Outillage pour le C

Matthias BRUN

24 janvier 2023

Groupe ESEO – Option SE

Plan

La programmation C

La programmation C

Le langage C

- développé dans les années 70 ;
- Dennis Ritchie, laboratoire Bell
  (implantation système Unix) ;
- 1978, première description publique
  (Brian Kernighan et Dennis Ritchie - K&R-C) ;
- code source portable ;
- code machine performant ;
- compilateurs pour tous les systèmes courants ;
- norme ANSI X3.159 (C95, C99) ($\sim$ ISO/IEC 9899) ;
- ANSI C95 reconnue par tous les compilateurs
  (ANSI C99 $\pm$) ;

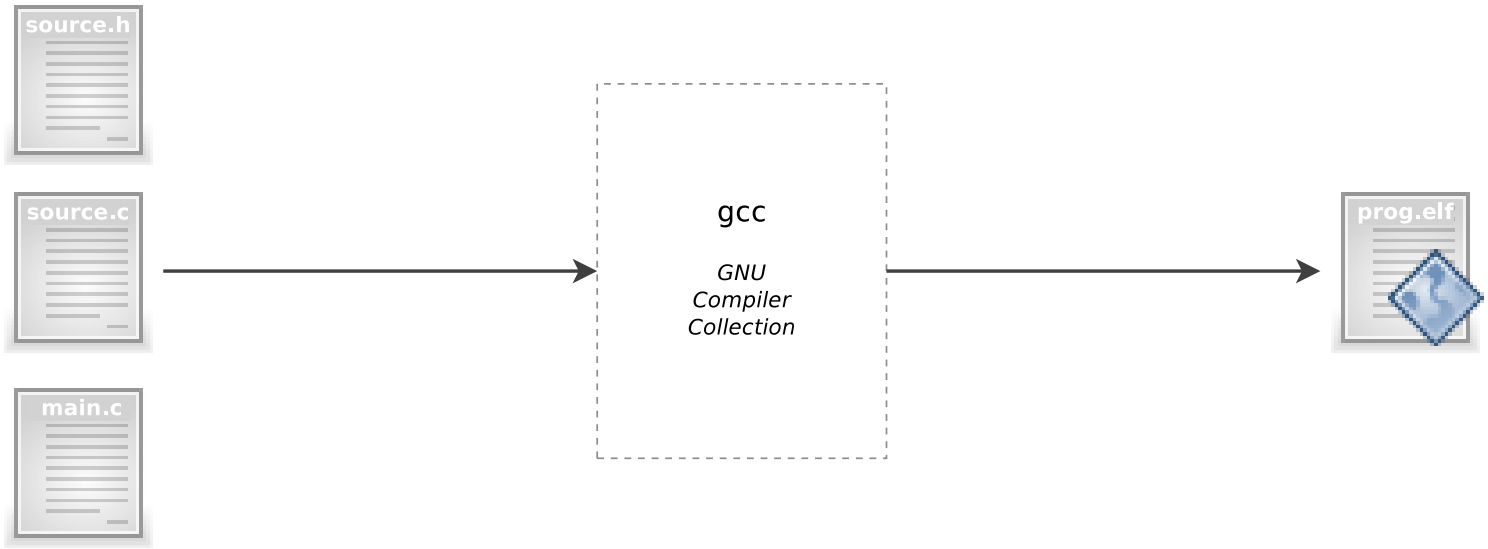
La programmation C


```
#ifndef _SOURCE_H
#define _SOURCE_H
/*
 * source.h
 *
 */

int a_function( int a_parameter, int * and_an_other );

int a_processing( int a_parameter );

int an_other_processing();


#endif /* _SOURCE_H */
```

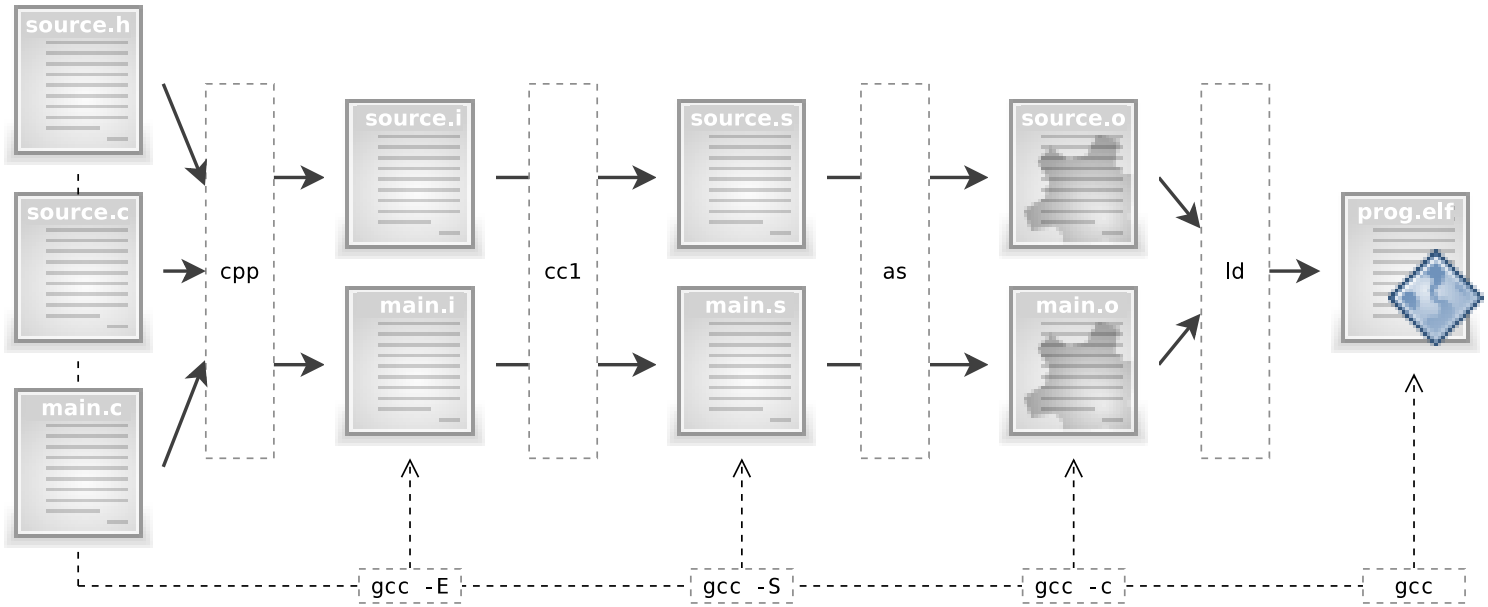
```
/*
 * source.c
 *
 */

#include "source.h"
#include <stdlib.h>

int a_function( int a_parameter, int * and_an_other )
{
      int a_result;

      a_result      = a_processing( a_parameter );
      *and_an_other = an_other_processing();

      return ( a_result ); /* and_an_other by pointer */
}

...

/* EOF */
```

```
/*
 * main.c
 *
 */

#include "source.h"

int a_global_value = 2;

int main ( int argc, char * argv[] )
{
     int a_local_value;
     int an_other_local_value;

     a_local_value = a_function( a_global_value, &an_other_local_value );

     return 0;
}

/* EOF */
```

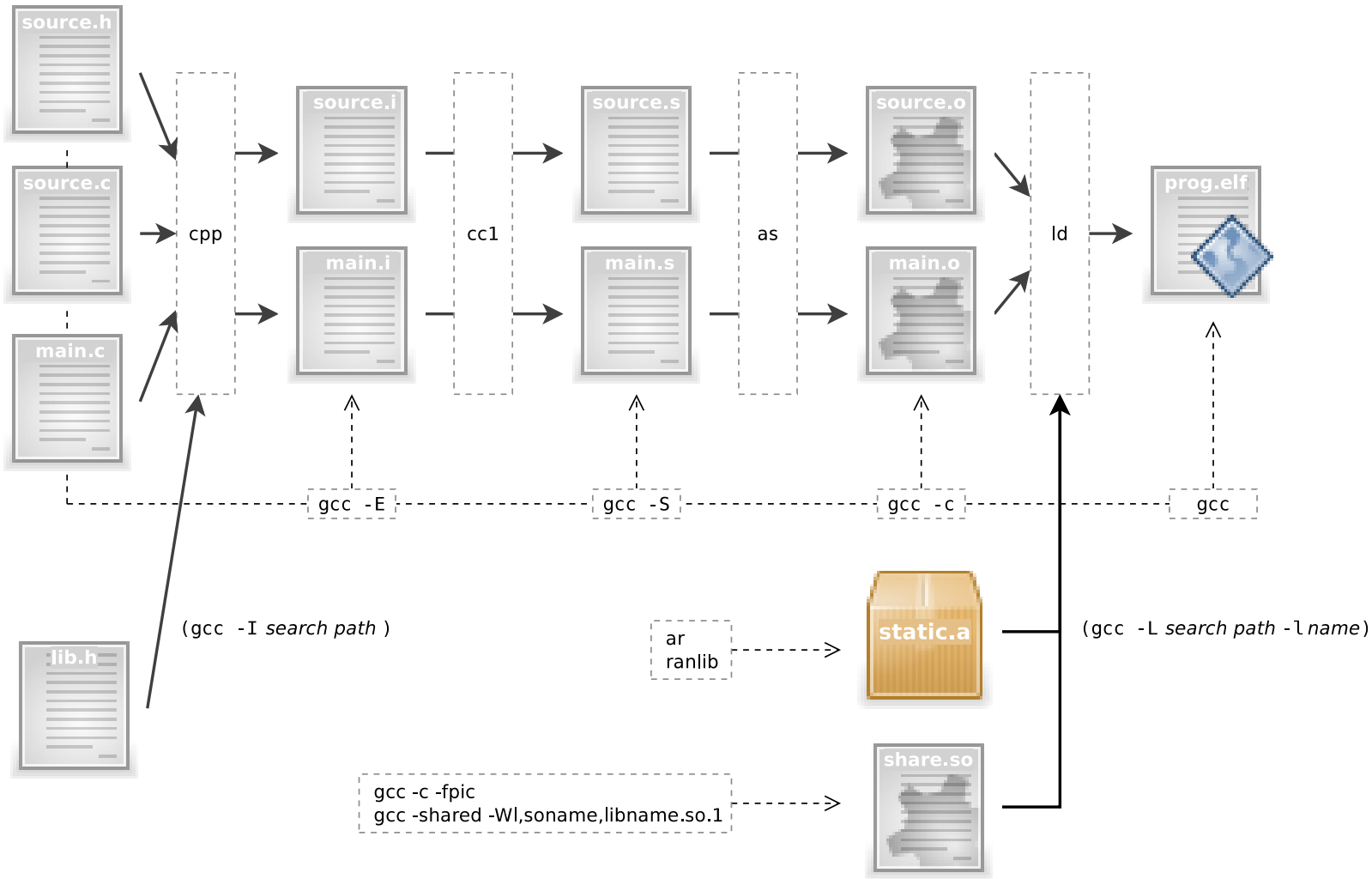
La compilation C



La compilation C



La compilation C



Outils de développement GNU

gcc	cpp, cc1, as, ld, *etc.*
gdb	*débugger un programme*
readelf	*informations sur les fichiers compilés au format* ELF readelf -x .text *source.o* (hexadécimal) readelf -h *source.o* (en-tête) readelf -S *source.o* (sections) readelf -s *source.o* (symboles)
objdump	*informations sur les fichiers objets* objdump -d *programme* (désassemble) objdump -d -j .text *source.o* (désassemble)
objcopy	*traduire un fichier objet* objcopy --output-target=binary *main.elf main.bin* ($\rightarrow$ binaire)
hexdump	*informations sur les fichiers objet*
nm	*lister les symboles d'un fichier objet*
strip	*supprimer les symboles d'un fichier objet*
size	*tailles des sections et taille totale d'un fichier objet*
strings	*caractères imprimables d'un fichier*
ldd	*dépendances de bibliothèques*

Plan

Makefile

- Principe : script de compilation (basé sur des règles)
- Syntaxe d'une règle :

```
target : dependencies
	commands # cette ligne commence par une tabulation !
```

- Évaluation récursive des règles :
  - $\forall$ dépendance, si dépendance = cible autre règle
    $\rightarrow$ évaluation autre règle.
  - dépendances analysées, si $\exists$ date dépendance $>$ date cible
    $\rightarrow$ exécution commandes.
- Lancement : `make [-f script] [target]`
- Scripts par défaut : `Makefile`, `makefile`
- Point d'entrée : première règle ou cible en paramètre

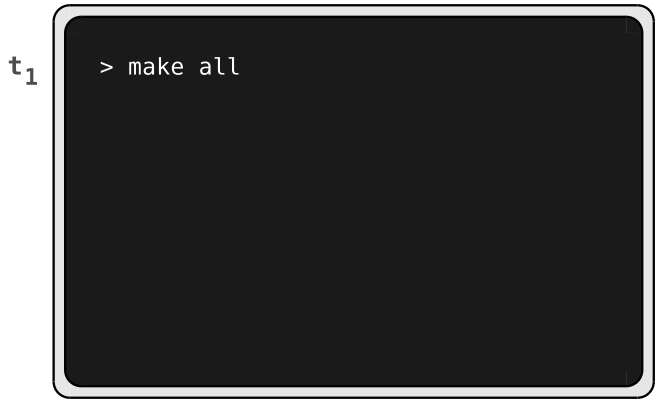
Makefile : exemple


```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple

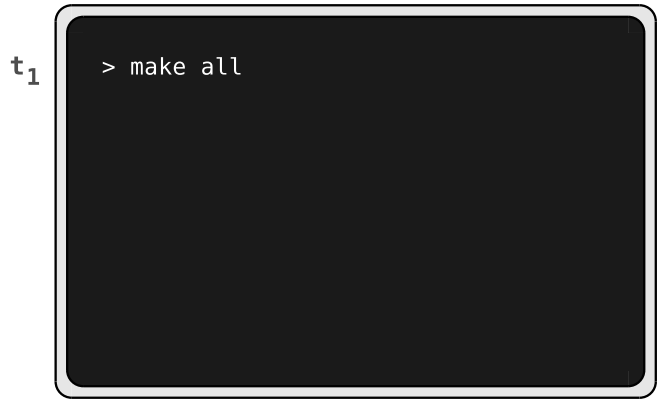

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple



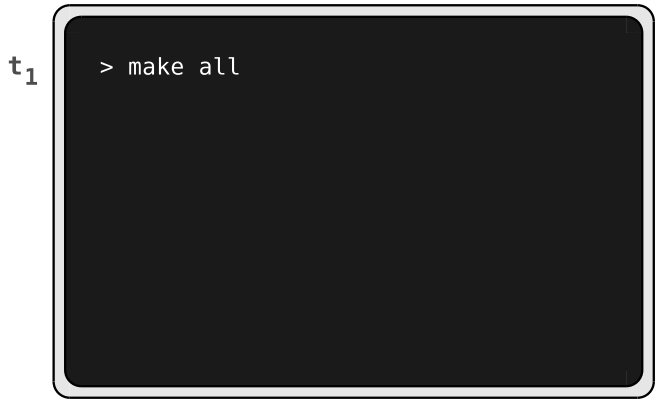

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple



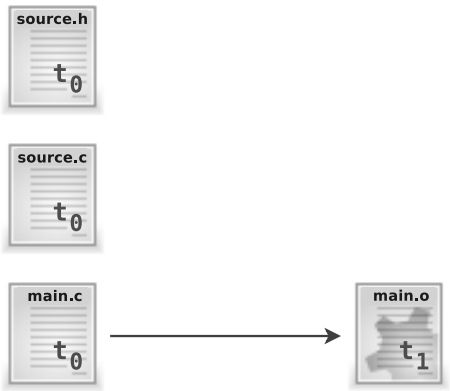

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple




```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple

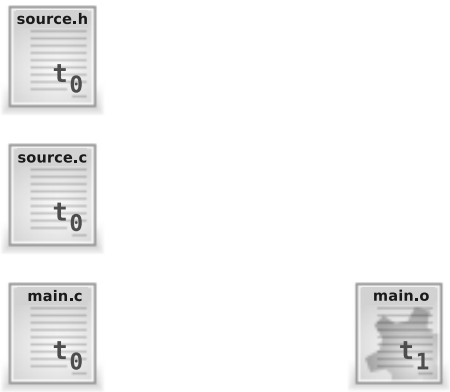


t_1

```
> make all
gcc -Wall -c main.c -o main.o
```

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
      gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
      gcc -Wall -c main.c -o main.o

source.o: source.c
      gcc -Wall -c source.c -o source.o

# Clean the project
clean:
      rm -f *.o prog.elf
```

Makefile : exemple

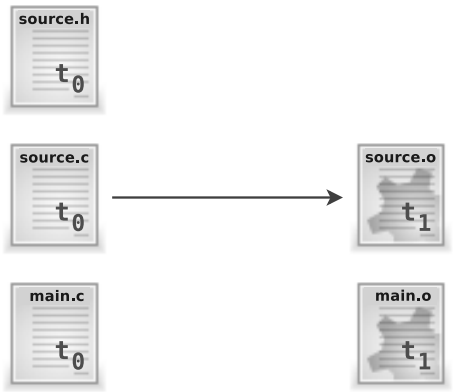


t_1

```
> make all
gcc -Wall -c main.c -o main.o
```

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple

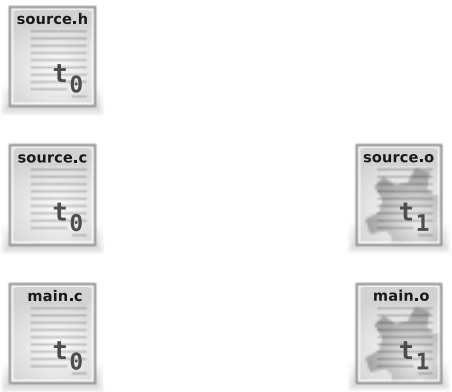


t_1

```
> make all
gcc -Wall -c main.c -o main.o
gcc -Wall -c source.c -o source.o
```

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple

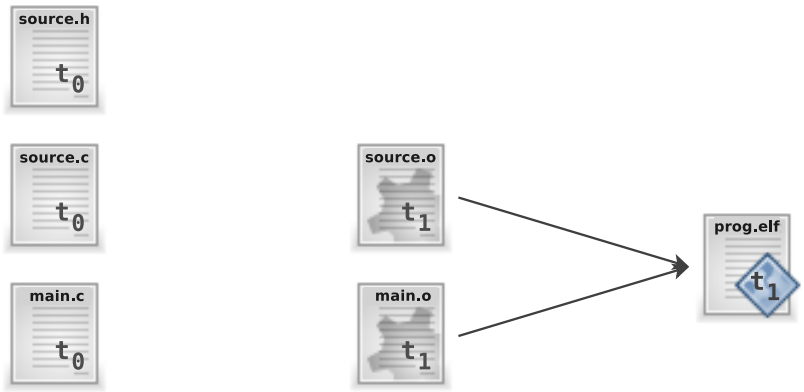


t_1

```
> make all
gcc -Wall -c main.c -o main.o
gcc -Wall -c source.c -o source.o
```

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple

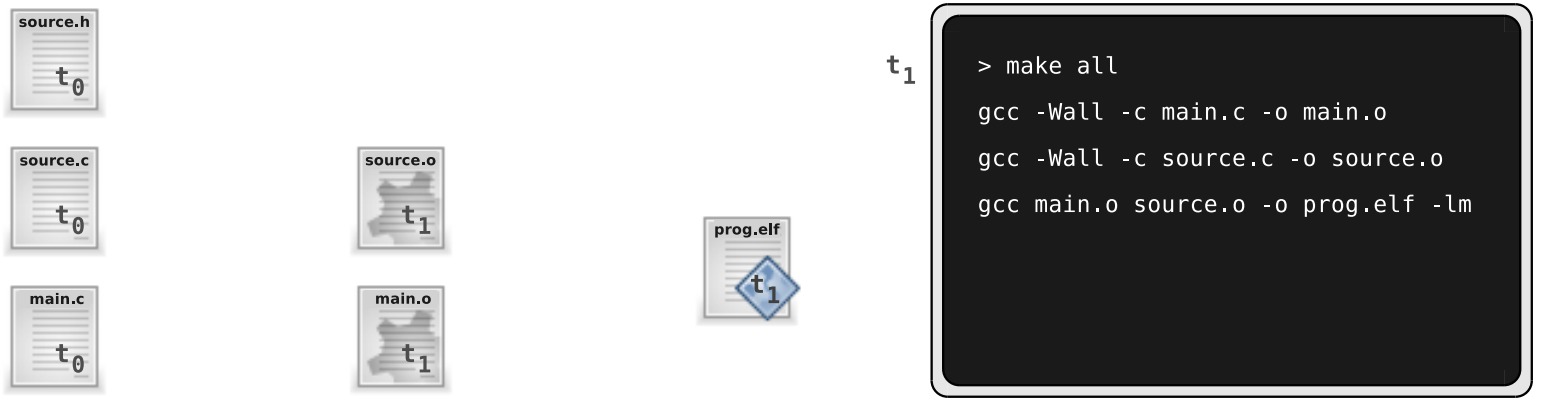


t_1

```
> make all
gcc -Wall -c main.c -o main.o
gcc -Wall -c source.c -o source.o
gcc main.o source.o -o prog.elf -lm
```

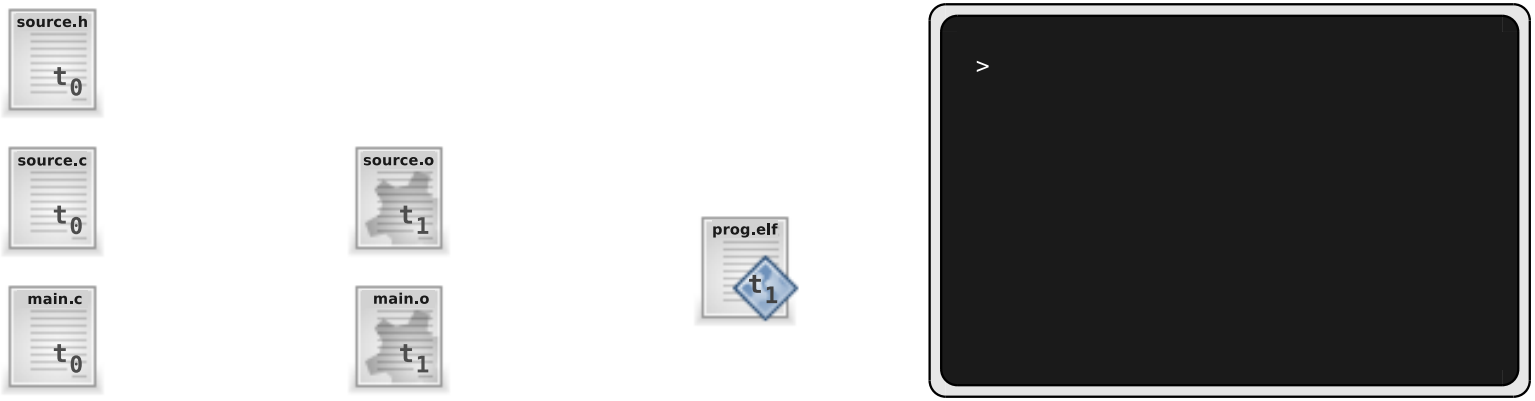
```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
      gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
      gcc -Wall -c main.c -o main.o

source.o: source.c
      gcc -Wall -c source.c -o source.o

# Clean the project
clean:
      rm -f *.o prog.elf
```

Makefile : exemple



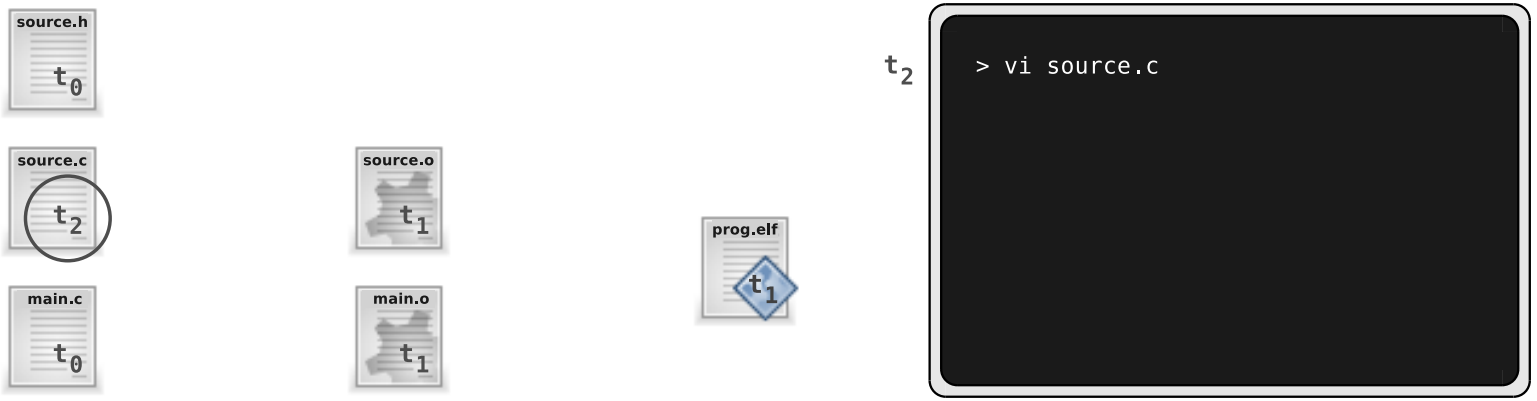

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple



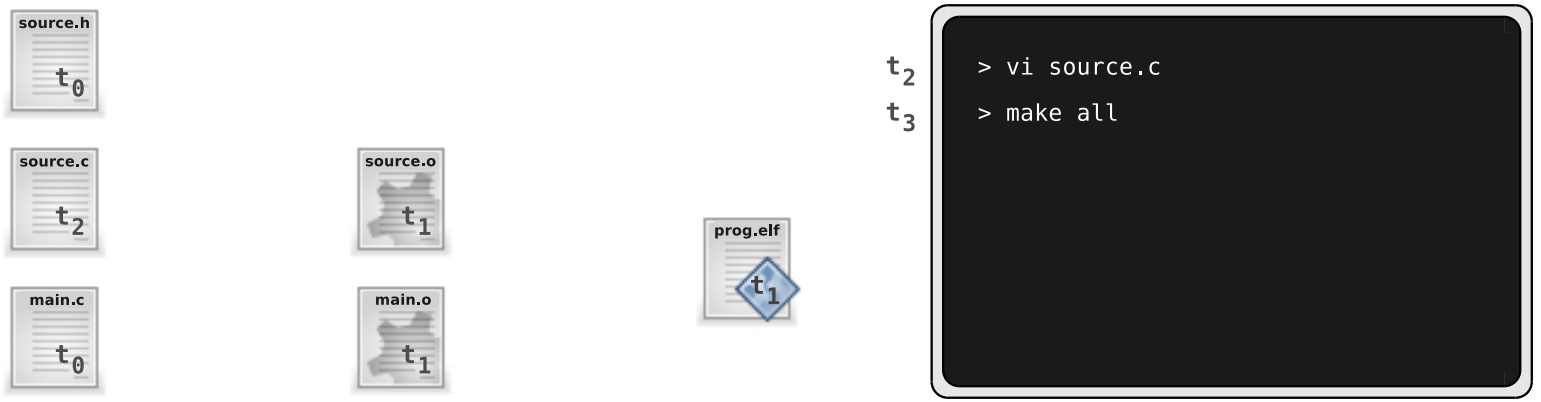

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple



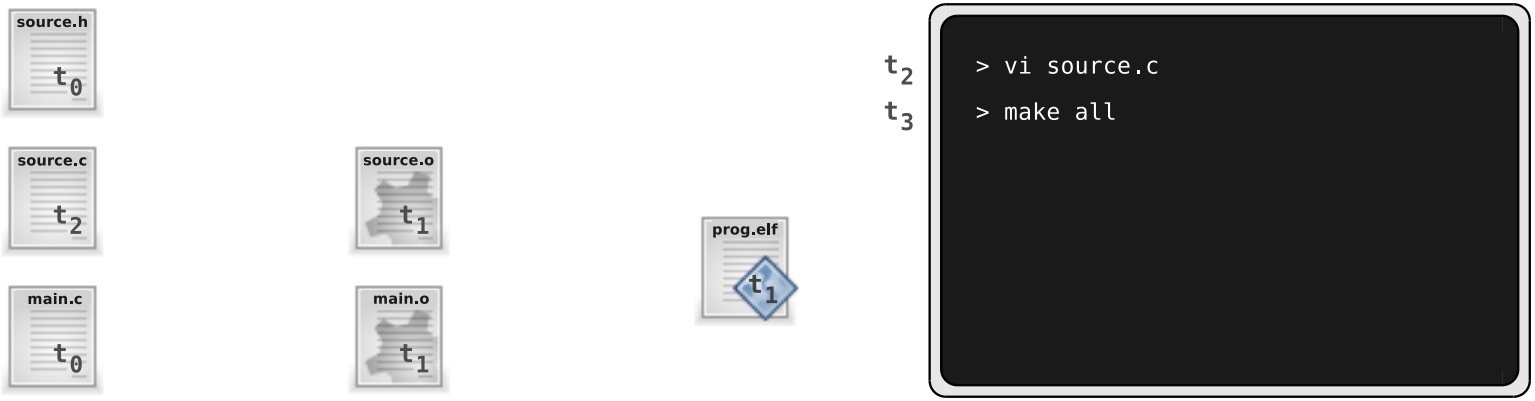

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple



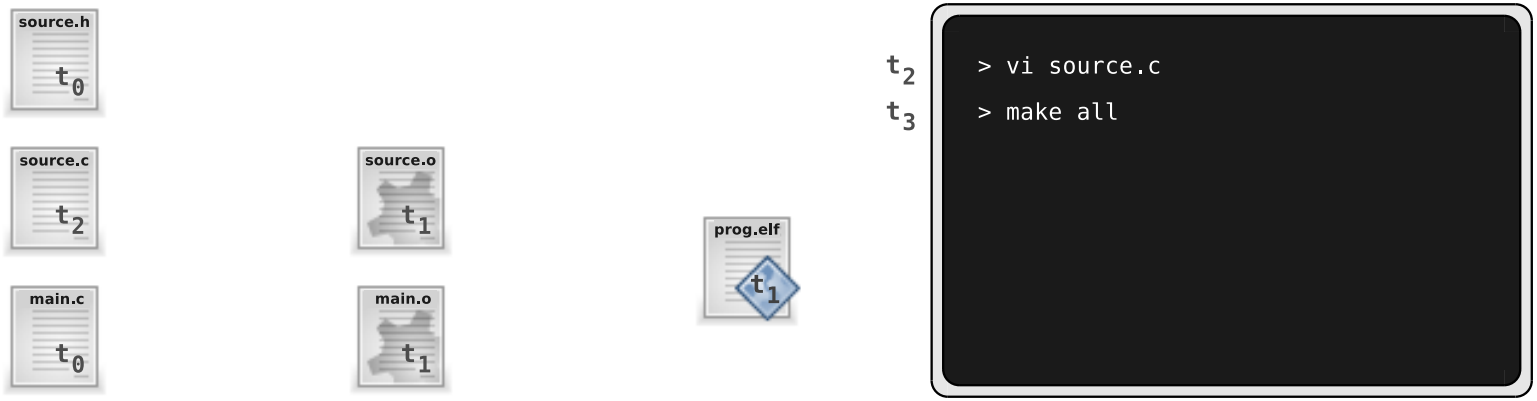

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple



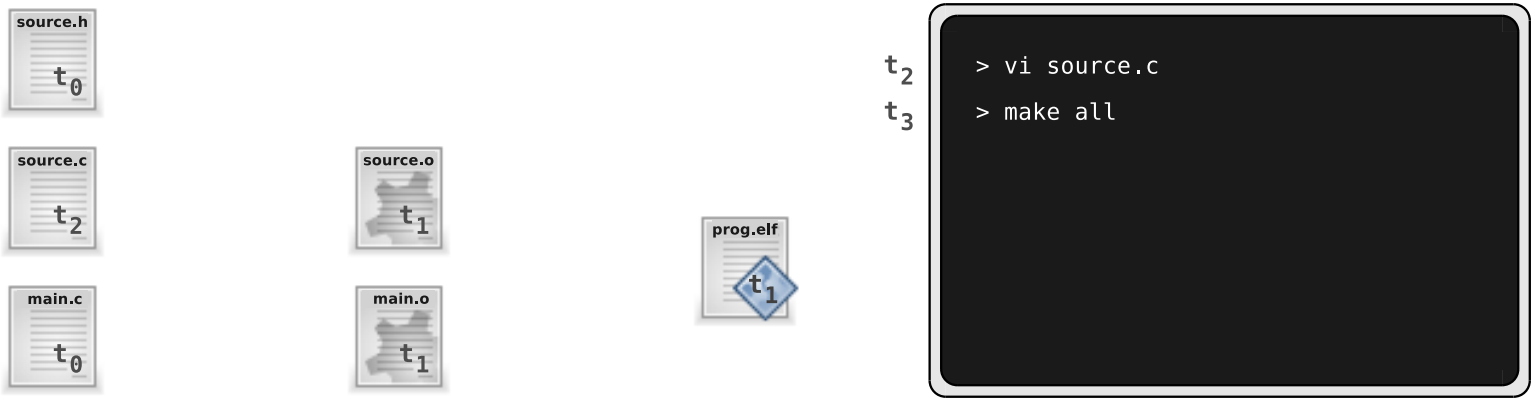

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple



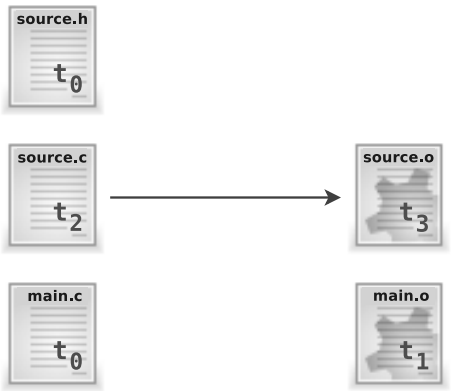

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

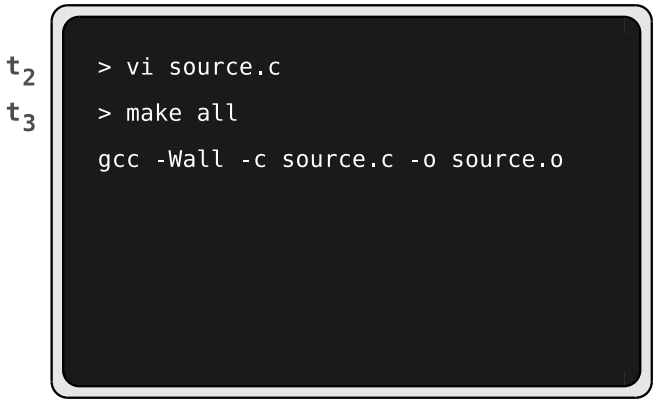
Makefile : exemple




```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

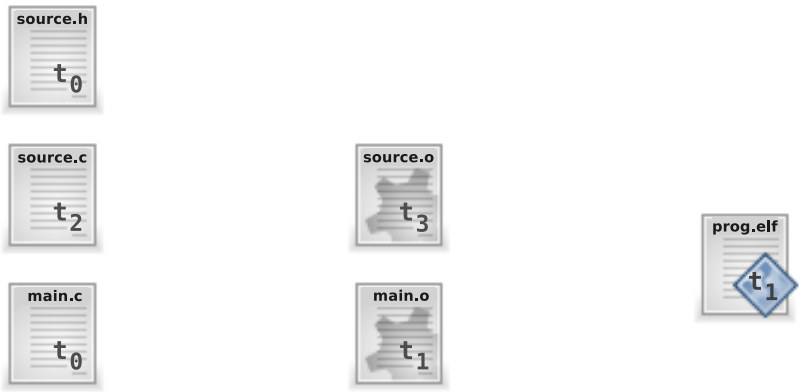
Makefile : exemple






```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
      gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
      gcc -Wall -c main.c -o main.o

source.o: source.c
      gcc -Wall -c source.c -o source.o

# Clean the project
clean:
      rm -f *.o prog.elf
```

Makefile : exemple

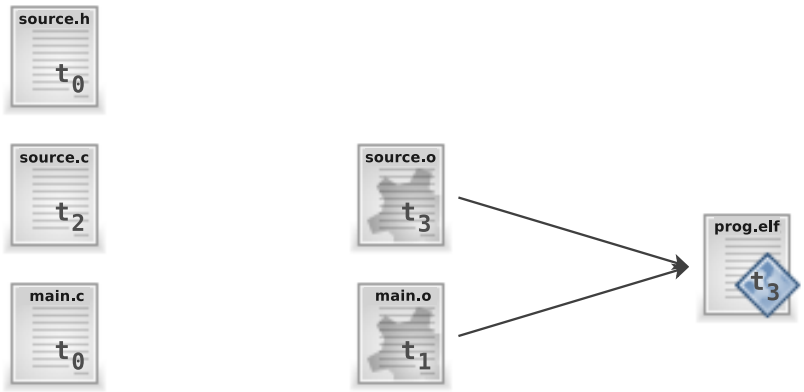


t_2
t_3

```
> vi source.c
> make all
gcc -Wall -c source.c -o source.o
```

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple

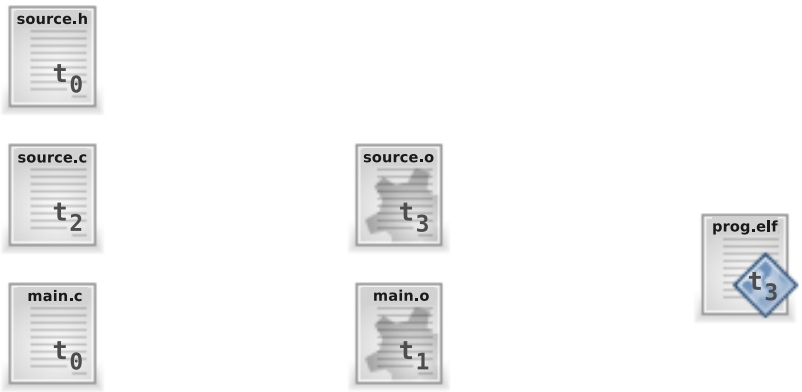


t_2
t_3

```
> vi source.c
> make all
gcc -Wall -c source.c -o source.o
gcc main.o source.o -o prog.elf -lm
```

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
      gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
      gcc -Wall -c main.c -o main.o

source.o: source.c
      gcc -Wall -c source.c -o source.o

# Clean the project
clean:
      rm -f *.o prog.elf
```

Makefile : exemple



t_2
t_3

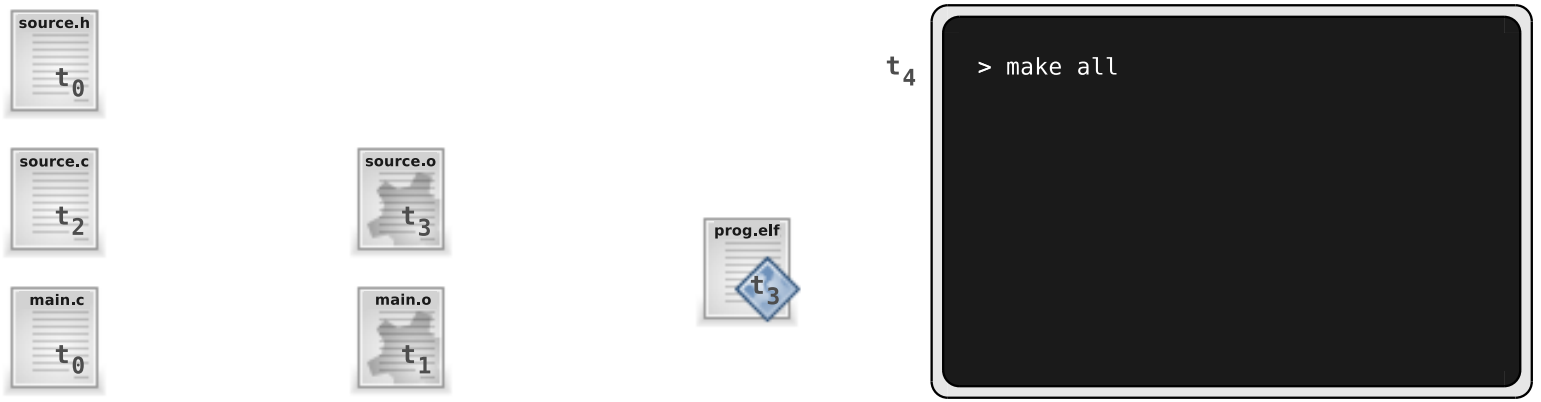
```
> vi source.c
> make all
gcc -Wall -c source.c -o source.o
gcc main.o source.o -o prog.elf -lm
```

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple

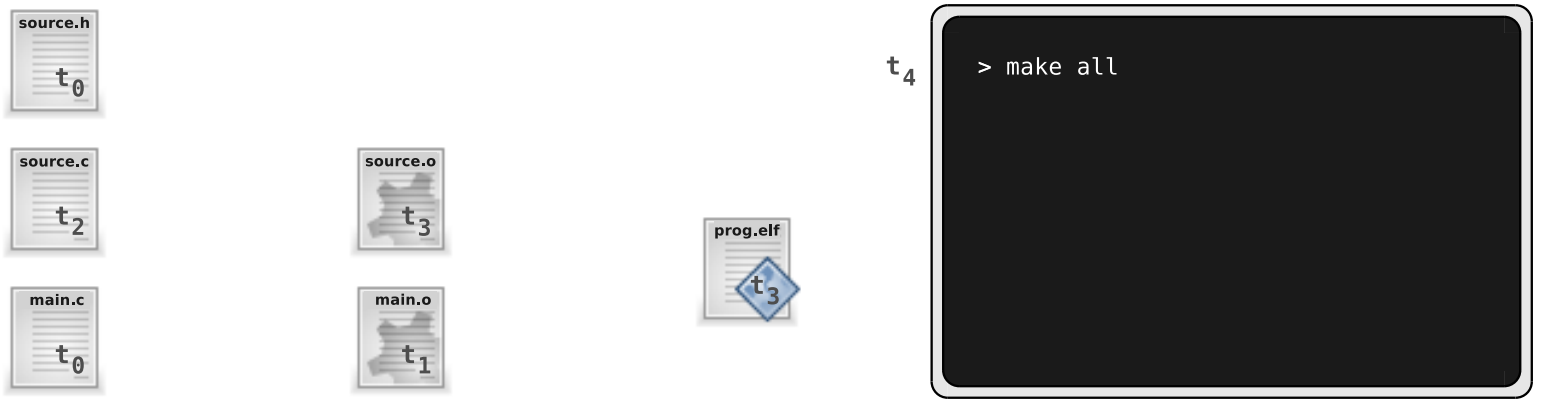

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple



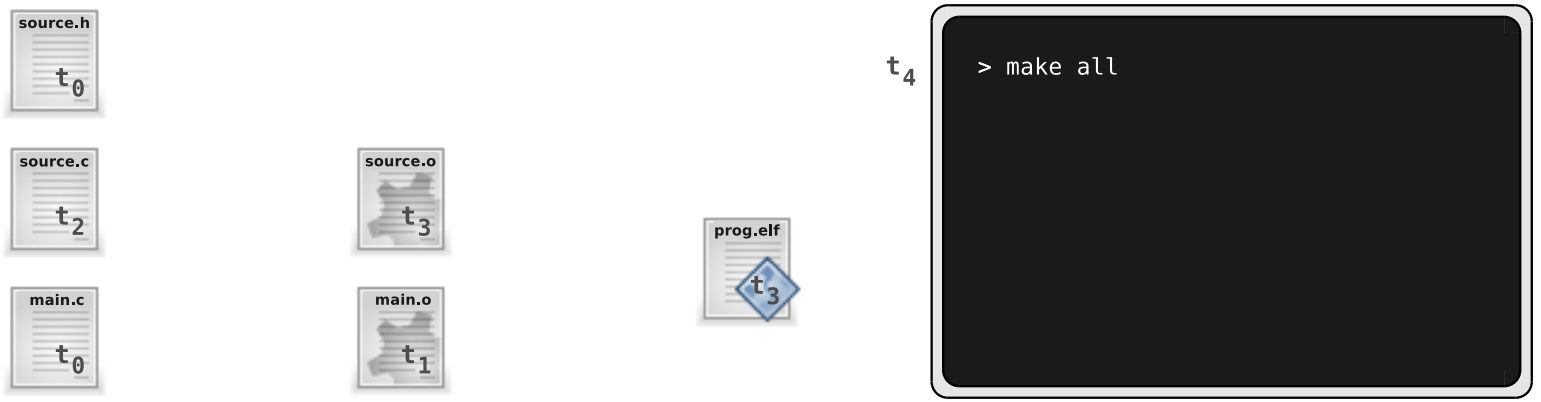

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple



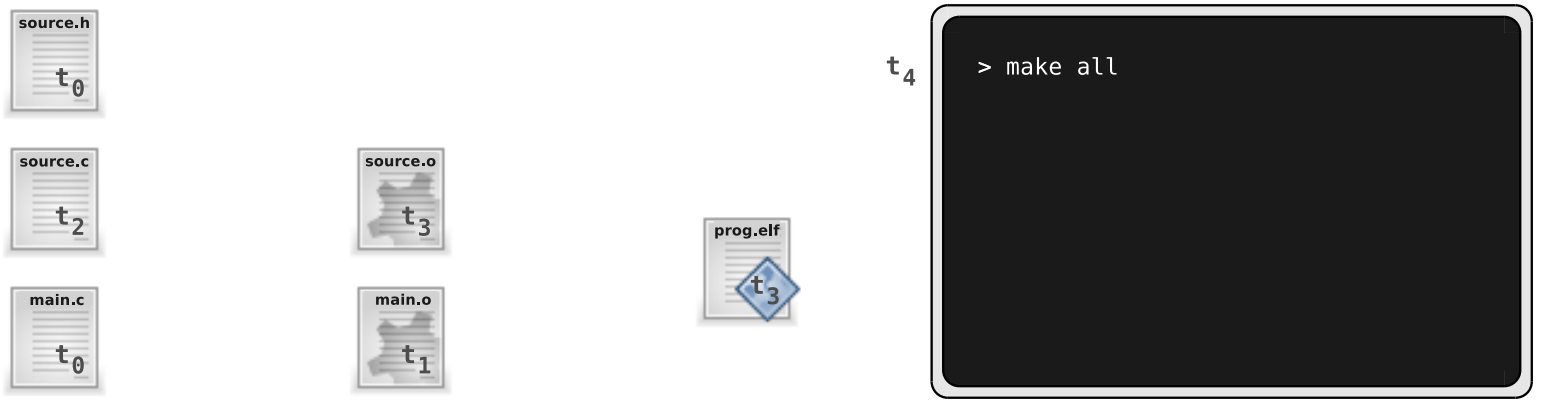

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple



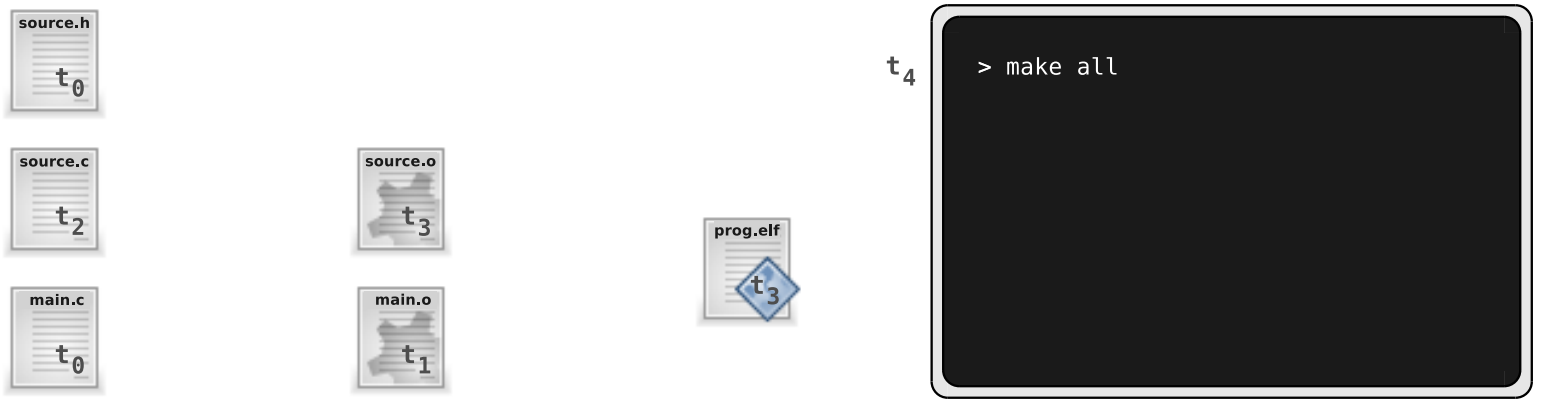

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple



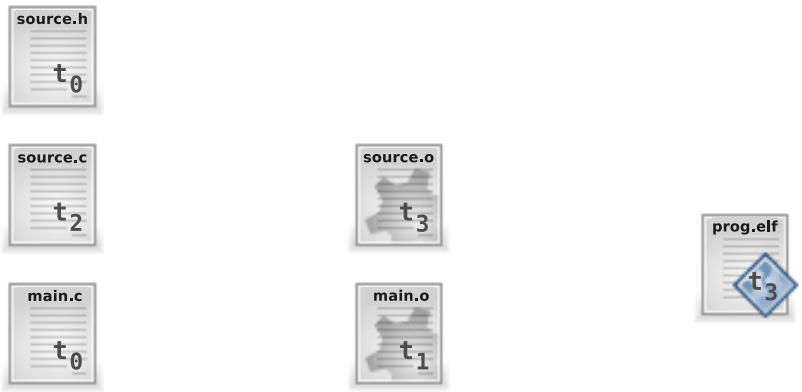

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
      gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
      gcc -Wall -c main.c -o main.o

source.o: source.c
      gcc -Wall -c source.c -o source.o

# Clean the project
clean:
      rm -f *.o prog.elf
```

Makefile : exemple




```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple



t_4

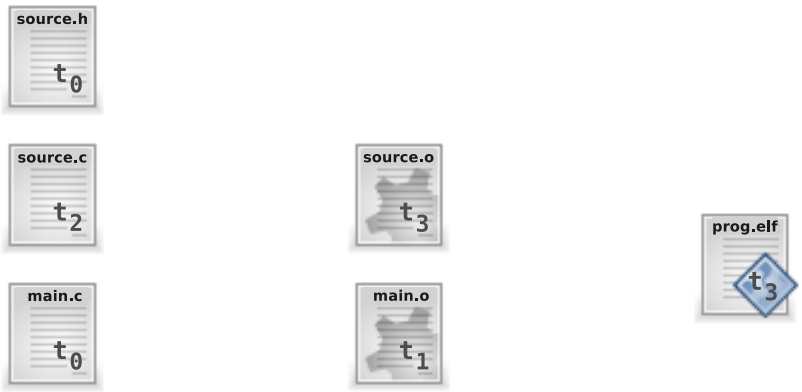
```
> make all
make: « prog.elf » est à jour.
```

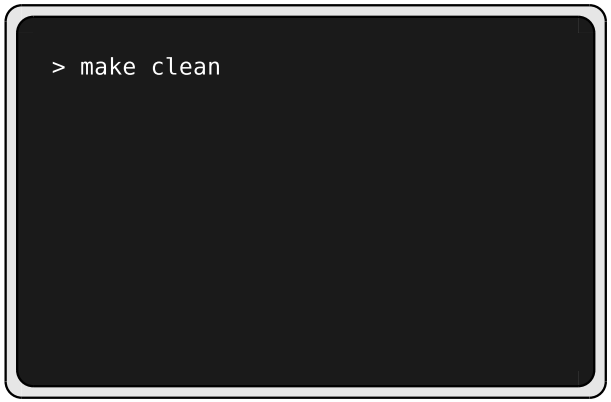
```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple


```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple






```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple


```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple


```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple (utilisation de variables)

```
# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple (utilisation de variables)

```
EXEC = prog.elf

CC = gcc
LD = gcc

CCFLAGS = -Wall
LDFLAGS = -lm

RM = rm -f



# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple (utilisation de variables)

```
EXEC = prog.elf

CC = gcc
LD = gcc

CCFLAGS = -Wall
LDFLAGS = -lm

RM = rm -f



# Compilation rules
all: prog.elf

prog.elf: main.o source.o
	gcc main.o source.o -o prog.elf -lm

main.o: main.c source.h
	gcc -Wall -c main.c -o main.o

source.o: source.c
	gcc -Wall -c source.c -o source.o

# Clean the project
clean:
	rm -f *.o prog.elf
```

Makefile : exemple (utilisation de variables)

```
EXEC = prog.elf

CC = gcc
LD = gcc

CCFLAGS = -Wall
LDFLAGS = -lm

RM = rm -f



# Compilation rules
all: $(EXEC)

$(EXEC): main.o source.o
	$(LD) main.o source.o -o $(EXEC) $(LDFLAGS)

main.o: main.c source.h
	$(CC) $(CCFLAGS) -c main.c -o main.o

source.o: source.c
	$(CC) $(CCFLAGS) -c source.c -o source.o

# Clean the project
clean:
	$(RM) *.o $(EXEC)
```

Makefile : exemple (utilisation de variables)

```
EXEC = prog.elf

CC = gcc
LD = gcc

CCFLAGS = -Wall
LDFLAGS = -lm

RM = rm -f



# Compilation rules
all: $(EXEC)

$(EXEC): main.o source.o
	$(LD) main.o source.o -o $(EXEC) $(LDFLAGS)

main.o: main.c source.h
	$(CC) $(CCFLAGS) -c main.c -o main.o

source.o: source.c
	$(CC) $(CCFLAGS) -c source.c -o source.o

# Clean the project
clean:
	$(RM) *.o $(EXEC)
```

Makefile : exemple (utilisation de variables)

```
EXEC = prog.elf

CC = gcc
LD = gcc

CCFLAGS = -Wall
LDFLAGS = -lm

RM = rm -f



# Compilation rules
all: $(EXEC)

$(EXEC): main.o source.o
	$(LD) main.o source.o -o $(EXEC) $(LDFLAGS)

main.o: main.c source.h
	$(CC) $(CCFLAGS) -c main.c -o main.o

source.o: source.c
	$(CC) $(CCFLAGS) -c source.c -o source.o

# Clean the project
clean:
	$(RM) *.o $(EXEC)
```

Makefile : exemple (utilisation de variables)

```
EXEC = prog.elf

CC = gcc
LD = gcc

CCFLAGS = -Wall
LDFLAGS = -lm

RM = rm -f



# Compilation rules
all: $(EXEC)

$(EXEC): main.o source.o
	$(LD) $^ -o $@ $(LDFLAGS)

main.o: main.c source.h
	$(CC) $(CCFLAGS) -c $< -o $@

source.o: source.c
	$(CC) $(CCFLAGS) -c $< -o $@

# Clean the project
clean:
	$(RM) *.o $(EXEC)
```

Makefile : exemple (utilisation de variables)

```
EXEC = prog.elf

CC = gcc
LD = gcc

CCFLAGS = -Wall
LDFLAGS = -lm

RM = rm -f



# Compilation rules
all: $(EXEC)

$(EXEC): main.o source.o
	$(LD) $^ -o $@ $(LDFLAGS)

main.o: main.c source.h
	$(CC) $(CCFLAGS) -c $< -o $@

source.o: source.c
	$(CC) $(CCFLAGS) -c $< -o $@

# Clean the project
clean:
	$(RM) *.o $(EXEC)
```

Makefile : exemple (utilisation de variables, règles d'inférence)

```
EXEC = prog.elf

CC = gcc
LD = gcc

CCFLAGS = -Wall
LDFLAGS = -lm

RM = rm -f



# Compilation rules
all: $(EXEC)

$(EXEC): main.o source.o
	$(LD) $^ -o $@ $(LDFLAGS)

main.o: main.c source.h
	$(CC) $(CCFLAGS) -c $< -o $@

source.o: source.c
	$(CC) $(CCFLAGS) -c $< -o $@

# Clean the project
clean:
	$(RM) *.o $(EXEC)
```

Makefile : exemple (utilisation de variables, règles d'inférence)

```
EXEC = prog.elf

CC = gcc
LD = gcc

CCFLAGS = -Wall
LDFLAGS = -lm

RM = rm -f



# Compilation rules
all: $(EXEC)

$(EXEC): main.o source.o
	$(LD) $^ -o $@ $(LDFLAGS)

main.o: source.h


%.o: %.c
	$(CC) $(CCFLAGS) -c $< -o $@

# Clean the project
clean:
	$(RM) *.o $(EXEC)
```

Makefile : exemple (utilisation de variables, règles d'inférence)

```
EXEC = prog.elf

CC = gcc
LD = gcc

CCFLAGS = -Wall
LDFLAGS = -lm

RM = rm -f



# Compilation rules
all: $(EXEC)

$(EXEC): main.o source.o
	$(LD) $^ -o $@ $(LDFLAGS)

main.o: source.h


%.o: %.c
	$(CC) $(CCFLAGS) -c $< -o $@

# Clean the project
clean:
	$(RM) *.o $(EXEC)
```

Makefile : exemple (utilisation de variables, règles d'inférence)

```
EXEC = prog.elf

CC = gcc
LD = gcc

CCFLAGS = -Wall
LDFLAGS = -lm

RM = rm -f



# Compilation rules
all: $(EXEC)

$(EXEC): main.o source.o
	$(LD) $^ -o $@ $(LDFLAGS)

main.o: source.h

%.o: %.c
	$(CC) $(CCFLAGS) -c $< -o $@

# Clean the project
clean:
	$(RM) *.o $(EXEC)
```

Makefile : exemple (utilisation de variables, règles d'inférence, wildcard)

```
EXEC = prog.elf

CC = gcc
LD = gcc

CCFLAGS = -Wall
LDFLAGS = -lm

RM = rm -f

SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)

# Compilation rules
all: $(EXEC)

$(EXEC): main.o source.o
	$(LD) $^ -o $@ $(LDFLAGS)

main.o: source.h

%.o: %.c
	$(CC) $(CCFLAGS) -c $< -o $@

# Clean the project
clean:
	$(RM) *.o $(EXEC)
```

Makefile : exemple (utilisation de variables, règles d'inférence, wildcard)

```
EXEC = prog.elf

CC = gcc
LD = gcc

CCFLAGS = -Wall
LDFLAGS = -lm

RM = rm -f

SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)

# Compilation rules
all: $(EXEC)

$(EXEC): main.o source.o
	$(LD) $^ -o $@ $(LDFLAGS)

main.o: source.h

%.o: %.c
	$(CC) $(CCFLAGS) -c $< -o $@

# Clean the project
clean:
	$(RM) *.o $(EXEC)
```

Makefile : exemple (utilisation de variables, règles d'inférence, wildcard)

```
EXEC = prog.elf

CC = gcc
LD = gcc

CCFLAGS = -Wall
LDFLAGS = -lm

RM = rm -f

SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)

# Compilation rules
all: $(EXEC)

$(EXEC): $(OBJ)
	$(LD) $^ -o $@ $(LDFLAGS)

main.o: source.h

%.o: %.c
	$(CC) $(CCFLAGS) -c $< -o $@

# Clean the project
clean:
	$(RM) *.o $(EXEC)
```

Makefile : exemple (utilisation de variables, règles d'inférence, wildcard)

```
EXEC = prog.elf

CC = gcc
LD = gcc

CCFLAGS = -Wall
LDFLAGS = -lm

RM = rm -f

SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)

# Compilation rules
all: $(EXEC)

$(EXEC): $(OBJ)
	$(LD) $^ -o $@ $(LDFLAGS)

main.o: source.h

%.o: %.c
	$(CC) $(CCFLAGS) -c $< -o $@

# Clean the project
clean:
	$(RM) *.o $(EXEC)
```

Makefile

- Concaténation de variables : `+=`
- Passage de variables en ligne de commande :

  exemple : `make TARGET=robot`
- Structures conditionnelles : `ifdef`, `ifeq`, `else`, `endif`

```
ifeq ($(TARGET), robot)
      # Compilation pour la cible
      $(CC) = arm-elf-gcc
else
      #Compilation pour le pc de dev
      $(CC) = gcc
endif
```

- Exécution systématique d'une commande : `.PHONY`

```
.PHONY: clean

clean:
      rm -f *.o programme
```

Makefile

- Visibilité d'une variable dans un autre makefile : `export`

```
export $(CC) = gcc

cible_x:
	# => CC = gcc pour make lors de
	# l'execution de Makefile.cible_x
	$(MAKE) -f Makefile.cible_x
```

- Makefiles récursifs

```
SUBDIRS = src test

.PHONY: all clean $(SUBDIRS)

all: $(SUBDIRS)

clean: $(SUBDIRS)

$(SUBDIRS):
	$(MAKE) $(MAKECMDGOALS) -C $@
```

- Empêcher affichage commande *stdout* : `@`*commande*
- Inclusion de fichier : `include`

Makefile

- Gcc pour gestion des dépendances, options `gcc` :
  - `-M` : générer règles de dépendances (remplacement sortie `cpp`).
  - `-MM` : idem sans considération des fichiers d'en-tête systèmes.
  - `-MF` : préciser un fichier de sortie.
  - `-MD` (avec `-o`) : `-M -MF $*.d` (avec concervation sortie `cpp`).
  - `-MMD` : idem sans considération des fichiers d'en-tête systèmes.
  - `-MP` : éviter message d'erreur après suppr. fichiers d'en-tête.

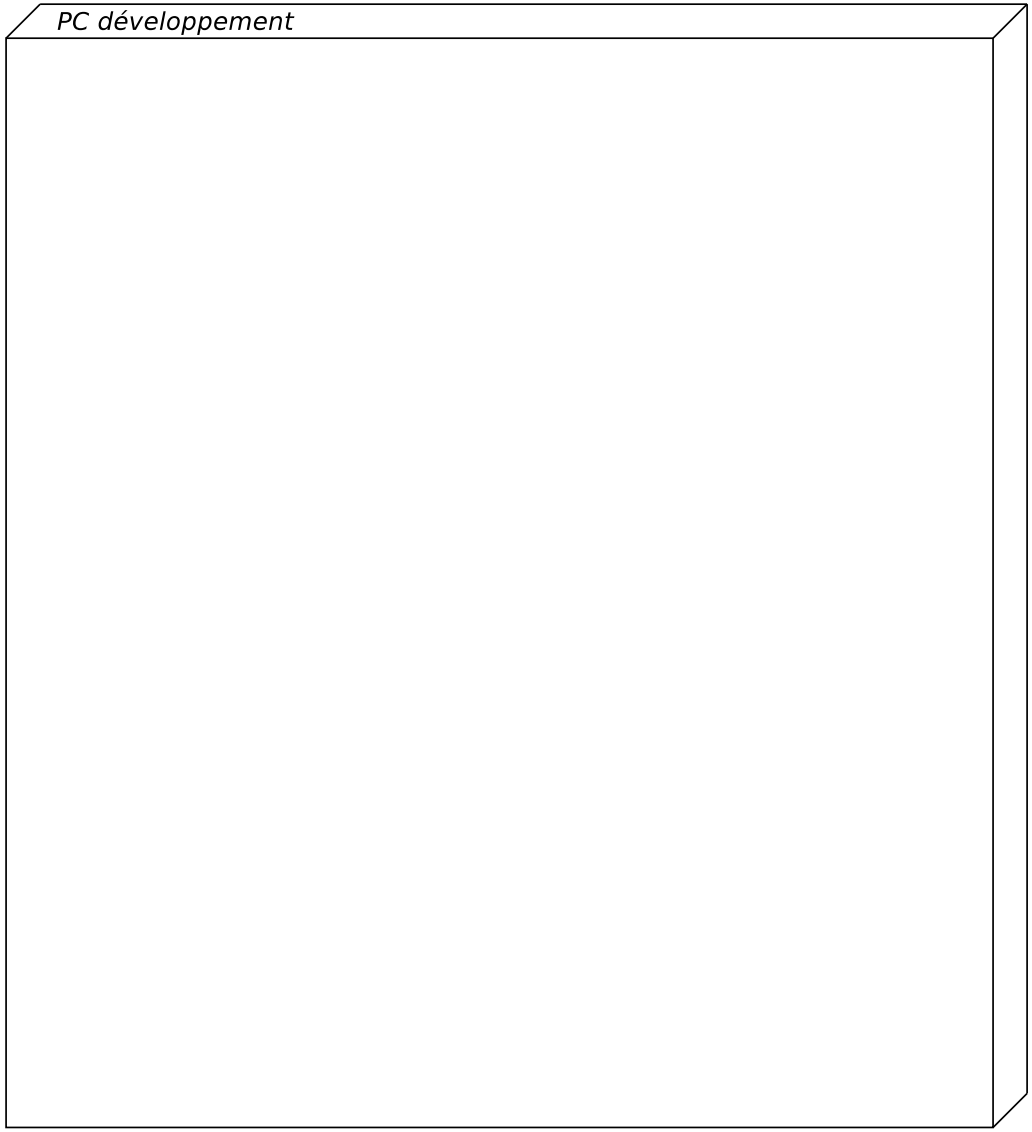
```
# Exemple de gestion automatique des dépendances
# à partir des instructions include de cpp.

SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)
DEP = $(SRC:.c=.d)

-include $(DEP)

.c.o:
	gcc -c -MMD -MP $< -o $@
```

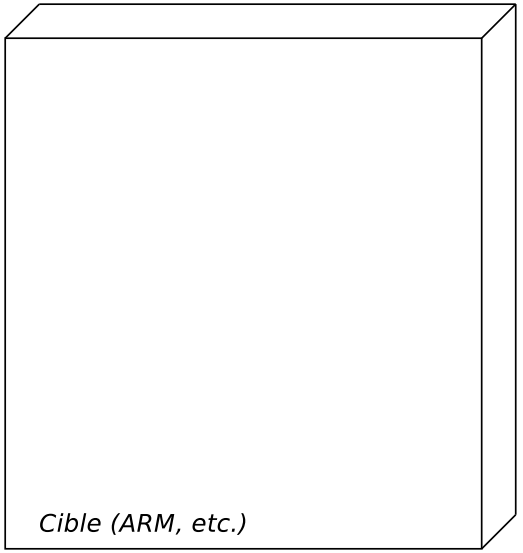
Makefile

- Plus d'informations : `man make`, `info make`

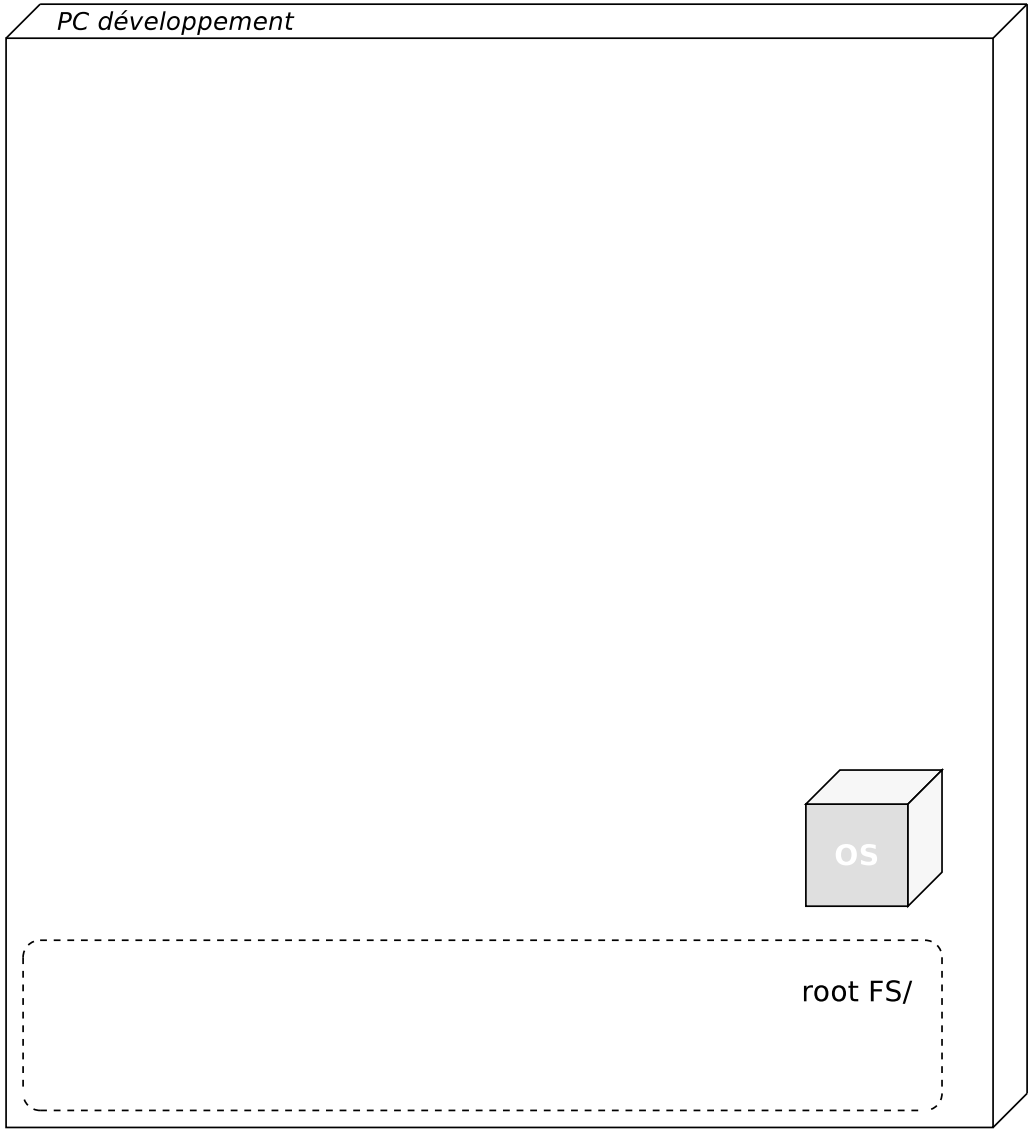
Plan

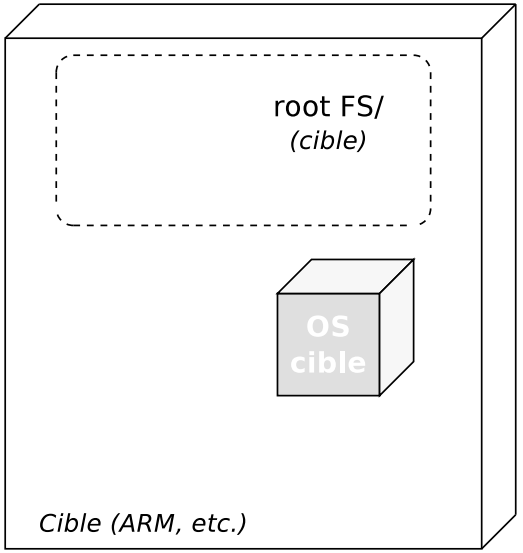
Développement croisé



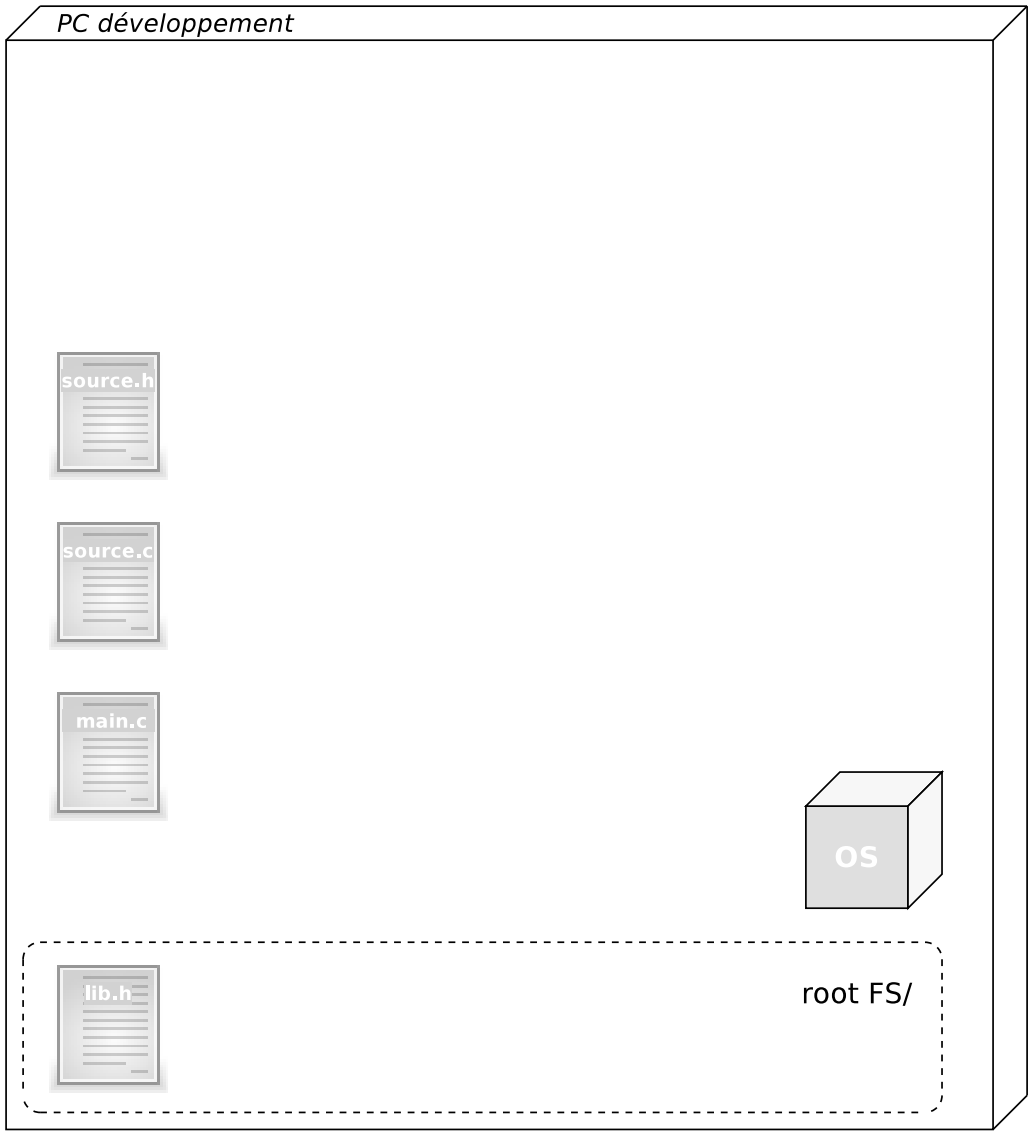


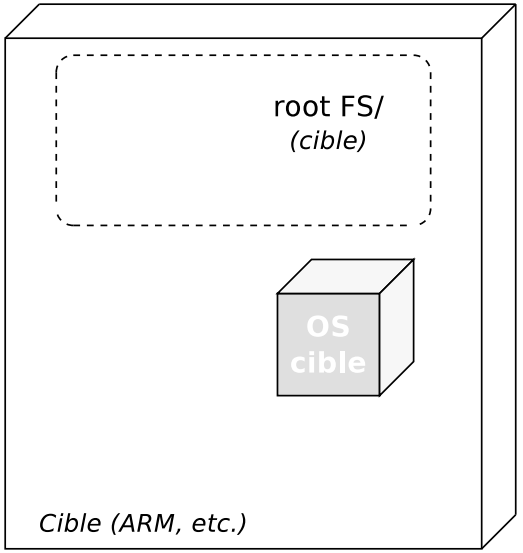
Développement croisé



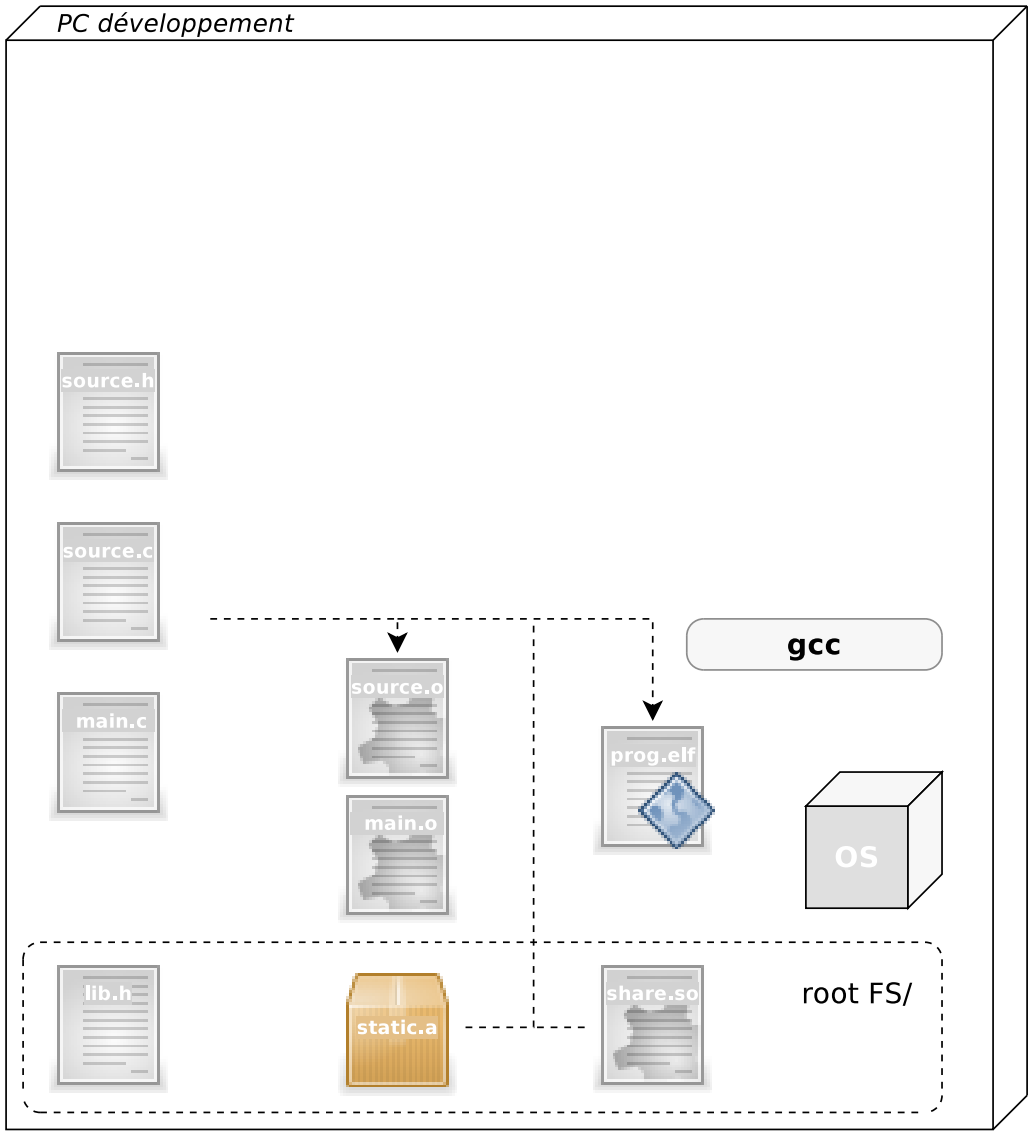


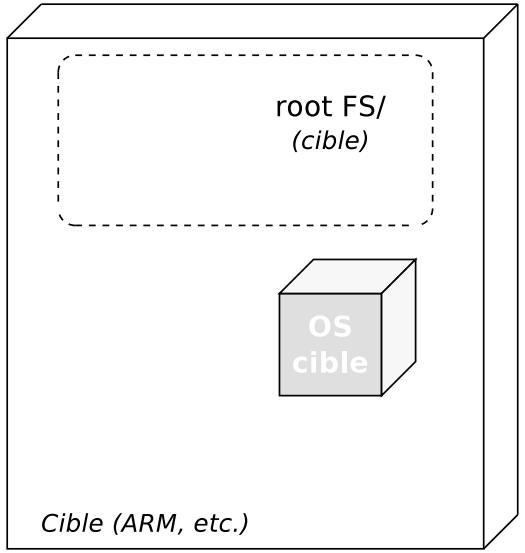
Développement croisé



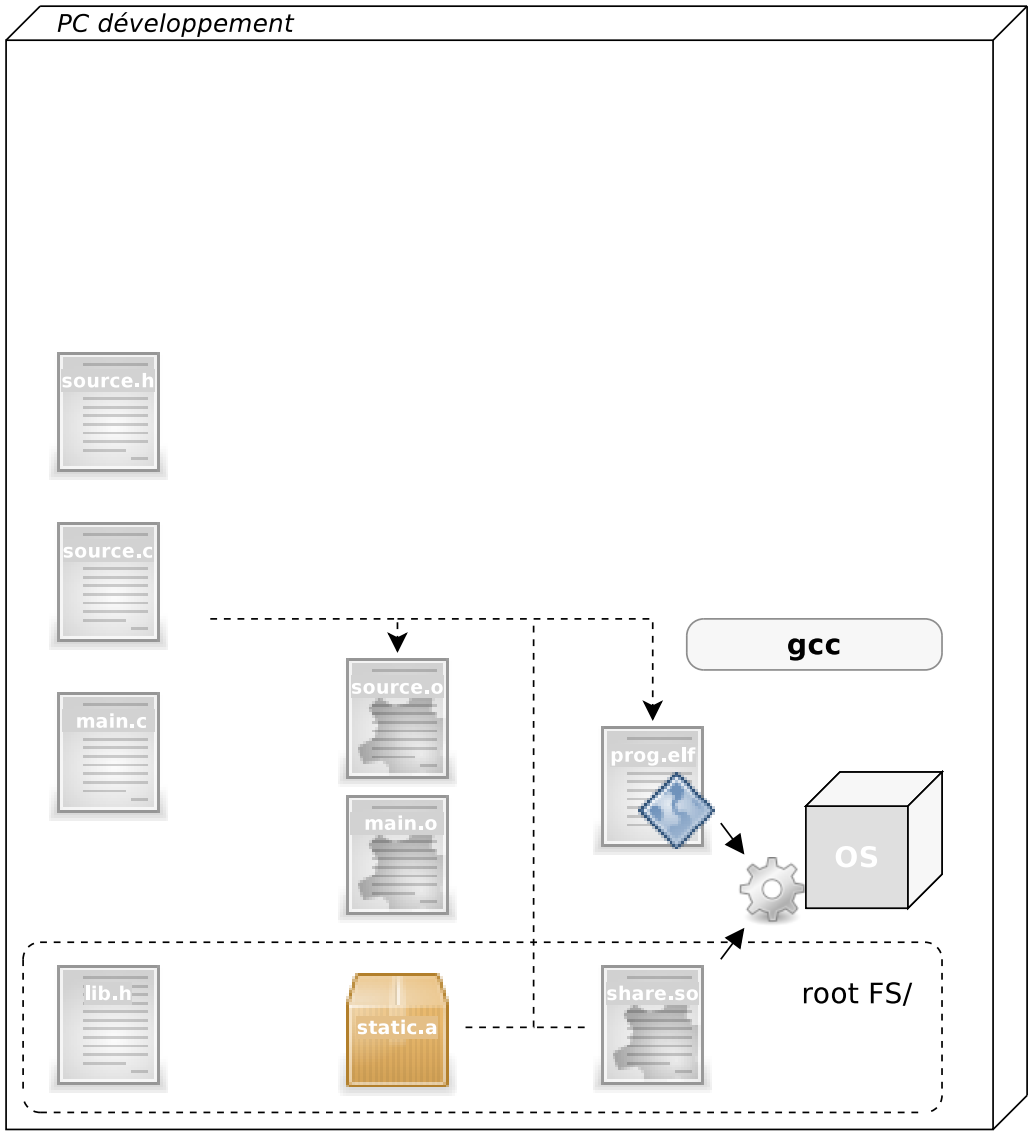


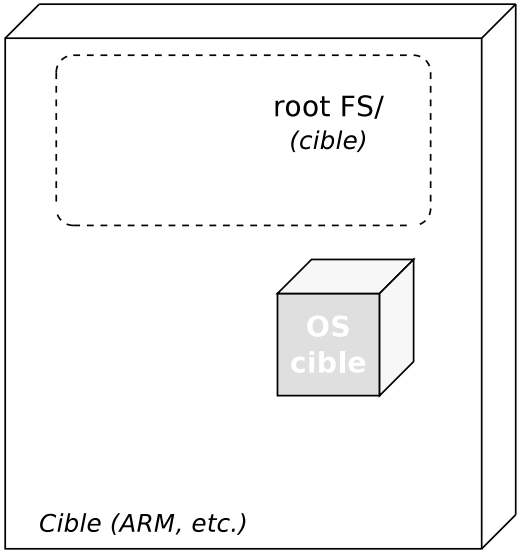
Développement croisé



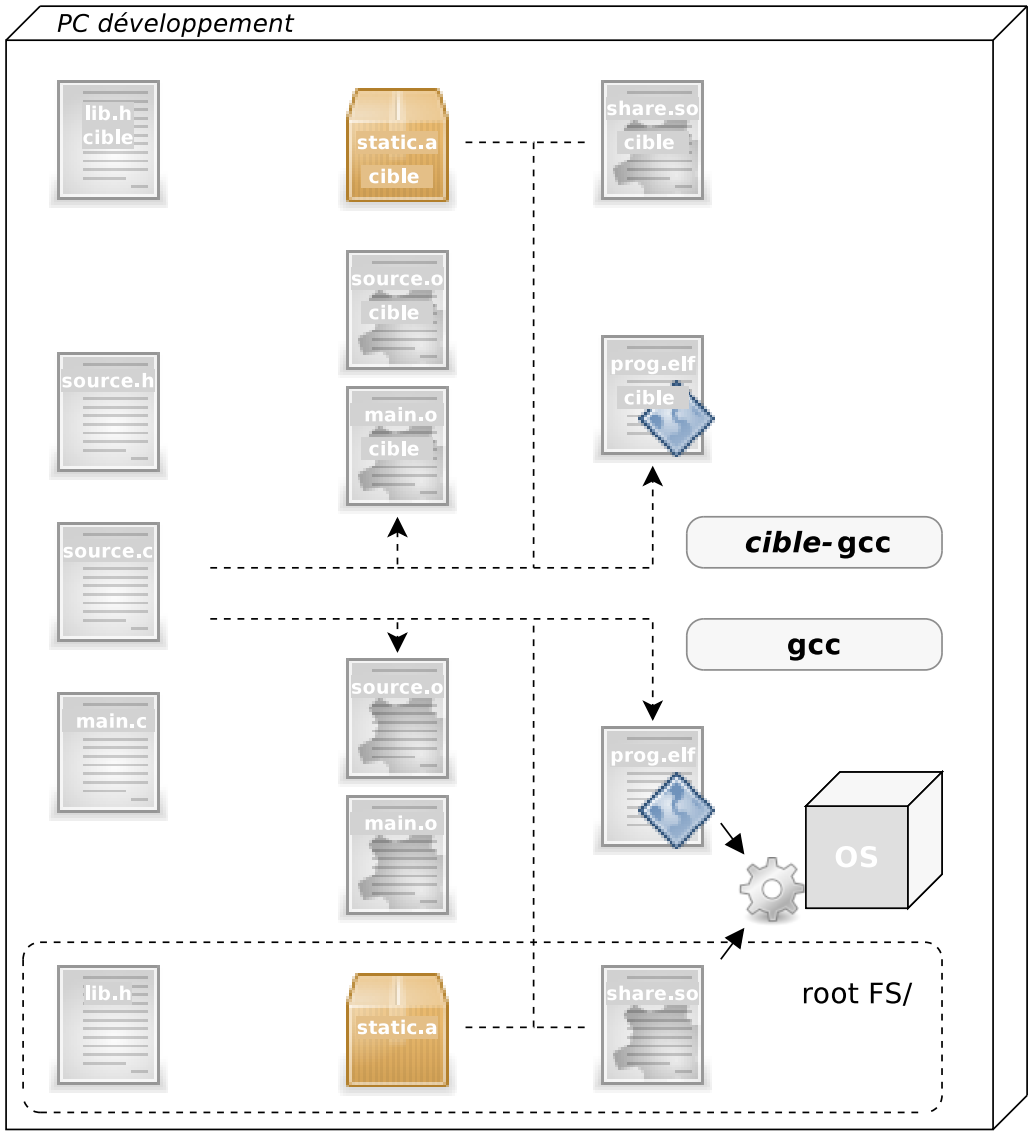


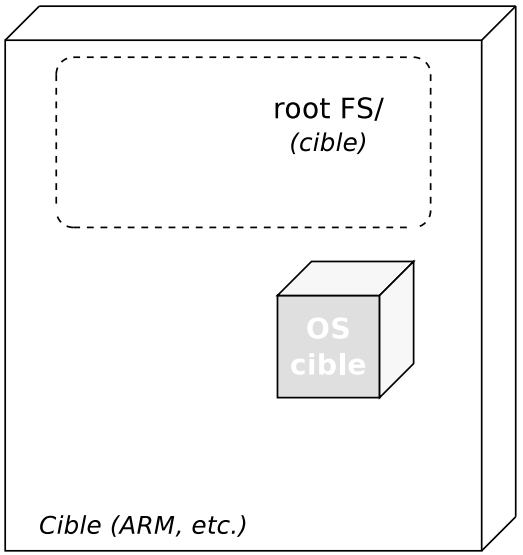
Développement croisé



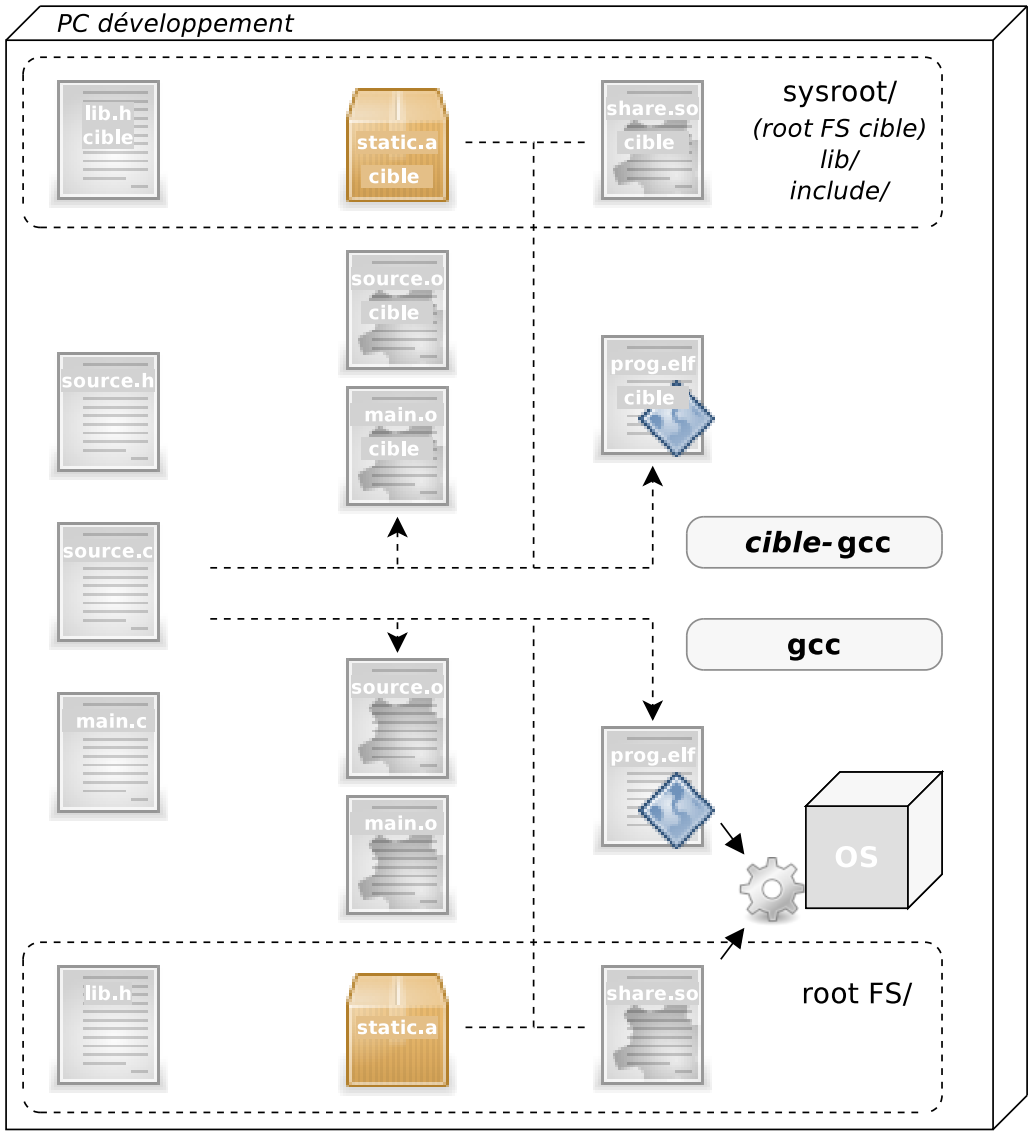


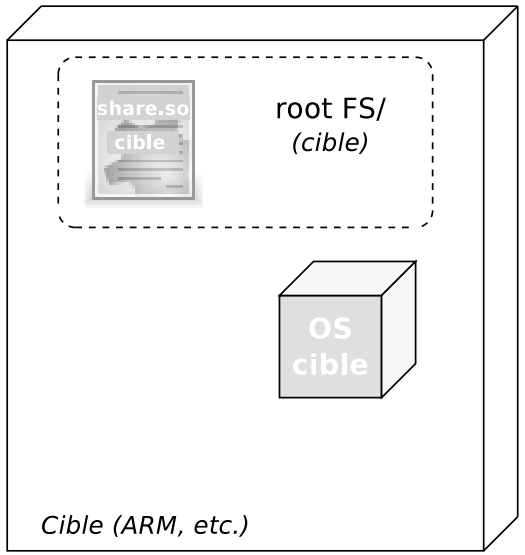
Développement croisé



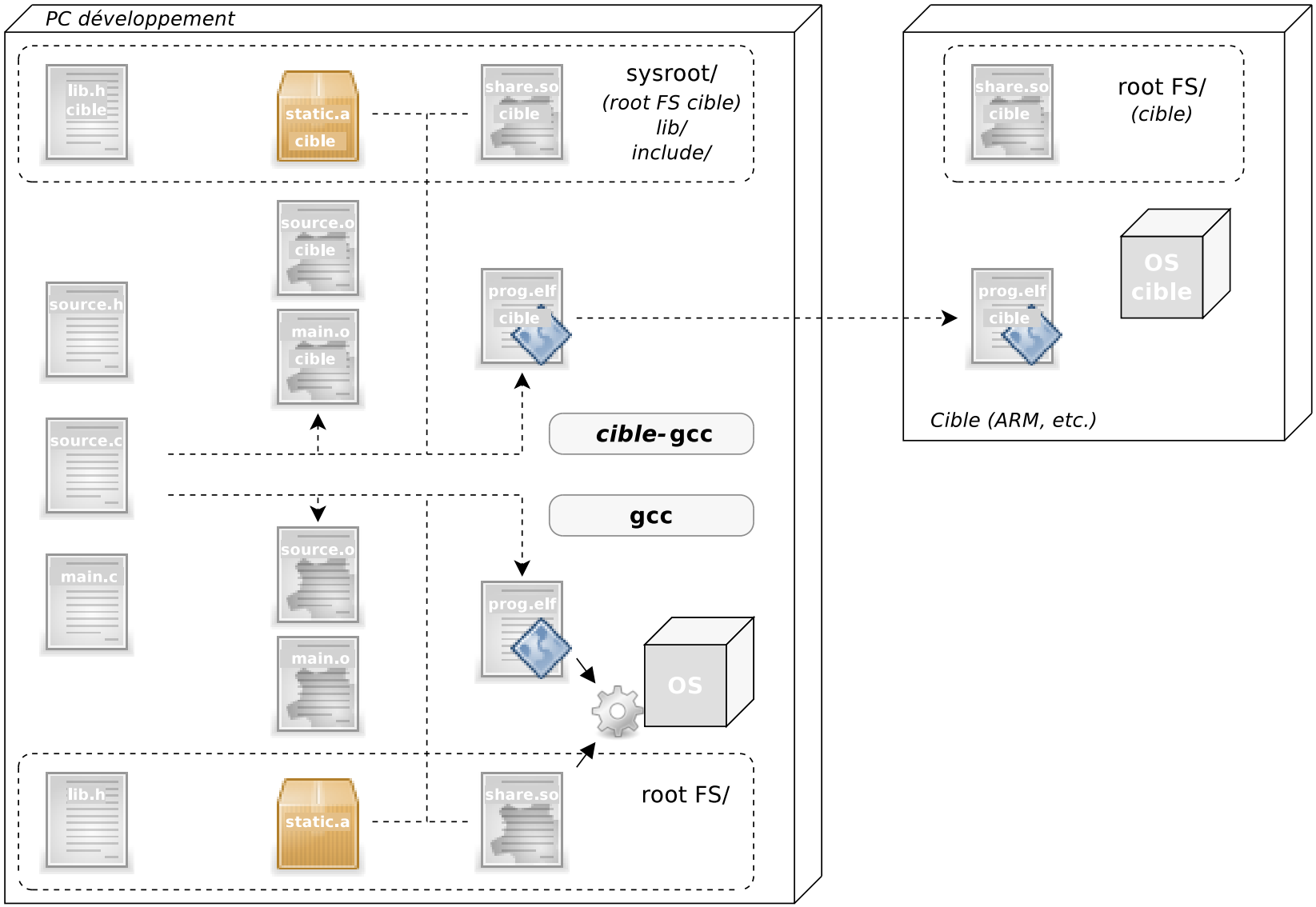


Développement croisé

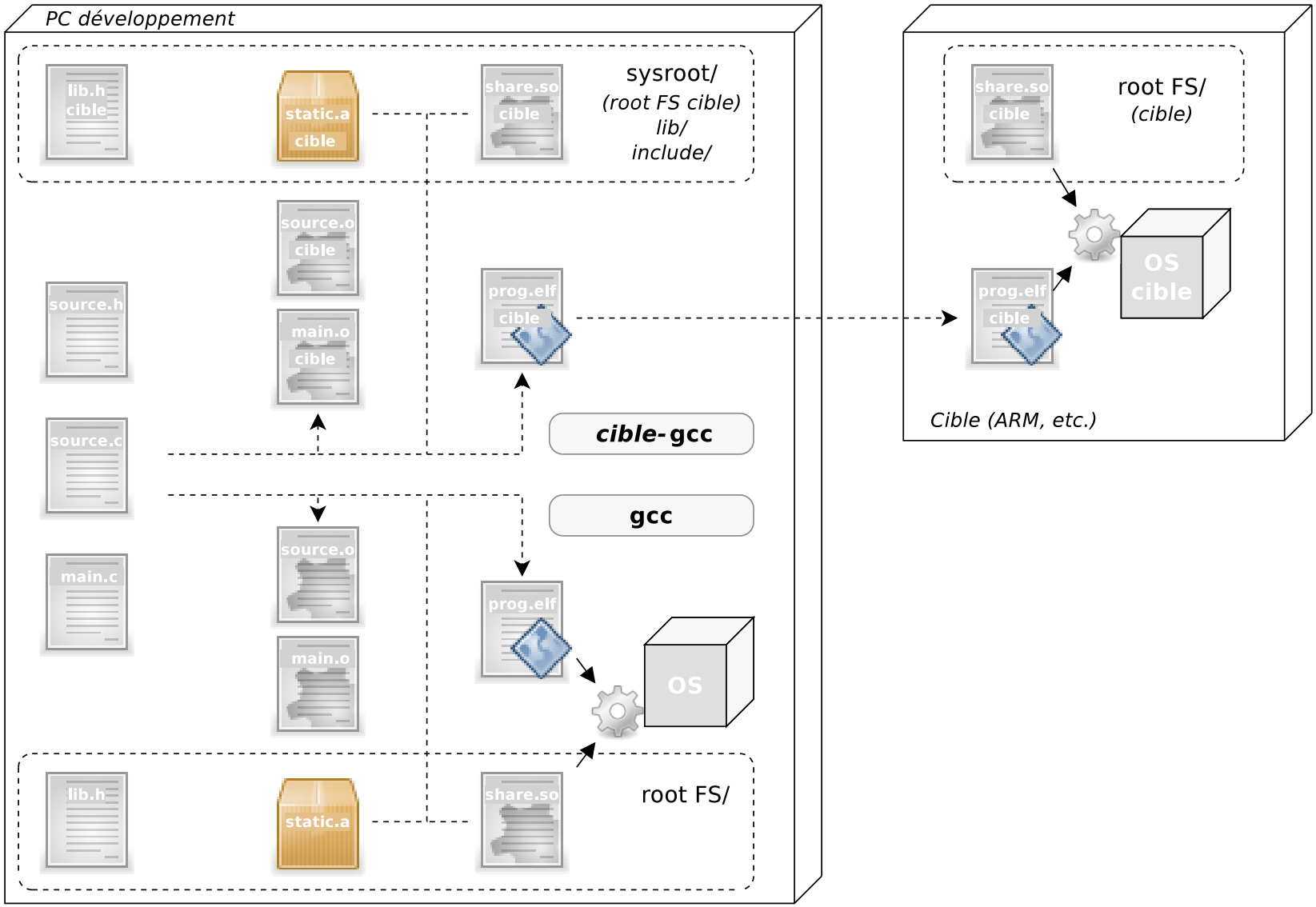




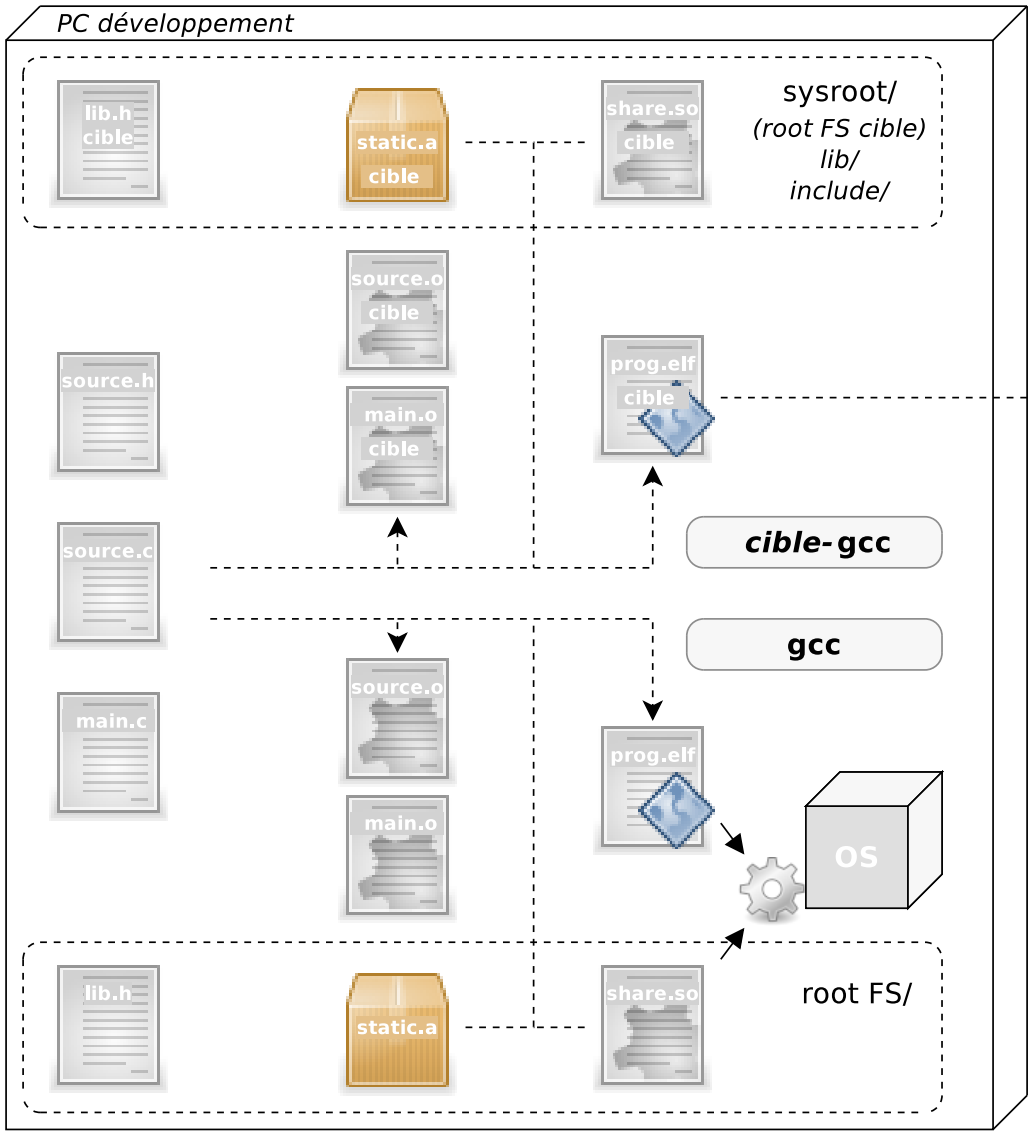
Développement croisé

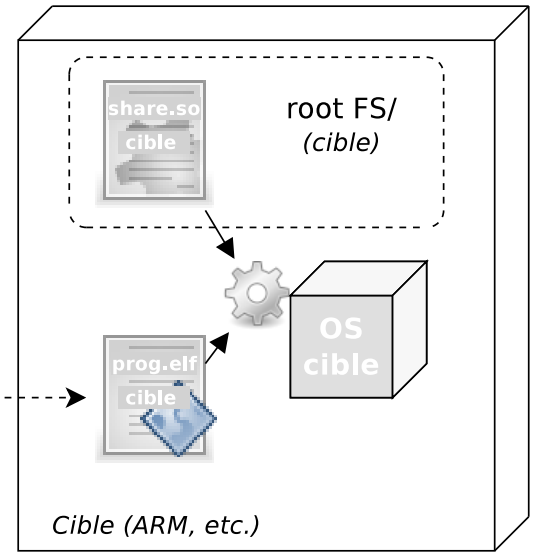


Développement croisé



Développement croisé





Attention :

- sysroot (*lib.so*, *lib.h*)
- ABI (*App. Binary Interface*)
- Makefile (multi-cibles)

Annexe

La compilation C

Quelques options de Gcc		
	`-D`	définition de macro (-D_MY_DEFINE=2 ⇔ #define _MY_DEFINE 2)
	`-I`	chemin de recherche des fichiers d'en-tête (en plus de /usr/include)
	`-L`	chemin de recherche des bibliothèques (en plus de /lib et /usr/lib)
	`-l`	nom d'une bibliothèque pour l'édition des liens (-lxxx pour libxxx.so)
	`-o`	nom de l'exécutabe à générer
	`-W`	avertissement (`-Wall`)
	`-pedantic`	avertissements rigoureux (portabilité)
	`-g`	inclusion d'informations de debuggage
	`-pg`	informations de profilage (fichier gmon.out créer à l'exécution, gprof)
	`-O`	optimisation (`-O0` à `-O3`) (incompatible avec `-g` !)
	`-std`	respect d'un standard (`-std=c90`, `-std=c99`, etc.)
	`-fpic`	(ou `-fPIC`) position indépendante du code (librairie)

La compilation C avec Makefile

Macros	`$@`	nom de la cible
	`$<`	nom de la première dépendance
	`$^`	liste des dépendances
	`$?`	liste des dépendances plus récentes que la cible
	`$*`	préfixe partagé par la source et la cible

Le debuggage

Quelques commandes de GDB		
	list	affiche le listing du code source
	run [argument]	lance le programme ($\rightarrow$ point d'arrêt)
	break <ligne>	insère un point d'arrêt à la ligne indiquée
	step	avance d'un pas ($\rightarrow$ sous-routine)
	next	avance jusqu'à la prochaine instruction ($\times$ sous-routines)
	cont	continue ($\rightarrow$ point d'arrêt)
	print <variable>	affiche le contenu d'une variable
	backtrace	affiche le contenu de la pile
	help	aide sur le debugger
	info	documentation sur le debugger
	quit	quitte le debugger