-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

```
#######################
# Biblioteca estatica #
#############################################################
comp-decomp-estatico: comp-decomp-driver.o libcomp-decomp.a
    $(CC) -static comp-decomp-driver.o -L . -lcomp-decomp -o comp-decomp-estatico

comp-decomp-driver.o: comp-decomp-driver.c
    $(CC) -c $(CFLAGS) comp-decomp-driver.c

libcomp-decomp.a: comp.o decomp.o
    $(AR) crs libcomp-decomp.a comp.o decomp.o

comp.o: comp.c
    $(CC) -c $(CFLAGS) comp.c

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`. `pic`: Position Independent Code.
- 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?

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

comp-decomp-driverdin.o: comp-decomp-driverdin.c
    $(CC) -fpic -c $(CFLAGS) comp-decomp-driverdin.c

libcomp-decomp.so: comp-pic.o decomp-pic.o
    $(CC) -shared -o libcomp-decomp.so comp-pic.o decomp-pic.o

comp-pic.o: comp.c
    $(CC) -c -o comp-pic.o -fpic $(CFLAGS) comp.c

decomp-pic.o: decomp.c
    $(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 dóna 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:

```
Executem libcomp-decomp.a

comp.o:
0000000000000000 t caracteres_iguales
0000000000000078 T comprime
                 U sprintf
                 U strcat
                 U strlen

decomp.o:
0000000000000000 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-decom.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-decom`,
  `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
    linux-vdso.so.1 =>  (0x00007fffa57ff000)
    libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f38c1b46000)
    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

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

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

  public void comprime (string s, ref string cs) {
    int l = 0, cont = 0;
    string num, s2;

    l = s.length;
    if (l < 3) cs += s;
    else {
      cont = caracteres_iguales (s);
      if (cont > 2) {
        num = cont.to_string();
        cs += num; cs += s[0].to_string();
      } else { cs += string.nfill (cont, s[0]); }
      s2 = s[cont:s.length];
      comprime (s2, ref cs);
    }
  }
}
```


El codi d'exemple en Vala II

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

int main(string[] args) {
  if (args.length != 3) {
    stdout.printf ("Us: 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;
}
```