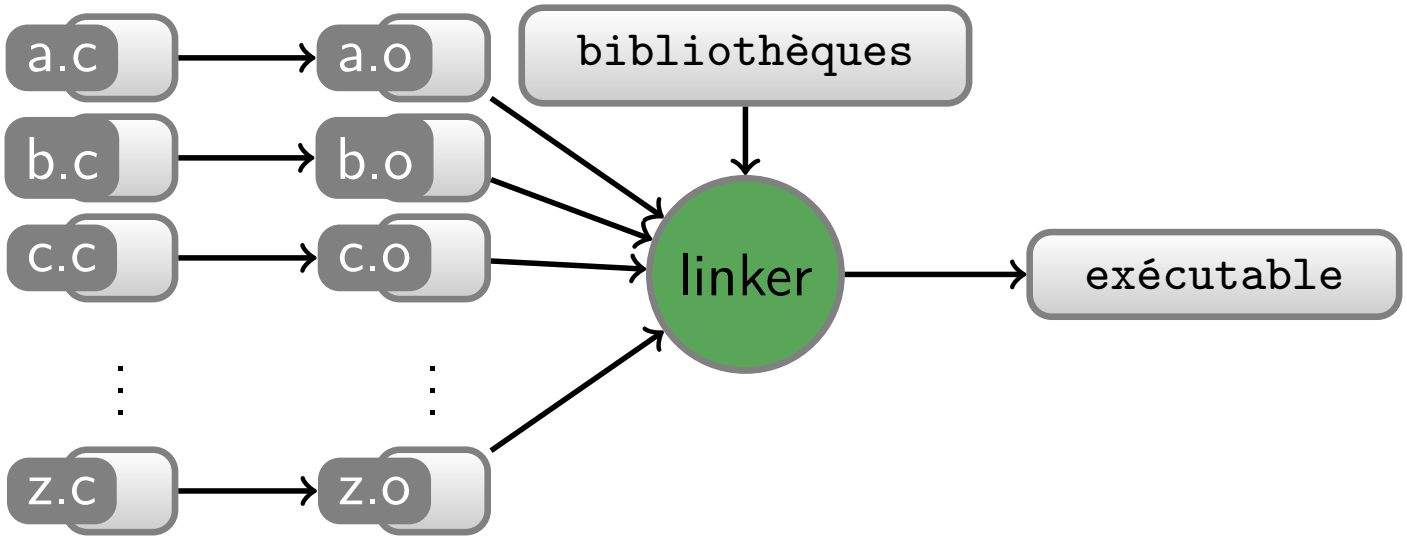
Langage C avancé
Utilisation des bibliothèques

Samuel KOKH

samuel.kokh@cea.fr

MACS 1 – Institut Galilée

A propos de la compilation séparée : jusqu'ici...



Jusqu'ici

La compilation séparée se résume à

- Produire une collection de fichier objets `*.o` à partir d'un ensemble de fichiers source `*.c` et de fichiers en-têtes `*.h`
- Produire un exécutable à partir des fichiers objets et de bibliothèques via l'éditeur de liens

Les bibliothèques (Libraries)

Qu'est-ce qu'une bibliothèque ?

une bibliothèque est un agrégat d'une collection de plusieurs fichiers objets `*.o`.

Comme pour les fichiers `*.o`, le contenu des bibliothèques est destiné à être utilisé via un éditeur de liens pour être combiné au code de l'exécutable.

Pour quoi faire ?

En associant les bibliothèques à des fichiers `*.h` bien définis :

- simplification de l'accès (pour les développeurs) à un ensemble de ressources (fonctions/variables/types/macros) définies dans la bibliothèque
- simplification de la distribution des éléments de code

Les types de bibliothèques

Deux catégories de bibliothèques

- bibliothèques statiques (static library)
- bibliothèques partagées (shared library)

Différence entre les deux catégories de bibliothèques

Leur contenu n'est pas combiné à celui de l'exécutable au même moment et l'édition de liens ne se fait pas de la même manière.

Bibliothèques à l'intérieur d'un projet de code

Il est rare qu'une application exécutable soit le seul produit d'un projet de code informatique.

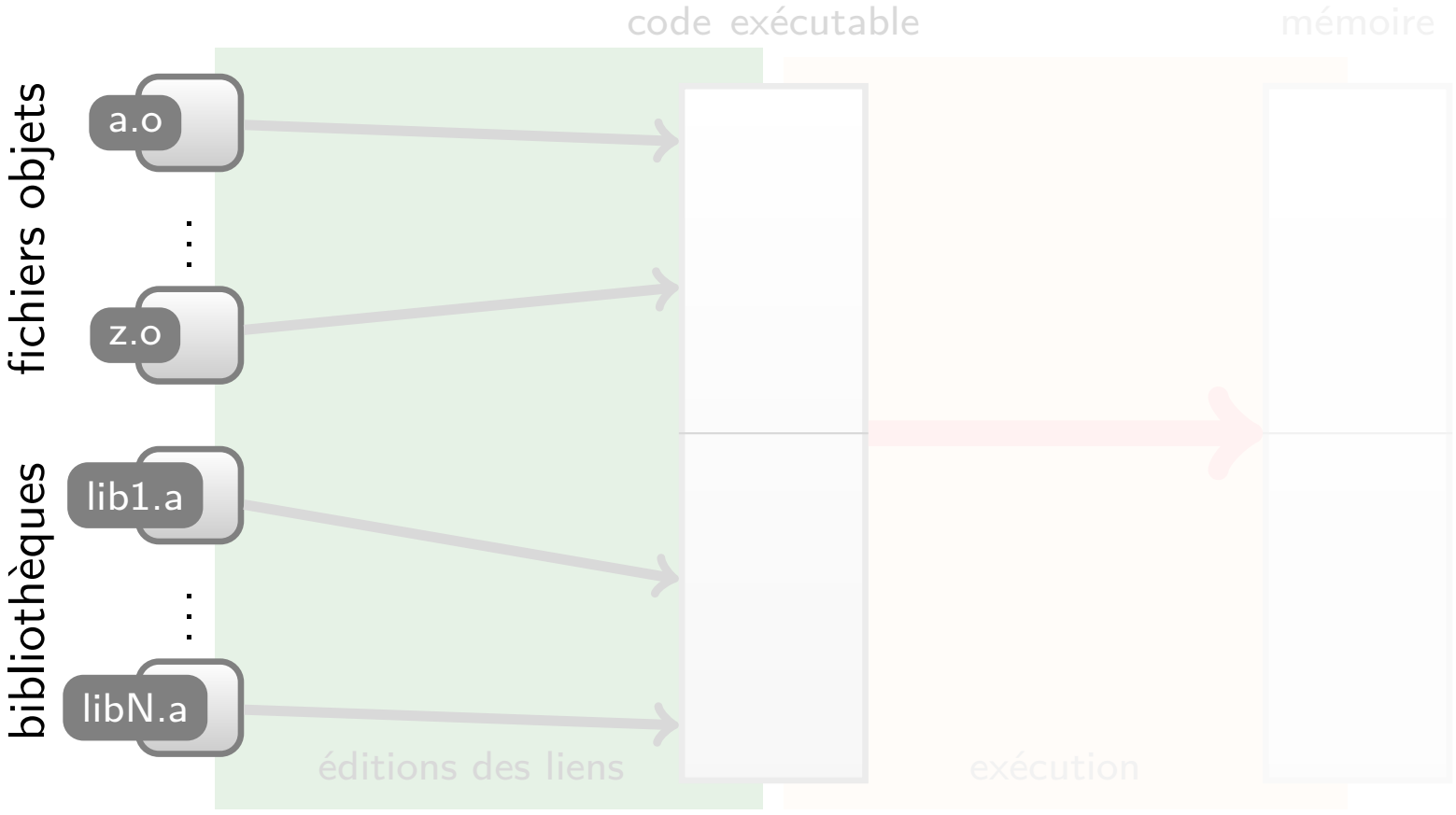
Structure (courante) d'un projet de code

- le projet produit des bibliothèques (statiques et/ou partagées)
- le projet produit des application exécutables (construites à partir des bibliothèques du projet)

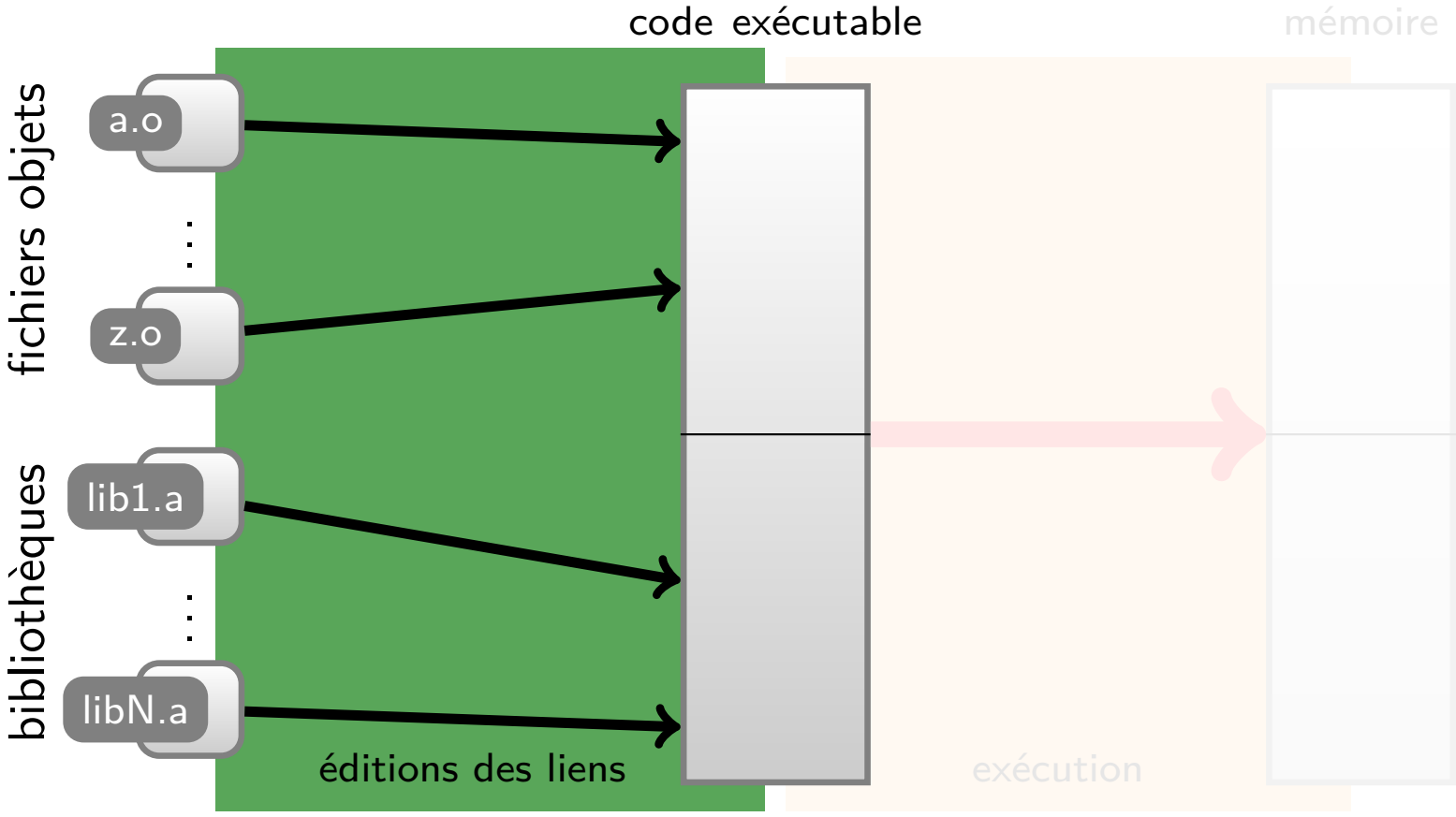
Les bibliothèques statiques

- le code de la bibliothèque est incorporé dans l'exécutable pendant la phase d'édition de liens
- l'exécutable résultant contient le code des ressources de la bibliothèque dont il a besoin
- après édition des liens la bibliothèque n'est plus nécessaire pour le l'exécution de l'application

Les bibliothèques statiques

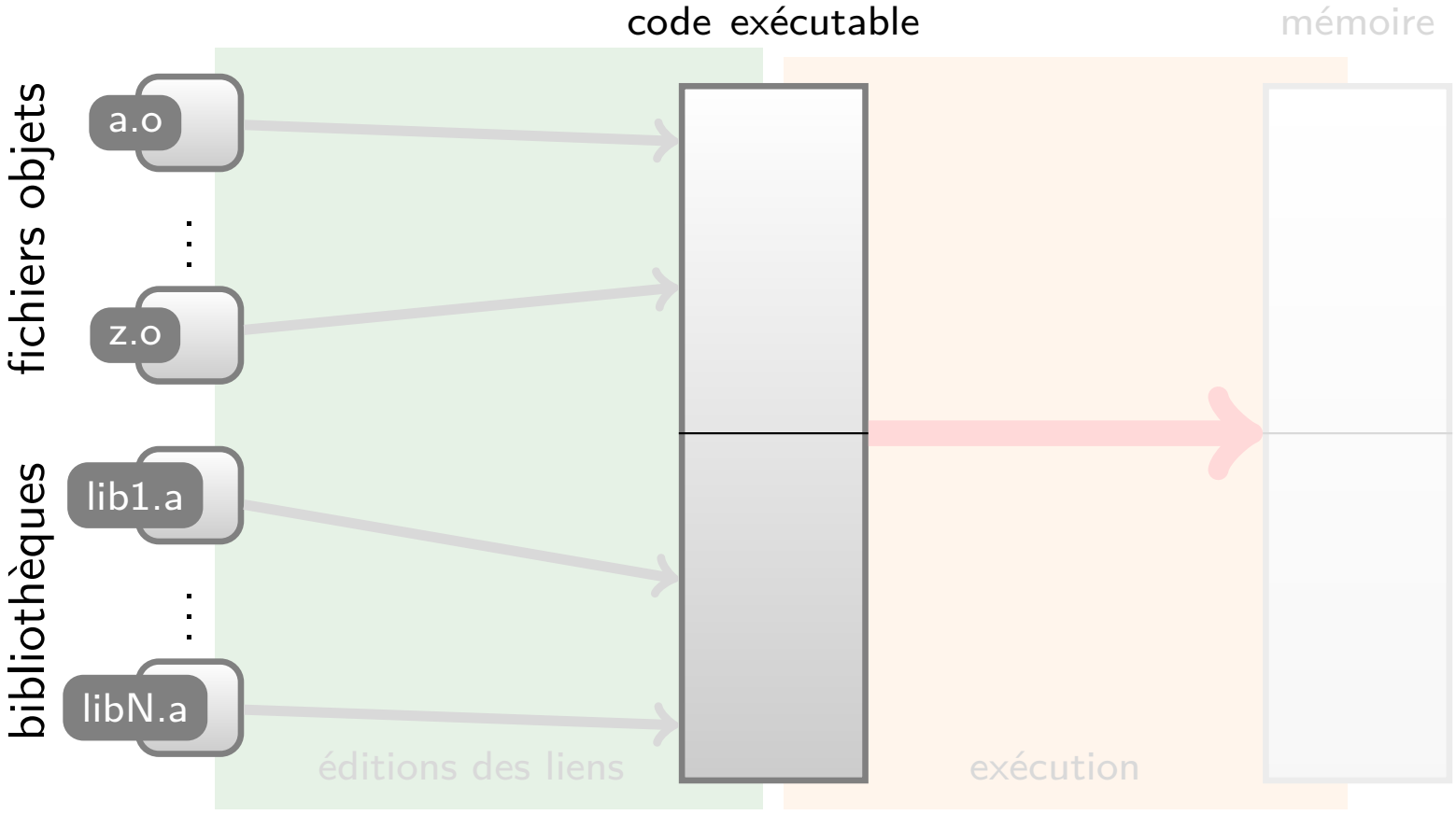


Les bibliothèques statiques



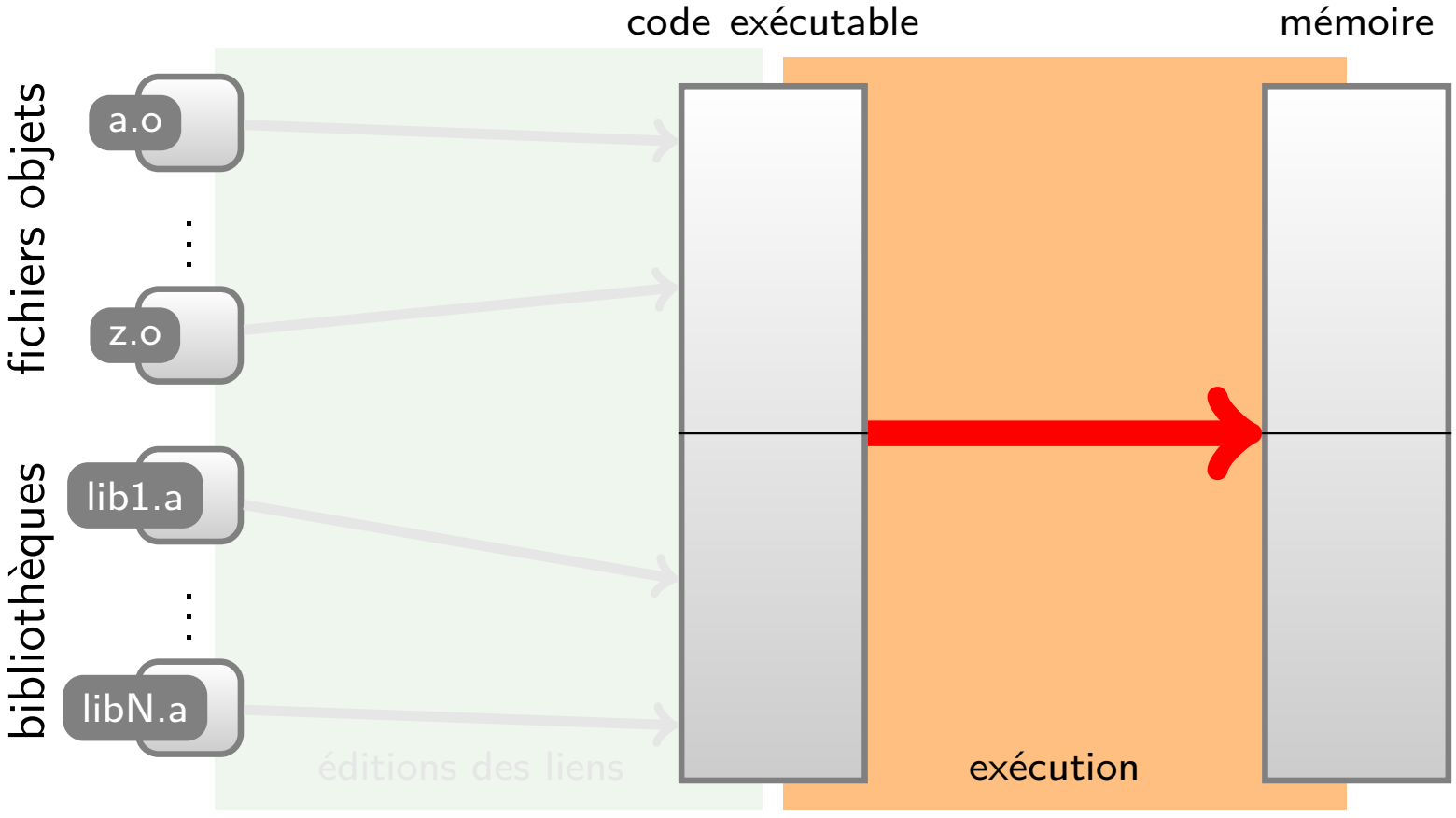
L'éditeur de lien produit un exécutable en agrégeant tous les fichiers objets et en leur en greffant les ressources nécessaires des bibliothèques statiques.

Les bibliothèques statiques



L'éditeur de lien produit un exécutable en agrégeant tous les fichiers objets et en leur en greffant les ressources nécessaires des bibliothèques statiques.

Les bibliothèques statiques



A l'exécution du programme le code objet de l'exécutable est « poussé » en mémoire.

Librairie statique

La bibliothèque statique est utilisée un peu à la manière « d'une greffe » qui serait effectuée pendant l'édition des liens.

- l'exécutable est « auto-porteur », il renferme tout le code dont il a besoin pour fonctionner
- cette méthode tend à générer des exécutable de taille importante
- pour tenir compte d'une mise-à-jour d'une bibliothèque à l'intérieur du code, il faut recompiler complétement l'exécutable

Les bibliothèques statiques sont des fichiers (traditionnellement) nommés `*.a` ou plus précisément la bibliothèque *NAME* est (traditionnellement) nommée `libNAME.a`.

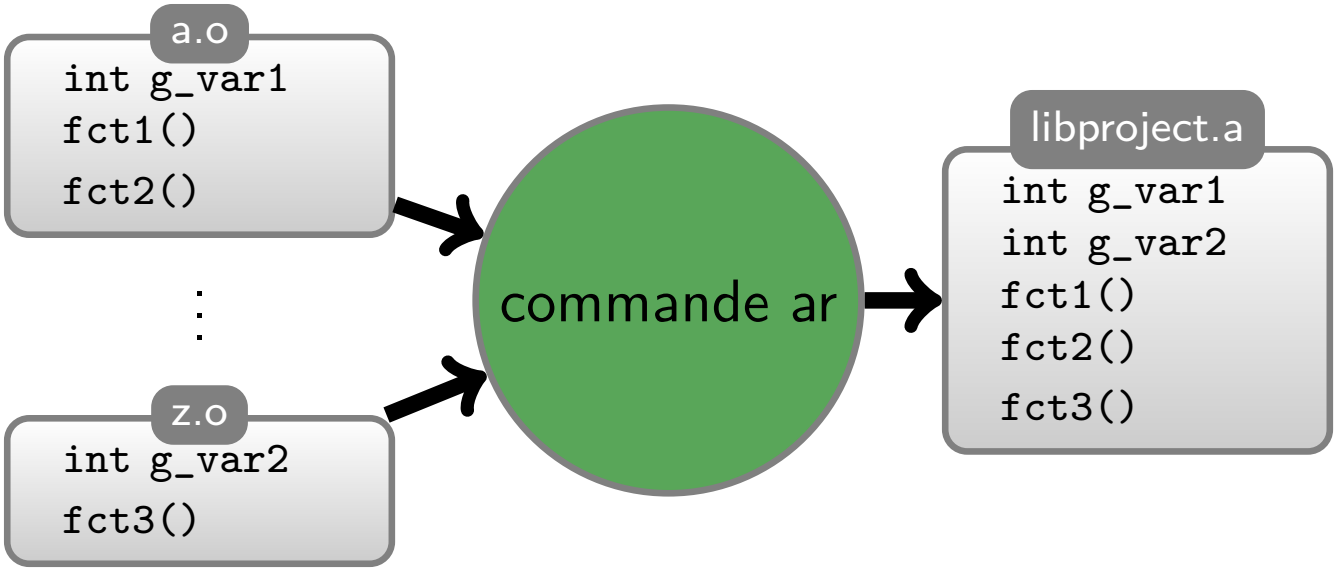
Exemple de bibliothèque statique : `libm.a`

- bibliothèque mathématiques standard du C
- définition des fonctions `cos`, `sqrt`, `log`, etc.
- sous UNIX, généralement : `/usr/lib/libm.a`
- appelée avec l'option de compilation `-lm`
- les prototypes associés sont regroupés dans le fichier `math.h`, situé généralement `/usr/include/math.h`

Exemple de bibliothèque statique : `libc.a`

- bibliothèque standard du C ANSI-ISO
- définition des fonctions `printf`, etc.
- sous UNIX, généralement : `/usr/lib/libc.a`
- elle est liée par défaut avec tous les programmes C

Création d'un bibliothèque statique



La création d'une bibliothèque statique à partir d'une collection de fichiers objet s'effectue via l'utilitaire ar (archive).

```
$ ar -cru libproject.a a.o ... z.o
```

Quelques options de ar

option -c

(create) : créer une bibliothèque (si la bibliothèque existe l'option est ignorée)

```
$ ar -c <bibliotheque>
```

option -r

(replace) : remplacer des objets dans une bibliothèque (si ils existent dans la bibliothèque)

```
$ ar -r <bibliotheque> <fichiers objets>
```

option -ru

ne remplacer les objets dans la bibliothèque que si ceux des fichiers `*.o` sont plus récents que ceux de la bibliothèque

```
$ ar -ru <bibliotheque> <fichiers objets>
```

Quelques options de ar

option -v

mode verbeux

```
$ ar -v
```

option -t (liste)

Liste le nom des fichiers objets contenus dans la bibliothèque

```
$ ar -t <bibliotheque>
```

```
$ ar -t /usr/lib/libc.a
init-first.o
libc-start.o
sysdep.o
version.o
check_fds.o
...
```

Quelques options de ar

option -d (delete)

Supprime les ressources d'un fichier objet d'une bibliothèque

```
$ ar -d <bibliotheque> <fichiers objet>
```

```
$ ar -t libproject.a
foo.o
bar.o
$ ar -d libproject.a bar.o
$ ar -t libproject.a
foo.o
```

Quelques options de ar

option -x (eXtraction)

Extraction d'un fichier objet d'une bibliothèque

```
$ ar -x <bibliotheque> <fichiers objet>
```

```
$ ar -t libproject.a
foo.o
bar.o
$ ar -x libproject.a bar.o
$ ls -rtlh *.o
...
bar.o
```

Utilitaire `ranlib`

Index d'une bibliothèque et utilitaire `ranlib`

L'outil `ranlib` permet de créer un index des ressources présentes dans une bibliothèque statique et d'incorporer cet index à l'intérieur de la bibliothèque.

Index et compilation

Une bibliothèque munie d'un index peut permettre d'accélérer substantiellement la phase d'édition des liens.

Utilisation de `ranlib`

```
$ ranlib libproject.a
```

Utilisation équivalente avec `ar`

```
$ ar -s libproject.a
```

Compilation à l'aide d'une bibliothèque statique

Utilisation d'une bibliothèque pour la compilation

L'idée consiste simplement à remplacer l'appel aux fichiers objet de la bibliothèque pour la bibliothèque.

On suppose que la fonction `main()` de notre programme est définie dans `prog.o`

Avec les notations précédentes, on remplace

```
$ gcc prog.o a.o ... z.o -o program.exe
```

par

```
$ gcc prog.o ./lib/libproject.a -o program.exe
```

dans l'exemple ci-dessus, on suppose que `libproject.a` est dans le répertoire courant.

Compilation à l'aide d'une bibliothèque statique

De manière générale.

```
$ gcc prog.o <chemin vers la bibliothèque> -o program.exe
```

comme par exemple

```
$ gcc prog.o /home/bob/project/lib/libproject.a -o program.exe
```

Compilation à l'aide d'une bibliothèque statique

```
$ gcc prog.o -L<directory> -lname -o program.exe
```

comme par exemple

```
$ gcc prog.o -L/home/bob/project/lib -lproject -o program.exe
```

Options de gcc

- l'option `-L` indique à `gcc` un directory dans lequel il faut chercher la bibliothèque (si elle ne se trouve pas dans les directories de recherche par défaut)
- l'option `-l` permet de raccourir l'écriture du nom des bibliothèques. La référence à une bibliothèque `libname.a` sera ainsi abrégée en `-lname`.

Les bibliothèques partagées

- le code de la bibliothèque n'est pas incorporé dans l'exécutable pendant la phase d'édition de liens
- l'éditeur de liens vérifie simplement que la bibliothèque contient toutes les ressources (fonctions, variables) nécessaires pour construire l'application
- c'est seulement à l'exécution de l'application (runtime) que le code des bibliothèques partagées est incorporé au code de l'application. Cette opération est effectuée à la volée par le chargeur dynamique de bibliothèque `ld` (Library Loader).

Les bibliothèques partagées

Les exécutables obtenus nécessite beaucoup moins d'espace de stockage.

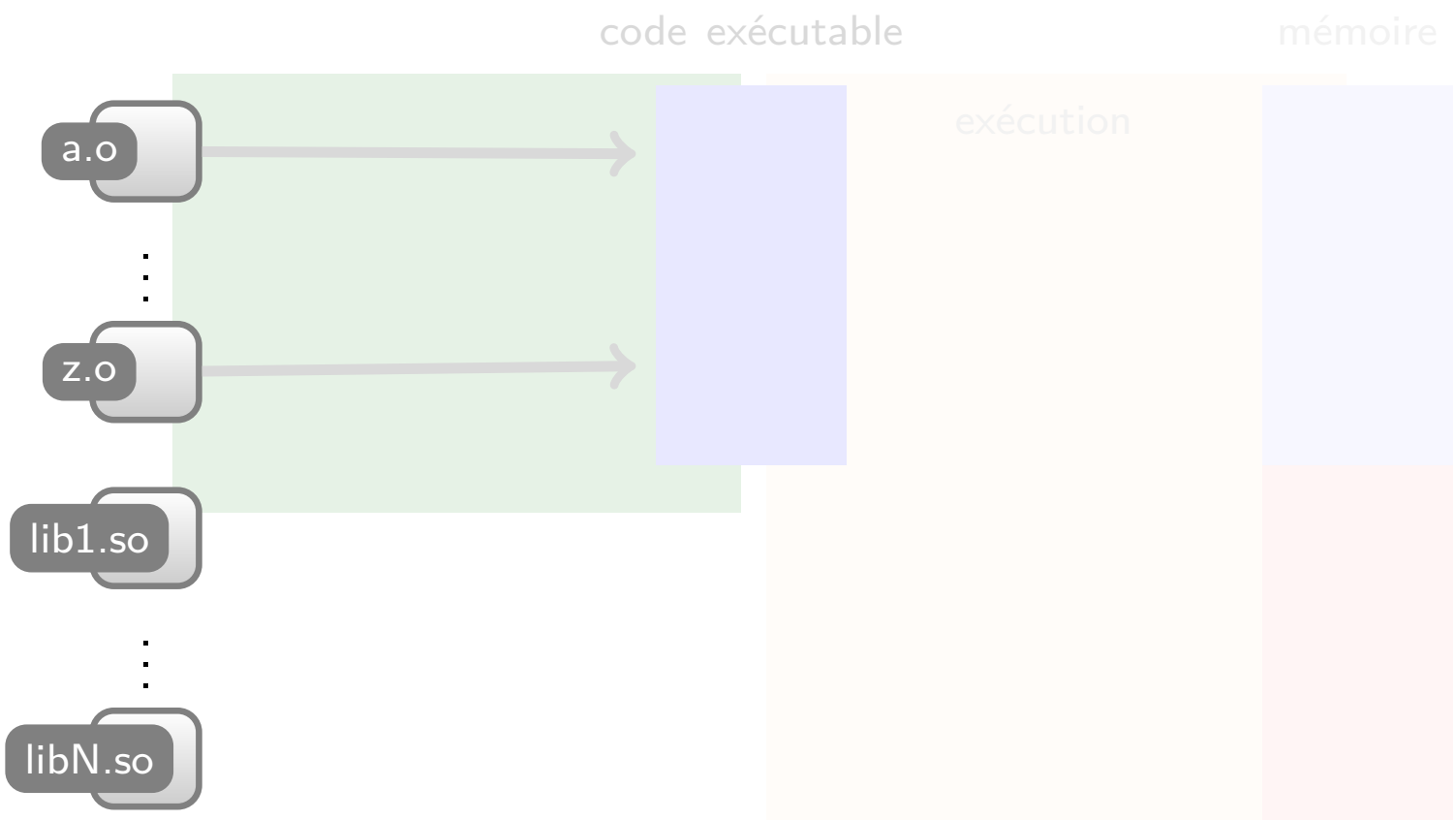
- Il est impératif que la bibliothèque soit présente dans le système au moment de l'exécution du programme !
- si la bibliothèque est mise à jour, la programme profite automatiquement de sa mise-à-jour (sans recompilation !).

Nommage des bibliothèques partagées

Le nom des bibliothèques partagées se termine par l'extension `*.so` (Shared Object).

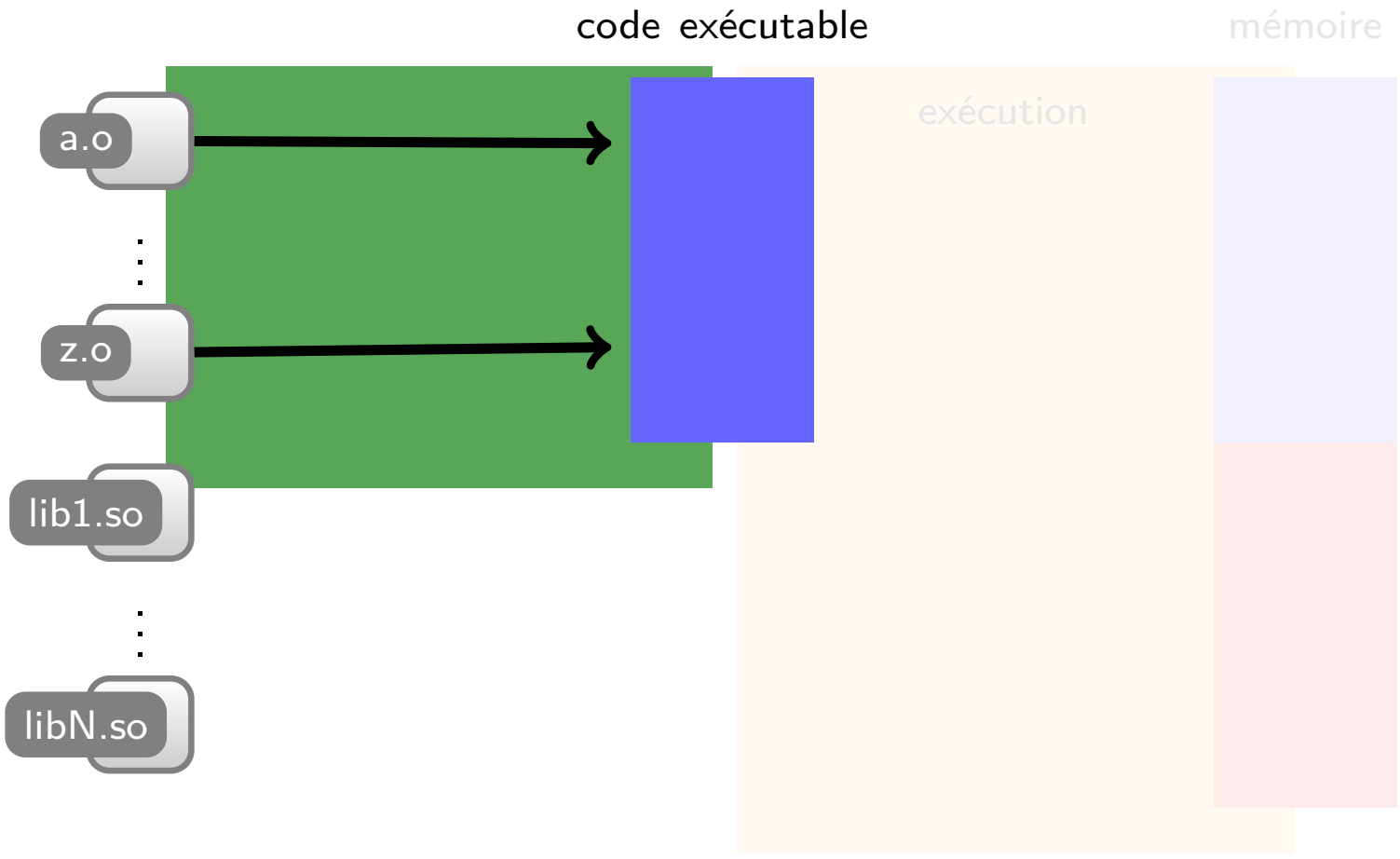
Comme par exemple : `/usr/lib/libm.so`, `/usr/lib/libc.so`

Les bibliothèques dynamiques



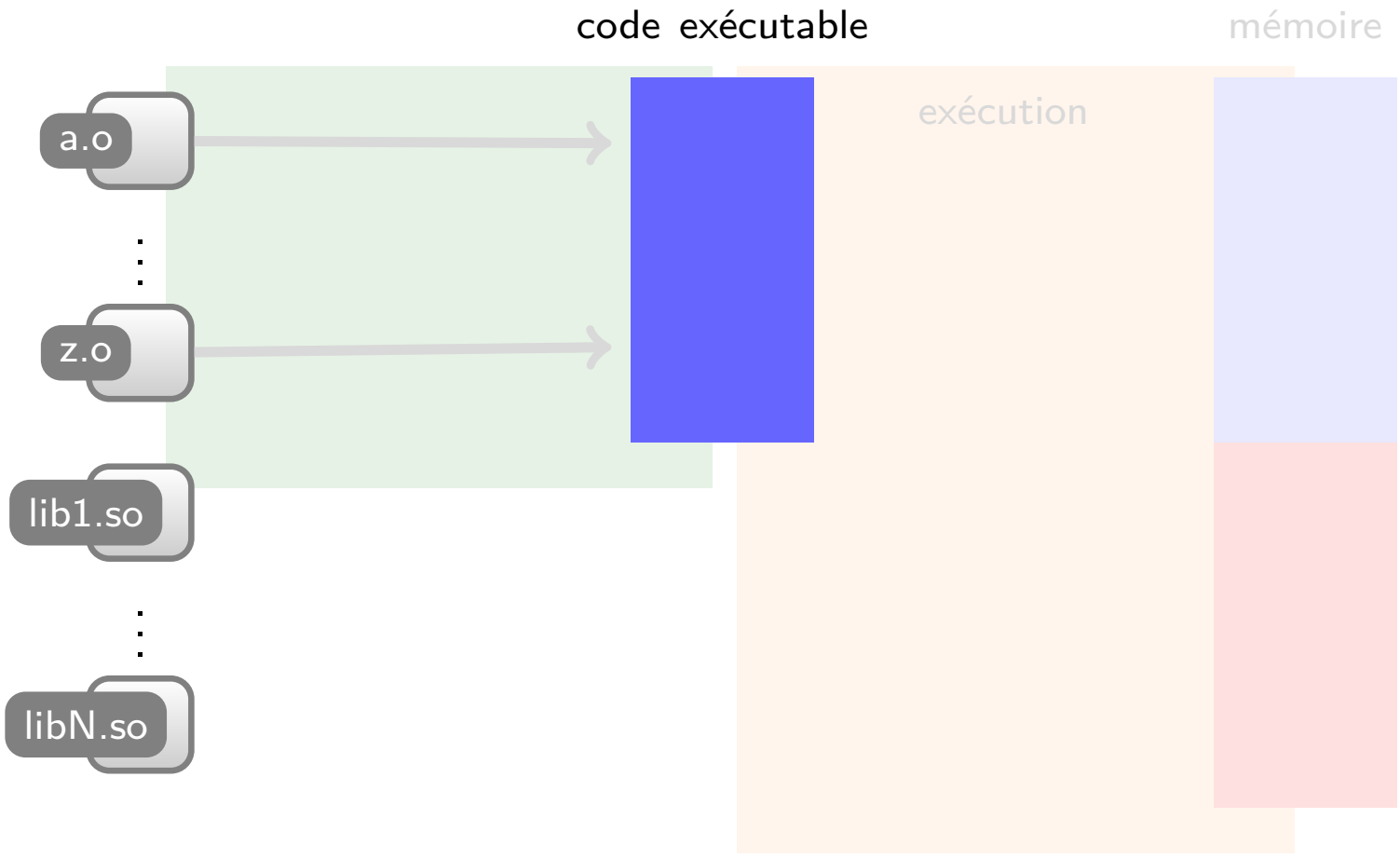
Les bibliothèques dynamiques

éditions des liens



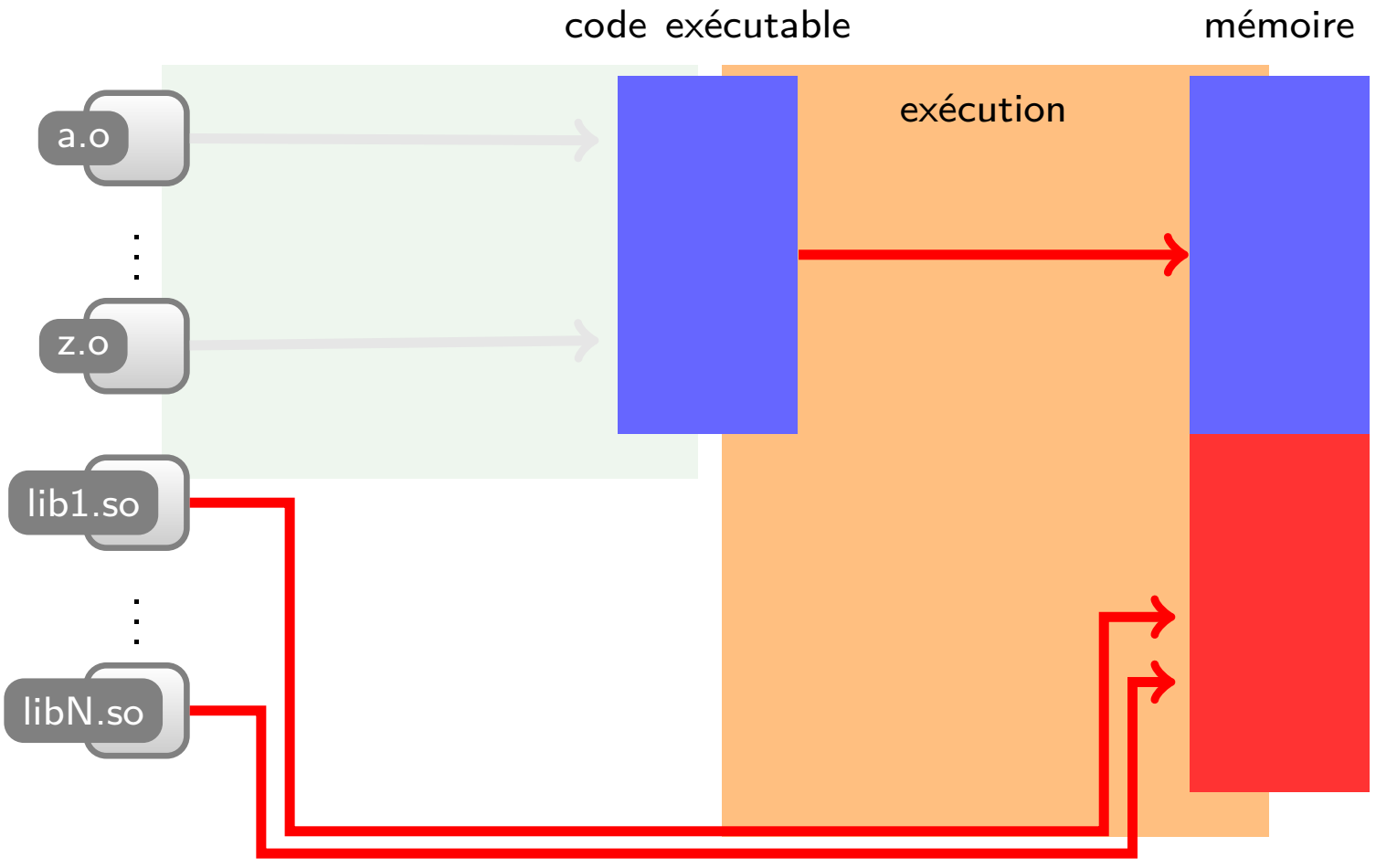
Les bibliothèques dynamiques

éditions des liens

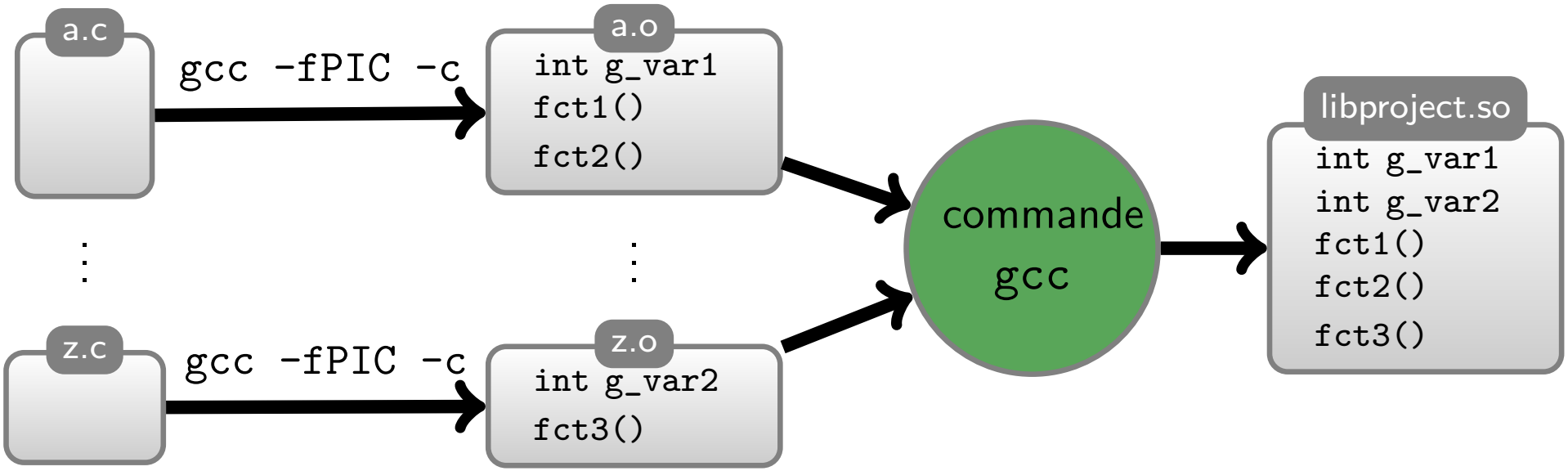


Les bibliothèques dynamiques

éditions des liens



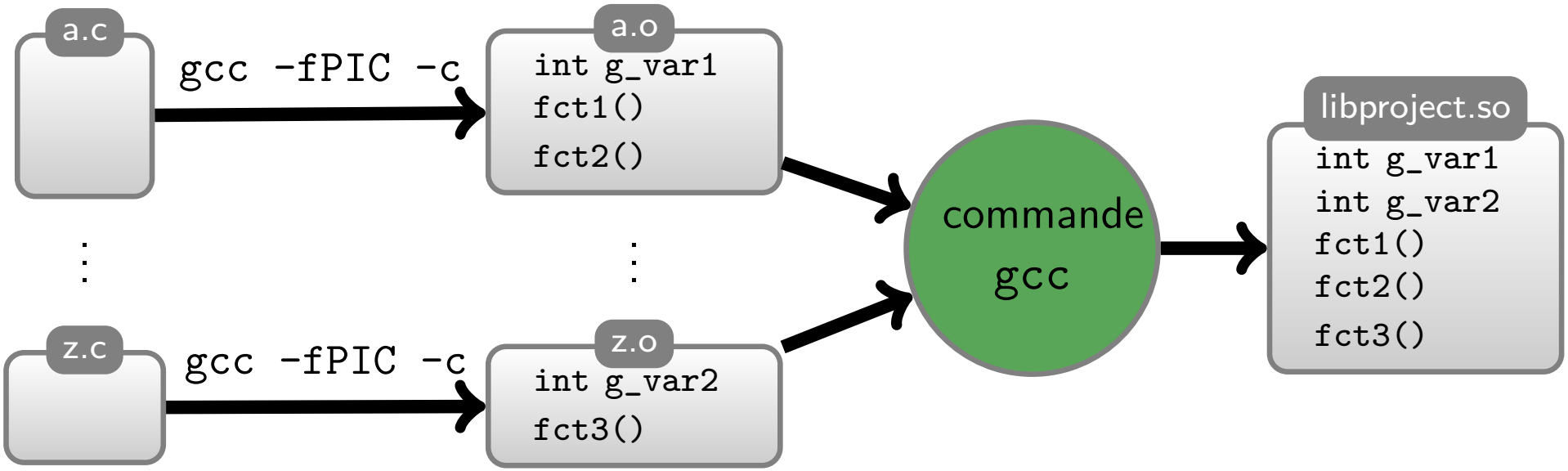
Création d'un bibliothèque dynamique



La création d'une bibliothèque statique nécessite d'utiliser l'option `-fPIC` (Position Independant Code) pour compiler les objets dont elle est composée.

```
$ gcc -Wall -fPIC -c a.c
...
$ gcc -Wall -fPIC -c z.c
```

Création d'un bibliothèque dynamique

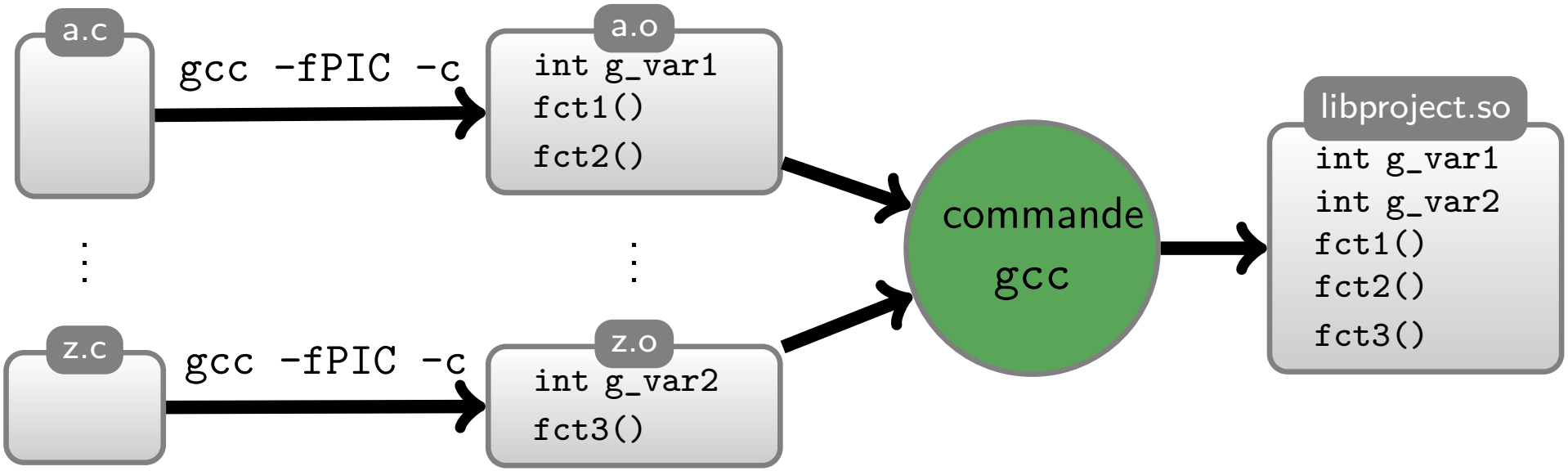


la production de la bibliothèque s'effectue via la commande gcc appelée avec des options dédiées.

```
$ gcc -shared   a.o ... z.o   -o libproject.so
```

(des options plus riches pour le nommage de la bibliothèques sont disponibles)

Création d'un bibliothèque dynamique



la production de la bibliothèque s'effectue via la commande gcc appelée avec des options dédiées.

```
$ gcc -shared <fichiers objets> -o <nom du fichier sortie>
```

A propos du nommage des bibliothèques partagées

Le nom d'une bibliothèque dyn. $\neq$ nom du fichier contenant les ressources objets.

Cohabitation de plusieurs versions d'une bibliothèque dyn. $\Rightarrow$ convention de nommage

Nom du fichier (nom réel)	:	`libproject.so.1.0.1`
Nom propre (soname)	:	`libproject.so.1`
« Nom d'édition »	:	`libproject.so`

- le chargeur utilise le soname pour désigner une bibliothèque dyn.
- le soname d'une bibliothèque dyn. lui est propre et est contenu dans ses ressources.
- la gestion des bibliothèques présente sur le système est réalisée par l'utilitaire `ldconfig`

Compilation à l'aide d'une bibliothèque dynamique

Les options de compilation avec les bibliothèques dynamiques sont les mêmes que pour les bibliothèques statiques.

On verra néanmoins que l'usage de bibliothèques dynamiques demande plus d'attention.

Fortement déconseillé (mais fonctionne)

```
$ gcc prog.o <chemin vers la bibliothèque> -o program.exe
```

comme par exemple

```
$ gcc prog.o /home/bob/project/lib/libproject.so -o program.exe
```

Compilation à l'aide d'une bibliothèque dynamique

```
$ gcc prog.o -L<directory> -lname -o program.exe
```

comme par exemple

```
$ gcc prog.o -L/home/bob/project/lib -lproject -o program.exe
```

Options de gcc

- l'option `-L` indique à `gcc` un directory dans lequel il faut chercher la bibliothèque (si elle ne se trouve pas dans les directories de recherche par défaut)
- l'option `-l` permet de raccourir l'écriture du nom des bibliothèques. La référence à une bibliothèque `libname.so` sera ainsi abrégée en `-lname`.

Compilation à l'aide d'une bibliothèque dynamique

```
$ gcc prog.o -L<directory> -lname -o program.exe
```

comme par exemple

```
$ gcc prog.o -L/home/bob/project/lib -lproject -o program.exe
```

libproject.so ou libproject.a ?

Que se passe-t'il si
/home/project/lib/libproject.a
et
/home/project/lib/libproject.so
existent toutes les deux ?

Compilation à l'aide d'une bibliothèque dynamique

```
$ gcc prog.o -L<directory> -lname -o program.exe
```

comme par exemple

```
$ gcc prog.o -L/home/bob/project/lib -lproject -o program.exe
```

libproject.so ou libproject.a ?

Que se passe-t'il si
`/home/project/lib/libproject.a`
et
`/home/project/lib/libproject.so`
existent toutes les deux ?

Par défaut gcc maximise la mutualisation et choisira la bibliothèque partagée.

Difficultés liées à l'utilisation des bibliothèques partagées

Localisation des bibliothèques statiques

Besoin de localiser précisément une bibliothèque statique **pendant la compilation** (phase d'édition des liens)

Localisation des bibliothèques partagées

- besoin de localiser précisément une bibliothèque partagée **pendant la compilation** (phase d'édition des liens)
- besoin de localiser précisément une bibliothèque partagée **à l'exécution de l'application** (phase chargement dynamique)

Difficultés liées à l'utilisation des bibliothèques partagées

Localisation des bibliothèques statiques

Besoin de localiser précisément une bibliothèque statique **pendant la compilation** (phase d'édition des liens)

Localisation des bibliothèques partagées

- besoin de localiser précisément une bibliothèque partagée **pendant la compilation** (phase d'édition des liens)
- besoin de localiser précisément une bibliothèque partagée **à l'exécution de l'application** (phase chargement dynamique)

Avant d'aller plus loin : utilitaire `ldd`

L'utilitaire `ldd` permet d'afficher les dépendances d'une bibliothèque partagée ou d'un exécutable vis-à-vis de bibliothèque partagées.

```
$ ldd /usr/lib/libm.so
 /lib/ld-linux.so.2 (0xb7f12000)
 linux-gate.so.1 =>  (0xb7f11000)
 libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7d87000)
```

```
$ ldd /bin/bash
  linux-gate.so.1 =>  (0xb7f85000)
  libncurses.so.5 => /lib/libncurses.so.5 (0xb7f3f000)
  libdl.so.2 => /lib/tls/i686/cmov/libdl.so.2 (0xb7f3b00
  libc.so.6 => /lib/tls/i686/cmov/libc.so.6 (0xb7deb000)
 /lib/ld-linux.so.2 (0xb7f86000)
```

Exemple

```
/home/bob/project/
|-- libfoo.so
|-- ...
|-- main.c
\-- main.o
```

contenu du directory projet

Exemple

```
/home/bob/project/
|-- libfoo.so
|-- ...
|-- main.c
\-- main.o
```

```
$ gcc main.o -lfoo -o prog.exe

/usr/bin/ld: cannot find -lfoo
collect2: ld returned 1 exit status
```

(il est possible de contourner le problème en utilisant la variable d'environnement `$LIBRARY_PATH`)

Exemple

```
/home/bob/project/
|-- libfoo.so
|-- ...
|-- main.c
\-- main.o
```

```
$ gcc main.o libfoo.so -o prog.exe
$ ldd prog.exe
/lib/ld-linux.so.2 (0xb7f9e000)
libfoo.so (0xb7f99000)
...
$ ./prog.exe
$ cp ./prog.exe ../otherdir; cd ../otherdir
$ ./prog.exe
error while loading shared libraries: libfoo.so: cannot
open shared object file: No such file or directory
```

Exemple

```
/home/bob/project/
|-- libfoo.so
|-- ...
|-- main.c
\-- main.o
```

```
$ gcc main.o $PWD/libfoo.so -o prog.exe
$ ldd prog.exe
/lib/ld-linux.so.2 (0xb7f9e000)
/home/bob/project/libfoo.so (0xb7f99000)
...
$ ./prog.exe
$ mv libfoo.so ../otherdir/
$ ./prog.exe
error while loading shared libraries:
/home/bob/project/libfoo.so: cannot
open shared object file: No such file or directory
```

Exemple

```
/home/bob/project/
|-- libfoo.so
|-- ...
|-- main.c
\-- main.o
```

```
$ gcc main.o -L. -lfoo
$ ldd prog.exe
/lib/ld-linux.so.2 (0xb7f9e000)
libfoo.so (0xb7f99000)
...
$ ./prog.exe
$ mv libfoo.so ../otherdir/
$ ./prog.exe
error while loading shared libraries:
/home/bob/project/libfoo.so: cannot
open shared object file: No such file or directory
```

Exemple

```
/home/bob/project/
|-- libfoo.so
|-- ...
|-- main.c
\-- main.o
```

- variable d'environnement LIBRARY_PATH
  spécifie une liste de directories de recherche pour les bibliothèques partagées à **la compilation**
  `LD_LIBRARY_PATH = DIR1:DIR2:..:DIRn`
- variable d'environnement LD_LIBRARY_PATH
  spécifie une liste de directories de recherche pour les bibliothèques partagées à **l'exécution**
  `LD_LIBRARY_PATH = DIR1:DIR2:..:DIRq`

```
$ export LIBRARY_PATH = /home/bob/project: $LIBRARY_PATH
$ export LD_LIBRARY_PATH = /home/bob/project: $LD_LIBRARY_PATH
```

Exemple

```
/home/bob/project/
|-- libfoo.so
|-- ...
|-- main.c
\-- main.o
```

Proposition de gestions de la localisation des bibliothèques (attention)

- copie des bibliothèques vers les directories standards (`/usr/local/lib`)
- liens symboliques dans les directories standards (`/usr/local/lib`)
- procédure d'installation avec un fichier d'environnement à charger avant exécution (script shell qui met à jour les variables d'environnement)

Exemple

```
 /home/bob/project/
|-- libfoo.so
|-- ...
|-- main.c
|-- main.o
 \-- environ.sh
```

fichier environ.sh

```
PRJT_INSTALL_DIR=/home/bob/project

export LIBRARY_PATH=${PRJT_INSTALL_DIR}:${LIBRARY_PATH}
export LD_LIBRARY_PATH=${PRJT_INSTALL_DIR}:${LD_LIBRARY_PATH}
```

Fichier est à charger dans le shell avant exécution ou travail sur le code.

Quelques généralités

Forcer l'édition statique

On peut forcer l'utilisation de bibliothèques statiques pour tous les liens avec l'option `-static`

```
$ gcc -static main.o -lm -lproject ... -o prog.exe
```

Quelques généralités

Combiner bibliothèques partagées et statiques

On laisse utiliser les bibliothèques partagées par défaut et on spécifie «en dur » le chemin vers les bibliothèques statiques.

```
$ gcc -static main.o ... \
-lm -lproject ... \
/opt/lib/libfoo.a /opt/lib/libbar.a ... \
-o prog.exe
```

Quelques généralités

Combiner bibliothèques partagées et statiques

On laisse utiliser les bibliothèques partagées par défaut et on spécifie «en dur » le chemin vers les bibliothèques statiques.

```
$ gcc -static main.o ... \
-lm -lproject ... \
/opt/lib/libfoo.a /opt/lib/libbar.a ... \
-o prog.exe
```