

Compilation et make

F. Gluck

v0.7

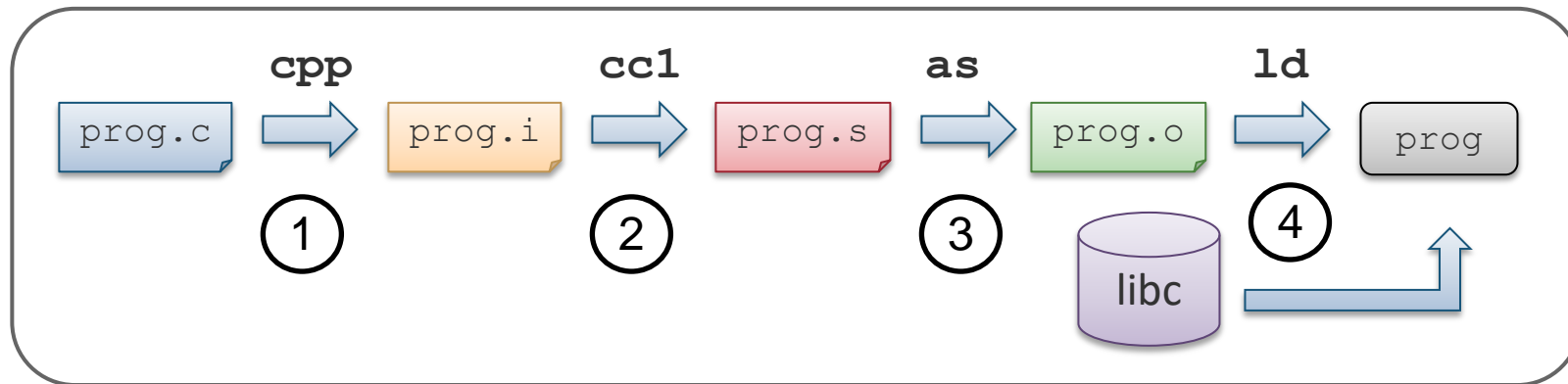
Etapes de génération d'un exécutable?

Le compilateur C utilisé ici est `gcc` (**GNU C compiler**). Générer un binaire exécutable avec `gcc` implique 4 étapes:

1. **Précompilation:** `gcc` appelle `cpp`, le **préprocesseur** qui effectue de la substitution de texte (defines, macros, includes, etc.) et génère du code C (texte) portant l'extension `.i`
2. **Compilation assembleur:** `gcc` appelle `cc1` qui compile le code C en code assembleur (texte) portant l'extension `.s`
3. **Compilation code objet:** `gcc` appelle `as`, l'**assembleur** qui compile le code assembleur en code machine (code objet) portant l'extension `.o`
4. **Edition des liens:** `gcc` appelle `ld`, l'**éditeur de liens** qui lie le code objet avec les bibliothèques (et potentiellement d'autres codes objets) pour produire le binaire exécutable final

Etapes de génération

Compilation de `prog.c` avec `gcc prog.c -o prog`



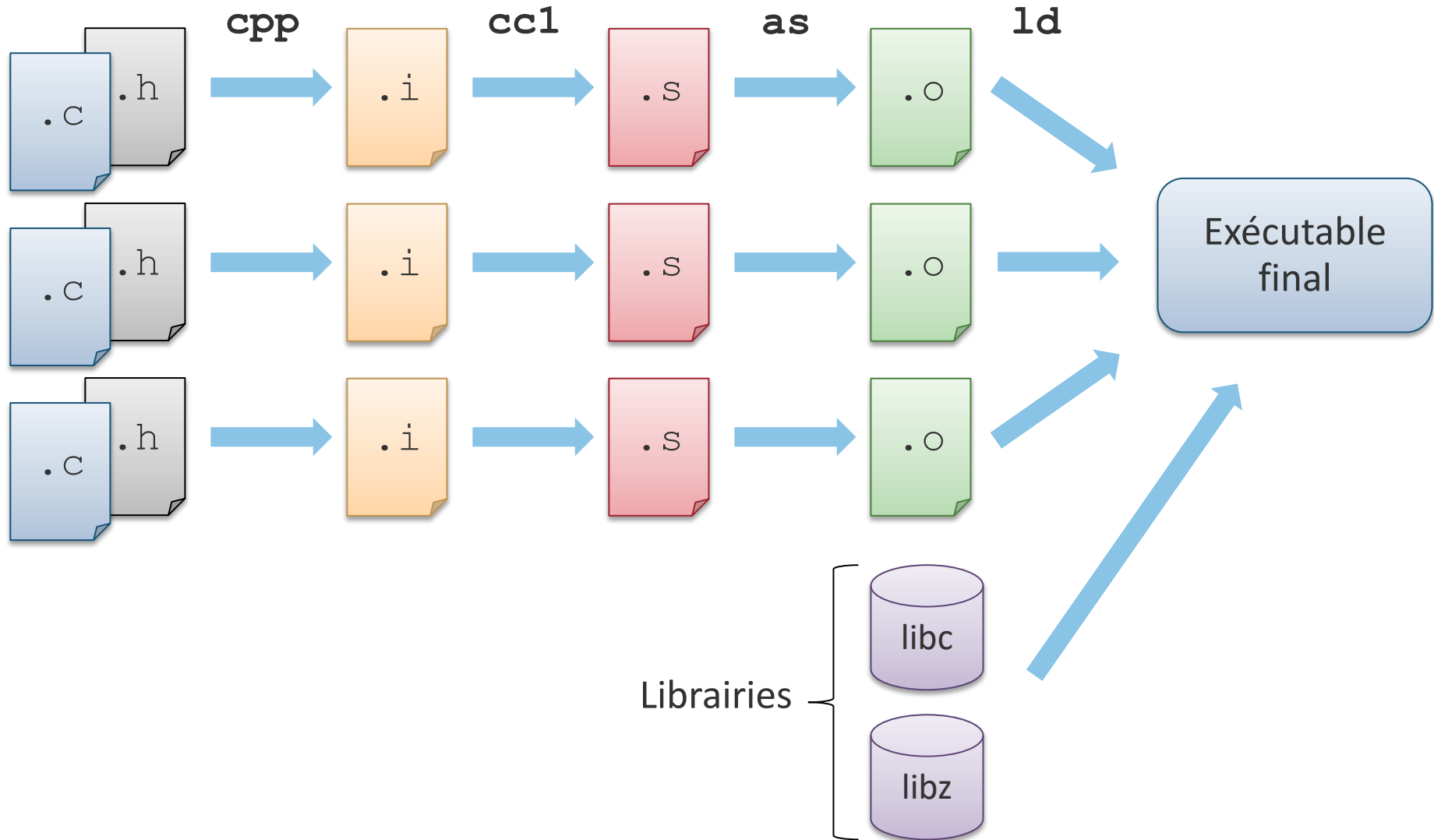
1. Précompilation
2. Compilation assembleur
3. Compilation code objet
4. Edition des liens

Remarque: les fichiers `.i` et `.s` sont effacés par défaut par le compilateur.

Options de compilation

- Compilation et édition des liens avec :
`gcc -g -Wall -Wextra -std=gnu11 prog.c -o prog`
- `-g` indique de générer les informations de débogage.
- `-Wall` et `-Wextra` indique au compilateur d'alerter le programmeur d'un maximum d'erreurs potentielles.
- `-std=gnu11` spécifie de compiler pour la variante GNU du standard C ISO 2011
- `-o` définit le fichier exécutable à produire en sortie ; si ce dernier est omis, alors un fichier `a.out` est automatiquement créé.

Plusieurs fichiers sources



Compilation séparée (1)

Projet contenant 3 fichiers source: **main.c**, **sum.c**, **sum.h**

```
#include <stdio.h>
#include "sum.h"

int main() {
    int tab[] = { 1,2,3,4 };
    int n = sizeof(tab)/sizeof(tab[0]);
    printf("sum: %d\n", sum(tab,n));
    return 0;
}
```

main.c

```
#include "sum.h"

int sum(int tab[], int length) {
    int s = 0;
    for (int i = 0; i < length; i++)
        s += tab[i];
    return s;
}
```

sum.c

```
#ifndef _SUM_H_
#define _SUM_H_

int sum(int tab[], int length);

#endif
```

sum.h

Compilation séparée (2)

- La compilation séparée est nécessaire dans le cas de plusieurs fichiers sources.
- L'option `-c` indique à `gcc` de ne pas appeler le linker `ld`
- Exemple pour les fichiers de la slide précédente:

- Compilation des fichiers sources en fichiers objets:

```
gcc -g -Wall -Wextra -std=gnu11 -c main.c
```

```
gcc -g -Wall -Wextra -std=gnu11 -c sum.c
```

- Edition des liens pour produire le fichier binaire final:

```
gcc -g main.o sum.o -o prog
```

Make

Make

Qu'est ce que **make**?

- Utilitaire permettant d'automatiser le processus de création de fichiers en gérant leurs dépendances.
- Souvent utilisé lors de la compilation de code source en code objet, puis pour l'édition des liens afin de créer le binaire final.
- Tout projet conséquent utilise un outil similaire à make.

Make permet de définir :

- des **cibles** (*targets*)
- les **dépendances** des cibles
- les **actions** permettant de construire les cibles

Makefile

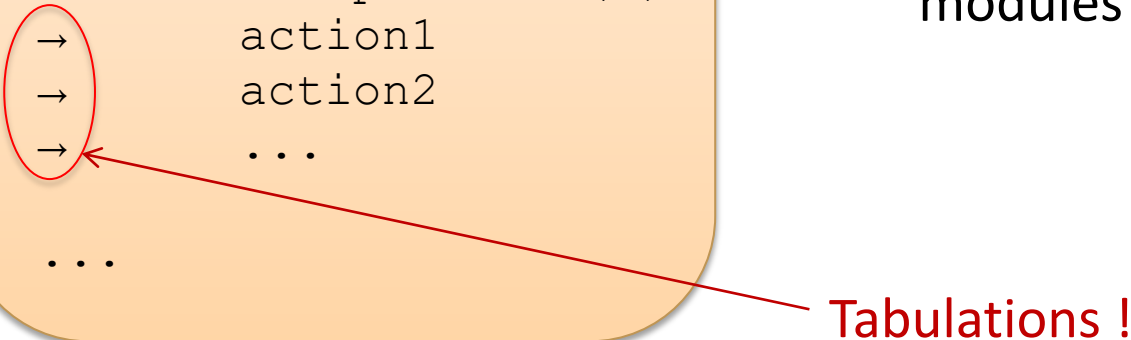
Syntaxe générale d'un makefile :

```
variables
...

cible1 : dépendance(s)
→      action1
→      action2
→      ...

cible2 : dépendance(s)
→      action1
→      action2
→      ...

...
```



- Un makefile est un fichier lu par make, contenant cibles, dépendances et actions.
- En général : makefile à la racine du projet et dans les sous-répertoires constituant les modules du projet.

Exemple (1)

example.c

```
#include <stdio.h>

int main(int argc, char **argv){
    printf("Programme très complexe...");
    return 0;
}
```

makefile

```
example: example.c
    gcc example.c -o example
```

```
> make
gcc example.c -o example
```

Exemple (2)

MyMakefile

```
# Cible principale
example: example.o
        gcc -o example example.o

# Deuxième cible
example.o: example.c example.h
        gcc -c example.c

# Efface fichiers objets et exécutable
clean:
        rm example.o example
```

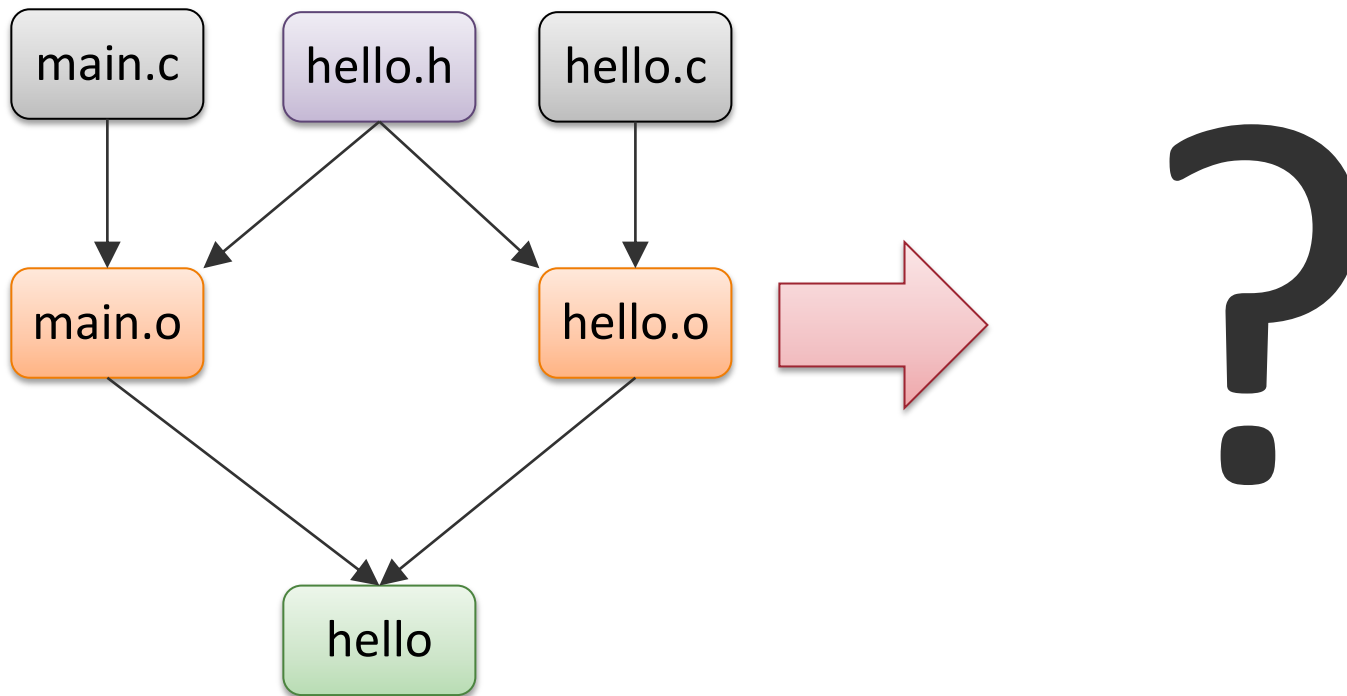
```
> make -f MyMakefile clean
rm example.o example
> make -f MyMakefile
gcc -c example.c
gcc -o example example.o
```

Principe

- make cherche un fichier **makefile** ou **Makefile** dans le répertoire courant. Sinon, spécifier le nom du fichier avec '-f'
- Décide si une cible doit être régénérée en comparant les dates de modification des fichiers ➡ **ne recompile que ce qui a été modifié.**
- Exécute la **première cible** ou celle spécifiée en ligne de commande.
- Si les dépendances sont satisfaites :
 - exécute les actions définies pour la cible.
- Sinon
 - prend la première dépendance pour cible, ensuite la 2ème, etc.
➡ processus **récuratif** !

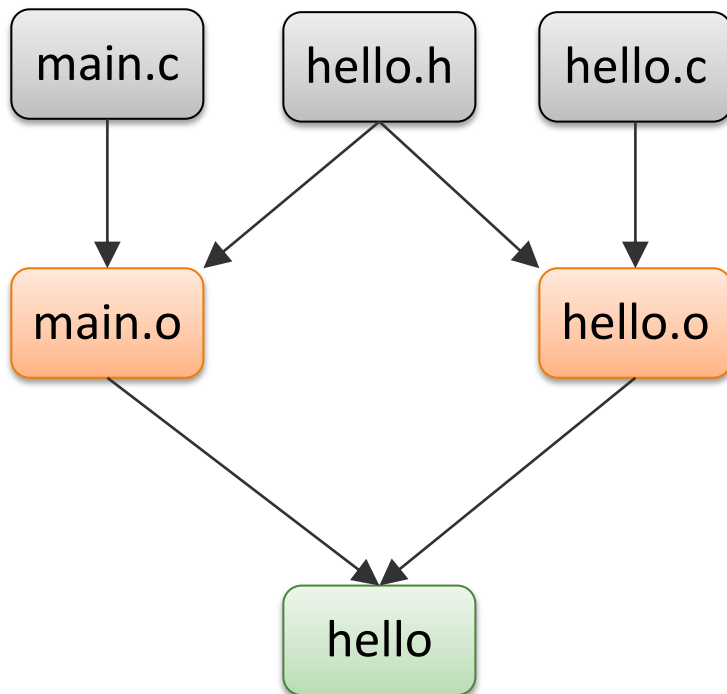
Quiz

Donnez un makefile permettant de construire le programme `hello` selon l'arbre de dépendances suivant :



Quiz

Donnez un makefile permettant de construire le programme `hello` selon l'arbre de dépendances suivant :



```
hello: hello.o main.o
    gcc hello.o main.o -o hello

hello.o: hello.c hello.h
    gcc -Wall -O2 -c hello.c

main.o: hello.h main.c
    gcc -Wall -O2 -c main.c

clean:
    rm -f *.o hello

rebuild: clean hello
```

Variables

Variables utilisateur

- Déclaration : `nom=valeur`
`nom=valeur1 valeur2 valeur3`
- Utilisation : `$ (nom)`
- Une variable peut aussi être déclarée en ligne de commande :
`make CFLAGS="-O2 -Wall"`

Variables internes

- `$@` : représente la cible
- `^` : représente la liste des dépendances
- `$<` : représente la première dépendance
- `$*` : représente le nom de la cible sans extension

Exemple

```
hello: hello.o main.o
    gcc hello.o main.o -o hello

hello.o: hello.c hello.h
    gcc -Wall -Wextra -c hello.c

main.o: main.c
    gcc -Wall -Wextra -c main.c

clean:
    rm -f *.o hello

rebuild: clean hello
```



```
CC=gcc -Wall -Wextra

hello: hello.o main.o
    $(CC) $^ -o $@

hello.o: hello.c hello.h
    $(CC) -c $<

main.o: main.c
    $(CC) -c $<

clean:
    rm -f *.o hello

rebuild: clean hello
```

Le manuel de GNU make

- <http://www.gnu.org/software/make/manual/make.html>

Quelques tutoriels

- <http://www.opussoftware.com/tutorial/TutMakefile.htm>
- <http://mrbook.org/tutorials/make/>