Διαδικαστικός Προγραμματισμός

Ενότητα 13: Αρθρωτή Ανάπτυξη Προγραμμάτων

Καθηγήτρια Μαρία Σατρατζέμη

Τμήμα Εφαρμοσμένης Πληροφορικής

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

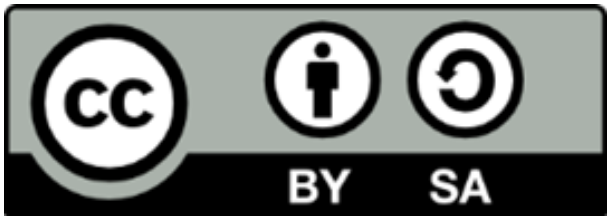
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Μακεδονίας» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σκοποί ενότητας

- Να αναγνωρίσετε ότι οι διασυνδέσεις στη C αναπαριστώνται με αρχεία κεφαλίδας.
- Να μάθετε τους συντακτικούς κανόνες που απαιτούνται για τη συγγραφή ενός αρχείου κεφαλίδας.
- Να κατανοήσετε την ανάγκη για διάσπαση του κώδικα σε πολλά αρχεία.
- Να μεταγλωττίζετε και συνδέετε πολλά αρχεία και βιβλιοθήκες κάνοντας χρήση του gcc compiler.

Προγράμματα πολλαπλών αρχείων

- Η ενότητα αυτή ασχολείται με προβλήματα που αντιμετωπίζουμε όταν γράφουμε σχετικά μεγάλα προγράμματα.
- Συνήθως τα μεγάλα προγράμματα είναι χωρισμένα σε **αρθρώματα** (**modules**), τα οποία βρίσκονται σε ξεχωριστά αρχεία πηγαίου κώδικα.
- Η συνάρτηση **main()** θα βρίσκεται σε ένα αρχείο, έστω **main.c**, πιθανώς με κάποιες άλλες συναρτήσεις, ενώ τα υπόλοιπα αρχεία θα περιέχουν άλλες συναρτήσεις, αλλά όχι τη συνάρτηση main().
- Επίσης μπορούμε να δημιουργήσουμε μια δική μας βιβλιοθήκη, γράφοντας μια ομάδα από συναρτήσεις (χωρίς να υπάρχει συνάρτηση **main()** σε αυτές), οι οποίες θα βρίσκονται σε ένα (ή περισσότερα) αρθρώματα.
- Τέτοιες βιβλιοθήκες συναρτήσεων μπορούν να ενσωματωθούν σε διάφορα προγράμματα, με τον ίδιο περίπου τρόπο που ενσωματώνουμε τις συναρτήσεις των πρότυπων βιβλιοθηκών της C.

Αρχεία κεφαλίδας ή διασυνδέσεις

- Σε περίπτωση που έχουμε πολλαπλά αρχεία, το κάθε ένα από αυτά θα περιέχει τους ορισμούς των σταθερών, μεταβλητών και συναρτήσεών του, αλλά θα πρέπει (πιθανά) να γνωρίζει και τους αντίστοιχους ορισμούς από τα υπόλοιπα αρχεία.
- Πώς επιτυγχάνεται η κοινοποίηση αυτών των ορισμών;
- Η καλύτερη λύση είναι να συλλέξουμε αυτές τις οδηγίες σε ένα αρχείο, και να το ενσωματώνουμε (**include**) σε καθένα από τα επιμέρους αρχεία.
- Ένα τέτοιο αρχείο λέγεται **αρχείο κεφαλίδας** (***header file***) ή ***διασύνδεση***.
- Συνήθως τα αρχεία αυτά έχουν την κατάληξη **.h**. Έχουμε ήδη συναντήσει αρχεία κεφαλίδας, όπως το **stdio.h**, η δε ενσωμάτωση επιτυγχάνεται με την οδηγία προεπεξεργαστή **#include**, π.χ.
  #include <stdio.h>
- Τα σύμβολα < > σημαίνουν ότι το αρχείο κεφαλίδας βρίσκεται στον προκαθορισμένο φάκελο των αρχείων κεφαλίδας .

Διαίρεση Προγράμματος σε Πολλαπλά Αρχεία(1)

- Οι προγραμματιστές συνήθως προσεγγίζουν ένα πρόβλημα με τη μέθοδο της **βηματικής εκλέπτυνσης** (***stepwise refinement***) όπου το πρόβλημα αναλύεται σταδιακά σε επιμέρους προβλήματα, αυτά με τη σειρά τους σε νέα επιμέρους προβλήματα κ.ο.κ. μέχρι να καταλήξουμε σε βήματα που είναι εύκολο να υλοποιηθούν.
- Ένα αρχείο μπορεί να περιέχει τις συναρτήσεις που ασχολούνται με την επίλυση ενός υπο-προβλήματος, και στη συνέχεια όλες οι σχετικές εντολές συνθέτουν μια βιβλιοθήκη.
- Γενικά η τεχνική της επιτυχημένης διαίρεσης απαιτεί τον προσεκτικό καθορισμό της διασύνδεσης του αρχείου (ή αρθρώματος ή βιβλιοθήκης) με τα υπόλοιπα προγράμματα έτσι ώστε:
  - το άρθρωμα να είναι εύκολα επαναχρησιμοποιήσιμο.
  - όλες οι σχετικές συναρτήσεις να βρίσκονται μαζί.
  - πιθανές μελλοντικές αλλαγές να επηρεάζουν το εσωτερικό του αρθρώματος και όχι τη διασύνδεση.
  - πιθανή νέα λειτουργικότητα να μην επηρεάζει την υπάρχουσα διασύνδεση.

Διαίρεση Προγράμματος σε Πολλαπλά Αρχεία (2)

- Σε κάθε αρχείο υλοποίησης αρθρώματος τα δεδομένα είναι οργανωμένα με συγκεκριμένο τρόπο. Τυπικά έχουμε:
- Αρχικά πιθανά **#define** σταθερών, **#include** άλλων αρχείων κεφαλίδας και δηλώσεις τύπων δεδομένων (**typedef**).
- Δηλώσεις καθολικών μεταβλητών. Οι καθολικές μεταβλητές μπορεί και να οριστούν εδώ.
- Μια ή περισσότερες συναρτήσεις.
- Η σειρά είναι σημαντική με την έννοια ότι όποια μεταβλητή ή συνάρτηση χρησιμοποιούμε πρέπει να δηλώνεται πρώτα. Για τις συναρτήσεις ισχύουν τα παρακάτω:
  - Αν η συνάρτηση ορίζεται και χρησιμοποιείται στο ίδιο αρχείο, ο ορισμός της συνάρτησης πρέπει να προηγείται της χρήσης.
  - Αν η συνάρτηση που χρησιμοποιείται ορίζεται σε άλλο αρχείο, πρέπει στο αρχείο που γίνεται η χρήση της συνάρτησης να συμπεριλαμβάνεται το αρχείο κεφαλίδας όπου δηλώνεται το πρωτότυπο της συνάρτησης.

Διαίρεση Προγράμματος σε Πολλαπλά Αρχεία (3)

- Για κάθε άρθρωμα δημιουργούμε δύο αρχεία.
- Αυτό με κατάληξη **.c** που περιλαμβάνει την υλοποίηση και αυτό με κατάληξη **.h** που περιλαμβάνει τις δηλώσεις: των συναρτήσεων (πρωτότυπα -prototypes), σταθερών, τύπων.
- Οταν κάποιο πρόγραμμα πρέπει να χρησιμοποιήσει μια συνάρτηση από το άρθρωμά μας, τότε το αρχείο με κατάληξη **.h** θα πρέπει να συμπεριληφθεί στο τμήμα με τα **#include** του προγράμματος.
- Το άρθρωμα είτε θα διατηρείται σε μορφή πηγαίου κώδικα (soure code) και θα μεταγλωττίζεται κάθε φορά ή θα βρίσκεται σε μορφή αντικείμενου κώδικα (object code) και θα συνδέεται μετά τη μεταγλώττιση του προγράμματος. Σε αυτή τη δεύτερη περίπτωση **ο αντικείμενος κώδικας μπορεί να φυλάσσεται σε ειδικό φάκελο αρθρωμάτων ή να βρίσκεται στον τρέχοντα φάκελο μεταγλώττισης**.

Σύνοψη

- Οι συναρτήσεις μιας βιβλιοθήκης γράφονται από **κατασκευαστές** (*implementers*) και καλούνται από **πελάτες** (*clients*). Στο σημείο στο οποίο "συναντώνται" οι πελάτες με τους κατασκευαστές ονομάζονται **διασυνδέσεις** (*interface*).
- Στη C οι διασυνδέσεις αναπαριστώνται με **αρχεία κεφαλίδας** (*header files*). Τα αρχεία κεφαλίδας που χρησιμοποιούνται ως διασυνδέσεις περιέχουν εκτενή τεκμηρίωση.
- Τα κριτήρια που πρέπει να πληρεί μια καλά σχεδιασμένη διασύνδεση είναι:
  - **Ενοποιημένο θέμα**. Οι συναρτήσεις μιας διασύνδεσης πρέπει να επιλεγούν με τέτοιο τρόπο ώστε να έχουν ένα ενοποιημένο θέμα. Πχ η **math.h** αποτελείται από μαθηματικές συναρτήσεις.
  - **Απλότητα και απόκρυψη πληροφοριών**. Περιορίστε όσο γίνεται το πλήθος των ορισμάτων (απλότητα) των συναρτήσεων. Μη αποκαλύπτετε τις λεπτομέρειες υλοποίησης
  - **Σταθερότητα**. Η διασύνδεση παραμένει σταθερή η υλοποίηση της μπορεί να μεταβάλλεται

Αρχεία κεφαλίδας - διασυνδέσεις (1)

Η βασική δομή μιας διασύνδεσης είναι

1. **#ifndef _*όνομα*_h**
2. **#define _ *όνομα*_h**
3. *Τυχόν απαραίτητες γραμμές #include*
4. *στοιχεία διασύνδεσης*
5. **#endif**

Όπου:

- *όνομα* είναι το όνομα της διασύνδεσης
- η ενότητα *γραμμές* ***#include*** χρησιμοποιείται μόνο αν η ίδια η διασύνδεση απαιτεί άλλες βιβλιοθήκες
- *στοιχεία διασύνδεσης* είναι τα πρωτότυπα των συναρτήσεων, οι σταθερές, και οι τύποι που εξάγονται από τη διασύνδεση

Για να διασφαλίσετε ότι ο μεταγλωττιστής θα διαβάσει μια διασύνδεση μόνο μια φορά, κάθε διασύνδεση θα πρέπει να περιλαμβάνει πριν από το πρώτο στοιχείο της τις δυο πρώτες γραμμές και στο τέλος του αρχείου διασύνδεσης τη γραμμή 5.

Αρχεία κεφαλίδας - διασυνδέσεις (2)

- Ένα παράδειγμα που δείχνει τη βηματική εκλέπτυνση του αρχικού κώδικα
- Αρχικός κώδικας hello.c

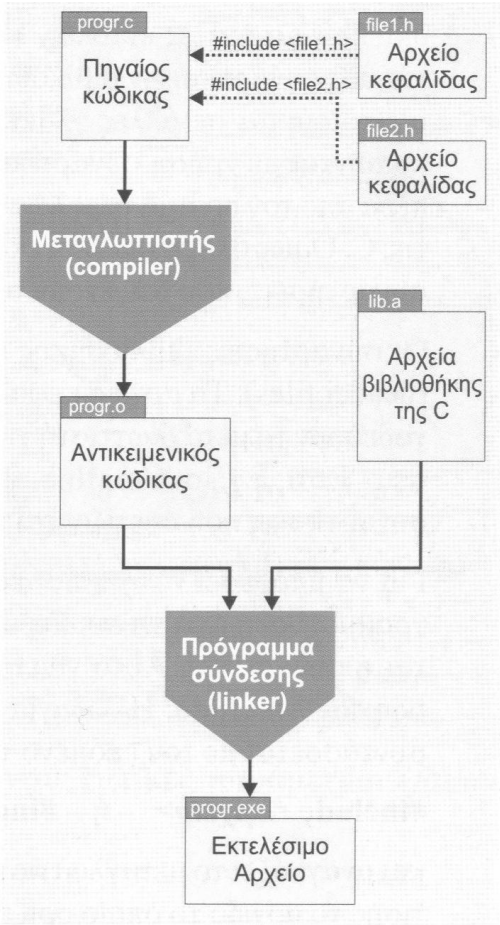
```
#include <stdio.h>
int main(int argc, char *argv[])
{
        // This is a VERY useful line.
        printf("hello, world\n");
        return 0;
}
```

Μεταγλώττιση & σύνδεση

Ο μεταγλωττιστής συμπεριλαμβάνει κατά τη μεταγλώττιση στον πηγαίο κώδικα όλα τα αρχεία συμπερίληψης (**#include**) (πχ **file1.h**, **file2.h**) και παράγει τον αντίστοιχο αντικειμενικό κώδικα (object code) (πχ **progr.o**)

Ο αντικειμενικός κώδικας είναι μεταγλωττισμένο σε γλώσσα μηχανής πηγαίο πρόγραμμα χωρίς όμως να συμπεριλαμβάνει και τον κώδικα των συναρτήσεων της βιβλιοθήκης που έχουν χρησιμοποιηθεί.

Το **πρόγραμμα σύνδεσης** (***linker***) προσθέτει στον αντικειμενικό κώδικα τον κώδικα των συναρτήσεων που χρησιμοποιήθηκαν. Ο κώδικας των συναρτήσεων βιβλιοθήκης (πχ. **lib.a**) βρίσκεται σε ένα ή περισσότερα αρχεία βιβλιοθήκης της C. Το λειτουργικό αποτέλεσμα του linker είναι ένα αυτόνομο εκτελέσιμο αρχείο (πχ **progr.exe**).



Αρχεία κεφαλίδας – διασυνδέσεις (1)

- Δημιουργία νέας συνάρτησης

hello.c

```
#include <stdio.h>
int main(int argc, char *argv[])
{
  // This is a VERY useful line.
      printf("hello, world\n");
   return 0;
}
```

```
#include <stdio.h>

void do_something_useful(void);

int main(int argc, char *argv[])
{
// Call the VERY useful function.
            do_something_useful();
            return 0;
}
void do_something_useful(void)
{
            printf("hello, world\n");
}
```

Αρχεία κεφαλίδας - διασυνδέσεις (2)

- Μεταφορά σε νέο αρχείο (1)

hello.c

```
#include <stdio.h>

void do_something_useful(void);

int main(int argc, char *argv[])
{
// Call the VERY useful function.
        do_something_useful();
        return 0;
}

void do_something_useful(void)
{
        printf("hello, world\n");
}
```

useful.h

```
void do_something_useful(void);
```

useful.c

```
#include <stdio.h>
void do_something_useful(void)
{
        printf("hello, world\n");
}
```

Αρχεία κεφαλίδας - διασυνδέσεις (3)

- Μεταφορά σε νέο αρχείο (2)

hello.c (***αρχείο πελάτης***)

```
#include "useful.h"

int main(int argc, char *argv[])
{
// Call the VERY useful function.
        do_something_useful();
        return 0;
}
```

useful.h (***διασύνδεση***)

```
void do_something_useful(void);
```

useful.c (***υλοποίηση διασύνδεσης***)

```
#include <stdio.h>

#include "useful.h"

void do_something_useful(void)
{
printf("hello, world\n");
}
```

Αρχεία κεφαλίδας –μεταγλώττιση & σύνδεση (1)

- Το αρχείο **hello.c** ενσωματώνει το αρχείο κεφαλίδας **stdio.h**, από το προκαθορισμένο φάκελο, και το αρχείο κεφαλίδας useful.h από τον φάκελο μεταγλώττισης.
- Η συνάρτηση **main()** καλεί τη συνάρτηση **do_something_useful()** που υλοποιείται στο αρχείο **useful.c** αλλά το πρωτότυπο της δηλώνεται στο αρχείο κεφαλίδας **useful.h**.
- Η μεταγλώττιση των αρχείων γίνεται είτε μέσω ενός Ολοκληρωμένου Περιβάλλοντος Προγραμματισμού (IDE)
  - Δημιουργείτε Project
  - Προσθέτετε στο project τα αρχεία σας (αρχεία κεφαλίδας, αρχεία υλοποίησης, αρχείο με τη main())
  - Ζητάτε **Rebuild All** ή **make** (με τη βοήθεια της οποίας γίνεται η μεταγλώττιση και σύνδεση).
- Μπορεί να γίνει και από τη κονσόλα (γραμμή εντολών) δίνοντας την εντολή

gcc -o helloAppl hello.c useful.c

Οπότε δημιουργείται το εκτελέσιμο αρχείο **helloAppl.exe**

Αρχεία κεφαλίδας –μεταγλώττιση & σύνδεση (2)

- Μεταγλώττιση (**compile**) & σύνδεση (**linking**) από την κονσόλα

1.gcc -c hello.c	μεταγλώττιση και δημιουργία του **hello.o**
2.gcc -c useful.c	μεταγλώττιση και δημιουργία του **useful.o**
3.gcc hello.o useful.o	σύνδεση (linking) των παραπάνω object files σε ένα εκτελέσιμο με το προκαθορισμένο όνομα **a.exe**

Ή σε μια εντολή μεταγλώττιση & σύνδεση πολλών αρχείων αντί για τις 1-3

4. **gcc hello.c useful.c** δημιουργείται το εκτελέσιμο **a.exe**

Εκτέλεση του **a.exe**

```
a
hello, world
Πιέστε ένα πλήκτρο για συνέχεια. . .
```

Δες το πχ στο φάκελο Hello (μεταγλώττιση, σύνδεση, εκτέλεση από κονσόλα /wxdevcpp

Ή αν θέλουμε πολλά αρχεία να μεταγλωττιστούν και να γίνει σύνδεση τους σε ένα εκτελέσιμο με άλλο όνομα εκτός από το προκαθορισμένο

gcc –o helloAppl hello.c useful.c

δημιουργείται το εκτελέσιμο **helloAppl.exe**

Μεταγλώττιση & σύνδεση ξεχωριστών αρχείων(1)

intro.c

```
#include <stdio.h>
#include <stdlib.h>

void book()
{
    printf ("H glossa C se bathos\n");
}
```

```
#include <stdio.h>
#include <stdlib.h>

void thanks(char *name)
{
   printf ("Thanks %s!\n", name);
}
```

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "myfunc.h"

int main()
{
  book();
  thanks("Nikos");
  system("pause");
  return 0;
}
```

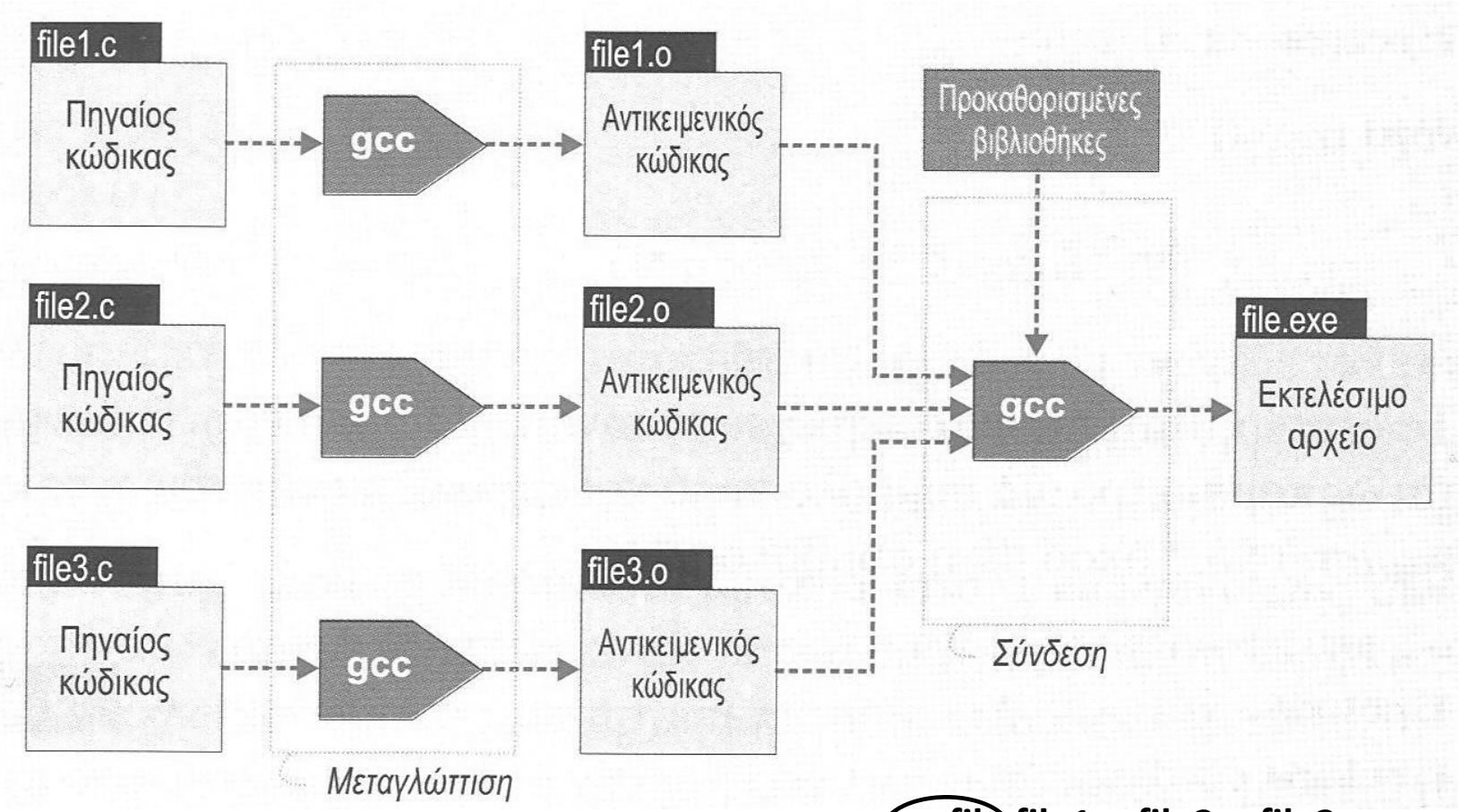
Δες το πχ στο φάκελο separate (μεταγλώττιση, σύνδεση, εκτέλεση από κονσόλα /wxdevcpp

Το εκτελέσιμο αρχείο gcc βρίσκεται στο φάκελο c:\dev-cpp\bin. Για να μπορούμε να το εκτελούμε απ' οποιοδήποτε φάκελο θα πρέπει να συμπεριλάβουμε στο path του περιβάλλοντος το συγκεκριμένο φάκελο

1. Η επιλογή **–c** εξαναγκάζει το μεταγλωττιστή να κάνει μόνο μεταγλώττιση (και όχι σύνδεση) του πηγαίου αρχείου
2. Την επιλογή **–o** ακολουθεί το όνομα του τελικού εκτελέσιμου αρχείου
3. Το **C:******* δηλώνει τη διαδρομή στον ΗΥ σας μέχρι το φάκελο separate

```
C:\>path %path%;c:\dev-cpp\bin
C:\>cd C:\*****\separate
C:\*****\separate>gcc -c main.c
C:\*****\separate>gcc -c intro.c
C:\*****\separate>gcc -c bye.c
C:\*****\separate>gcc -o all main.o intro.o bye.o
C:\*****\separate>all
H glossa C se bathos
Thanks Nikos!
Πιέστε ένα πλήκτρο για συνέχεια...
```

Μεταγλώττιση & σύνδεση ξεχωριστών αρχείων(2)



gcc -c file1.c παράγεται το **file1.o**

gcc –o file file1.o file2.o file3.o παράγεται το **file.exe**

Δημιουργία δικών μας βιβλιοθηκών

- Η δημιουργία και μεταγλώττιση προγραμμάτων που έχουν αναπτυχθεί σε πολλαπλά αρχεία είναι ένα βήμα πριν τη δημιουργία δικών μας βιβλιοθηκών.
- Η βασική διαφοροποίηση είναι ότι ενώ στη περίπτωση των πολλαπλών αρχείων όλα τα τμήματα του δικού μας προγράμματος μεταγλωττίζονται και επικαιροποιούνται κάθε φορά με την εκτέλεση της εφαρμογής make, στη περίπτωση των δικών μας βιβλιοθηκών, ένα τμήμα του προγράμματός μας (η βιβλιοθήκη μας) λαμβάνεται ως δεδομένη και απλά συνδέεται με τα υπόλοιπα αρθρώματα, ακριβώς όπως και οι πρότυπες βιβλιοθήκες.

Δημιουργία δικών μας βιβλιοθηκών (1ο πχ)(1)

- Αυτός είναι ο κώδικας της συνάρτησης που θα γίνει βιβλιοθήκη. Είναι μια απλή συνάρτηση που επιστρέφει το μέσο όρο δύο πραγματικών αριθμών.
- Το όνομα του αρχείου του πηγαίου κώδικα είναι **calc_mean.c**

```
#include <stdio.h>
	double mean(double a, double b)
	 {
		return (a+b) / 2;
	}
```

- Το αντίστοιχο αρχείο κεφαλίδας λέγεται **calc_mean.h**

```
double mean(double, double);
```

Δες το πχ στο φάκελο Lib1 (μεταγλώττιση, σύνδεση, εκτέλεση από κονσόλα /wxdevcpp

Δημιουργία δικών μας βιβλιοθηκών (1ο πχ)(2)

- Μια βιβλιοθήκη είναι βασικά ένα σύνολο από αρχεία αντικειμενικού κώδικα που έχουν μεταγλωττιστεί (ή αντιγραφεί) σε ένα ενιαίο αρχείο. Αυτό το ενιαίο αρχείο είναι η βιβλιοθήκη. Το αρχείο δημιουργείται με την εφαρμογή του **archiver (ar)**
- Πρώτα το αρχείο του πηγαίου κώδικα (υλοποίηση της διασύνδεσης) μεταγλωττίζεται σε αρχείο αντικείμενου κώδικα, όπως ακριβώς και στη περίπτωση των πολλαπλών αρχείων:

gcc -c calc_mean.c

- Κατόπιν, η καλείται η εφαρμογή **ar** για να παράγει τη βιβλιοθήκη με όνομα πχ. **libmean.a** με βάση το αρχείο του αντικείμενου κώδικα **calc_mean.o**.

ar rcs libmean.a calc_mean.o

- Το όνομα της βιβλιοθήκης ξεκινά με το πρόθεμα **lib** και έχει επέκταση **.a**. Είναι προφανές ότι στα δύο παραπάνω βήματα θα μπορούσαμε να έχουμε περισσότερα από ένα αρχεία πηγαίου ή αντικείμενου κώδικα ως είσοδο των εφαρμογών **gcc** και **ar** αντίστοιχα.

Χρήση της βιβλιοθήκης (1)

- Το όνομα του αρχείου του πηγαίου κώδικα είναι **main.c** (αρχείο πελάτης που χρησιμοποιεί τη βιβλιοθήκη).

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

int main(int argc, char *argv[]) {
      double v1, v2, m;
      v1 = 5.2;
      v2 = 7.9;
      m = mean(v1, v2);
     printf("The mean of %3.2f and %3.2f is %3.2f\n", v1, v2, m);
     return 0;
}
```

  - Σύνδεση με τη βιβλιοθήκη:

    gcc main.c libmean.a -o statically_linked

  - Δημιουργείται το εκτελέσιμο αρχείο **statically_linked.exe**

Χρήση της βιβλιοθήκης (2)

- Σύνδεση με τη βιβλιοθήκη :

gcc main.c libmean.a -o statically_linked

ή

gcc main.c -lmean -o statically_linked

- Η συντόμευση **-l** μπορεί να χρησιμοποιηθεί για να μη χρειάζεται να προσδιοριστεί πλήρως το όνομα της βιβλιοθήκης (πρόθεμα **lib** και επέκταση **.a**), δηλαδή γράφουμε αντί για ***lib*mean*.a*** -> -**lmean**

-**L*ibrary search path*** :

- Η επιλογή **-L** μπαίνει πριν από τη διαδρομή ***library search path*** που περιγράφει τον φάκελο που βρίσκεται το αντικειμενικό αρχείο της εξωτερικής βιβλιοθήκης (**.a**), εφόσον βέβαια δεν βρίσκεται είτε στο προκαθορισμένο φάκελο εγκατάστασης βιβλιοθηκών του gcc (πχ. c:\dev-cpp\lib φάκελος βιβλιοθηκών) είτε στον τρέχοντα φάκελο.

-l*include path* :

- Η επιλογή -I μπαίνει πριν από τη διαδρομή ***include path*** που περιγράφει τον φάκελο που βρίσκεται το αρχείο κεφαλίδας της εξωτερικής βιβλιοθήκης (.h), εφόσον βέβαια δεν βρίσκεται είτε στο προκαθορισμένο φάκελο εγκατάστασης αρχείων κεφαλίδας του gcc (πχ. c:\dev-cpp\include φάκελος αρχείων κεφαλίδας) είτε στον τρέχοντα φάκελο.

Δημιουργία δικών μας βιβλιοθηκών (2ο πχ)

intro.c

```
#include <stdio.h>
#include <stdlib.h>

void book()
{
    printf ("H glossa C se bathos\n");
}
```

bye,c

```
#include <stdio.h>
#include <stdlib.h>

void thanks(char *name)
{
   printf ("Thanks %s!\n", name);
}
```

test.c

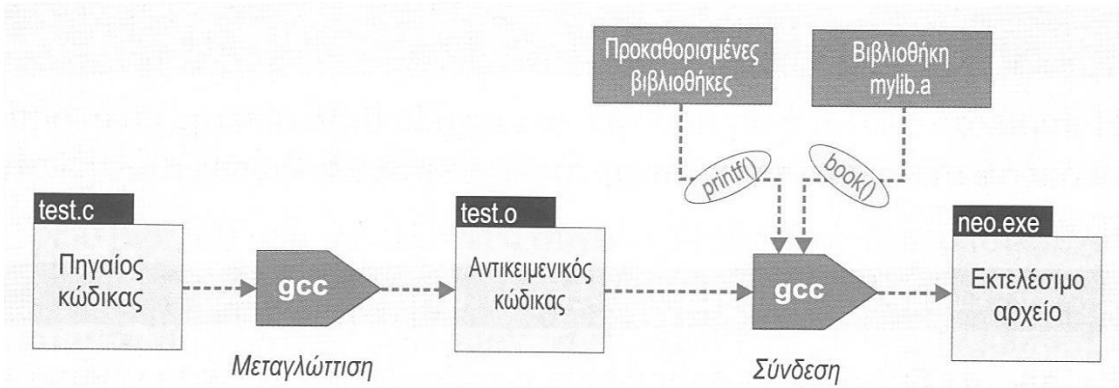
```
#include <stdio.h>
#include <stdlib.h>
#include "myfunc.h"

int main()
{
    int i;
    for (i=1;i<=5;i++) book();
    return 0;
}
```

myfunc.h

```
void book();
void thanks(char *name);
```

```
ar cr mylib.a intro.o bye.o
gcc test.c mylib.a -o neo
```

```
neo
H glossa C se bathos
H glossa C se bathos
H glossa C se bathos
H glossa C se bathos
H glossa C se bathos
```



gcc compiler

- Για μια πλήρη αναφορά στις επιλογές και στη χρήση του μεταγλωττιστή gcc, μπορείτε να ανατρέξετε στο δικτυακό τόπο

An Introduction to GCC- Network Theory Ltd

Εφαρμογή …

- Στο φάκελο ch08 σας δίνετε το αρχείο **random.c** που αποτελεί την υλοποίηση της διασύνδεσης **random.h** .
- Γράψτε το κώδικα για τη διασύνδεση **random.h**
- Το **crapts.c** χρησιμοποιεί τη διασύνδεση **random.h**. Δημιουργήστε το εκτελέσιμο αρχείο του **crapts.c**
  - Με εντολές από τη κονσόλα
  - Μέσω του περιβάλλοντος wx-devcpp
- Δημιουργήστε τη βιβλιοθήκη με όνομα **libmyrandom.a**
- Δημιουργήστε το εκτελέσιμο αρχείο από το **crapts.c** κάνοντας χρήση της βιβλιοθήκης **libmyrandom.a**
  - Με εντολές από τη κονσόλα
  - Μέσω του περιβάλλοντος wx-devcpp

*Αποθηκεύστε το αρχείο κεφαλίδας στο φάκελο **myincludedir** και τη βιβλιοθήκη **libmyrandom.a** στο φάκελο **mylibdir***

Παράμετροι γραμμής εντολής

- Οι συναρτήσεις στη C μπορούν να έχουν παραμέτρους μέσω των οποίων η συνάρτηση που τις καλεί μπορεί να μεταβιβάσει κάποιες πληροφορίες.
- **Τι γίνεται με τη συνάρτηση main() ;**
- **Μπορεί να έχει παραμέτρους;**
- **Πως θα μεταβιβάσουμε τιμές σ' αυτές τις παραμέτρους;**

Η συνάρτηση **main()** μπορεί να έχει ΜΟΝΟ ΔΥΟ συγκεκριμένες παραμέτρους οι οποίες παίρνουν τιμές από τη γραμμή εκτέλεσης του προγράμματος στο περιβάλλον του λειτουργικού συστήματος

Οι παράμετροι αυτοί είναι οι **arg argv[]**

main(int *argc*, char **argv[]*)

Παράμετροι της main(1)

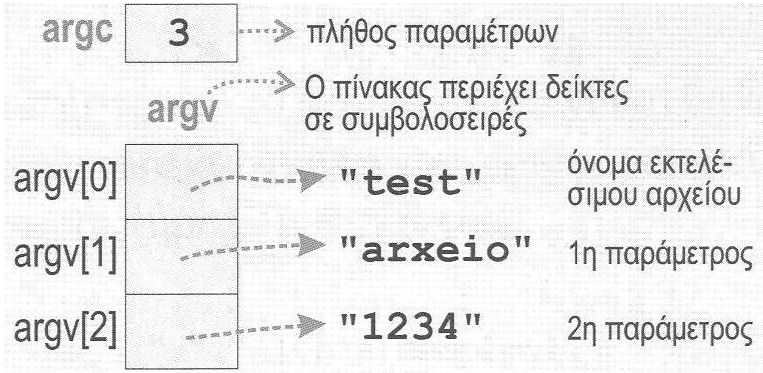
main(int argc, char *argv[])

- Η παράμετρος ***argc*** μεταβιβάζει στη main() το πλήθος των παραμέτρων με τις οποίες καλείται το πρόγραμμα. Η τιμή της argc είναι τουλάχιστον 1, δεδομένου ότι στο πλήθος των παραμέτρων υπολογίζεται και το όνομα του προγράμματος.
- Η παράμετρος ***argv[]*** είναι ένας πίνακας που περιέχει δείκτες σε char. Η πρώτη θέση μνήμης του πίνακα argv[] δείχνει σε ένα σύνολο χαρακτήρων το οποίο περιέχει το όνομα του εκτελέσιμου αρχείου του προγράμματος, όπως αυτό δίνεται στη γραμμή εντολών (π.χ. test). Οι υπόλοιποι δείκτες στις θέσεις μνήμης του πίνακα argv[], δείχνουν σε σύνολα χαρακτήρων που περιέχουν τις παραμέτρους με τις οποίες έχει κληθεί από τη γραμμή εντολών το πρόγραμμα.

Παράμετροι της main(2)– Ένα παράδειγμα

Έστω το παρακάτω πρόγραμμα **test.c**. Μετά τη μεταγλώττιση προκύπτει το εκτελέσιμο **test.exe**. Αν το εκτελέσουμε

```
test arxeio 1234
Edoses 3 parametrous
test
arxeio
1234
Πιέστε ένα πλήκτρο για συνέχεια. . .
```




```
#include <stdio.h>
#include <stdlib.h>

main(int argc,char *argv[])
{
		int i;
		printf("Edoses %d parametrous\n",argc);
		for(i=0;i<argc;i++)
		{
			//puts(argv[i]);
			printf("%s\n",argv[i]);
		}
		system("pause");
}
```

Δοκιμάστε τα πχ στο φάκελο main_parameters

1. To test.c
2. To 15_space. Το 15_space διαβάζει κείμενο από ένα αρχείο εισόδου και το γράφει σε ένα αρχείο εξόδου απαλείφοντας τα κενά διαστήματα. (στο wx-devcpp Execute->Parameters εισάγετε τις 2 παραμέτρους και μετά Run)

Τέλος Ενότητας

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης