Makefile

Un fichier **Makefile**, est un fichier de spécifications qui décrit les dépendances des objets (commande **make** pour l'appeler).

Exemple d'un MakeFile :

Sens de lecture ↑

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

/* Fichier main.c */	/* Fichier hello.c */

#include "hello.h"
 	
int main(void){
  Hello(); return 0;
}
	#include <stdio.h>
	#include "hello.h"
	
	void Hello(void){
	  printf("Hello, world!\n");
	}

Un Makefile peut définir des macros sous la forme « macro = valeur », on peut alors lire ces macros avec la syntaxe $(macro).

Exemple :

```
OBJ      = main.o hello.o
CC       = gcc
CFLAGS= -DDEBUG -g
        main:  $(OBJ)
               $(CC) -o main $(OBJ)
        main.o: main.c hello.h
               $(CC) $(CFLAGS) -c main.c
        hello.o: hello.c hello.h
               $(CC) $(CFLAGS) -c hello.c
```

Macros spéciales :

$@	Nom de la cible
$<	Nom de la première dépendance
$^	Liste des dépendances
$?	Liste des dépendances plus récentes que la cible
$*	Nom du fichier sans suffixe

```
main.o: main.c hello.h                    main.o: main.c hello.h
    $(CC) $(CFLAGS) -c main.c      ⇨          $(CC) $(CFLAGS) -c $*.c
```

Makefile générique :

```
CC=gcc
CFLAGS=-Wall -std=gnu99 –g

SRC=$(wildcard *.c)
EXE=$(SRC:.c=)

all: $(EXE)

clean:
        rm -f $(EXE) *~
```

Permet de se servir de patterns (ici tous les .c)

```
CC=gcc
CFLAGS=-Wall -Wextra -std=gnu99 -g -I

all: main
main: main.o fonction.o
main.o: main.c fonction.o
fonction.o: fonction.c fonction.h

clean:
        rm -rf *.o
```

Dans l'ordre des dépendances (.h → .c →.o).