Makefiles

a.h

```h
#ifndef A_H_
#define A_H_

void f();


#endif /* A_H_ */
```

a.c

```c
#include "a.h"
void f() {...}
...
```

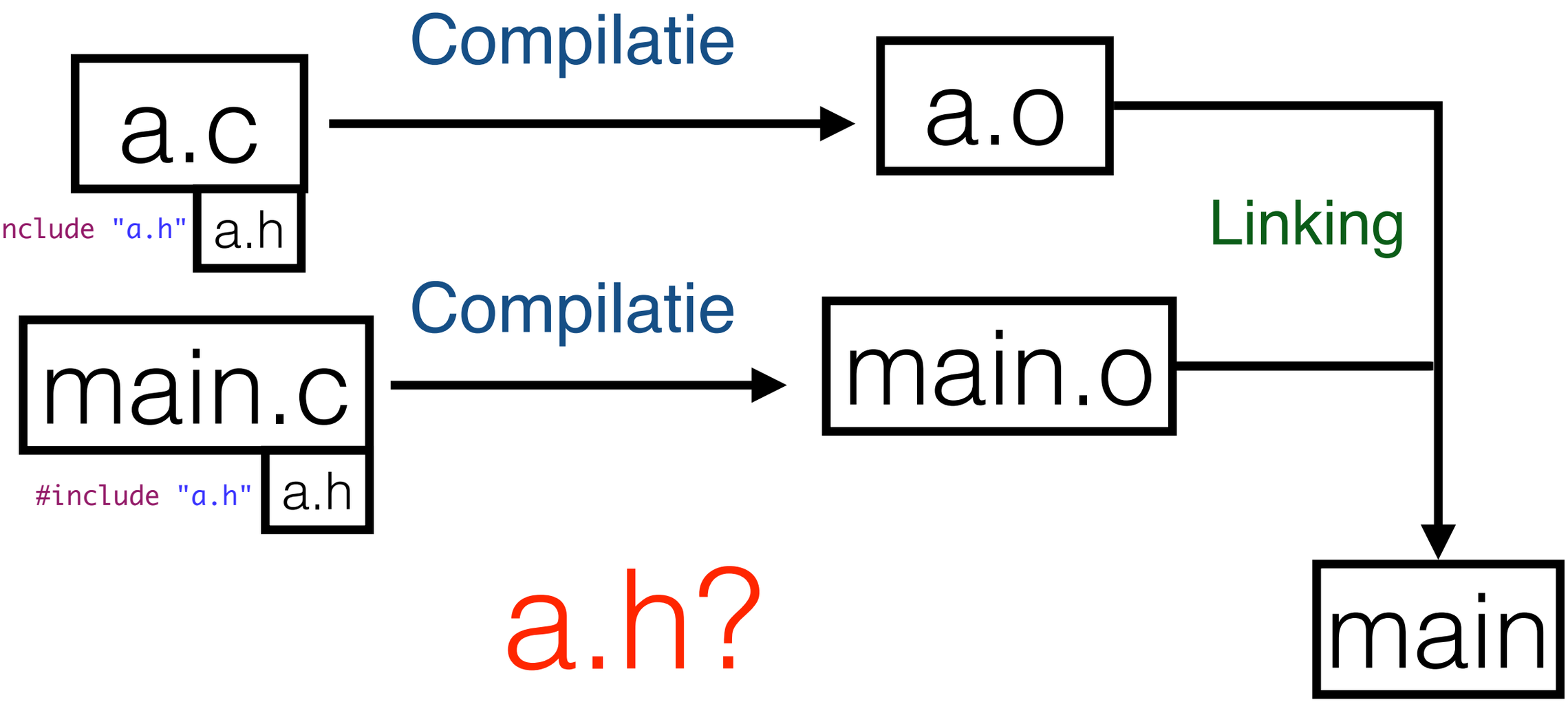
main.c

```c
#include <stdio.h>
#include "a.h"

int main() {
   ...
   f();
   return 0;
}
```

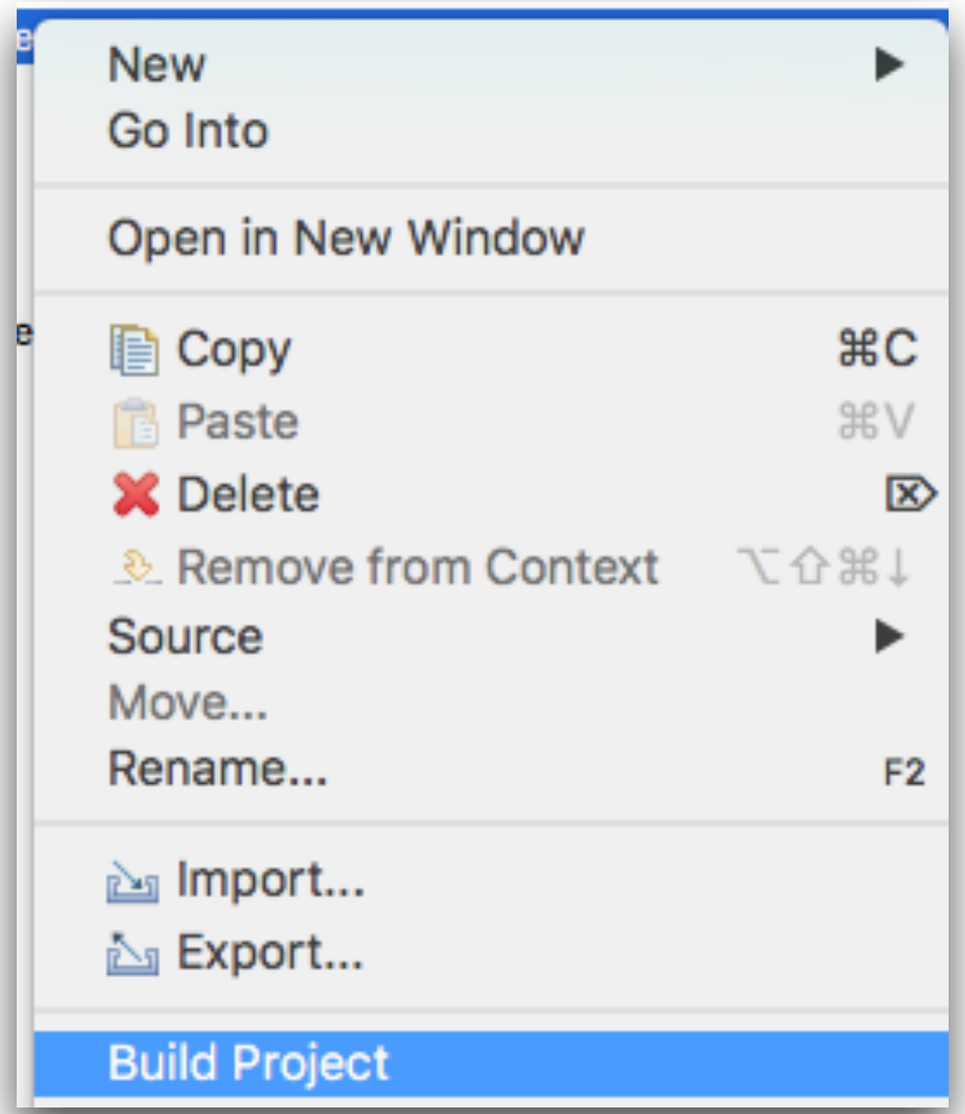
Recap

Executable maken = Compilatie + Linking



`#include "a.h"` → Gecompileerd samen met a.c/main.c

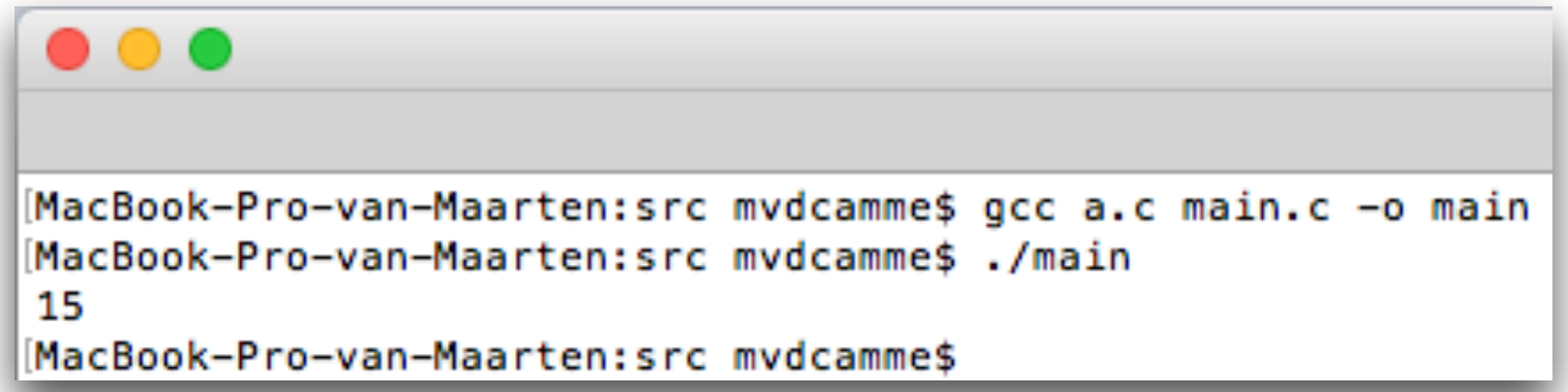
Compileren via Eclipse



Compileren via Eclipse

Maar:

1. Eclipse is traag
2. Programma installeren -> code compileren
3. Eclipse niet altijd beschikbaar
4. …



https://lifehacker.com/5633909/who-needs-a-mouse-learn-to-use-the-command-line-for-almost-anything
https://www.imore.com/how-use-terminal-mac-when-you-have-no-idea-where-start

Compileren via de terminal

```
[MacBook-Pro-van-Maarten:src mvdcamme$ gcc a.c main.c -o main
[MacBook-Pro-van-Maarten:src mvdcamme$ ./main
 15
[MacBook-Pro-van-Maarten:src mvdcamme$
```

gcc a.c main.c -o main

- Compileer a.c en main.c
- Link de object files
- Noem de output (de executable) “main”

Object-files apart compileren

gcc a.c -c -o a.o

gcc main.c -c -o main.o

-c: Compileer enkel een object-file, zonder te linken
-o X: Noem de output (de object-file) X

Linken:

gcc a.o main.o -o main

Object-files apart compileren

gcc a.c -c -o a.o

Voordeel hiervan?

gcc main.c -c -o main.o

sneller

-c: Compileer enkel een object-file, zonder te linken

-o X: Noem de output (de object-file) X

Linken:

gcc a.o main.o -o main

Makefiles

aaa.c

aab.c

. . .

ijk.c

. . .

zzz.c

Makefiles

aaa.c

aab.c

. . .

ijk.c

. . .

zzz.c

Code veranderd —> hercompileren

Alle files samen hercompileren?
Neen, te traag

Alle files een-voor-een hercompileren?
Neen, niet praktisch

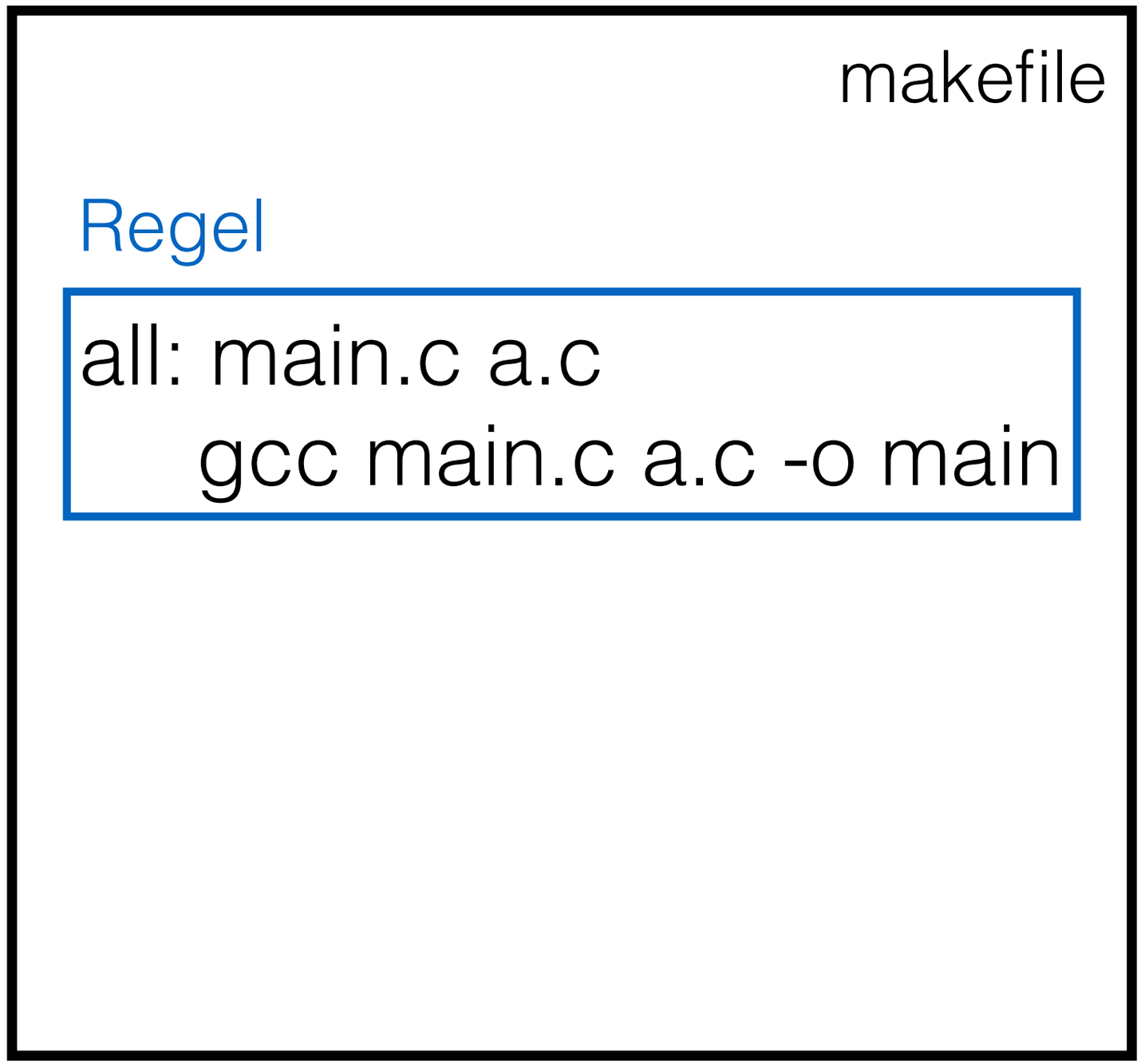
Makefile

Makefiles: basis

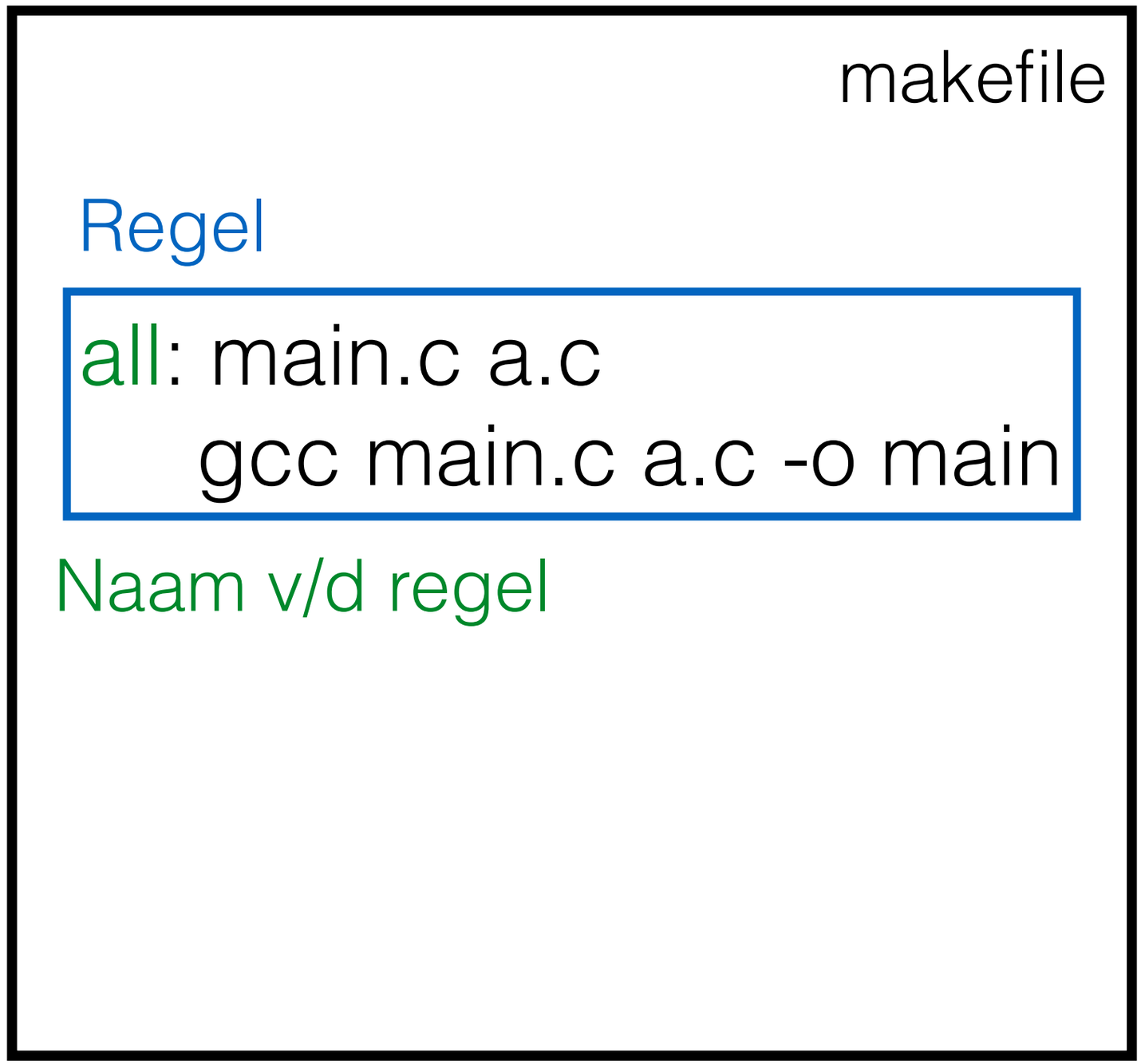
makefile

```
all: main.c a.c
	gcc main.c a.c -o main
```

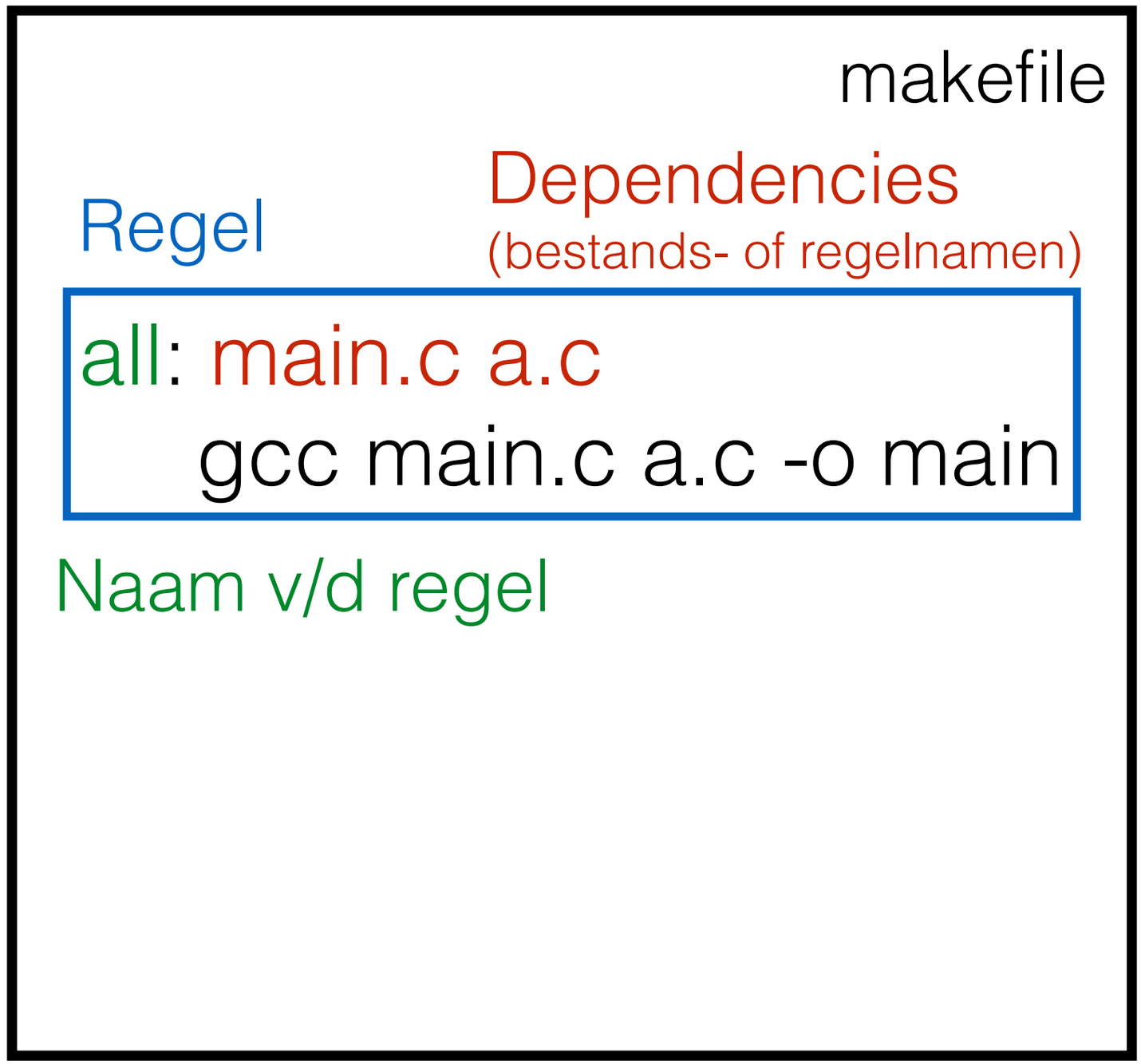
Makefiles: basis



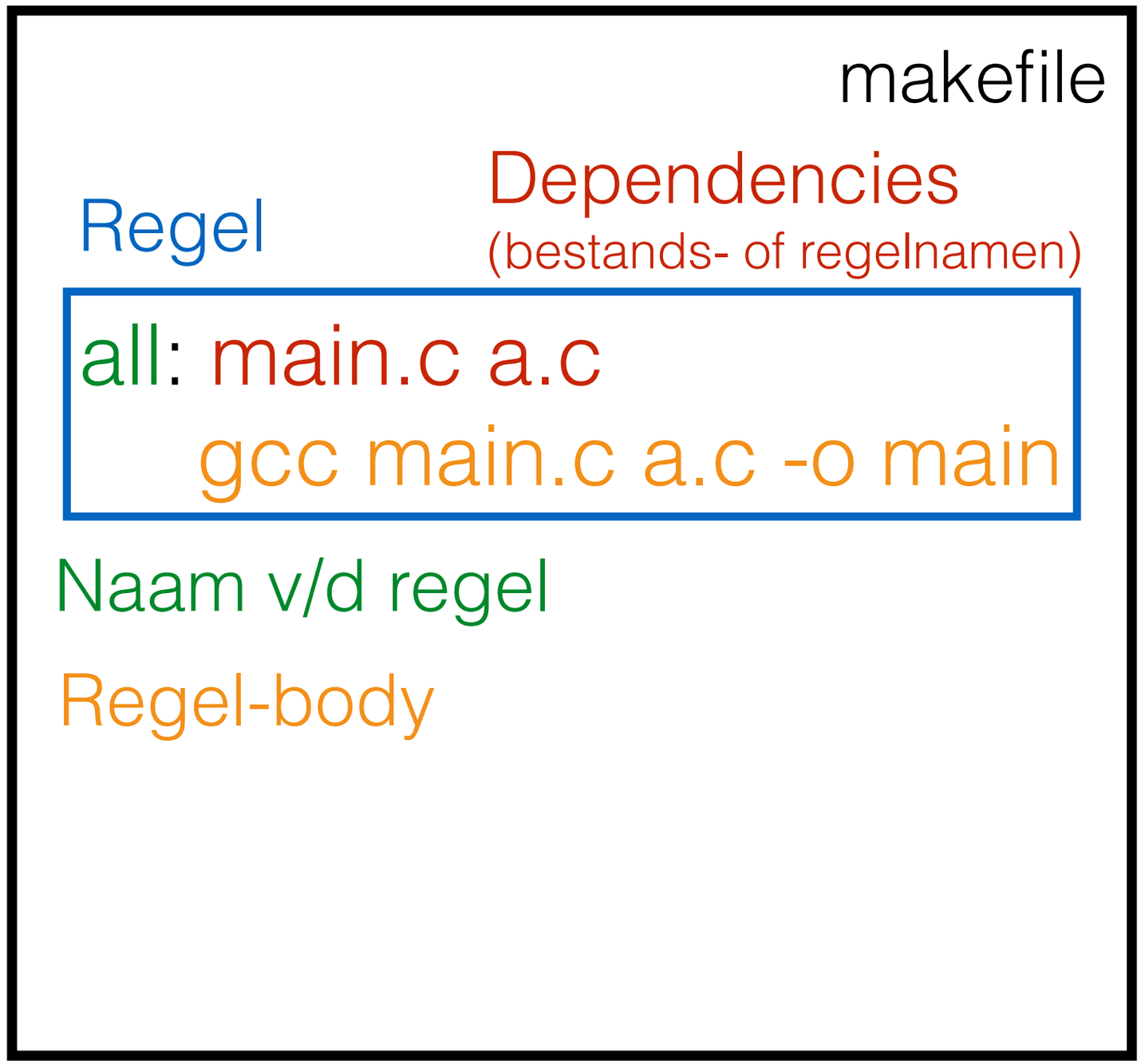
Makefiles: basis



Makefiles: basis



Makefiles: basis



Makefiles: basis

makefile

Regel

Dependencies
(bestands- of regelnamen)

```
all: main.c a.c
    gcc main.c a.c -o main
```

Naam v/d regel

Regel-body

Een dependency veranderd?
Voer regel-body opnieuw uit

Makefiles: basis

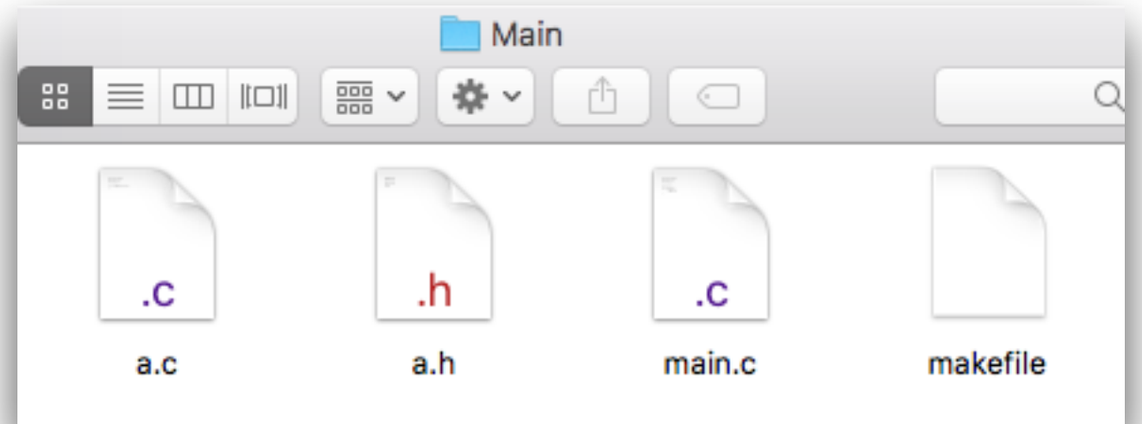
makefile

```
all: main.c a.c
    gcc main.c a.c -o main
```

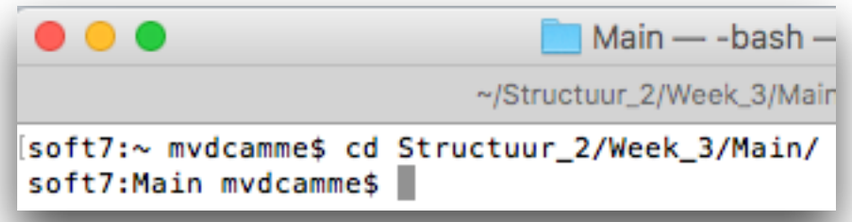
Syntax:
‘:’ na de naam
tab voor de regel-body

Makefiles: praktisch

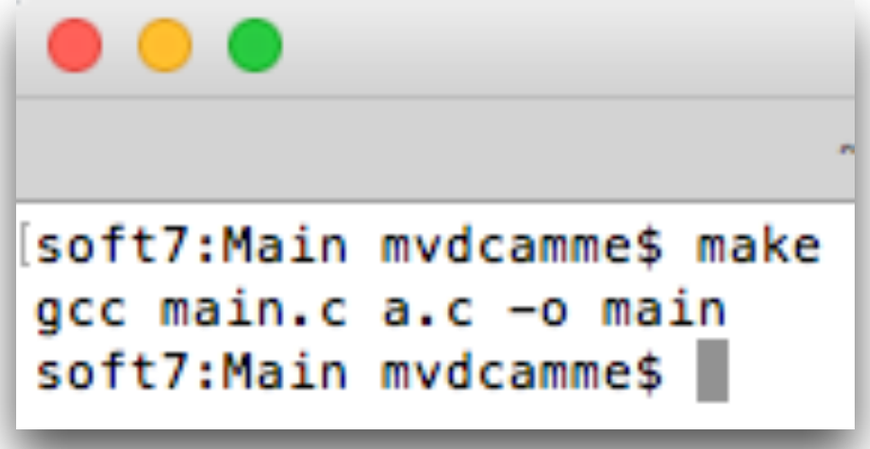
1. Folder met code + bestand “makefile”



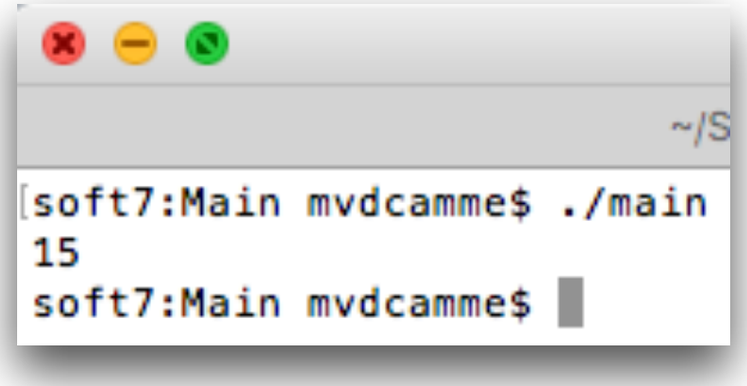
2. Open terminal en ga naar folder:
cd pad/naar/folder



3. Voer de makefile uit
make



4. Voer het nieuwe programma uit
./main



Object-files compileren

Makefile uitvoeren = bovenste regel uitvoeren

a.h

a.c

main.c

```
all: main.o a.o
	gcc main.o a.o -o main

main.o: main.c a.h
	gcc main.c -c -o main.o

a.o: a.c a.h
	gcc a.c -c -o a.o
```

Object-files compileren

Code veranderd?
Voer enkel relevante regels opnieuw uit

a.h

a.c

main.c

```
all: main.o a.o
	gcc main.o a.o -o main

main.o: main.c a.h
	gcc main.c -c -o main.o

a.o: a.c a.h
	gcc a.c -c -o a.o
```

Makefiles: extra

```
OBJ=main.o a.o
HEADERS=a.h
CC=gcc
TARGET=main

# Comments schrijven via ‘#’

all: $(OBJ)
	@echo “Linking all object files”
	$(CC) main.o a.o -o $(TARGET)

main.o: main.c $(HEADERS)
	@echo “Compiling main.c”
	$(CC) main.c -c -o main.o

a.o: a.c $(HEADERS)
	@echo “Compiling a.c”
	$(CC) a.c -c -o a.o
```

Makefiles: extra

```
OBJ=main.o a.o
HEADERS=a.h
CC=gcc
TARGET=main

all: $(OBJ)
	$(CC) $^ -o $(TARGET)

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

%.o = elke file met extensie.o
%.c = overeenkomstige file met extensie.c
$@ = naam v/h linkerelement
$< = naam v/h eerste rechterelement
$^ = alle rechterelementen

https://www.cs.jhu.edu/~langmea/resources/lecture_notes/130_makefiles.pdf

http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/