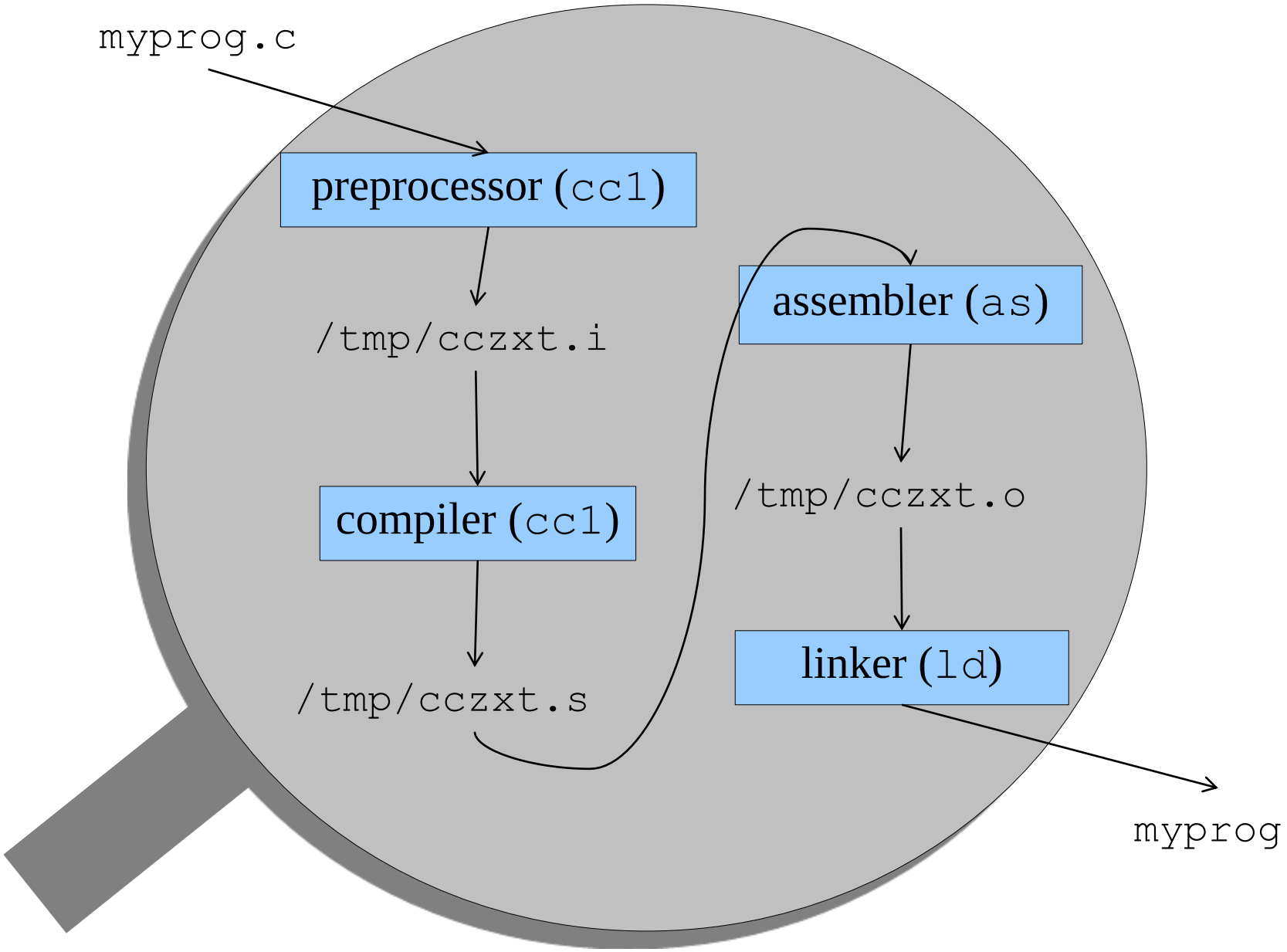
make


myprog.c
preprocessor (cc1)
/tmp/cczxt.i
compiler (cc1)
/tmp/cczxt.s
assembler (as)
/tmp/cczxt.o
linker (ld)
myprog

Δοκιμάστε

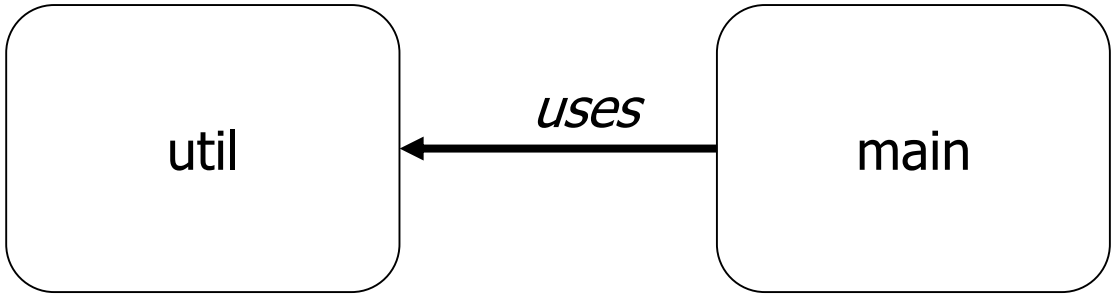
- `gcc -E test.c > test.i`
- `gcc -S test.i` (παράγει αρχείο `.s`)
- `gcc -c test.s` (παράγει αρχείο `.o`)
- `gcc test.o -o test`

Διαδικασία μεταγλώττισης

- Η διαδικασία μεταγλώττισης ενός συστήματος λογισμικού μπορεί να είναι αρκετά πολύπλοκη
- Μεταγλώττιση από πολλά ξεχωριστά αρχεία
- Με συγκεκριμένη σειρά
- Σύνδεση με πολλές και διαφορετικές βιβλιοθήκες

- Οι παραπάνω επιλογές μπορεί να εξαρτώνται από διάφορες παραμέτρους
  - π.χ. το σύστημα προορισμού (target platform)
- Δε χρειάζεται να μεταφράζονται εκ νέου αρχεία το περιεχόμενο των οποίων δεν έχει αλλάξει

Παράδειγμα



- `util.h`: prototypes βοηθητικών συναρτήσεων
- `util.c`: υλοποιήσεις των παραπάνω συναρτήσεων

- `main.c`: κυρίως πρόγραμμα, που χρησιμοποιεί συναρτήσεις του `util.h`

main.c

```
#include "util.h"

int main (int argc, char *argv[])
{
   …
   lowercase(…);
   …
}
```

util.h

```
char lowercase(char c);
```

util.c

```
#include "util.h"
#include<stdio.h>

char lowercase(char c) {
  …
}
```

compile, link, exec

```
> gcc -Wall -c util.c
> gcc -Wall -c main.c
> gcc main.o util.o -o all
> ./all
```

make

- Το εργαλείο (πρόγραμμα) make μπορεί να απαλλάξει τον προγραμματιστή από το να πρέπει να σκέφτεται και να πληκτρολογεί τις σωστές εντολές, **κάθε φορά** που πρέπει να παραχθεί κώδικας

- Το make μπορεί να εκτελέσει μια ή περισσότερες λειτουργίες (π.χ. μεταγλώττιση) με **αυτόματο** τρόπο
- Με βάση συγκεκριμένους **κανόνες**
- Οι κανόνες δίνονται σε ένα αρχείο, το **Makefile**
- Περιγράφονται με μια συγκεκριμένη σύνταξη
  - μια γλώσσα προγραμματισμού ειδικού σκοπού

Στοιχεία Makefile

- **Μεταβλητές:** χρησιμοποούνται για ονόματα αρχείων, προγραμμάτων, flags κτλ. που εμφανίζονται σε πολλά σημεία του Makefile

- **Κανόνες:** αποτελούνται από το όνομα του **στόχου** (**τι** θέλουμε να δημιουργήσουμε) και τις σχετικές **συνταγές παραγωγής** (**πώς** το δημιουργούμε)
- Οι συνταγές χρησιμοποιούν shell commands

- **Σχόλια:** για τον συνηθισμένο λόγο

- ... και πολλά άλλα (βλέπε manual)

Εκτέλεση Makefile

- `make <goal name>`
- Το make ψάχνει στο τρέχοντα κατάλογο να βρει ένα αρχείο με όνομα Makefile (ή makefile)
- Ψάχνει να βρει τον στόχο μέσα στο Makefile
- Εκτελεί τις εντολές για το συγκεκριμένο στόχο
- Αφού πρώτα εκτελέσει ότι εντολές αντιστοιχούν σε **προαπαιτούμενους** στόχους (αναδρομή)

- `make`
- Όπως παραπάνω, αλλά εκτελεί τις εντολές για τον **πρώτο** στόχο που εμφανίζεται στο Makefile

Κανόνες

```
<target>:  <dependencies>
        <command1>
        <command2>
        ...
        <command>
```

- `<target>` (στόχος): αρχείο προς κατασκευή (συνήθως εκτελέσιμο) ή ενέργεια προς ολοκλήρωση
- `<dependencies>` (εξαρτήσεις): ένα ή περισσότερα αρχεία ή ενέργειες από τις οποίες εξαρτάται ο στόχος
- `<command>`: εντολή που στέλνει το make προς το shell για να δημιουργηθεί ο στόχος (το σύνολο των εντολών για ένα στόχο ονομάζεται και «συνταγή»).
- **Προσοχή:** tab πριν από **κάθε** εντολή

```c
#include "util.h"

int main (int argc, char *argv[])
{
    …
    lowercase(…);
    …
}
```

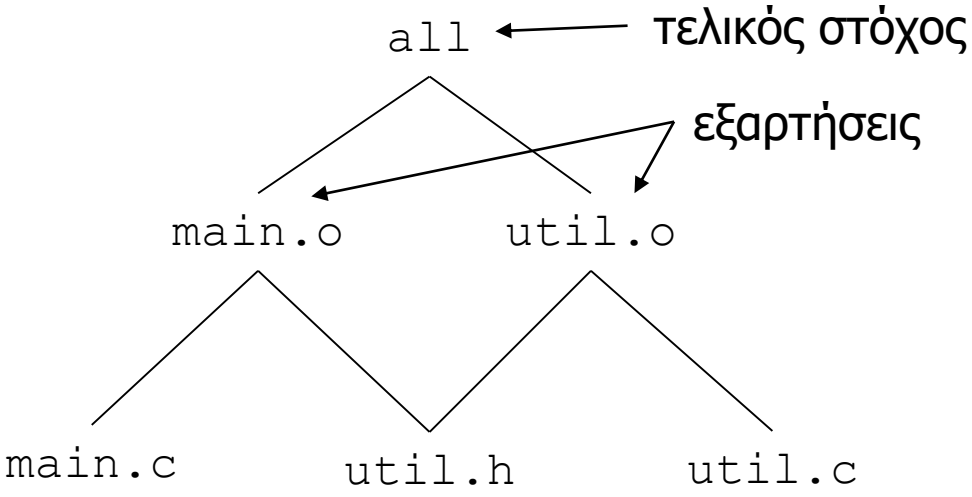
main.c

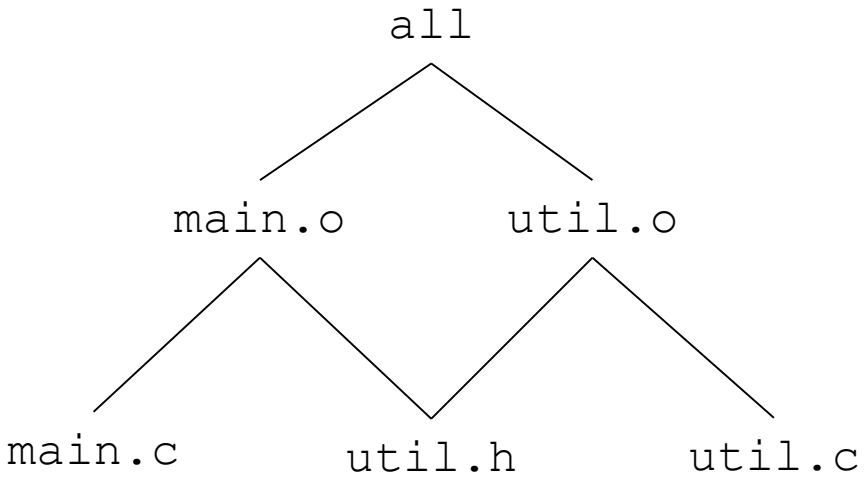
util.h

```c
char lowercase(char c);
```

util.c

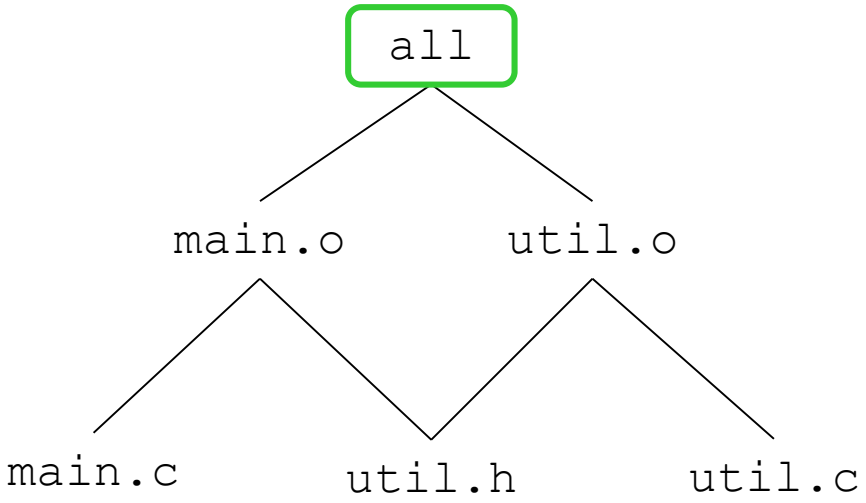
```c
#include "util.h"
#include<stdio.h>

char lowercase(char c) {
   …
}
```





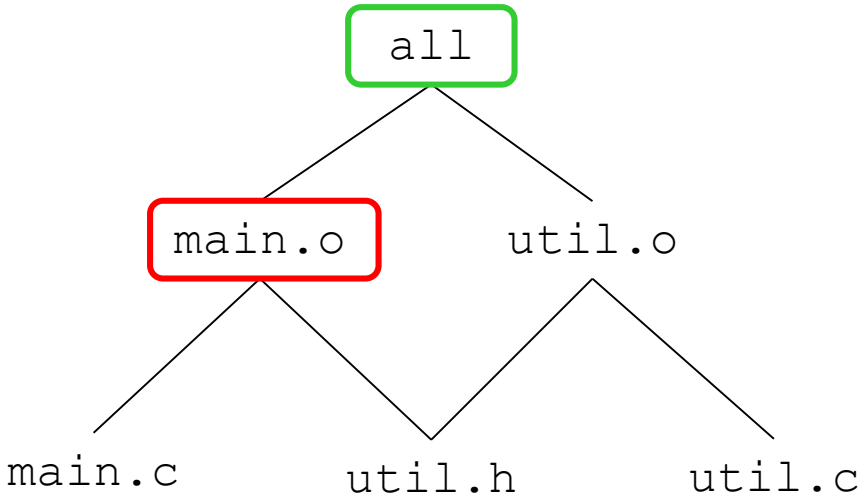
Makefile

```
all: main.o util.o
	gcc -Wall -g main.o util.o -o all

main.o: main.c util.h
	gcc -Wall -g -c main.c

util.o: util.c util.h
	gcc -Wall -g -c util.c
```



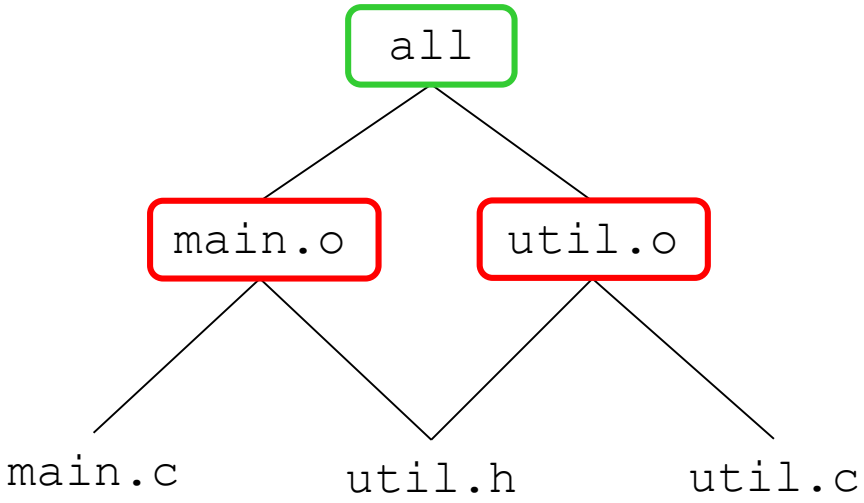
Makefile

```
all: main.o util.o
	gcc -Wall -g main.o util.o -o all

main.o: main.c util.h
	gcc -Wall -g -c main.c

util.o: util.c util.h
	gcc -Wall -g -c util.c
```



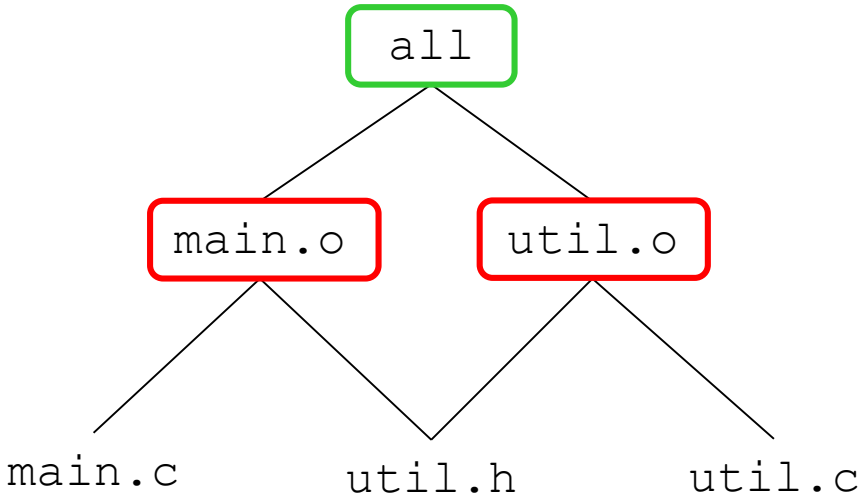
Makefile

```
all: main.o util.o
	gcc -Wall -g main.o util.o -o all

main.o: main.c util.h
	gcc -Wall -g -c main.c

util.o: util.c util.h
	gcc -Wall -g -c util.c
```



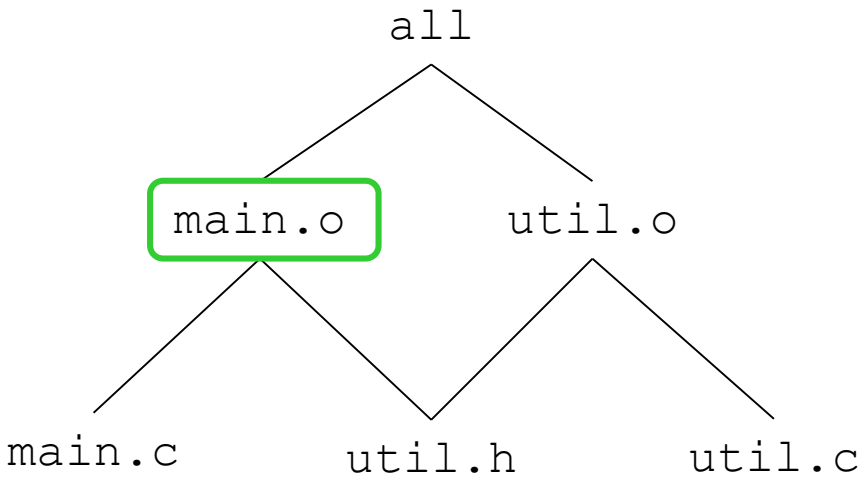
Makefile

```
all: main.o util.o
	gcc -Wall -g main.o util.o -o all

main.o: main.c util.h
	gcc -Wall -g -c main.c

util.o: util.c util.h
	gcc -Wall -g -c util.c
```



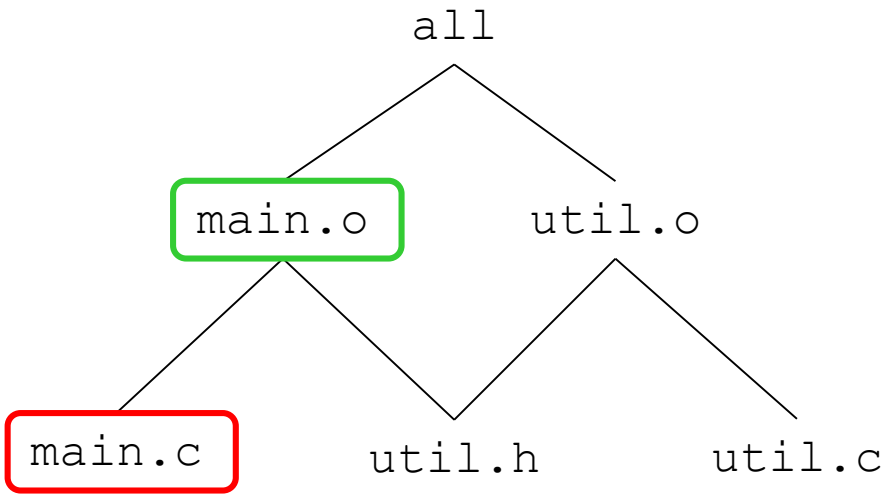
Makefile

```
all: main.o util.o
	gcc -Wall -g main.o util.o -o all

main.o: main.c util.h
	gcc -Wall -g -c main.c

util.o: util.c util.h
	gcc -Wall -g -c util.c
```



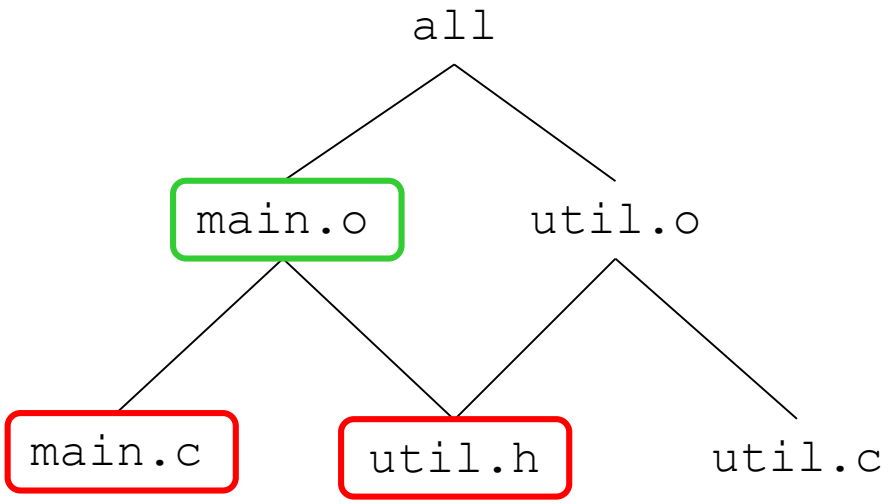
Makefile

```
all: main.o util.o
	gcc -Wall -g main.o util.o -o all

main.o: main.c util.h
	gcc -Wall -g -c main.c

util.o: util.c util.h
	gcc -Wall -g -c util.c
```



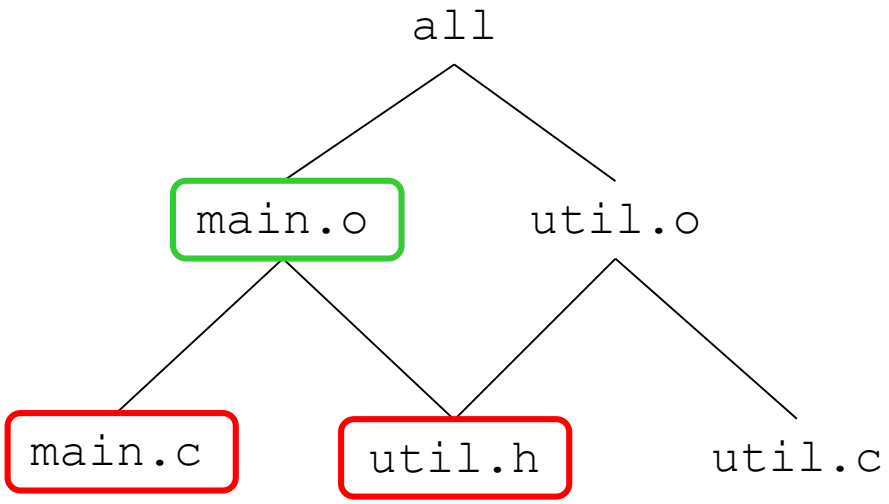
Makefile

```
all: main.o util.o
	gcc -Wall -g main.o util.o -o all

main.o: main.c util.h
	gcc -Wall -g -c main.c

util.o: util.c util.h
	gcc -Wall -g -c util.c
```



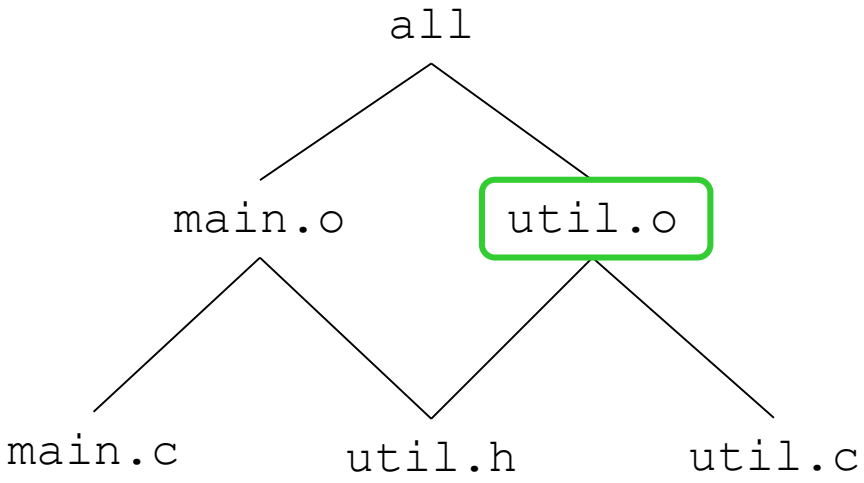
Makefile

```
all: main.o util.o
	gcc -Wall -g main.o util.o -o all

main.o: main.c util.h
	gcc -Wall -g -c main.c

util.o: util.c util.h
	gcc -Wall -g -c util.c
```



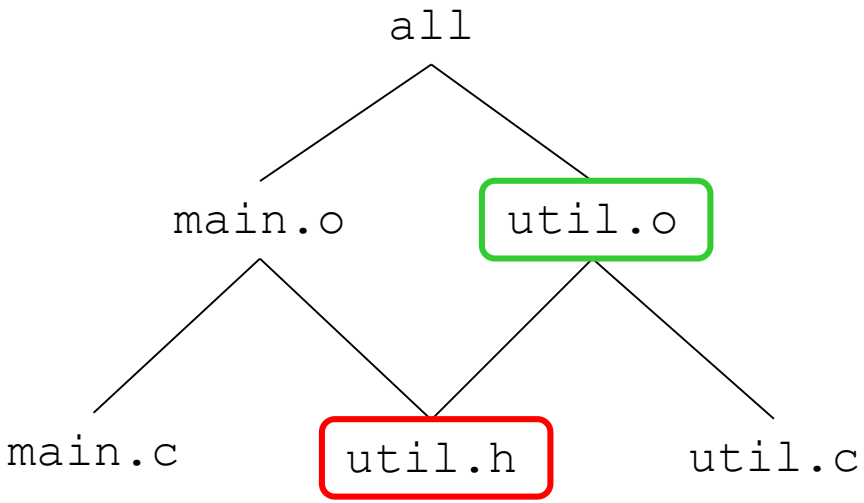
Makefile

```
all: main.o util.o
	gcc -Wall -g main.o util.o -o all

main.o: main.c util.h
	gcc -Wall -g -c main.c

util.o: util.c util.h
	gcc -Wall -g -c util.c
```



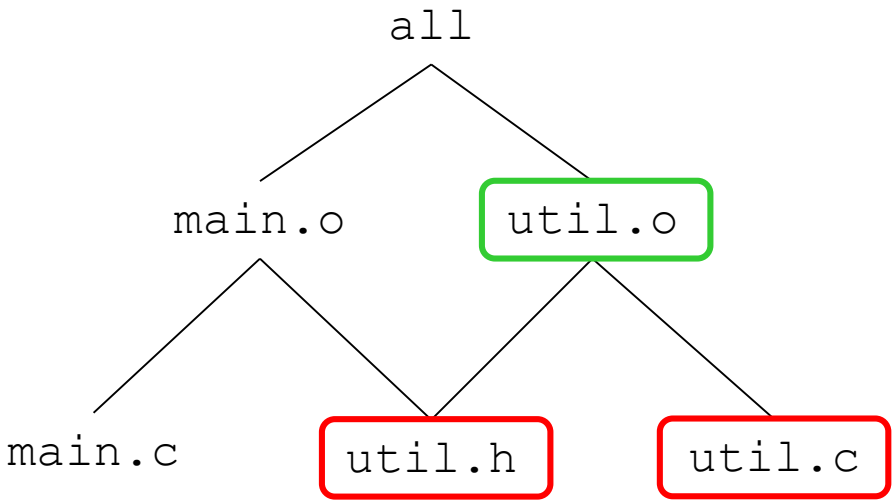
Makefile

```
all: main.o util.o
	gcc -Wall -g main.o util.o -o all

main.o: main.c util.h
	gcc -Wall -g -c main.c

util.o: util.c util.h
	gcc -Wall -g -c util.c
```



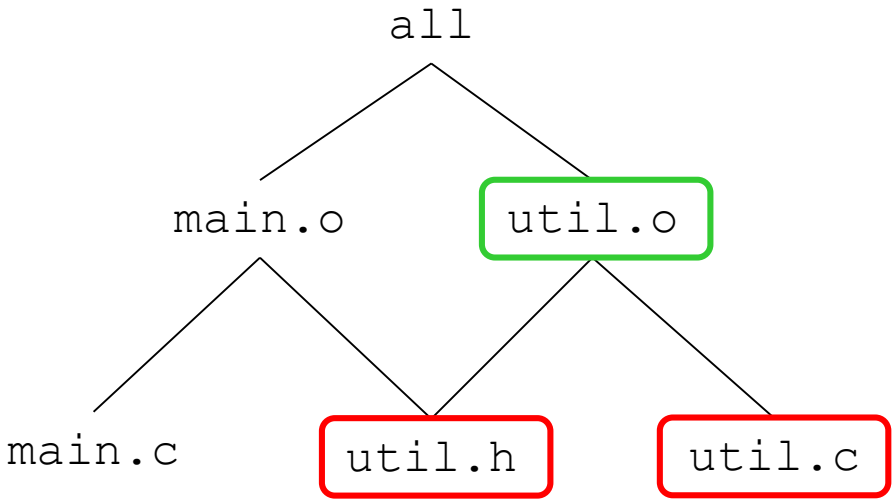
Makefile

```
all: main.o util.o
	gcc -Wall -g main.o util.o -o all

main.o: main.c util.h
	gcc -Wall -g -c main.c

util.o: util.c util.h
	gcc -Wall -g -c util.c
```



Makefile

```
all: main.o util.o
	gcc -Wall -g main.o util.o -o all

main.o: main.c util.h
	gcc -Wall -g -c main.c

util.o: util.c util.h
	gcc -Wall -g -c util.c
```



Makefile

```
all: main.o util.o
	gcc -Wall -g main.o util.o -o all

main.o: main.c util.h
	gcc -Wall -g -c main.c

util.o: util.c util.h
	gcc -Wall -g -c util.c
```

Μεταβλητές

```
<name> = <value>
```

- `<name>` (όνομα): το όνομα της μεταβλητής
- `<value>` (τιμή): ένα ή περισσότερα αρχεία ή ενέργειες από τις οποίες εξαρτάται ο στόχος

- αναφορές στο όνομα μιας μεταβλητής (μέσα σε άλλες εκφράσεις) γίνονται με `$()`

```
CC = gcc
CFLAGS = -Wall -g
OBJ = main.o util.o

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

main.o: main.c util.h
	$(CC) $(CFLAGS) -c main.c

util.o: util.c util.h
	$(CC) $(CFLAGS) -c util.c
```

Σχόλια

- Ξεκινούν με # και τελειώνουν στο τέλος της γραμμής
- Μπορούν να τοποθετηθούν (σχεδόν) οπουδήποτε

```
CC = gcc
CFLAGS = -Wall -g
OBJ = main.o util.o #all object files

#top-level goal
all: $(OBJ)
	$(CC) $(OBJ) -o all

main.o: main.c util.h
	$(CC) $(CFLAGS) -c main.c

util.o: util.c util.h
	$(CC) $(CFLAGS) -c util.c
```

Patterns και αυτόματες μεταβλητές

- Το σύμβολο `$@` αναπαριστά το όνομα του στόχου
- Το σύμβολο `%` μπορεί να χρησιμοποιηθεί στο στόχο ή/και στις εξαρτήσεις) για να αναπαραστήσει μέρος του ονόματος ενός αρχείου
- Το σύμβολο `$*` αναπαριστά το κομμάτι του ονόματος που αντιστοιχεί στο `%`

```
CC = gcc
OBJ = foo.o bar.o

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

foo.o: foo.c foo.h
	$(CC) -c foo.c

bar.o: bar.c bar.h
	$(CC) -c bar.c
```

```
CC = gcc
OBJ = foo.o bar.o

all: $(OBJ)
	$(CC) $(OBJ) -o $@

%.o: %.c %.h
	$(CC) -c $*.c
```

Στόχοι που δεν είναι αρχεία

```
.PHONY: clean
clean:
	rm *.o
```

- Ένας κανόνας μπορεί οδηγεί σε **οποιαδήποτε πράξη**, όχι απαραίτητα στην μεταγλώττιση αρχείων
- Τότε ο στόχος **δεν** είναι όνομα αρχείου
- Συνήθως χρησιμοποιούμε το χαρακτηρισμό `.PHONY` για να αποφύγουμε «συγκρούσεις» με αρχεία που πιθανώς έχουν ίδιο όνομα με το στόχο
- Τυπική εφαρμογή: κανόνας για το σβήσιμο περιττών αρχείων μετά την ολοκλήρωση της μεταγλώττισης

Στόχοι που δεν είναι αρχεία

```
test%: test%.data myprog
	./myprog < test$*.data

testall: test1 test2 test3
```

- Ένας κανόνας μπορεί να αποτελείται **μόνο** από το στόχο και τις εξαρτήσεις
- Να μην έχει κάποιες εντολές (συνταγή)

Σειρά εκτέλεσης

```
all: target1
@echo zero

target1: target2 target3
@echo one

target2:
@echo two

target3:
@echo three
```

output

Σειρά εκτέλεσης

```
all: target1
@echo zero

target1: target2 target3
@echo one

target2:
@echo two

target3:
@echo three
```

output

Σειρά εκτέλεσης

```
all: target1
@echo zero

target1: target2 target3
@echo one

target2:
@echo two

target3:
@echo three
```

output

Σειρά εκτέλεσης

```
all: target1
@echo zero

target1: target2 target3
@echo one

target2:
@echo two

target3:
@echo three
```

output

Σειρά εκτέλεσης

```
all: target1
@echo zero

target1: target2 target3
@echo one

target2:
@echo two

target3:
@echo three
```

output

Σειρά εκτέλεσης

```
all: target1
@echo zero

target1: target2 target3
@echo one

target2:
@echo two

target3:
@echo three
```

output

Σειρά εκτέλεσης

```
all: target1
@echo zero

target1: target2 target3
@echo one

target2:
@echo two

target3:
@echo three
```

output

```
two
```

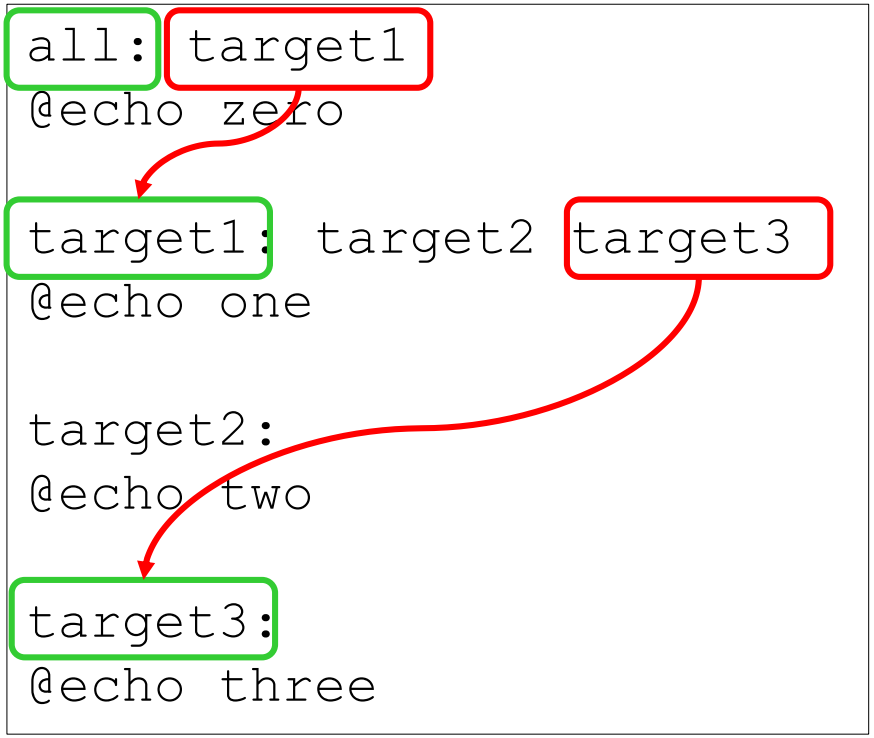
Σειρά εκτέλεσης

```
all: target1
@echo zero

target1: target2 target3
@echo one

target2:
@echo two

target3:
@echo three
```

output

```
two
```

Σειρά εκτέλεσης



output

```
two
```

Σειρά εκτέλεσης

```
all: target1
@echo zero

target1: target2 target3
@echo one

target2:
@echo two

target3:
@echo three
```

output

```
two
three
```

Σειρά εκτέλεσης

```
all: target1
@echo zero

target1: target2 target3
@echo one

target2:
@echo two

target3:
@echo three
```

output

```
two
three
one
```

Σειρά εκτέλεσης

```
all: target1
@echo zero

target1: target2 target3
@echo one

target2:
@echo two

target3:
@echo three
```

output

```
two
three
one
zero
```

Σειρά εκτέλεσης

```
all: target1
@echo zero

target1: target2 target3
@echo one

target2:
@echo two

target3:
@echo three
```

output

```
two
three
one
zero
```

output

```
three
two
one
zero
```

Εντοπισμός εξαρτήσεων

- Βασική προϋπόθεση για να φτιαχτεί ένα σωστό Makefile είναι να καταγραφούν σωστά οι εξαρτήσεις για την κατασκευή ενός στόχου
- Π.χ., προϋπόθεση για την μεταγλώττιση ενός προγράμματος που χρησιμοποιεί μια βοηθητική βιβλιοθήκη, είναι το αντίστοιχο `.h` αρχείο

- `gcc -MM <file names>`
- Επιστρέφει μια λίστα εξαρτήσεων για τα αρχεία
- για να συμπεριληφθούν και τα system headers `-M`

Συνηθισμένα λάθη

- `Makefile:4: *** missing separator.`
  - δεν υπάρχει tab στην αρχή εντολής
- `make: *** No rule to make target 'all'.`
  - δεν υπάρχει ο στόχος
- `Makefile:7: warning: overriding commands for target 'test1'`
  - πολλές «συνταγές» για τον ίδιο στόχο
- `Makefile:4: warning: ignoring old commands for target 'test1'`
  - εκτελείται όποια συνταγή είναι τελευταία
  - επιτρέπεται να υπάρχουν πολλοί κανόνες για τον ίδιο στόχο (αλλά τότε πρέπει να έχουν μόνο εξαρτήσεις όχι συνταγές)