

Chaîne de compilation

TL;DR de la séance 1 · aperçu de la séance 2

Samuel Tardieu

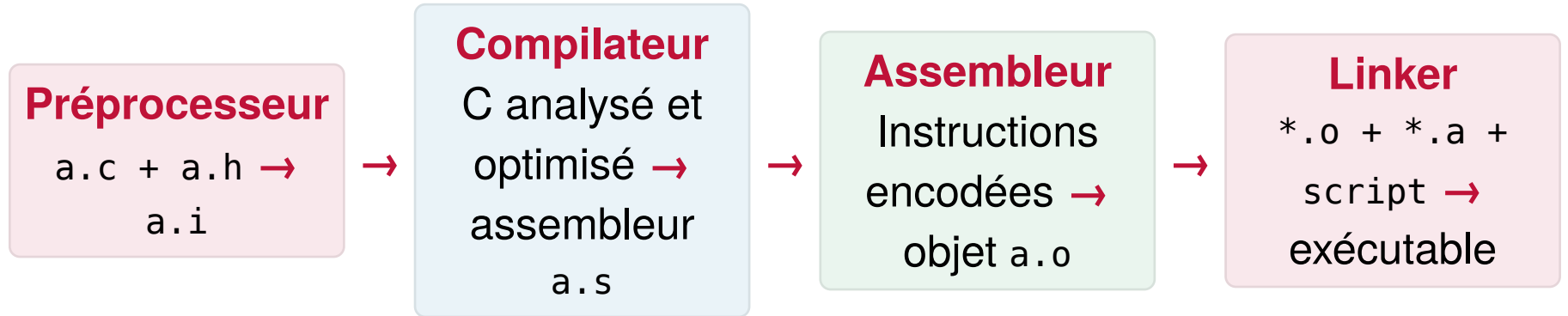
4SE03 / 4SE07

Télécom Paris



Rappel : du C aux objets

La carte du territoire



GCC n'est pas « le compilateur »

C'est un **driver** qui orchestre cpp, le compilateur, as et ld, avec les bibliothèques et fichiers de démarrage.

Cross-compilation

La chaîne s'exécute sur l'**hôte** mais produit du code pour une autre **cible**. Le pipeline reste le même.

Observer chaque étape : gcc -E · gcc -S · gcc -c · gcc -v

C : les réflexes à conserver

Types

- char, int, long n'ont pas une largeur universelle.
- Si la représentation compte : uint32_t, int64_t, etc.
- Si seule une largeur minimale compte : types least / fast.

Qualificateurs

- const exprime ce qui ne doit pas être modifié.
- volatile impose des accès observables ; il est adapté aux registres matériels, **pas** à la synchronisation multicœur.

Interfaces

- Tout header possède une garde d'inclusion.
- Une signature correcte de main est obligatoire.
- Les goto restent utiles pour sorties imbriquées et nettoyage d'erreurs.

Registre mappé en mémoire

```
#define REG (*(volatile uint32_t  
*) 0xff00ff00)
```

Le type est un contrat avec le compilateur — et souvent avec le matériel.

Du C au code machine : perdre la « magie »

Le front-end

1. analyse syntaxe et sémantique ;
2. construit une représentation intermédiaire — **GIMPLE** chez GCC ;
3. simplifie et optimise ;
4. confie le résultat au générateur de code.

Le back-end

- transforme les expressions en opérations élémentaires ;
- alloue les registres ;
- produit prologues et épilogues ;
- abaisse boucles et conditions en comparaisons et branchements.

Sur une architecture load/store

adresse → LDR → **registre** → calcul → STR → mémoire

Une variable C désigne sa **valeur** ; pour l'assembleur et le linker, son symbole désigne son **adresse**.

Le code machine ne connaît ni for, ni expression complexe : seulement des effets élémentaires.

Séparer pour compiler : déclarations et définitions

Déclarer

Associer un **nom** à un **type**, afin que chaque unité de compilation puisse générer du code.

```
// module.h
extern int counter;
int update(int delta);
```

Définir

Réserver le stockage d'une variable ou fournir le corps d'une fonction — **une seule fois**.

```
// module.c
#include "module.h"
int counter;
int update(int d) { return
counter += d; }
```

Règle d'architecture

- objet privé à un .c → `static` ;
- objet exporté → déclaration dans le .h, définition dans le .c ;
- dans un header : déclarations, types, macros, enum, fonctions `static inline` — **aucune autre définition** ;
- symbole `weak` : définition par défaut remplaçable.

Compilation séparée : la question laissée ouverte

Ce que sait le compilateur

Pour compiler un appel, il suffit de connaître :

- les types des paramètres ;
- le type de retour ;
- les conventions d'appel de l'ABI.

L'adresse finale peut rester inconnue.

Ce que contient alors a.o

- du code et des données par morceaux ;
- une table de symboles ;
- des références externes non résolues ;
- des informations indiquant où et comment corriger ces références.

```
// a.c
extern void f(int);
void g(void) { f(42); }
```

```
// b.c
void f(int x) { /* ... */ }
// Où se trouvera f en mémoire ?
```

Le compilateur peut promettre : « appeler f ici ».

**Mais qui transforme cette promesse en une adresse
réelle ?**

L'éditeur de liens.

Aujourd'hui : objets, ELF et mémoire

Le linker : résoudre, reloger, assembler

1 · Résoudre

Associer chaque référence externe à une définition de symbole. L'ordre des objets et bibliothèques peut compter.

2 · Reloger

Attribuer les adresses finales et **patcher** code et données grâce aux entrées de relocation.

3 · Assembler

Fusionner les sections d'entrée selon une politique de placement.

4 · Optimiser

Avec `-flto`, retrouver une vision globale du programme au moment du lien.

Options à reconnaître : `-lfoo · -L chemin · -Wl, ... · -nostartfiles · -nodefaultlibs · -nostdlib`

Question directrice : comment passer d'objets « à l'adresse 0 » à une image mémoire cohérente ?

ELF : deux vues du même fichier

Sections — vue du linker

- `.text` : code
- `.rodata` : constantes
- `.data` : données initialisées
- `.bss` : données nulles, sans octets dans le fichier
- tables de symboles et relocations

Outils : `readelf -S`, `objdump -h -t -r`

Segments — vue du loader

Regroupent les sections par droits et mode de chargement :

- lecture + exécution ;
- lecture seule ;
- lecture + écriture.

Outils : `readelf -l`, `objdump -p`

Le point subtil à maîtriser

VMA : adresse d'exécution · **LMA** : adresse de stockage/chargement.

En embarqué, `.data` est stockée en ROM à sa LMA, puis copiée en RAM à sa VMA ; `.bss` est seulement réservée puis mise à zéro.

Linker script + démarrage : fabriquer la mémoire voulue

```
MEMORY {  
    rom (rx) : ORIGIN = 0x00000000,  
    LENGTH = 256K  
    ram (wx) : ORIGIN = 0x20000000,  
    LENGTH = 64K  
}  
SECTIONS {  
    .text : { *(.text*)  
    *(.rodata*) } > rom  
    .data : { *(.data*) } > ram AT>  
    rom  
    .bss : { *(.bss*) } > ram  
}
```

Le script de lien décide

- régions mémoire et contraintes ;
- regroupement et placement des sections ;
- point d'entrée ;
- symboles frontières (`_data`, `_edata`, `_bstart`, ...).

Puis `crt0` prépare le C

1. initialise la pile ;
2. copie `.data` de ROM vers RAM ;
3. met `.bss` à zéro ;
4. appelle `main`.

Enfin : garder l'ELF pour analyser et déboguer, puis produire l'image flashable avec `objcopy -O binary`.

À la fin de la séance, vous saurez...

Lire

les sections, segments, symboles et relocations d'un objet ou exécutable ELF.

Expliquer

où résident code, constantes, données initialisées, données nulles, pile et tas.

Piloter

le linker avec ses options et un script simple décrivant ROM et RAM.

Relier

le script, les adresses LMA/VMA, le travail de `crt0` et l'image binaire flashée.

Fil rouge : source C → objet relogeable → ELF lié → état mémoire au démarrage

Source : *Chaîne de compilation — Genèse et autopsie des exécutables*, A. Polti & S. Tardieu, 2020–2024.

Synthèse diffusée dans le même cadre académique que le support d'origine.