

Systemes Informatiques

Construction d'un programme et exécution d'un processus

G.BERTHELOT

Compilation : historique

```
int globA;  
int globC =12;  
int exA=128;
```

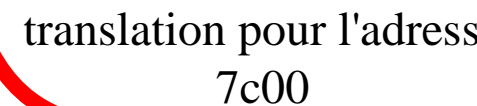
```
int main (){  
globA= exA+globC;  
return globA;  
}
```

compilation => fichier translatable

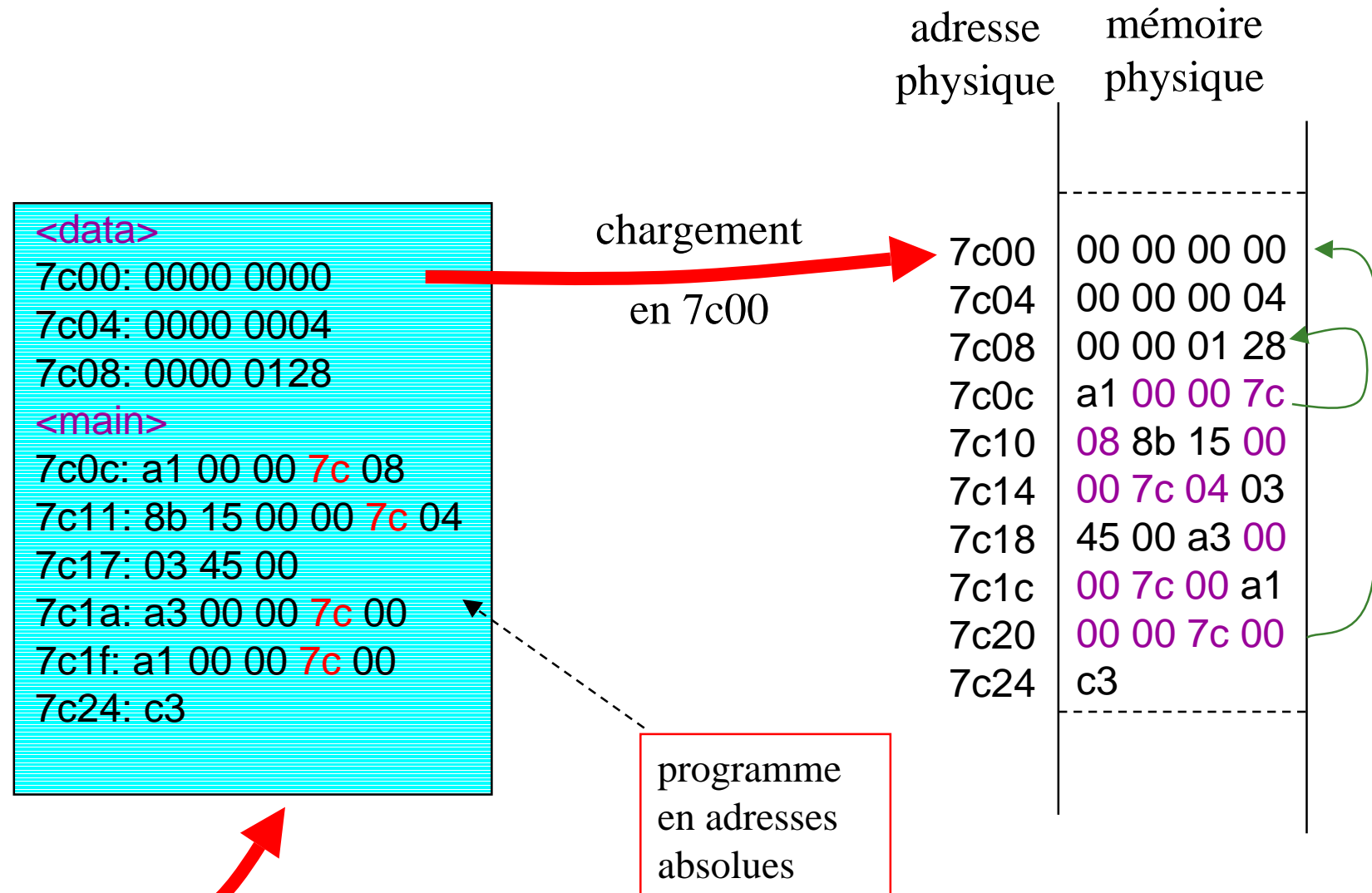


```
<data>  
00: 0000 0000      globA  
04: 0000 0004      globC  
08: 0000 0128      exA  
<main>  
0c: a1 00 00 00 08      mov  exA,%eax  
.11: 8b 15 00 00 00 04      mov  globC, %edx  
17: 03 45 00              add  %edx,%eax  
1a: a3 00 00 00 00      mov  %eax, globA  
1f: a1 00 00 00 00      mov  globA,%eax  
24: c3                  ret
```

translation pour l'adresse
7c00



Chargement : historique

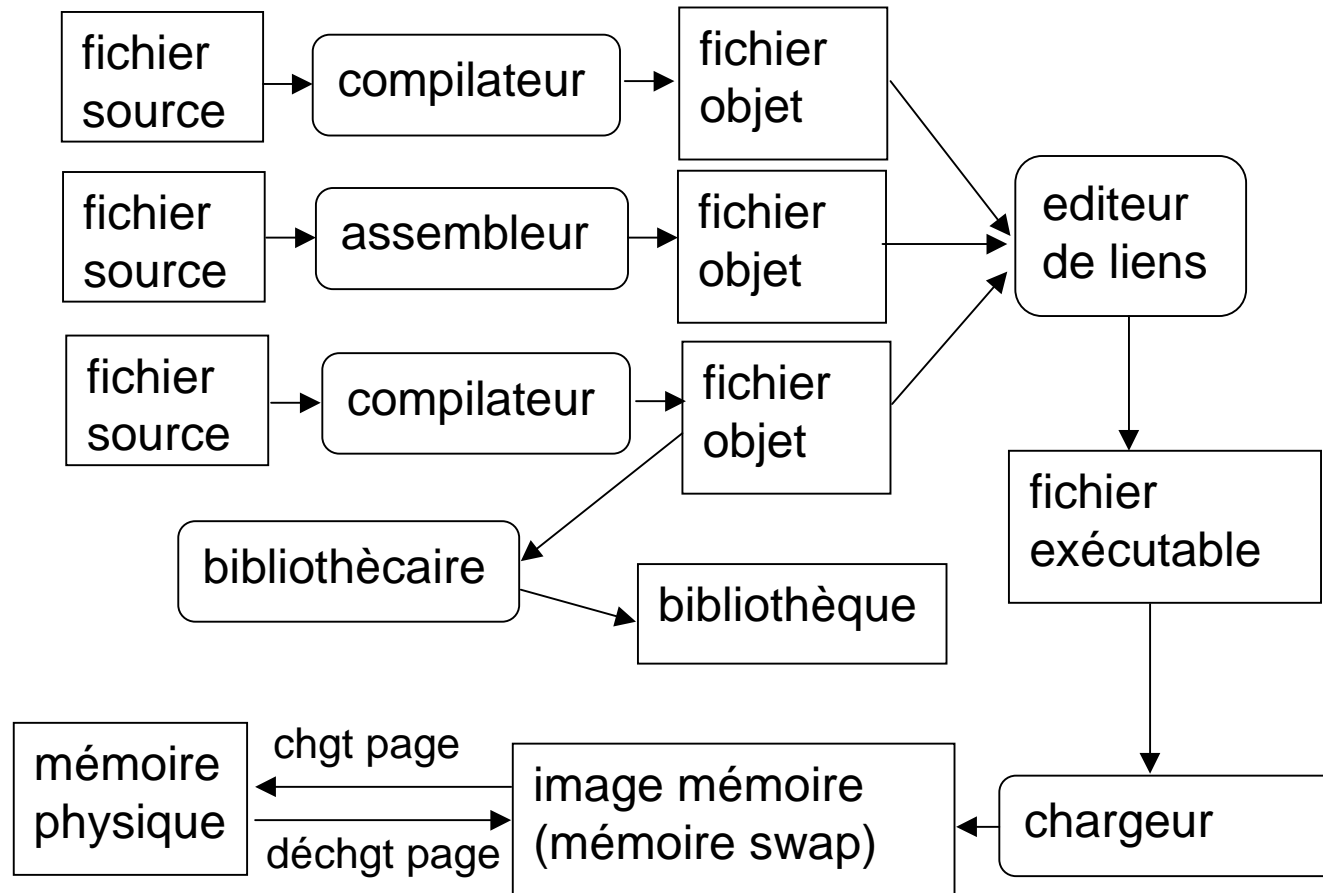


mode historique : conclusion

■ Remarques

- pas d'appel de fonctions non définies
- échange avec les périphériques inclus dans le programme ou si BIOS (ou BIOS+noyau)) installé, le programme doit contenir les trappes appel BIOS (ou appel noyau)
- méthode toujours utilisée pour le démarrage (boot) sur machine nue
 - pas de mémoire virtuelle (mode réel)
 - installation d'une gestion minimum des IT

Schéma général



exécutable, construction

- plusieurs fichiers sources => un fichier exécutable :
- 4 étapes (regroupées dans CC)
 - preprocessing
 - compilation
 - assembleur
 - édition des liens

- Préprocessing (commande cpp)
 - inclusion des fichiers (pas seulement xxx.h) aux endroits indiqués par `#include xxx`
 - remplacement des CdC concernées par les `#define`
 - ex `#define TVA 19,6`
 - traitement des inclusions conditionnelles
 - `#ifdef TVA`
 - `#endif`
 - `#undef TVA`
 - `#if <val_bool>`
 - ...
 - `#else` (ou `#elif`)
 - ...
 - `#end if`

exécutable, construction

■ Préprocessing

□ principales options cpp

- -D name=définition

ajoute une définition comme #define name définition

- -I dir

spécifie un répertoire de recherche pour les includes

ex #include <string.h>

- I /usr/include/

recherchera le fichier stdio.h dans /usr/include

(rappel : "mon_entete.h" est recherché dans le répertoire courant)

construction d'un fichier objet (relogeable)

■ 2 : compilation (commande cc1)

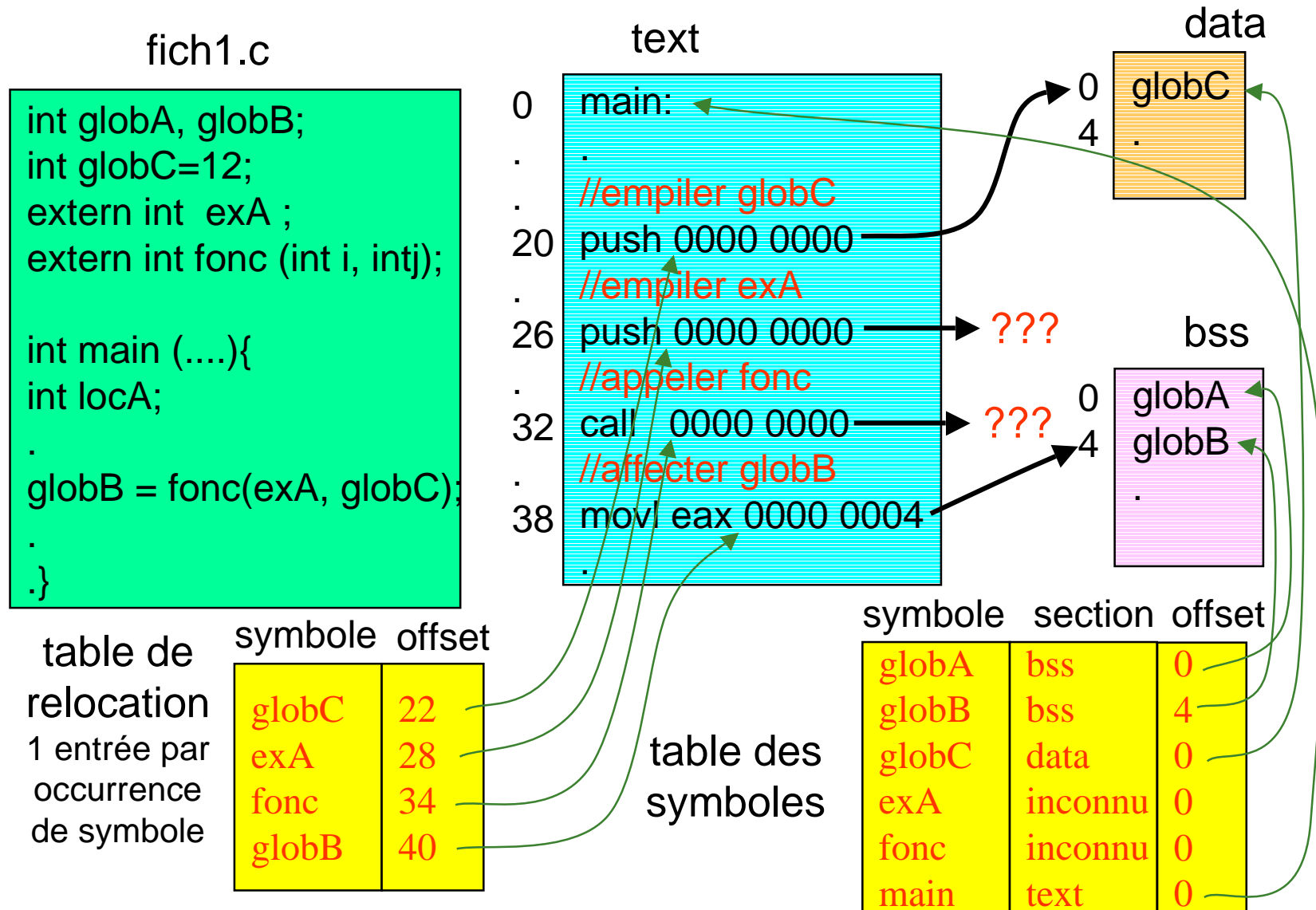
prend un fichier source en langage c et produit un fichier en assembleur

traduction de la suite d'instructions en langage de haut niveau en une suite équivalente d'instructions en langage assembleur,

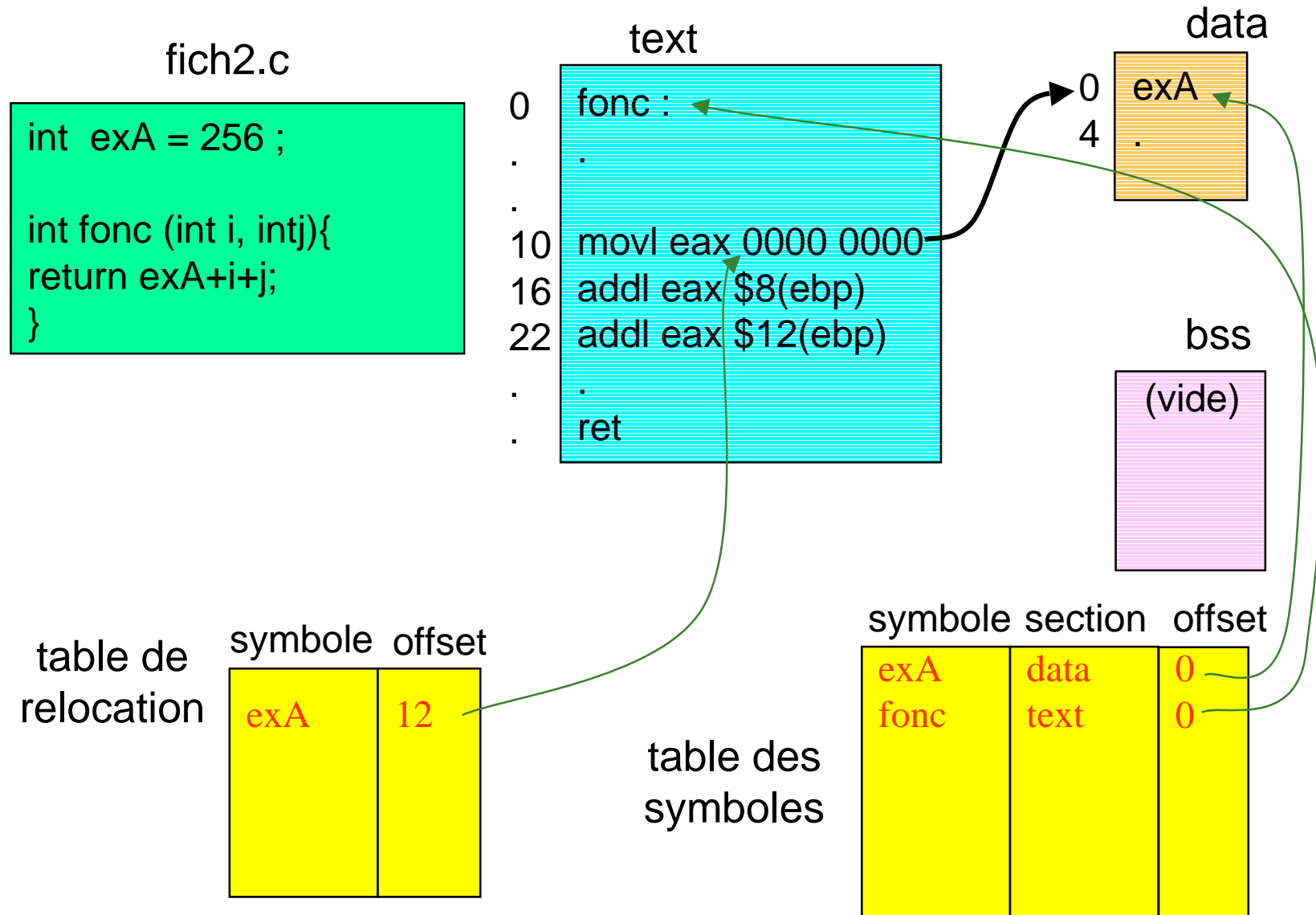
■ 3 : assemblage (commande as)

prend un fichier en assembleur et produit un fichier objet
traduction des instructions assembleur en instructions machine

sections et tables



sections et tables



■ 4 : édition des liens (commande ld)

rassemble différents fichiers objets ou bibliothèques pour faire un programme exécutable

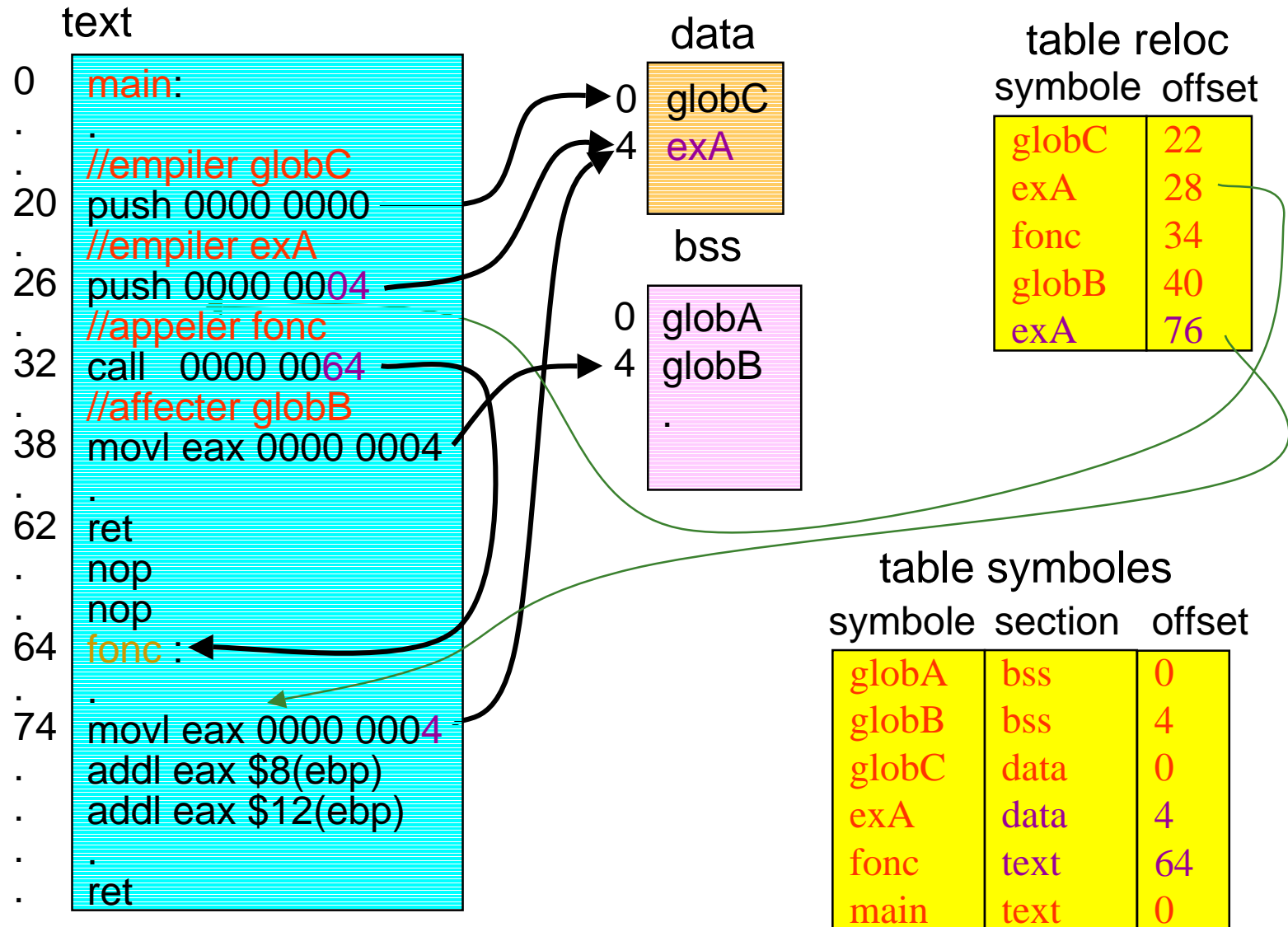
□ principales options

- -entry spécification du point d'entrée
- -L ou --library-path spécification d'un répertoire de recherche pour les bibliothèques (librairies)
- -T ou --section-start positionnement de sections

□ bibliothèques

- collection de fichiers objets produit par l'utilitaire ar l'éditeur de liens n'y prend que ce qui est nécessaire.
- ordre de parcours important

fichier exécutable

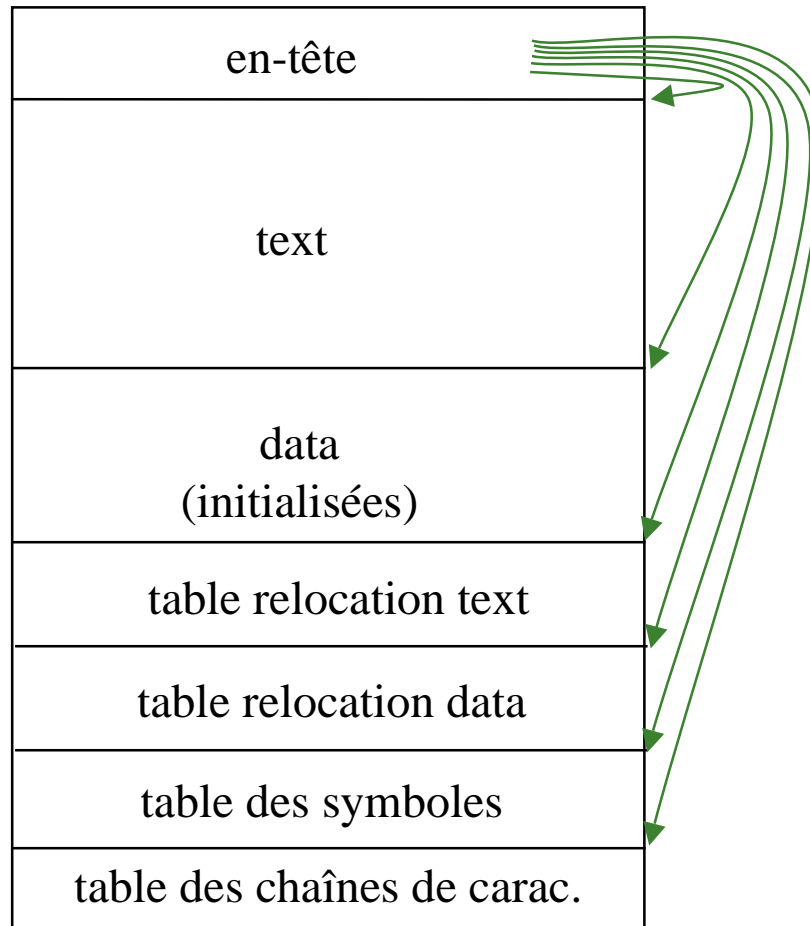


■ commandes globales pour gcc

- ❑ -E stoppe après le preprocessing
- ❑ -S stoppe après la compilation proprement dite (résultat en assembleur)
- ❑ -c stoppe après l'assemblage
- ❑ -o fich place le résultat dans le fichier fich, quelle que soit l'étape d'arrêt
- ❑ -I répertoire de recherche des #include
- ❑ -L répertoire de recherche des librairies
- ❑ -D pour définir une CdC
- ❑ -static (voir plus loin)
- ❑ -shared (voir plus loin)

exécutable, structure

structure d'un exécutable
selon le format **a.out.h**



Remarques

- l'en-tête contient la taille de chaque section et un pointeur vers son début
- l'en-tête contient un **magic number**, vérif exécutabilité
- l'en-tête contient les points d'entrée (ex: start)
- pas de sections pour les data non initialisées ni pour la pile
- les diverses tables sont conservées pour une possible future édition de liens
- elles peuvent être enlevées par la commande **strip**

■ édition de liens statique

- les symboles qui manquent dans la table des symboles après l'édition des fichiers objets fournis dans la ligne de commande sont recherchés dans la liste des bibliothèques par défaut ou fournies dans la ligne de commande.
- l'éditeur de liens prend ce qui est nécessaire et l'**intègre** au fichier exécutable

création d'un fichier exécutable lié statiquement:

- `cc -static xxx xxx xxx xxx xxx`
- `ld -static xxx xxx xxx xxx xxx`

exécution

- lorsque l'appel noyau `execve` (`fich,...`) est exécuté, le système construit sur la mémoire de réserve (swap) une **"image mémoire"** à partir du fichier exécutable `fich`. L'image mémoire contient l'ensemble des sections contenues dans le fichier exécutable, y compris les tables des symboles et les tables de relocation.
- Il y ajoute une section `bss` pour les données non initialisées et une section pour la `pile`.

image mémoire d'un processus

image mémoire

en-tête
text
data (initialisées)
table relocation text
table relocation data
table des symboles
table des chaînes de carac.
bss
pile

Remarques

- *l'image mémoire contient les parties utilisées de l'espace virtuel d'un processus*
- *elle est paginée et ses pages sont amenées en mémoire centrale au fur et à mesure des demandes*

- édition de liens statique : inconvénients
 - exemple : libc.a occupe 12 578 424 octets

Si on la lie systématiquement à tous les fichiers on ajoute au moins 1,9 Mo à tous les fichiers
 - disque encombré
 - deux pages de même contenu peuvent être chargées simultanément si deux processus s'exécutent en même temps. Donc mémoire mal utilisée
- => *insertion à partir des bibliothèques seulement au moment de l'exécution*
- => *partage par les processus en cours d'exécution*

exécution : Partage de code

- ❑ contrainte : pas de variables incluses

Si une variable est incluse dans du code partagé, alors toute modification de cette variable par un processus peut avoir des conséquences pour un autre processus partageur.

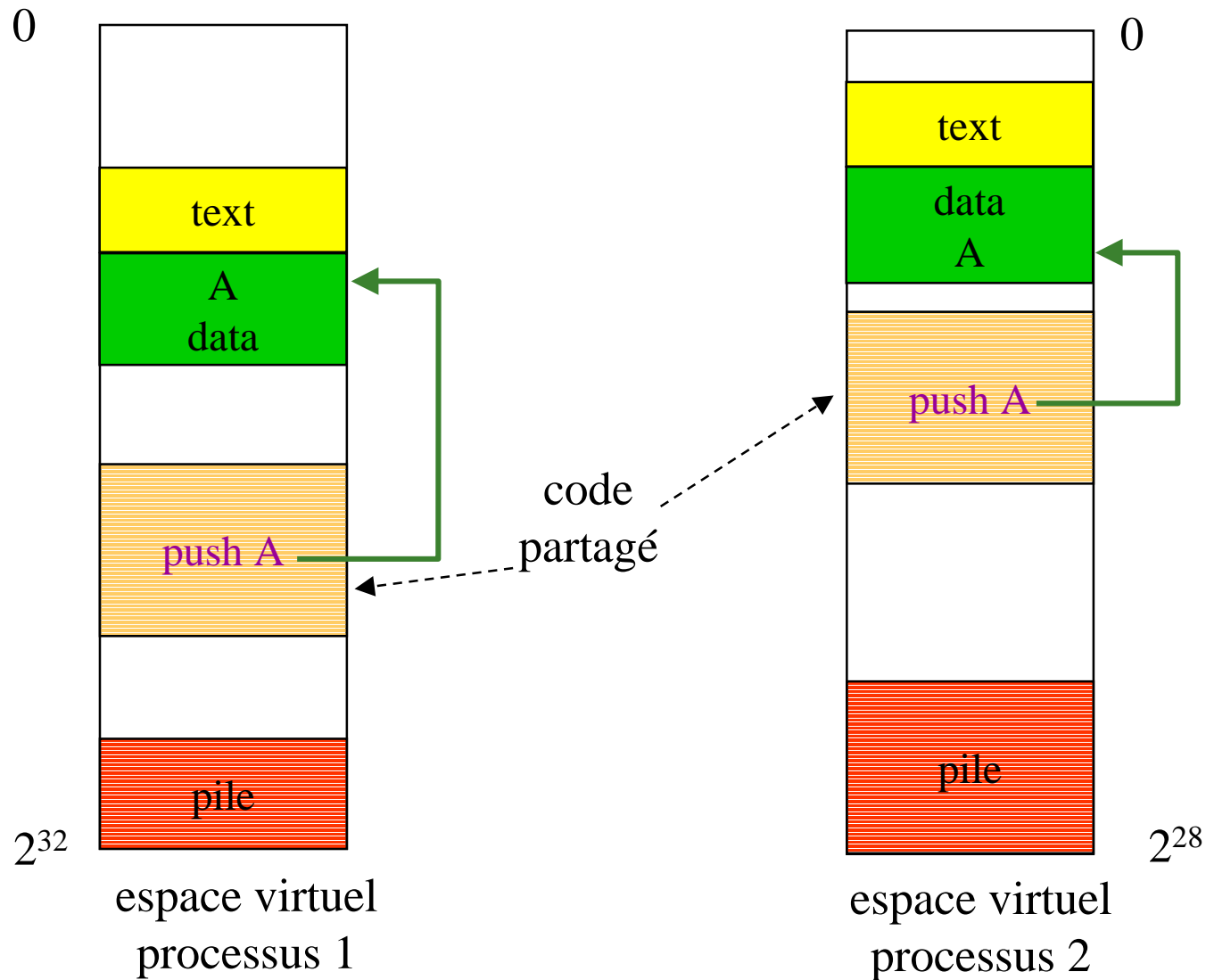
=> code sans variable propre

- les instructions doivent faire appel à des variables externes au code partagé

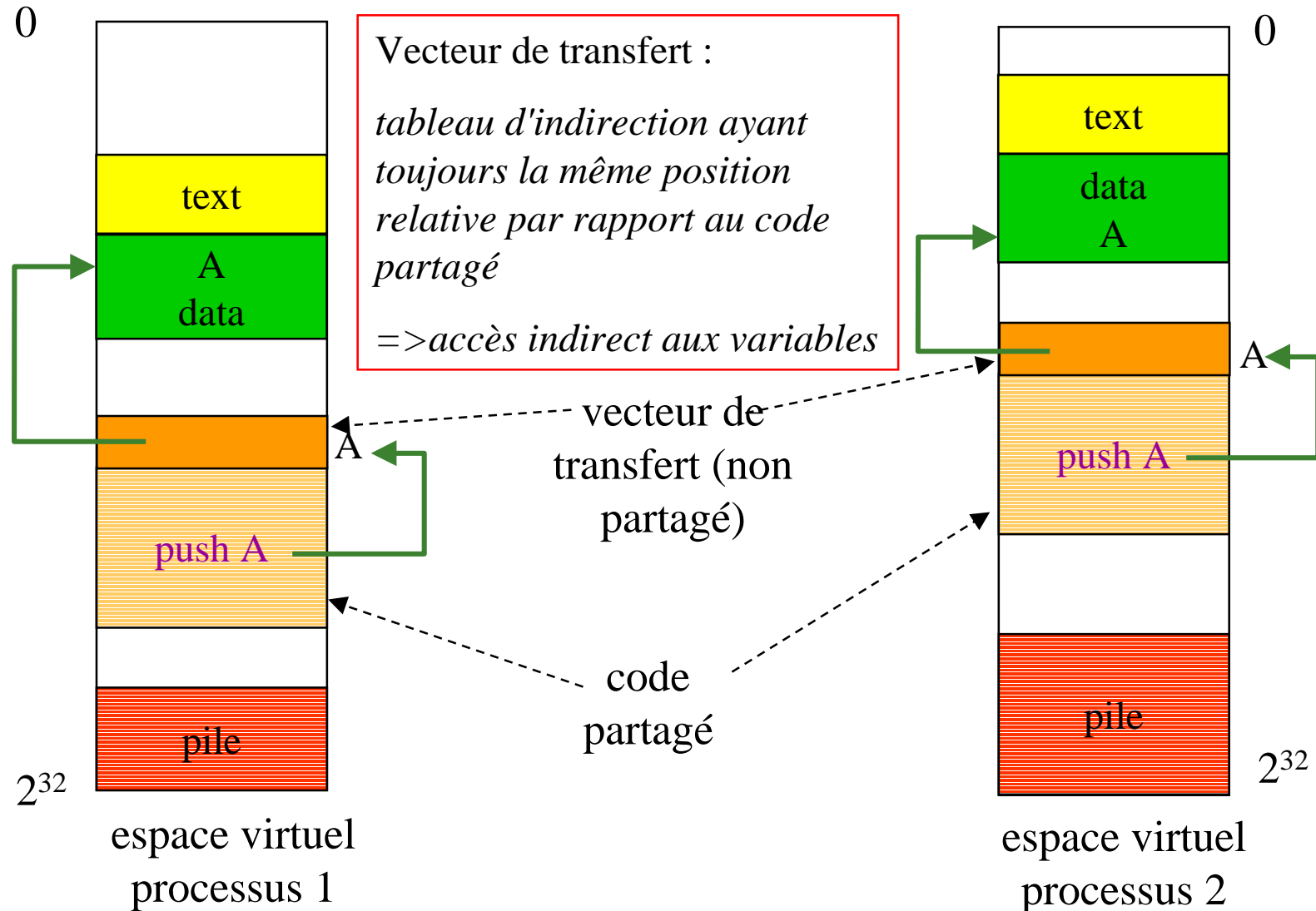
=> problème des adresses différentes

- ❑ problème: insertion à des adresses différentes dans les différents espaces virtuels

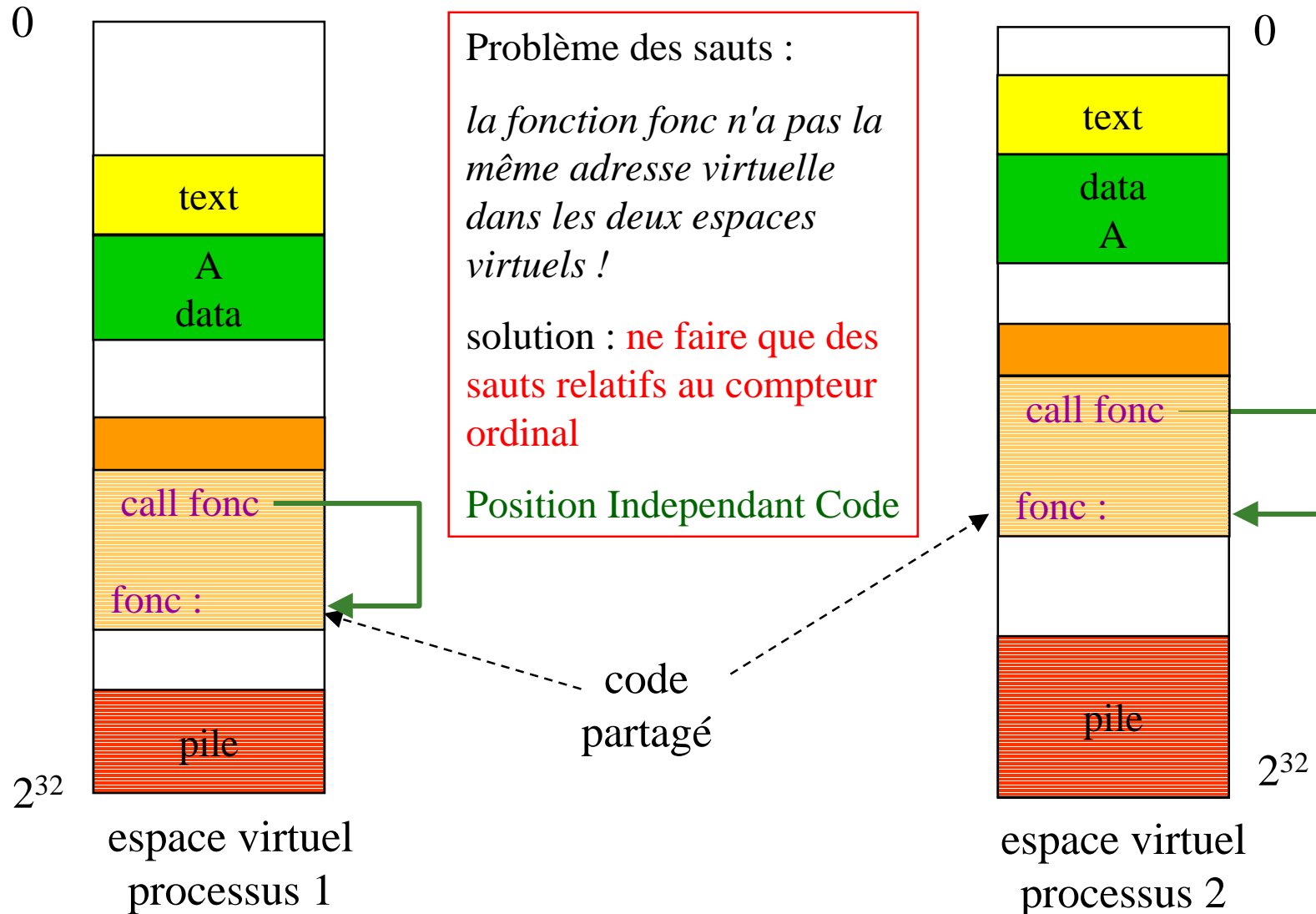
problème des codes partagés



problème des codes partagés



problème des codes partagés



■ édition de liens dynamique

- les symboles manquant dans la table des symboles seront recherchés dans la liste des bibliothèques partageables par défaut ou fournies dans la ligne de commande.
- les codes nécessaires seront **insérés juste avant** l'exécution ou même seulement **lors de l'appel** (lazy binding)

création de fichier objet ou librairies partageables:

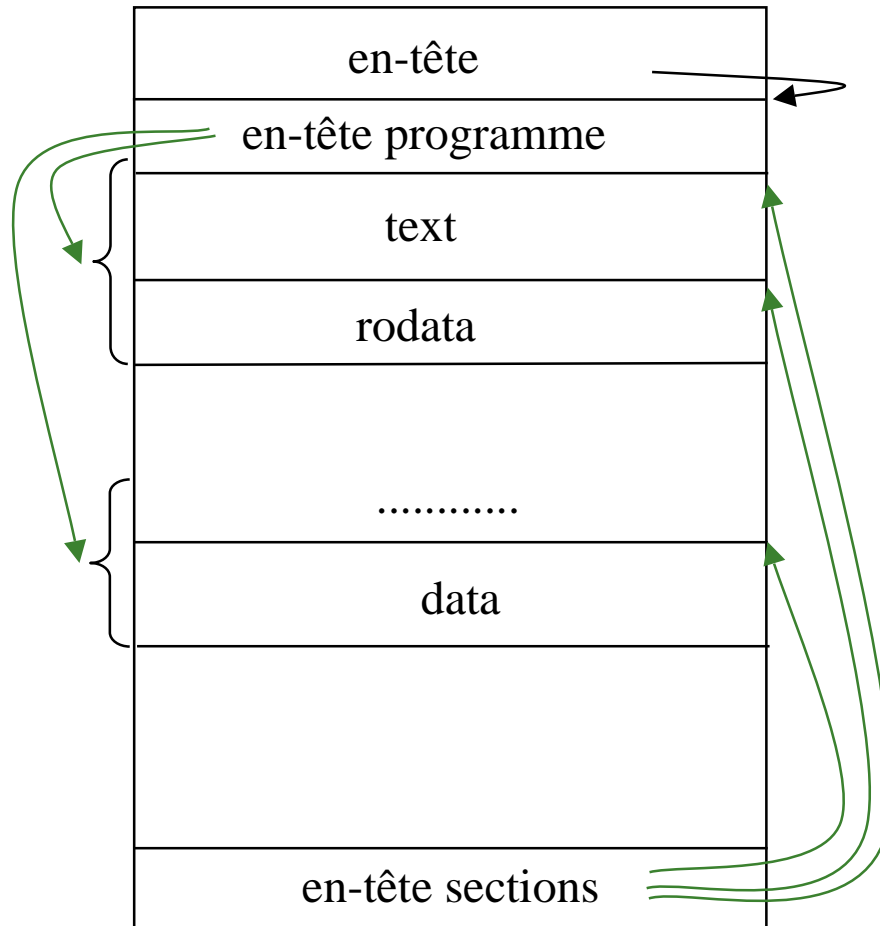
`cc -shared xxx xxx xxx xxx xxx) => <fichier>.so`

édition liens avec des librairies partageables:

`ld -shared xxx xxx xxx xxx xxx) => <fichier>.so`

fichier exécutable, structure

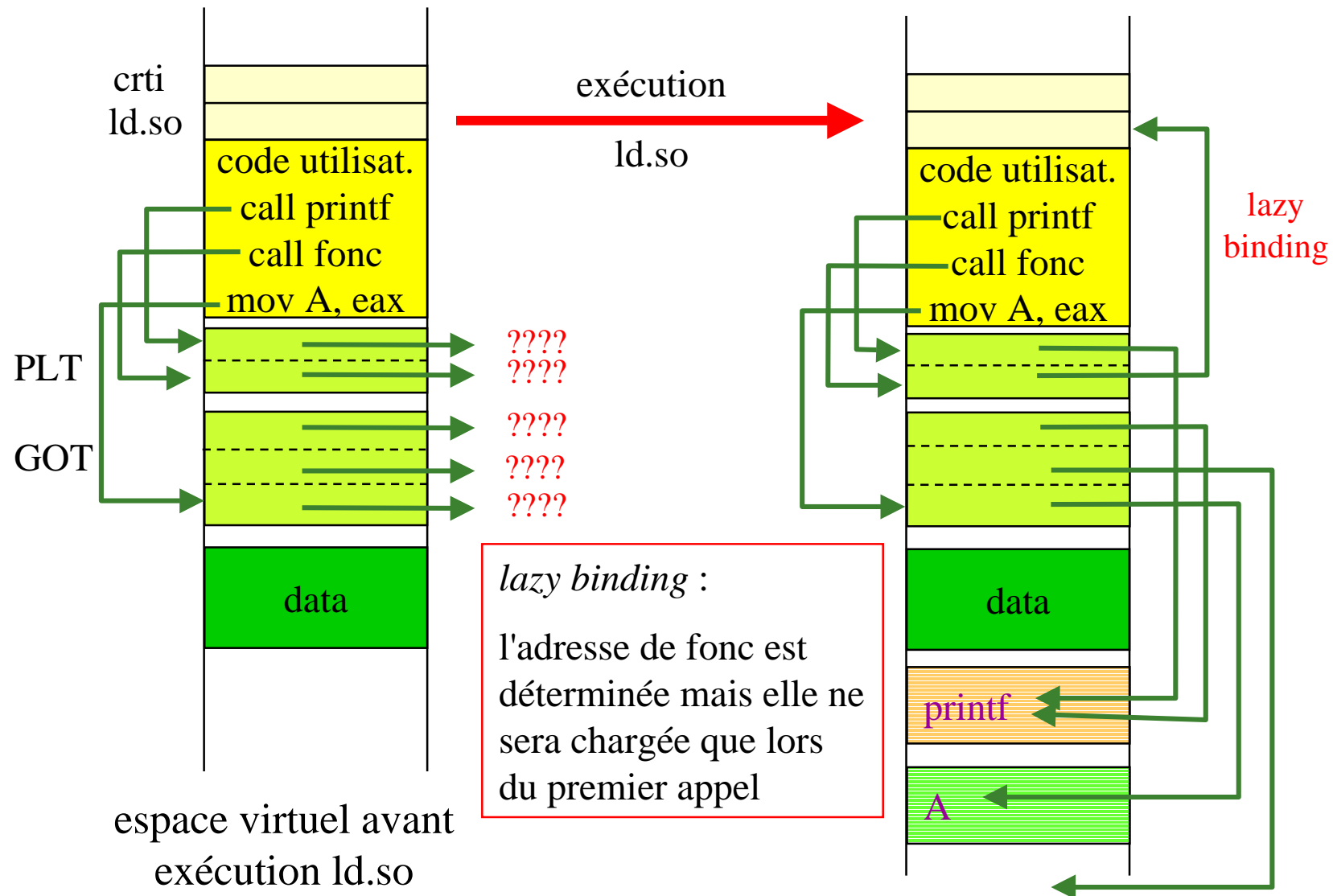
structure d'un ELF (Executable and Linkable Format)



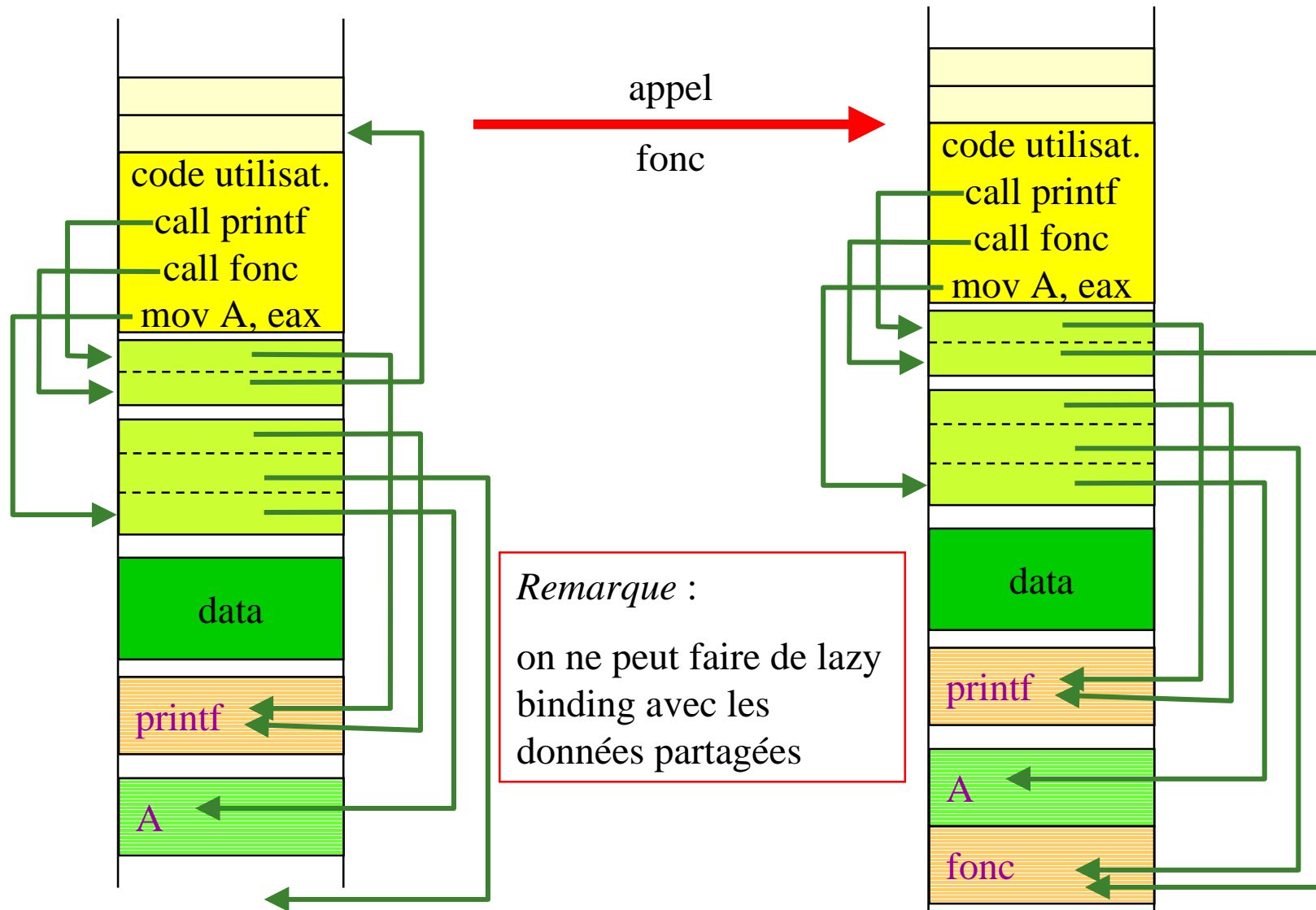
Remarques

- l'en-tête programme décrit les segments, utilisés lors de l'exécution (ex segment text + rodata, ne peut être écrit)
- segments pour les bibliothèques dynamiques
- l'en-tête sections décrit les sections, utilisées pour l'édition de liens et la relocation (semblables au format a.out.h)
- format Portable Executable pour windows

démarrage de code lié dynam.



démarrage de code lié dynam.

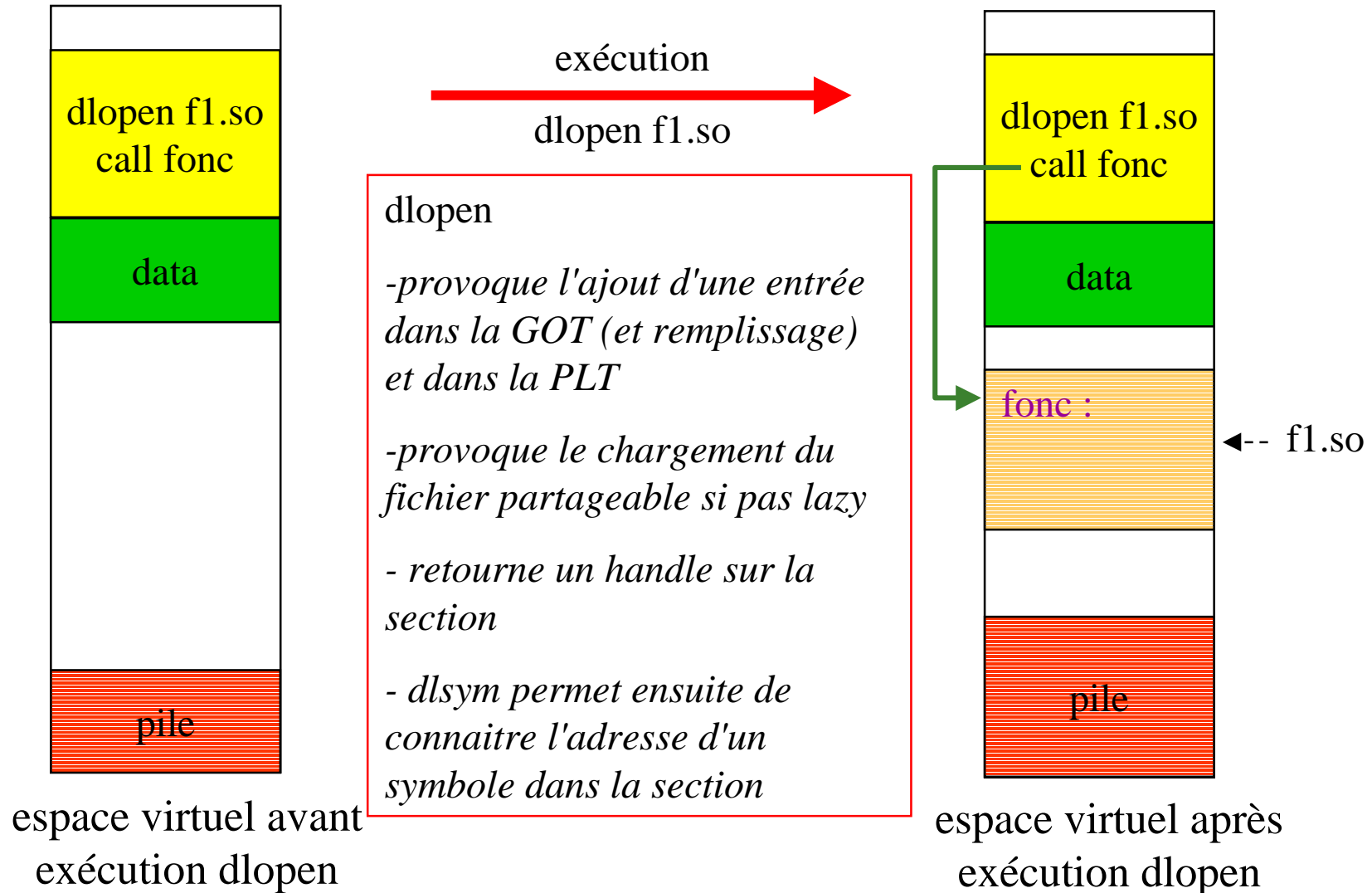


- GOT (Global Offset Table) :
 - *Table indiquant les positions des symboles liés dynamiquement. Table d'indirect. pour les data*
 - *Complétée par ld.so lors du démarrage*
- PLT (Procedure Linkage Table) :
 - *table d'indirection pour les appels de procédure,*
 - *non complétée pour les procédure chargée à la demande seulement (lazy binding)*
 - *le premier appel d'une proc. provoque l'appel de ld.so qui complète l'entrée à partir de la GOT*

démarrage de code lié dynam.

- paramétrage de ld.so
 - variable d'environnement LD_LIBRARY_PATH
 - /etc/ld.so.cache

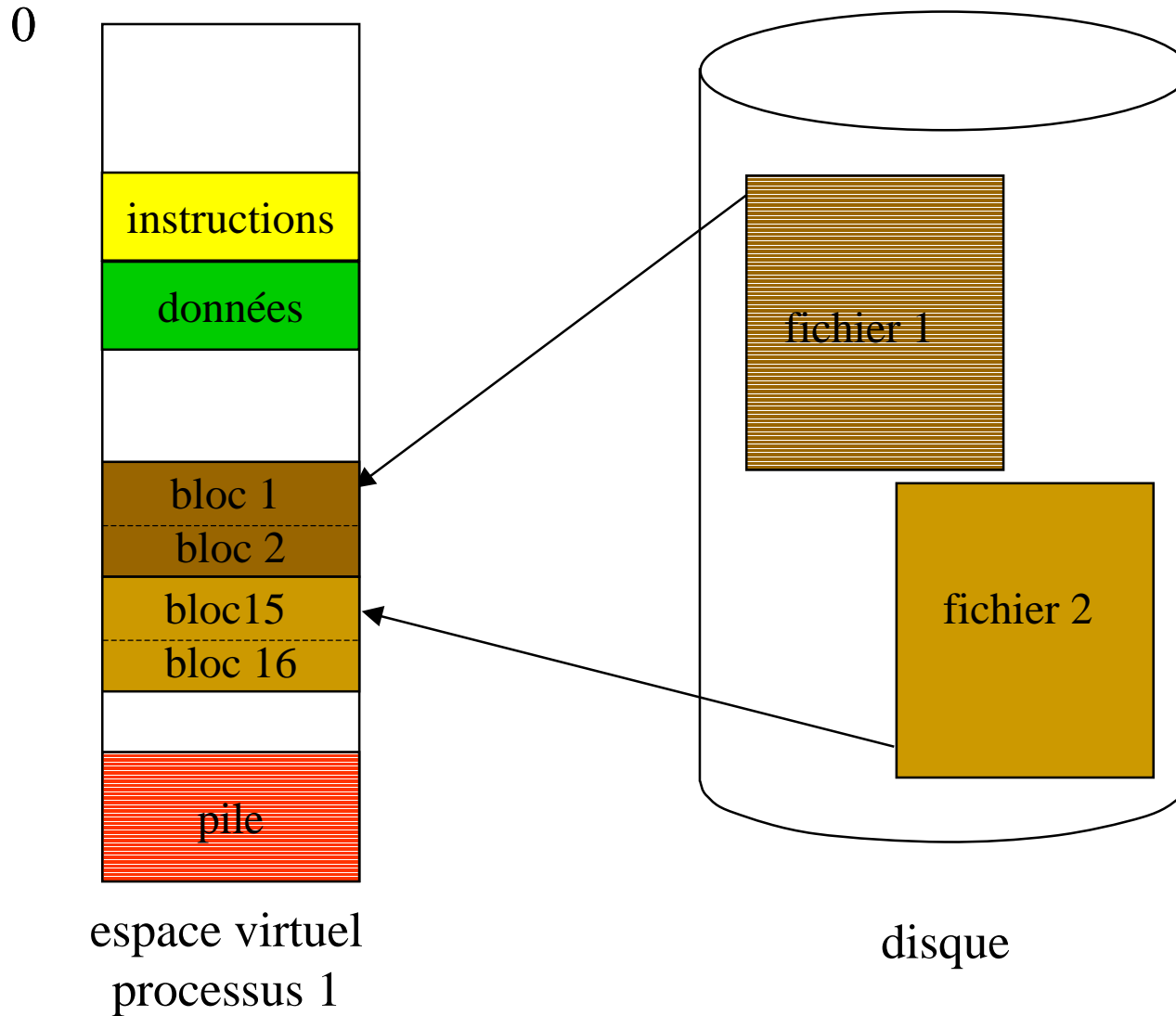
demande explicite d'ajout de code



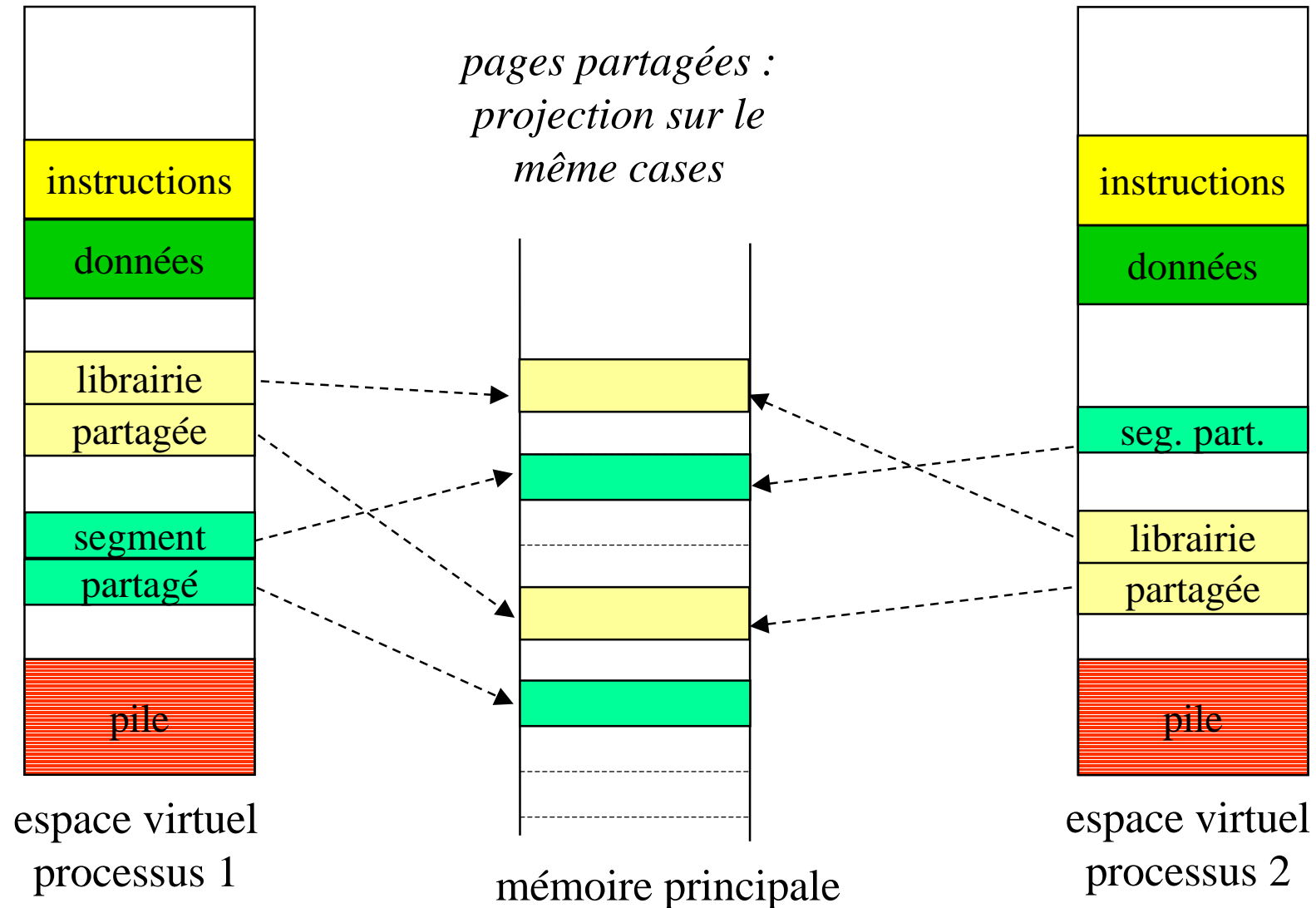
exécution : ajout dynamique de données

- d'autres régions peuvent être ajoutées
 - projection de fichier en mémoire (accès direct au contenu d'un fichier sans passer par read ou fread) :
appel noyau mmap
 - segments de mémoire partagés :
appel noyau shmget

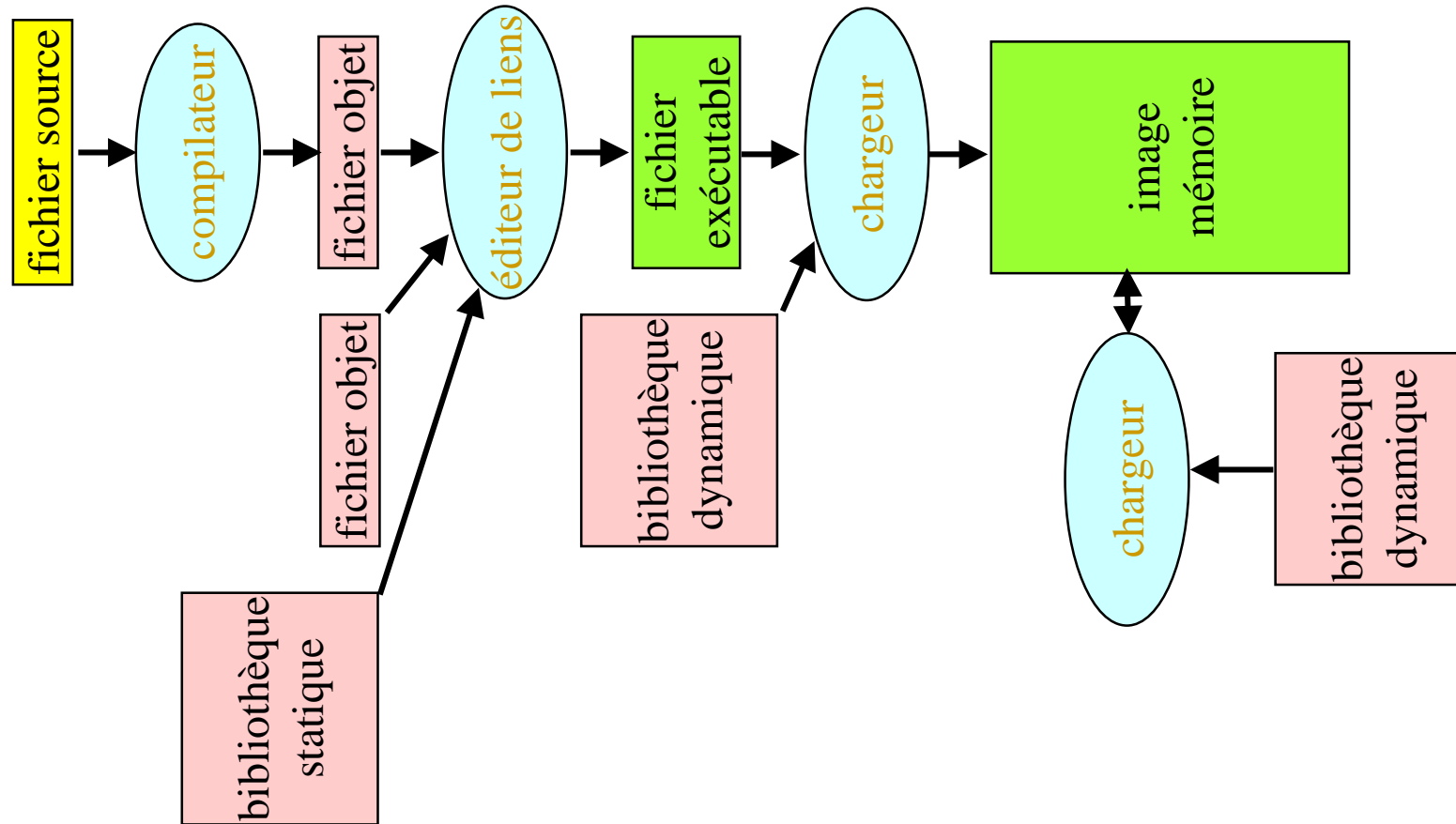
Projection de fichiers (mapping)



segments partagés



résumé



commandes aide diverses

- ❑ objdump : lecture d'objet ou d'exécutable
 - -t pour les symboles
 - -d pour de-assembler
- ❑ readelf : lecture d'objet ou d'exécutable
- ❑ nm : donne la liste des symboles
- ❑ ldd : donne les bibliothèques dynamiques d'un fichier
- ❑ time mesure les temps d'exécution
- ❑ strace : affiche les appels systèmes d'une exécution
- ❑ cat /proc/<pid>/smaps donne l'image mémoire du processus de numéro <pid>