

Compilation séparée, Makefile

INF304

# Rappels – Modules

Programmes composés de *modules* :

- Structuration du programme
- Abstractions
- Passage à l'échelle

# Rappels – Modules

Programmes composés de *modules* :

- Structuration du programme
- Abstractions
- Passage à l'échelle

Modules **indépendants** du programme les utilisant

# Compilation séparée

Modularité  $\implies$  compilation *séparée*.

# Compilation séparée

Modularité  $\implies$  compilation *séparée*.

Les modules peuvent être compilés séparément (et en parallèle, `make -j4`)

- Compilation : transformation *source*  $\Rightarrow$  *langage machine*
- Commande correspondante : `clang -c test.c`

# Compilation séparée

Modularité  $\implies$  compilation *séparée*.

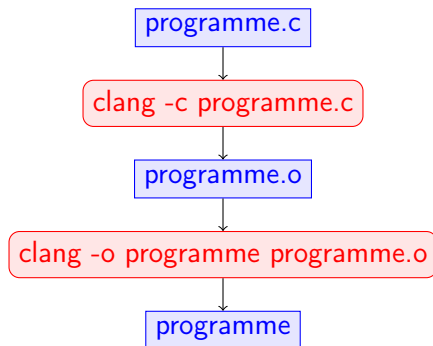
Les modules peuvent être compilés séparément (et en parallèle, `make -j4`)

- Compilation : transformation *source*  $\Rightarrow$  *langage machine*
- Commande correspondante : `clang -c test.c`

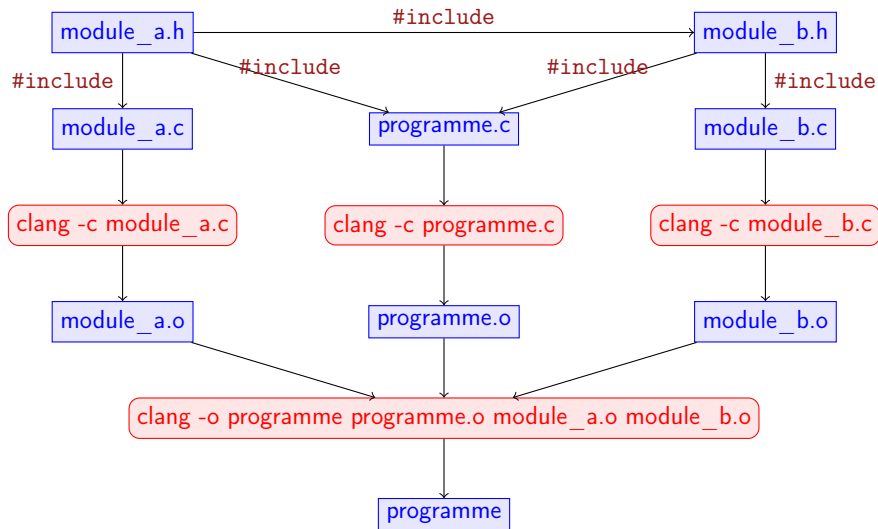
L'édition de liens (finale) n'est pas modulaire : combinaison de tous les résultats de la compilation.

- Commande correspondante : `ld -o prog test.o`

# Exemple – Un seul module



# Exemple – Plusieurs modules





# Makefile

- Définition de **règles** (*rules*) pour construire des **cibles** (*targets*)

```
cible: dépendance_1 ... dépendance_n  
    commande <arg1> ... <argn>
```

commande est précédée par une **tabulation** !

# Makefile

- Définition de **règles** (*rules*) pour construire des **cibles** (*targets*)

```
cible: dépendance_1 ... dépendance_n  
    commande <arg1> ... <argn>
```

commande est précédée par une **tabulation** !

Exécution avec :

```
# Fabrication d'une règle
```

```
$ make cible
```

```
# Fabrication de la première règle du fichier
```

```
$ make
```

# Exécution d'un Makefile

Une règle est exécutée si :

- Le fichier cible n'existe pas
- Le fichier cible est plus **ancien** que l'une de ses *dépendances*

# Exécution d'un Makefile

Une règle est exécutée si :

- Le fichier cible n'existe pas
- Le fichier cible est plus **ancien** que l'une de ses *dépendances*

Variables :

- `$<` : nom de la première dépendance
- `$@` : nom de la cible
- `$^` : liste des des dépendances

`CC=clang`

`VARIABLE=test`

```
cible: $(VARIABLE).c $(VARIABLE).h
    $(CC) -o $@ $<
```