

**programme C divisé en plusieurs
fichiers**

Makefile

Compilation

Pour compiler le fichier liste_lignes.c en ligne de commande :

```
gcc -Wall -g -c -o liste_lignes.o liste_lignes.c
```

L'option **-c** indique qu'il faut juste compiler et produire un fichier *.o , un fichier compilé mais pas exécutable. L'option -o donne le nom de fichier .o compilé

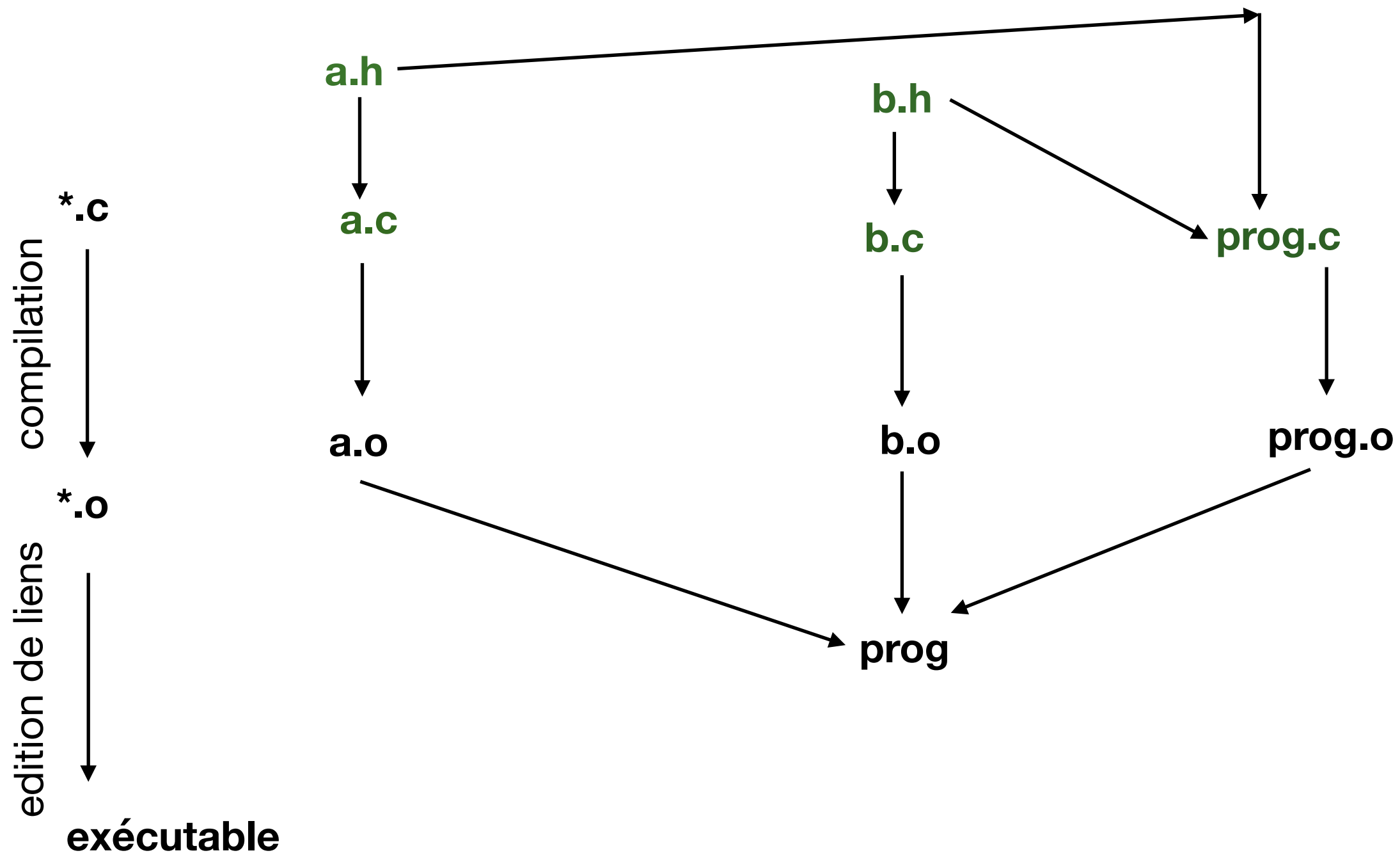
Pour obtenir un exécutable à partir de tous les fichiers compilés *.o :

```
gcc liste_lignes.o editeur.o affichage.o prog.o -o prog  
-lm -lreadline
```

-l pour indiquer une bibliothèque utilisée

On suppose que prog.c contient main(), C'est le seul fichier sans .h correspondant.

graphe de dépendances




prog.c contient la fonction main(),
prog.c est le seul fichier sans prog.h correspondant

Makefile minimal

Le fichier Makefile est composé de dépendances et de commandes:

```
prog : prog.o a.o b.o    #dépendance
      gcc -o prog prog.o a.o b.o (-lm)    #commande
prog.o : prog.c a.h b.h
      gcc -c -o prog.o prog.c
a.o : a.c a.h
      gcc -c -Wall -g -pedantic -o a.o a.c
b.o : b.c b.h
      gcc -c -Wall -g -pedantic -o b.o b.c
```



The diagram shows a box containing the text `-lm` from the first rule's command line. An arrow points from this box to a rectangular box containing the text "si bibliothèque math".

Makefile minimal

Ligne de dépendance :

```
target : dep1 dep2 dep3 ... depN
```

dit que

- si un de fichiers dep1 dep2 etc depN a changé
 - ou si le fichiers target n'existe pas
- alors il faut exécuter la commande qui suit.

La ligne de commande commence par le caractère de tabulation TAB.

La première dépendance concerne l'exécutable.

Makefile minimal

```
prog.o : prog.c a.h b.h
```

```
 gcc -c -o prog.o prog.c
```

parce que prog.c fait include de a.h et b.h

les *variables* de make

Makefile avec de variables

CC=gcc

CFLAGS= -Wall -pedantic -g

LDLIBS= -lm

prog : prog.o a.o b.o

\$(CC) -o prog prog.o a.o b.o \$(LDLIBS)

prog.o : prog.c a.h b.h

\$(CC) \$(CFLAGS) -c -o prog.o prog.c

a.o : a.c a.h

\$(CC) -c \$(CFLAGS) -o a.o a.c

b.o : b.c b.h

\$(CC) -c \$(CFLAGS) -o b.o b.c

CC : le nom de compilateur

CFLAGS : les options de compilation

LDLIBS : les bibliothèques

Variables internes

`$@` : le nom de cible (à gauche de `<< : >>`)
`$<` : le nom de la première dépendance
`$^` : la liste de dépendances
`$?` : la liste de dépendances plus récentes que la cible
`$*` : le nom de fichier sans suffixe

`CC=gcc`

`CFLAGS= -Wall -pedantic -g`

`LDLIBS= -lm`

`prog : prog.o a.o b.o`
`$(CC) -o $@ $^ $(LDLIBS)`

`prog.o : prog.c a.h b.h`
`$(CC) -o $@ -c $< $(CFLAGGS)`

`a.o : a.c a.h`
`$(CC) -o $@ -c $< $< $(CFLAGS)`

`b.o : b.c b.h`
`$(CC) -o $@ -c $< $(CFLAGS)`

Supprimer les commandes par défaut et insérer les cibles fictives

```
CC=gcc
CFLAGS=    -Wall -g -pedantic
LDLIBS=    -lm
```

```
prog: prog.o a.o b.o
```

```
a.o : a.c a.h
```

```
b.o : b.c b.h
```

```
prog.o : prog.c a.h b.h
```

```
clean:
```

```
    rm -rf *~
```

```
cleanall:
```

```
    rm -rf *~ *.o prog
```

**pour compiler la commande juste
le nécessaire :**

make

pour recompiler tout :

make cleanall

make

créer la liste de fichier objet

```
CC=gcc
CFLAGS= -Wall -g -pedantic
LDLIBS= -lm
```

```
SRC= prog.c a.c b.c
```

```
OBJ= $(SRC:.c=.o)
```

```
prog: $(OBJ)
```

```
a.o : a.c a.h
```

```
b.o : b.c b.h
```

```
prog.o : prog.c a.h b.h
```

```
clean:
```

```
rm -rf *~
```

```
cleanall:
```

```
rm -rf *~ *.o prog
```

**pour compiler la commande juste
le nécessaire :**

make

pour recompiler tout :

make cleanall

make

plusieurs fichiers avec la fonction main()

```
CC=gcc
CFLAGS= -Wall -g -pedantic
LDLIBS= -lm
```

```
ALL= prog1 prog2
```

```
all: $(ALL)
```

```
prog1: prog1.o a.o b.o
```

```
prog2: prog2.o a.o b.o
```

```
a.o : a.c a.h
```

```
b.o : b.c b.h
```

```
prog1.o : prog1.c a.h b.h
```

```
prog2.o : prog2.c a.h b.h
```

```
clean:
```

```
rm -rf *~
```

```
cleanall:
```

```
rm -rf *~ *.o $(ALL)
```

prog1.c et prog2.c contiennent la fonction main()

D'autres options et variables

```
CC=gcc
CFLAGS= -std=c11 -Wall -O3 -g -I../include -I/home/toto/include
LDLIBS= -lm -lreadline
```

CFLAGS – les drapeaux de compilation+préprocesseur

Pour les fichiers *.h qui ne se trouvent pas dans le répertoire courant on indique leur emplacement avec l'option -I (si nécessaire plusieurs fois)

Certains utilisent

```
CPPFLAGS=-I/home/toto/include -I../incl
```

```
CFLAGS= -Wall -g -O3
```

pour séparer les flags de préprocesseur et les flags de compilateur.

LDLIBS les noms de bibliothèques, chaque nom précédé par -l

LDLIBS - les flags pour l'éditeur de liens, en particulier -L pour donner l'emplacement non-standard de bibliothèques