# Compilation

Romaric DAVID

École d'Automne Informatique Scientifique
1er Octobre 2008

- Compilation séparée
- Bibliothèques statiques
- Bibliothèques dynamiques
- Paramétrage de l'édition de liens
- Étude de cas

Romaric DAVID

ulp UNIVERSITÉ LOUIS PASTEUR STRASBOURG

# Plan I



Romaric DAVID

ulp UNIVERSITÉ LOUIS PASTEUR STRASBOURG

# Compilation séparée I

- Compilation séparée : création de plusieurs fichiers objets (un par ensemble de fonctions)
- Assemblage de ces fichiers en un exécutable : `g77 a.o b.o c.o -o monprog`
- = Édition de liens

# Distribution de code objet : un procédé simple I

- Problématique : rassembler les différents fichiers objet.
- Un procédé simple : la bibliothèque statique (archive)
- Assemblage de fichiers objets à intégrer lors de l'édition de liens.
- Édition de liens : résolution des branchements dans le code (appel de fonction = branchement) à la compilation

Utilisation de `ar`

- `ar cr archive.a fichier1.o fichier2.o`
- Construction d'un index des fonctions présentes dans l'archive, afin d'accélérer l'édition de liens.
- La mise à jour des composants de l'archive (fichiers objet) est possible individuellement : `ar r archive.a objet.o`

# Plan I



Romaric DAVID

ulp UNIVERSITÉ LOUIS PASTEUR STRASBOURG

# Utilisation de la bibliothèque I

- Utilisation du nom/chemin complet à la compilation
- Indication à l'éditeur de liens : emplacement et nom ($\Rightarrow$ convention sur le nom)

Résultat identique : recopie de l'ensemble du code objet dans l'exécutable.

- Convention de nommage : on parle de library/bibliothèque
- Un nom de fonction présent dans une bibliothèque est appelé symbole.
- Si l'on nomme l'archive `lib`**`mesfonctions`**`.a`
- $\Rightarrow$Utilisation par `-lmesfonctions`
- Indiquer l'emplacement : `-L/chemin/vers/la/bibliothèque`

# Utilisation de la bibliothèque II

Le compilateur utilise certains chemins de recherche des bibliothèques par défaut. Par exemple, pour le compilateur intel, on trouve entre autres /opt/intel/cc/10.1.008/lib /usr/lib /usr/local/lib

# Quelques ennuis... I

- Ordre à l'édition de liens
- Dépendances circulaires
- Utiliser le groupement de bibliothèques -Wl,–start-group a.a b.a -Wl,–end-group (peut être utile pour lapack/scalapack)

# Caractéristiques des bibliothèques statiques I

- Exécutable auto-suffisant (appelé exécutable statique)
- Plutôt indépendant de l'état du système (et de l'environnement du processus)
- Inconvénient : modification dans la bibliothèque (correction de bugs . . . ) $\Rightarrow$ recompilation de code
- Si plusieurs codes utilisent les mêmes bibliothèques, le code de ces bibliothèques est recopié plusieurs fois.

# Plan I



Romaric DAVID

ulp UNIVERSITÉ LOUIS PASTEUR STRASBOURG

# Distribution de code objet : un procédé élégant I

- Création de bibliothèques dynamiques (option `-shared` du compilateur)
- Compilation du code avec références à ces bibliothèques (options `-L` et `-l` , identiques à la compilation statique
- -L indique l'emplacement des bibliothèques lors de la compilation du programme
- -ltruc $\Rightarrow$ Fichier libtruc.so
- À l'exécution, calcul des branchements
- Si plusieurs programmes utilisent les mêmes bibliothèques, ils se partagent le code de ces bibliothèques.

# Exécution d'un programme dépendant de bibliothèques dynamiques I

- Visualisation des dépendances : `ldd`
- Variables d'environnement LD_LIBRRAY_PATH (utilisateur)
- Chemins de recherche standard du système : `/etc/ld.so.conf`
- `/usr/lib puis /lib`

# Exécution d'un programme dépendant de bibliothèques dynamiques II

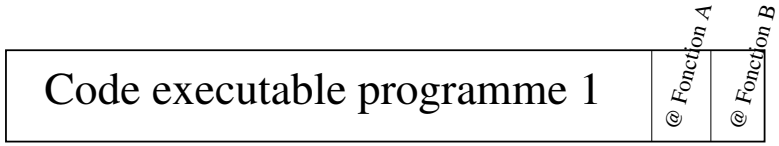
ld.so parcourt successivement tous les emplacements indiqués à la recherche des bibliothèques dont dépend le programme (cette information provient de la compilation du programme). Si une bibliothèque vient à manquer, un message du type `./adyn : error while loading shared libraries : liba.so : cannot open shared object file : No such file or directory` est affiché et l'exécution du programme s'arrête.

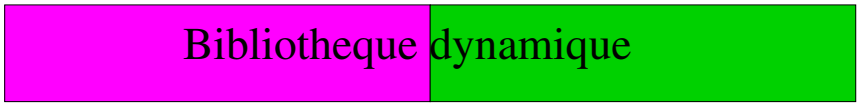
# Environnement d'exécution I

Problème : comment fixer à l'exécution le chemin de recherche des bilbiothèques ?

- Pour le chargeur dynamique (`LD_LIBRARY_PATH` ), dans l'environnement utilisateur
- Embarqué dans l'exécutable :
  - `-rpath` à l'édition de liens
  - Plus directement `-Wl,-rpath=....` à la compilation
  - ou variable `LD_RUN_PATH`

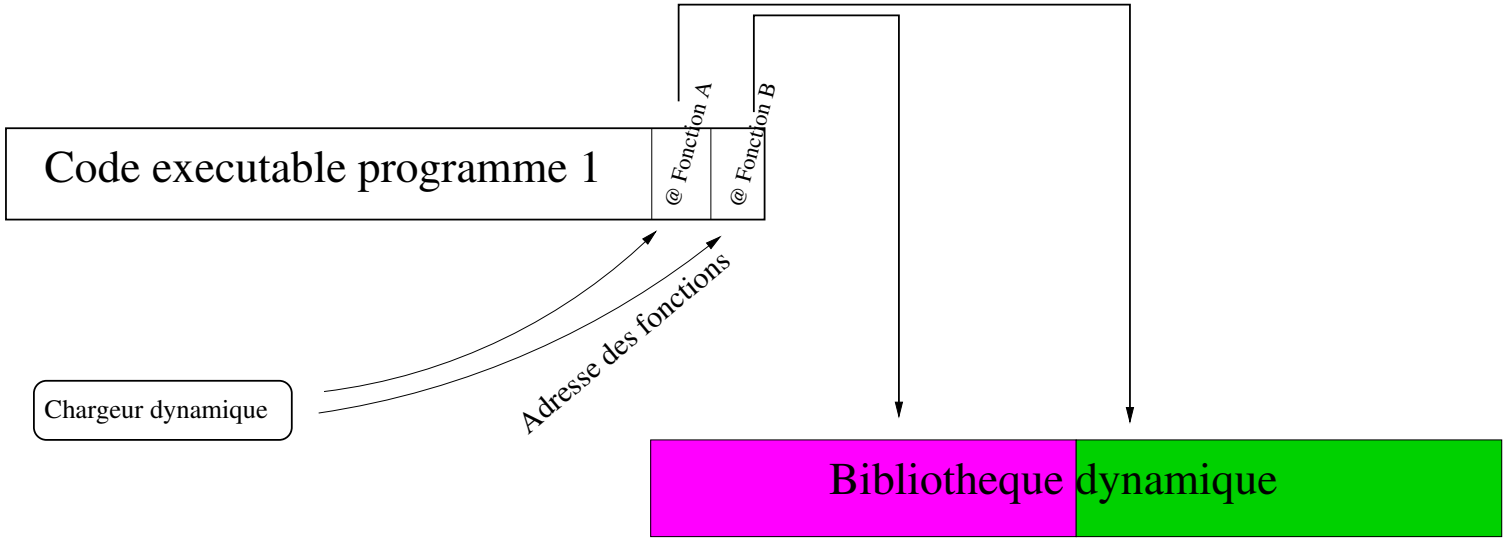
# The big picture



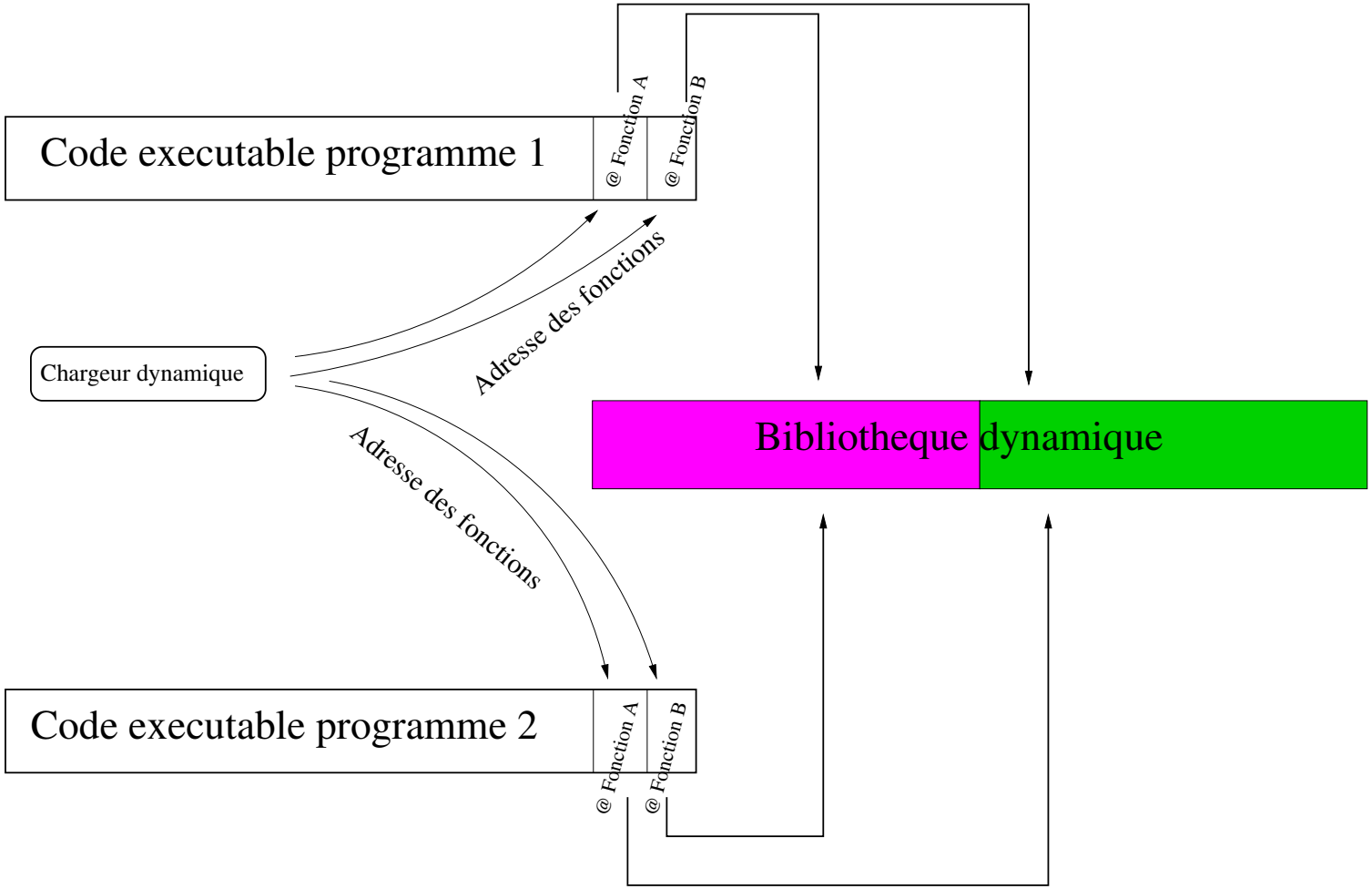

Romaric DAVID

# The big picture




Romaric DAVID

# The big picture




Romaric DAVID

# Plan I



Romaric DAVID

ulp UNIVERSITÉ LOUIS PASTEUR STRASBOURG

# Statique ou dynamique ? I

- Par défaut, choix de la bibliothèque dynamique
- Option -Wl,-Bstatic : les bibliothèques indiquées après cette option seront liées statiquement.

# Fonctionnalités avancées I

- Mise à jour du système facilitée
- Bibliothèques d'interposition : permettent d'intercepter les appels (debug, profiling). Exemple : vampirtrace

# Scripts du linker I

- Tout fichier n'étant pas une archive ou une bibliothèque dynamique sera interprété comme un script
- Script : ensemble de commandes destinées à l'éditeur de liens
- Commandes `INCLUDE` ou `GROUP`

Romaric DAVID

# Plan I



Romaric DAVID

ulp UNIVERSITÉ LOUIS PASTEUR STRASBOURG

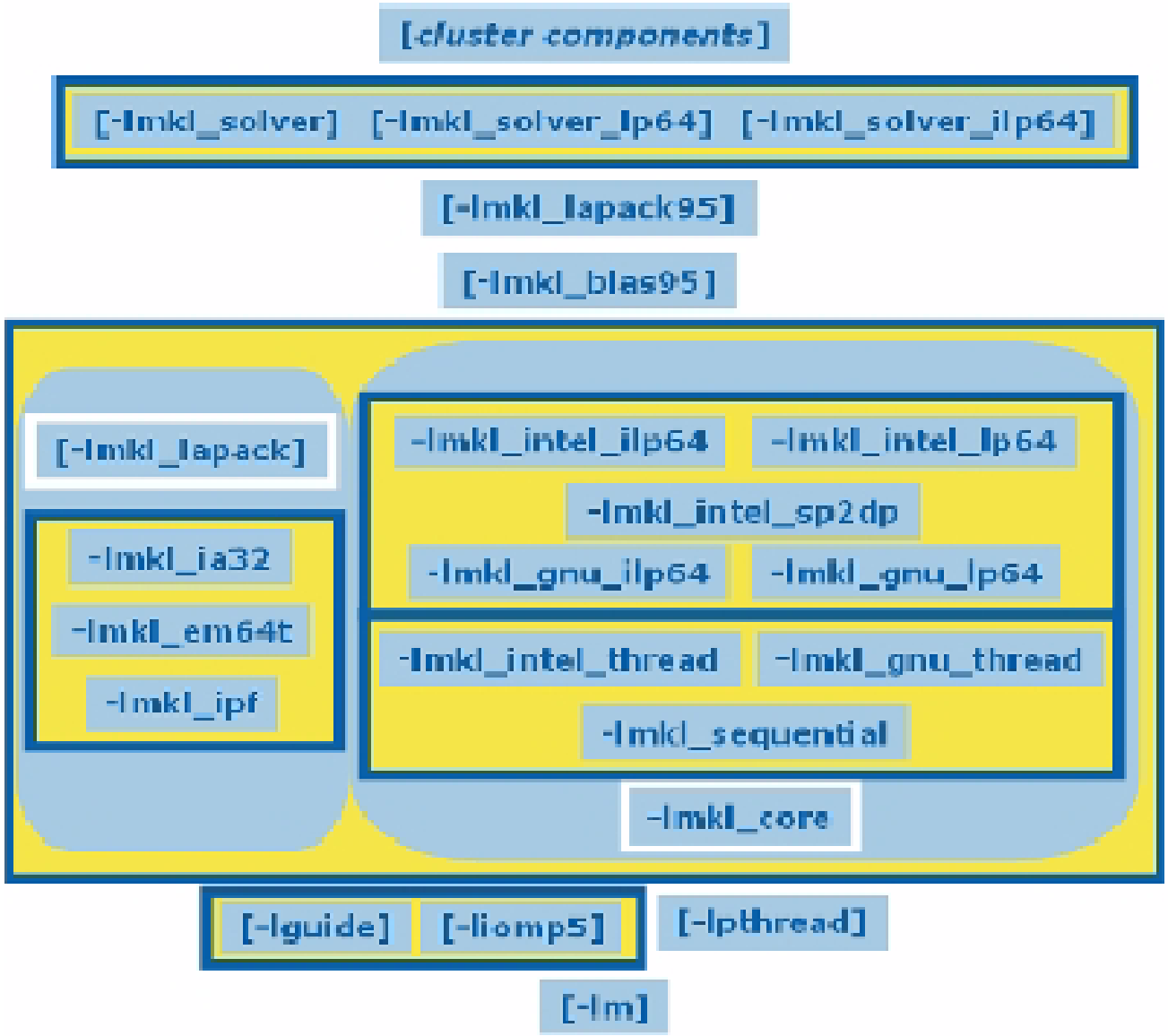
# Aperçu de la bibliothèque I

Les dépendances entre les différentes composantes peuvent se représenter comme ceci :

Romaric DAVID

# Aperçu de la bibliothèque II

## Intel MKL 10.0




Romaric DAVID

# Deux exemples de compilation I

- Version parallèle (threads) de la bibliothèque avec le compilateur intel : `-L/chemin-mkl -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core`
- Version séquentielle `-L/chemin-mkl -lmkl_intel_lp64 -lmkl_sequential -lmkl_core`
- La première version est accessible sous la forme d'un linker script : `-L/chemin-mkl -lmkl`

Romaric DAVID

ulp UNIVERSITÉ LOUIS PASTEUR STRASBOURG

# Plan I

# En résumé...

- Construction de bibliothèques : tâche facile
- Codes parallèles : préférer les bibliothèques dynamiques
- Nécessite environnement utilisateur bien huilé
- Porte ouverte à l'interfaçage avec d'autres langages