Computadors
Fonaments de la Programació (2/2)

Grau en Ciència i Enginyeria de Dades

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC)

Creative Commons License

Aquest document utilitza Creative Commons Attribution 3.0 Unported License

Els detalls d'aquesta licencia es poden trovar a
https://creativecommons.org/licenses/by-nc-nd/4.0

Index

- Gestió de Projectes de Software
- **Programació i Compilació**

Index

- Introducció
- Compiladors i el mecanisme de traducció
- Estructura del fitxer executable
- Debugger
- Intèrprets vs compiladors

Definim les bases de l'exemple

Codi d'Alt Nivell (hello.c)

```
#include <unistd.h>
#include <stdio.h>

int global = 4;

int main(int argc, char **argv){
        char buf[512];
        int num;

        num = sprintf(buf, "Hello World %d!\n", global);
        write(1, buf, num);

        return 0;
}
```

El Codi de Llenguatge d'Alt Nivell no pot ser executat directament per la CPU perquè no ho entén

Analitzem el codi de l'exemple

```
#include <unistd.h>
#include <stdio.h>

int global = 4;

int main(int argc, char **argv){
        char buf[512];
        int num;

        num = sprintf(buf, "Hello World %d!\n", global);
        write(1, buf, num);

        return 0;
}
```

Fitxers Capçalera

Variable Global

Variables Locals

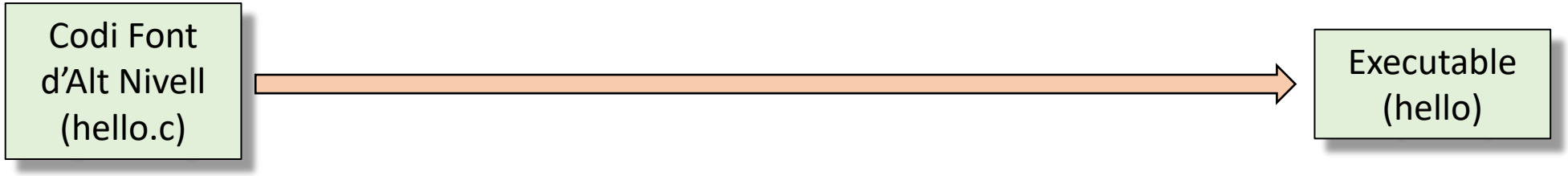
Crida a Funcions Externes

Valor que retorna el programa quan finalitza

Els Fitxers Capçalera (Header files) tenen l'información necesaria pel compilador sobre els components externs utilitzats en aquest codi, com ara les crides a "sprintf" i "write"

De codi d'alt nivell a codi executable

- **Com podem transformar codi font a un fitxer executable?**

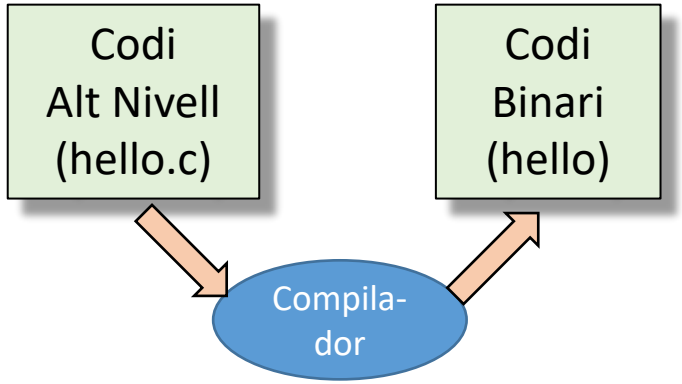


- **Codi font:** programa que pot ser escrit/modificat i entès per un humà
- **Fitxer executable:** programa que pot ser executat per un ordinador

De codi d'alt nivell a codi executable

- **Compilador:** software que tradueix un codi implementat amb un llenguatge a un altre llenguatge
  - Normalment tradueix de codi d'alt nivell a un de baix nivell

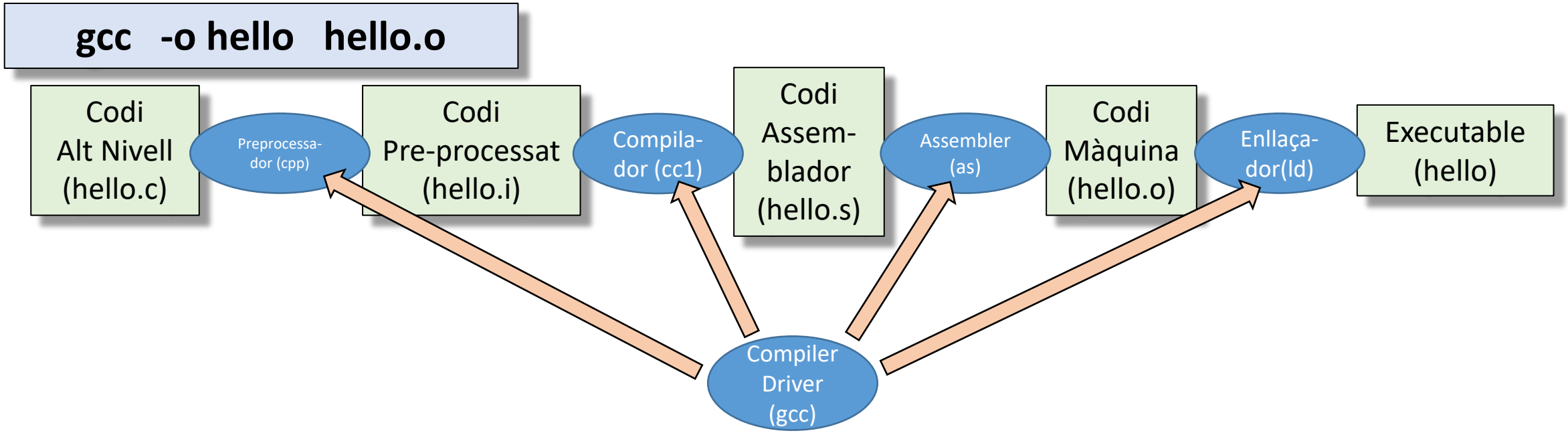
```
gcc   -o hello   hello.c
```



  - **Passos de compilació:** tasques complexes que poden ser dividides en passos més petits

De codi d'alt nivell a codi executable

- **Etapes de la compilació: preprocessar, compilar, assemblar i enllaçar**



- **Compiler driver:** programa que invoca els components de les etapes de compilació

De codi d'alt nivell a codi executable

- **Pre-processador:** processa directives del llenguatge (#include, #define…)
- **Compilador:** genera codi assemblador
- **Assemblador:** software que tradueix codi assemblador a codi màquina
  - El resultat es diu **fitxer objecte** (amb extensió **.o**)
  - **Fitxer Objecte:** combinació de instruccions codi máquina, dades i informació necessaria per posar les instruccions a memòria
    - Codi relocatable (ho veurem en properes transparències)
  - **Fitxer Biblioteca (Library):** combinació de fitxers objecte
- **Enllaçador:** programa que combina codis independents (fitxer objecte i biblioteques) per crear un executable, mentres resol tots els símbols sense definir
  - Fitxer Executable: programa que pot ser executat en un ordinador. Té el format d'un fitxer objecte, però les referències están resoltes. No hi han símbols sense definir.

Exemple de Codi Assemblador

```
...
        xorl    %eax, %eax
        movl    global(%rip), %edx
        leaq    -528(%rbp), %rax
        leaq    .LC0(%rip), %rsi
        movq    %rax, %rdi
        movl    $0, %eax
        call    sprintf@PLT
        movl    %eax, -532(%rbp)
        movl    -532(%rbp), %eax
        movslq  %eax, %rdx
        leaq    -528(%rbp), %rax
        movq    %rax, %rsi
        movl    $1, %edi
        call    write@PLT
...
```

Optimitzacions del Compilador

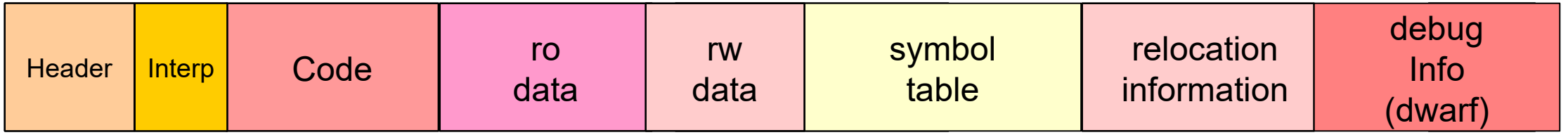
- -O**x** on **x** és un nombre que indica el tipus d'optimització
- Per exemple, el compilador gcc/g++ té les següents opcions:
  - -O0: sense optimitzacions. Redueix el temps de compilació i preserva el comportament per depurar
  - -O1, or -O: intenta reduir la mida del codi i el temps d'execució, sense fer optimitzacions que agafin molt de temps de compilació
  - -O2: fa gairebé totes les optimitzacions possibles que no impliquin un compromís entre espai i velocitat. Comparat amb –O1, aquesta opció augmenta tant el temps de compilació com el rendiment del codi generat
  - -O3: habilita totes les optimitzacions anteriors, així com altres de més avançades: function inlining, register renaming i loop unrolling entre altres

https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

https://gcc.gnu.org/onlinedocs/gcc/gcc-command-options/options-that-control-optimization.html

Fitxers Objecte / executables (codi màquina)

- Fitxers objecte/executables per SO tipus UNIX inclouen:
  1) **Capçalera del fitxer Object:** mida i localització de les altres regions del fitxer objecte
  2) **Segment de Text:** codi máquina (instruccions)
  3) **Segment Estàtic de Dades:** dades que duran tota la vida del programa (e.g. variables globals)
  4) **Taula de Símbols:** una taula que ajuda a saber l'ubicació de variables i funcions que poden ser referenciades des d'altres fitxers objecte
  5) **Registres/Informació de reubicació (relocation):** informació sobre adreces referenciades en el fitxer objecte
     1) L'enllaçador ho assigna una vegada es resol les ubicacions de memòria
     2) Mentre es carrega l'executable, el carregador aplica reubicacions si les adreces han canviat
  6) **Informació de Depuració:** dades addicionals per una eina que es diu depurador (debugger), que serveixen per relacionar instruccions máquina a codi d'alt nivell



Exemple de Fitxer Objecte

- Contingut de hello.o (od –t x1c hello.o)

```
0000000  7f  45  4c  46  02  01  01  00  00  00  00  00  00  00  00  00
       177   E   L   F 002 001 001  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000020  01  00  3e  00  01  00  00  00  00  00  00  00  00  00  00  00
       001  \0   >  \0 001  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
0000040  00  00  00  00  00  00  00  00  20  03  00  00  00  00  00  00
        \0  \0  \0  \0  \0  \0  \0  \0      003  \0  \0  \0  \0  \0  \0
0000060  00  00  00  00  40  00  00  00  00  00  40  00  0d  00  0c  00
        \0  \0  \0  \0   @  \0  \0  \0  \0  \0   @  \0  \r  \0  \f  \0
0000100  ...
0000200  48  65  6c  6c  6f  20  57  6f  72  6c  64  20  25  64  21  0a
         H   e   l   l   o       W   o   r   l   d       %   d   !  \n
0000220  00  00  47  43  43  3a  20  28  53  55  53  45  20  4c  69  6e
        \0  \0   G   C   C   :       (   S   U   S   E       L   i   n
0000240  75  78  29  20  38  2e  31  2e  30  00  00  00  00  00  00  00
         u   x   )       8   .   1   .   0  \0  \0  \0  \0  \0  \0  \0
0000260  14  00  00  00  00  00  00  00  01  7a  52  00  01  78  10  01
       024  \0  \0  \0  \0  \0  \0  \0 001   z   R  \0 001   x 020 001
```

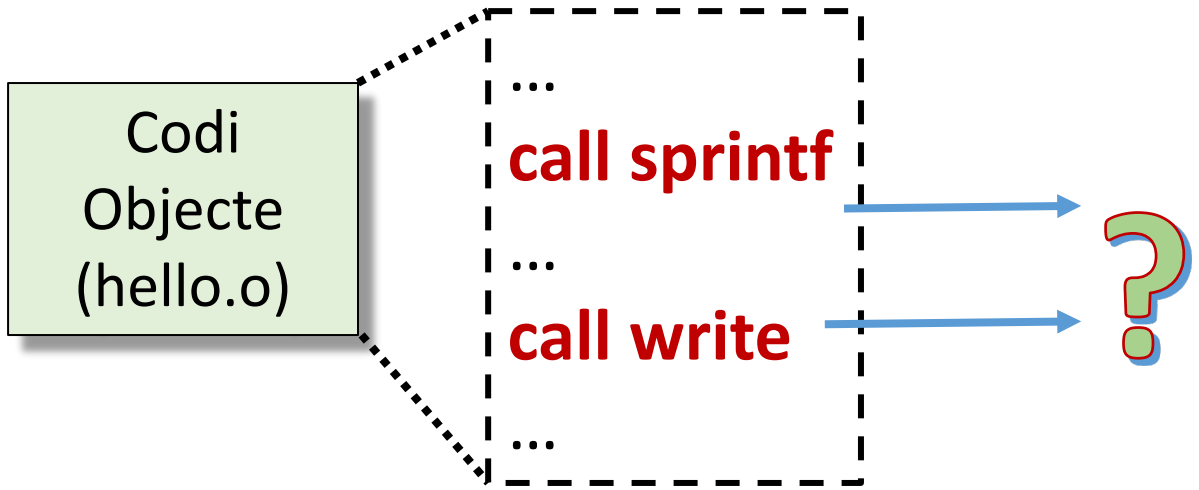
Exemple de Fitxer Objecte

- objdump -xs hello.o

```
Contents of section .text:
 0000 4881ec08 0200008b 15000000 00be0000  H...............
 0010 00004889 e7b000e8 00000000 4863d048  ..H.........Hc.H
 0020 89e6bf01 000000e8 00000000 b8000000  ................
 0030 004881c4 08020000 c3                 .H.......
Contents of section .data:
 0000 04000000                             ....
Contents of section .rodata.str1.1:
 0000 48656c6c 6f20576f 726c6420 2564210a  Hello World %d!.
 0010 00                                   .
Contents of section .comment:
 0000 00474343 3a202853 55534520 4c696e75  .GCC: (SUSE Linu
 0010 78292038 2e312e30 00                 x) 8.1.0.
Contents of section .eh_frame:   // exception support
 0000 14000000 00000000 017a5200 01781001  .........zR..x..
```

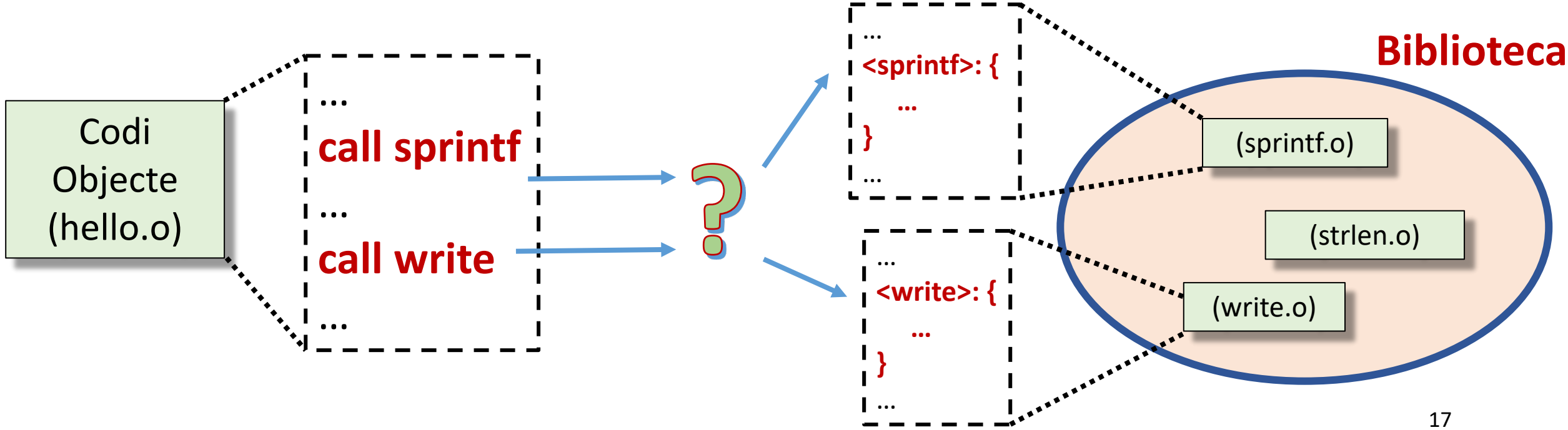
De codi d'alt nivell a codi executable

- Els codis objecte acostumen a fer crides a funcions que estan en altres fitxers objecte o inclús en biblioteques (libraries)



Biblioteques

- **Biblioteca (library):** fitxer que encapsula un conjunt de fitxers objecte
  - Les biblioteques ofereixes **independència**
    - Un codi d'alt nivel és independent del **SO** i de **l'arquitectura del Hw**, però …
    - … els executables es crean per un **SO** i una **arquitectura de Hw** en particular



Tipus de Biblioteques

- Biblioteques de Llenguatge
  - Proporciona funcions del llenguatge de programació
  - Relaciona un llenguatge a un SO en particular, però és independent de l'arquitectura Hw
  - De vegades són auto-contingudes (e.g. **math**): independent del SO
  - De vegades demana serveis al SO (e.g. **sprintf**, cin, cout…)

- Biblioteques del Sistema
  - Inclouen funcions que invoquen serveis/rutines del SO (e.g. **write**, open, …)
    - **Crides a Sistema (syscalls)** que les veurem en el tema de SO
  - Relaciona un SO a una arquitectura Hw en particular (independent del codi d'alt nivell)
    - Depèn del tipus de SO i de l'arquitectura de la CPU

- Exemples d'incompatibilitat
  - Linux 32 vs 64 bits, CPU Intel vs ARM, Ubuntu vs OpenSUSE, …

Linux i386 (32bits) vs Linux x64 (64bits)

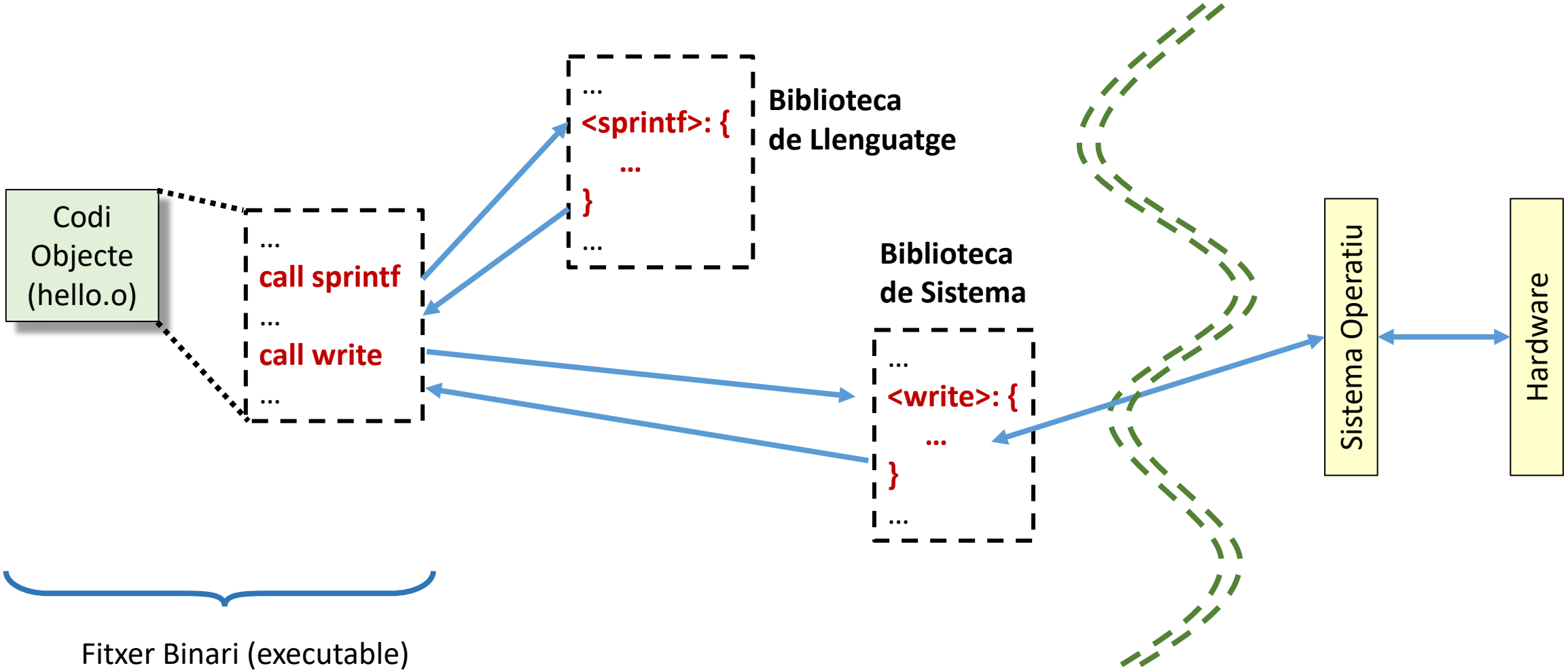
Linux i386

```
...
movl  $4, %eax      ; use the write syscall
movl  $1, %ebx      ; write to stdout
movl  $msg, %ecx ; use string "Hello World!\n"
movl  $13, %edx   ; write 13 characters
int     $0x80        ; make syscall
...
```

Linux x64

```
...
movq $1, %rax     ; use the write syscall
movq $1, %rdi     ; write to stdout
movq $msg, %rsi ; use string "Hello World!\n“
movq $13, %rdx   ; write 13 characters
syscall                ; make syscall
...
```

Analitzem les crides a funcions externes

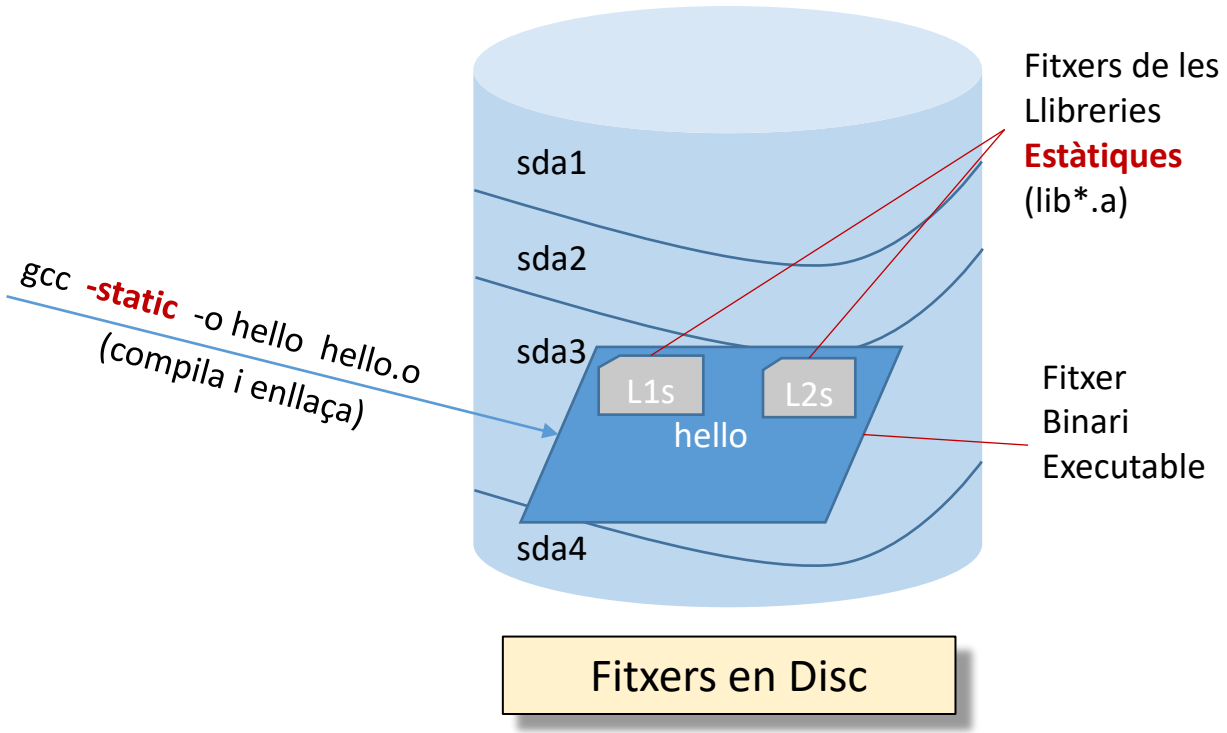


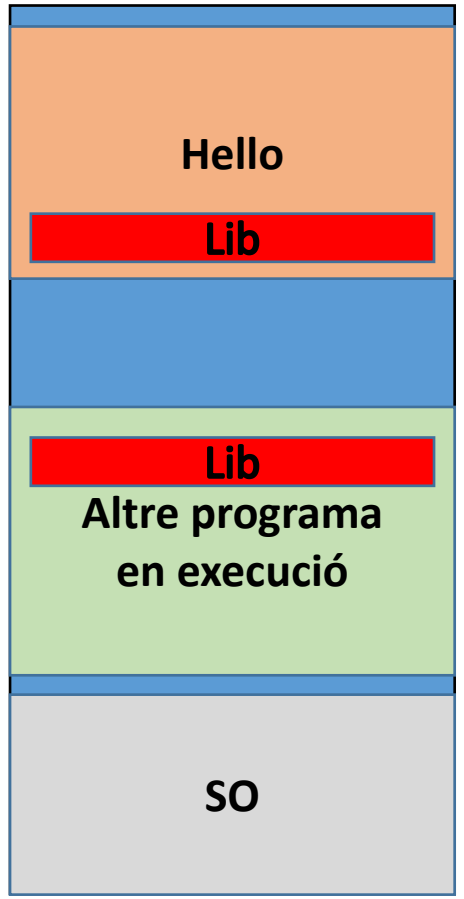
Enllaçament Estàtic versus Dinàmic

- Enllaçar
  - Fer accessible a un codi compilat el codi pre-compilat que inclou les llibreries
- Enllaçament Estàtic
  - El codi de les llibreries s'inclouen DINS de l'executable
  - Malgasta espai (en disc quan es guarden, i en memoria quan s'executen)
    - Diferents programes poden necesitar la mateixa llibreria, però es carreguen múltiples vegades (dins de cada executable)
  - Llibraries estàtiques: "**.a**" en SO basats en UNIX; "**.lib**" en SO basats en Windows
- Enlaçament Dinàmic
  - Part del procés d'enllaçament es endarrerit fins al moment d'execució. El codi no s'inclou en l'executable
  - Compartició de llibreries
  - Estalvia espai tant en disc com memòria
    - Diferents programes poden accedir a una mateixa llibreria compartida carregada una única vegada en memòria
  - Llibreries dinàmiques: "**.so**" en SO basats en UNIX; "**.dll**" en SO basats en Windows

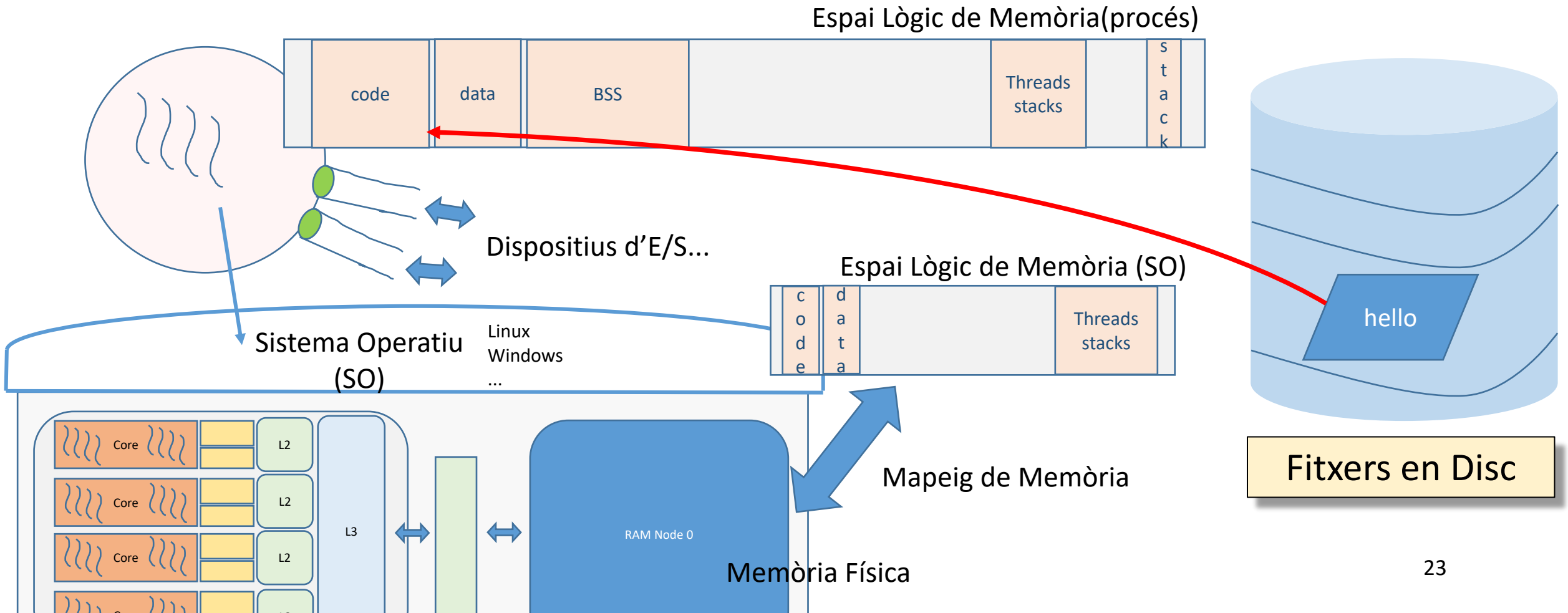
Gestió de biblioteques enllaçades estàticament

- Programes en execució que han sigut compilats amb enllaçament estàtic no comparteixen les biblioteques
  - Les biblioteques en realitat estan incrustades dins del programa



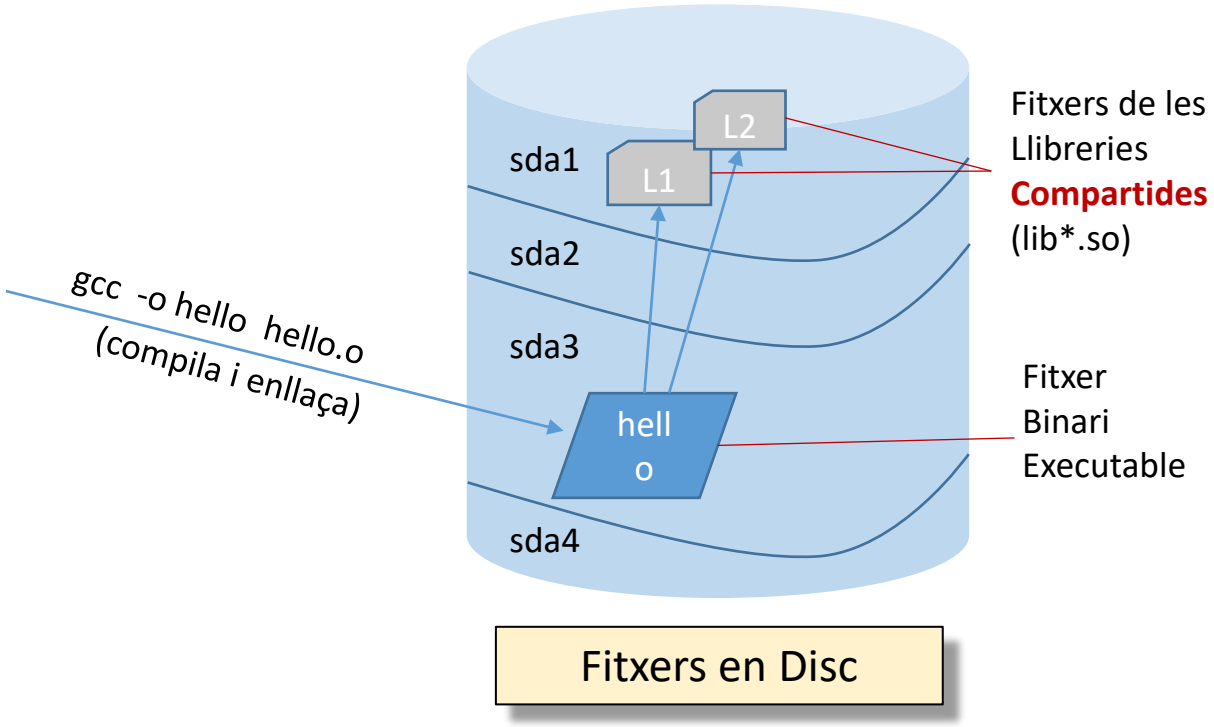


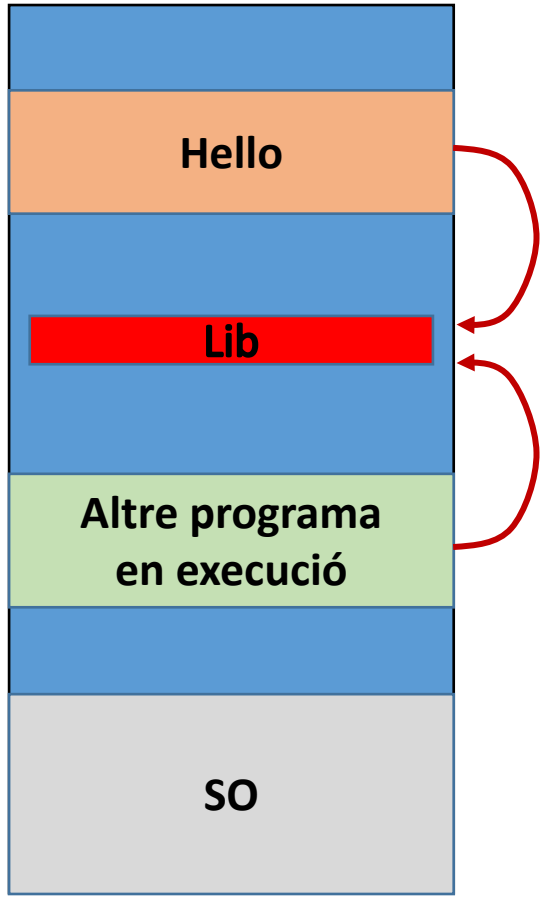
Procés amb llibreries estàtiques



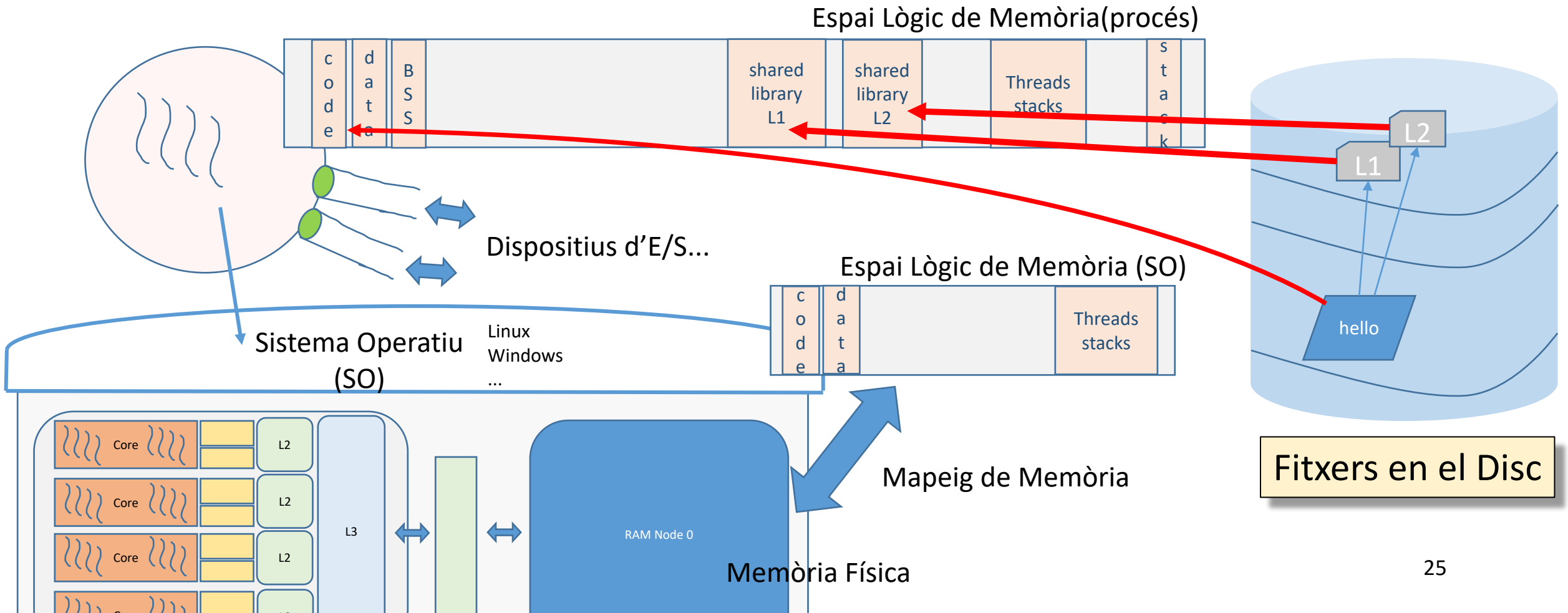
Gestió de biblioteques enllaçades dinàmicament

- Programes en execució que han sigut compilats amb enllaçament dinàmic poden compartir les biblioteques
  - Les biblioteques es carreguen una única vegada





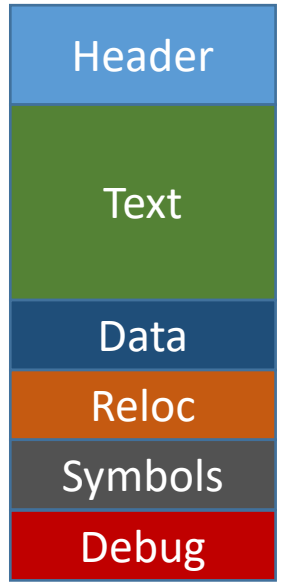
Procés amb biblioteques compartides (Enllaçat Dinàmic)



Fitxers Objecte / executables (codi màquina)

- **Fitxer Executable:** programa que pot ser executat en un ordinador. Té el format d'un fitxer objecte, però les referències estàn resoltes. No hi han símbols sense definir.
  - Seccions vistes en el fitxer objecte
    - Header, text (code), data, relocation information, symbol table, debug info
  - És possible que hi hagin adreces no resoltes
    - Apunten a rutines de biblioteques

X està definit a @1 en Data
Y està definit a @2 en Data
A està definit a @3 en Text
B està definit a @4 en Text



Fitxers executables són reubicables amb símbols associats a adreces
Taula de Símbols té símbols amb adreces de memòria assignades
L'enllaçador ha resolt adreces no definides, excepte pels símbols de biblioteques que s'enllacen dinàmicament

Executables

- Estructures específiques que depenen del SO
  - Windows – PE32 Portable Executable, PE32+ per 64 bits
  - Linux – ELF Executable and Linkable Format
    - x86_64 dades de 64-bits i adreces de 64-bits
    - x32 dades de 64-bits i adreces de 32-bits
    - i386 dades de 32-bits i adreces de 32-bits (compatible amb sistemes 32-bits)
- La capçalera (header) de l'executable té informació sobre el programa
  - Utilitzat per...
    - ...les eines ELF per llegir/entendre/escriure aquest tipus de fitxers
    - ...el SO per carregar els fitxers binaris i biblioteques per procedir amb l'execució

https://en.wikipedia.org/wiki/Portable_Executable

https://en.wikipedia.org/wiki/Executable_and_Linkable_Format

Exemple de Fitxer Executable

```
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x400470
  Start of program headers:          64 (bytes into file)
  Start of section headers:          6672 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         9
  Size of section headers:           64 (bytes)
  Number of section headers:         31
  Section header string table index: 28
```

Informació de Depuració

- DWARF (http://dwarfstd.org): per mantenir la temàtica de noms… ELF
  - Debugging With Attributed Record Formats
- S'inclou amb l'opció **–g** de gcc/g++
- Varies subseccions
  - Números de línia de codi font
  - Informació de crida a funcions
  - Tipus de dades
  - Enumeracions
  - Strings
  - Estructures, unions, classes
  - Funcions
  - …

Depurar el Codi (Debugging)

- Un depurador (debugger) és una eina que executa un procés sota unes condicions controlades per ajudar als programadors a trovar errors (bugs)
  - Pot pausar/continuar l'execució i analitzar valors, com ara variables, parametres i adreces de memòria, entre altres
  - Però té certes limitacions i requisits, com ara dificultats per analizar execucions paral·leles, dependència del llenguatge i complexitat per saber fer-ne ús correcte de l'eina

- Un programa…
  - Sense información de depuració en l'executable pot ser depurat, però és més difícil perquè no hi ha información addicional per entendre més fàcilment les instruccions de baix nivell
  - Amb información de depuració en l'executable pot ser depurat més fàcilment pel depurador, ja que mostra información addicional que relaciona codi de baix nivel amb línies de codi d'alt nivell

- Examples: **GDB – The GNU Debugger**, PDB (The Python Debugger), Visual Studio

https://www.gnu.org/software/gdb/

Intèrpret vs. Compilador

- Compilador: les etapes de compilació es fan per separat de l'execució
  - Compilar una única vegada, però executar moltes

- Intèrpret:
  - AOT: Ahead-of-Time
    - Compilar el programa abans d'executar: es fa en temps de compilació
  - JIT: Just-in-Time
    - Compilar el programa mentre s'executa: es fa en temps d'execució
  - Passos per executar un programa interpretat:

    Source code →AOT→ **Bytecode** (codi intermedi) →JIT→ **Native code**

Intèrpret vs. Compilador

- Casos d'ús
  - Java
    - Java source code -> Compilat a -> Java Bytecode -> Interpretat per: Java Virtual Machine
  - **Python**
    - **Python -> Compilat a -> CPython -> Interpretat per: Python Interpreter**
  - Microsoft .NET
    - Any .NET language -> Compilat a -> CIL (Common Intermediate Language) -> Interpretat per: CLR (Common Language Runtime)

Exemple de biblioteca estàndard de Python

- Distribuït amb l'entorn d'execució de Python
  - Strings
  - Números i matemàtiques
  - Accés a fitxers i directoris
  - Compressió de dades
  - Serveis del SO
  - Comunicacions i protocols
  - ...

https://docs.python.org/2/library/

https://docs.python.org/3/library/

Intèrpret vs. Compilador

- Intèrpret
  - Executa instruccions directament des del codi font
  - Poc temps per analitzar el codi font, però el programa executa més lent
  - No es crea un fitxer objecte
  - Executa fins el primer error → fàcil per depurar
  - E.g. Python, MATLAB, R, Javascript
- Compilador
  - Escaneja tot el codi i tradueix a codi máquina
  - Molt de temps per analitzar el codi font, però el programa executa més ràpid
  - Genera fitxers objecte
  - Missatges d'error després d'escanejar tot el programa
  - E.g. C, C++, FORTRAN

Bibliografia

- GCC (The GNU Compiler Collection) Tutorial
  - http://gcc.gnu.org/
  - https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html
- GDB (The GNU Project Debugger) Tutorial
  - https://www.sourceware.org/gdb/
    - Manual (Web & PDF)
  - https://gcc.gnu.org/onlinedocs/gcc/Debugging-Options.html
- Python Tutorial
  - https://docs.python.org/3/tutorial/
- PDB (The Python Debugger) Tutorial
  - https://docs.python.org/3/library/pdb.html