Προγραμματισμός Ι

Πολλαπλά Αρχεία

Δημήτρης Μιχαήλ

Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο

Πολλαπλά Αρχεία

Όταν γράφουμε μεγάλα προγράμματα θέλουμε να έχουμε ανεξάρτητα κομμάτια κώδικα ξεχωριστά. Έτσι μπορούμε να κρατήσουμε τον κώδικα μας διαχειρίσιμο παρά το μέγεθος του.

Η C μας επιτρέπει να γράψουμε τον κώδικα μας σε διαφορετικά αρχεία τα οποία μεταγλωττίζονται ξεχωριστά.

Μετά την μεταγλώττιση το πρόγραμμα σύνδεσης (linker) είναι υπεύθυνο για την ένωση των διαφορετικών αρχείων σε ένα μεγάλο πρόγραμμα.

Καθολικές Μεταβλητές

Έχουμε ήδη συναντήσει τις έννοιες της κλάσης αποθήκευσης και της εμβέλειας.

Μεταβλητές που δηλώνονται έξω από οποιονδήποτε ορισμό συνάρτησης αναφέρονται ως *καθολικές μεταβλητές* και είναι της κλάσης αποθήκευσης static ως προεπιλογή.

Οι καθολικές μεταβλητές είναι προσπελάσιμες από

1. οποιαδήποτε συνάρτηση ορίζεται στο ίδιο αρχείο, μετά την δήλωση της μεταβλητής
2. από συναρτήσεις σε άλλα αρχεία.

Καθολικές Μεταβλητές

Για να προσπελάσουμε όμως μια καθολική μεταβλητή από ένα άλλο αρχείο πρέπει να δηλώσουμε την μεταβλητή αυτή και στο δεύτερο αρχείο.

π.χ

εαν ορίσουμε μια καθολική ακέραια μεταβλητή `flag` σε ένα αρχείο και το αναφέρουμε σε ένα δεύτερο αρχείο, το δεύτερο αρχείο πρέπει να περιέχει την δήλωση

```
extern int flag;
```

Λέει στον μεταγλωττιστή πως η μεταβλητή `flag` ορίζεται είτε αργότερα στο ίδιο αρχείο, είτε σε διαφορετικό αρχείο.

```
extern int flag;
```

Ο μεταγλωττιστής πληροφορεί το πρόγραμμα σύνδεσης ότι εμφανίζονται πρόωρες αναφορές προς την μεταβλητή `flag` στο αρχείο (ο μεταγλωττιστής δεν γνωρίζει που ορίζεται η `flag`, κι έτσι, αφήνει το πρόγραμμα σύνδεσης να προσπαθήσει να βρει την `flag`).

Εάν το πρόγραμμα σύνδεσης δεν μπορεί να εντοπίσει έναν ορισμό της `flag`, δημιουργείται ένα σφάλμα σύνδεσης, αλλιώς το πρόγραμμα σύνδεσης αντιστοιχεί τις αναφορές δηλώνοντας την θέση της `flag`.

Εμβέλεια Συναρτήσεων

Η εμβέλεια των συναρτήσεων μπορεί να επεκταθεί πέραν του αρχείου που ορίζεται με την χρήση αρχέτυπων.

Ένα αρχέτυπο συνάρτησης δηλώνει στον μεταγλωττιστή ότι η καθορισμένη συνάρτηση ορίζεται είτε στην συνέχεια το ίδιο αρχείο, είτε σε διαφορετικό αρχείο.

Ο μεταγλωττιστής αφήνει την αντιστοίχιση των κλήσεων της συνάρτησης στο πρόγραμμα σύνδεσης.

Αρχεία Επικεφαλίδας

Παράδειγμα της χρήσης αρχέτυπων συνάρτησης είναι οι συναρτήσεις `printf`() και `scanf`() που χρησιμοποιούμε διαρκώς.

Κάνοντας `#include` <stdio.h> δηλώνουμε με αρχέτυπα τις συναρτήσεις `printf`() και `scanf`(). Οι συναρτήσεις αυτές είναι ορισμένες σε άλλο αρχείο κώδικα, το οποίο έχει μεταγλωττιστεί ήδη από τον κατασκευαστή.

Ο μεταγλωττιστής όποτε βλέπει κλήση της `printf`() ενημερώνει το πρόγραμμα σύνδεσης. Αφού μεταγλωττιστεί ο κώδικας, το πρόγραμμα σύνδεσης είναι υπεύθυνο ώστε να αντιστοιχίσει τις κλήσεις αυτές με την πραγματική θέση της συνάρτησης.

Παράδειγμα

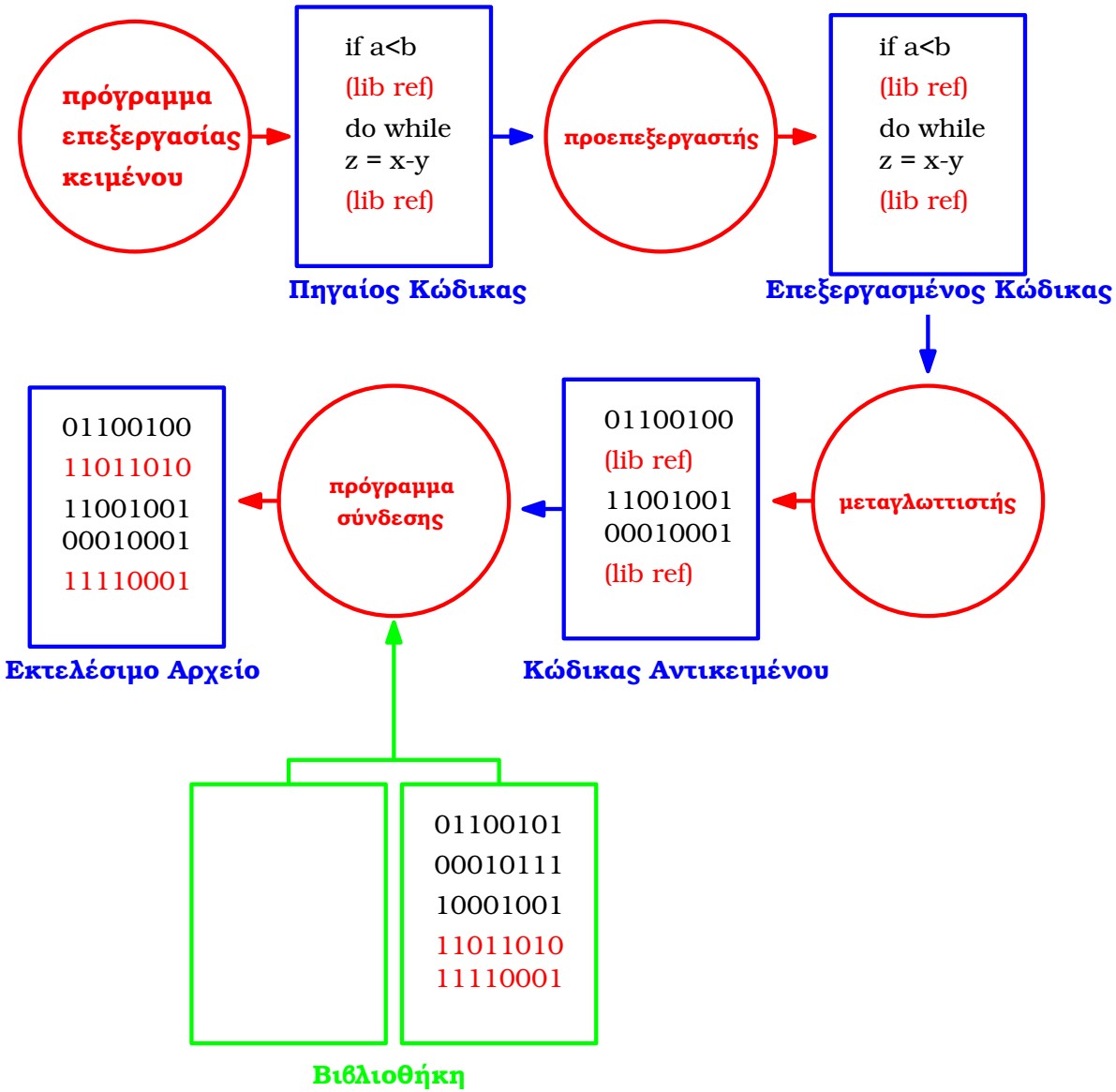
αρχείο **counter.c**

```c
#include <stdio.h>

int shared;

void increase( int i )
{
    shared = shared + i;
}

void init()
{
    shared = 0;
}
```

αρχείο **program.c**

```c
#include <stdio.h>

extern int shared;

void increase( int i );
void init();

main()
{
    int i;

    init();
    for( i = 0; i < 100; i++ )
        increase( 1 );

    printf("%d\n", shared);
}
```

Διαδικασία Μεταγλώττισης



Η μεταγλώττιση προγραμμάτων με πολλά αρχεία είναι πολύπλοκη διαδικασία.

Το προηγούμενο παράδειγμα που είδαμε θέλει τις εξής εντολές για να μεταγλωττιστή με το χέρι (σε περιβάλλον Unix με τον μεταγλωττιστή GCC).

```
gcc -c counter.c
```

Η παραπάνω εντολή παράγει το αρχείο `counter.o`. Στην συνέχεια τρέχουμε:

```
gcc program.c counter.o -o execfile
```

Η παραπάνω εντολή μεταγλωττίζει το πρόγραμμα `program.c` και μετά καλεί το πρόγραμμα σύνδεσης για να ενώσει τα δύο προγράμματα σε ένα.

Εργαλεία Μεταγλώττισης

Με το χέρι

Φανταστείτε ένα πρόγραμμα που αποτελείται από τα αρχεία `program.c` και `f1.c`, `f2.c` έως `fn.c` όπου $n = 100$.

Τότε θα έπρεπε να γράψουμε

```
gcc -c f1.c
gcc -c f2.c
...
gcc -c fn.c
gcc program.c f1.o f2.c f3.o ... fn.o -o program
```

Θα θέλαμε αυτή η διαδικασία να είναι αυτόματη.

Εργαλεία Μεταγλώττισης
IDE

Όλα τα σύγχρονα περιβάλλοντα προγραμματισμού (IDE - Integrated Development Environment) όπως:

- Eclipse
- Visual Studio
- KDevelop
- Dev-C++

μας παρέχουν την δυνατότητα να μεταγλωττίζουμε προγράμματα με πολλά αρχεία.

Μεταγλώττιση Πολλών Αρχείων

Η μεταγλώττιση πολλών αρχείων είναι σοβαρή υπόθεση γιατί:

1. όταν κάνουμε μια μικρή αλλαγή σε ένα αρχείο θέλουμε να μεταγλωττιστεί μόνο το αρχείο που άλλαξε και στην συνέχεια το πρόγραμμα σύνδεσης να δημιουργήσει το καινούριο πρόγραμμα,
2. αλλαγές σε ένα σημείο ενός προγράμματος, μέσω αλληλοεξάρτησης, μπορεί να επιφέρουν την ανάγκη μεταγλώττισης διαφόρων αρχείων.

Θέλουμε όλα αυτά να συμβαίνουν αυτόματα.

Μεταγλώττιση Πολλών Αρχείων

Σε περίπτωση όμως που έχουμε

- ένα μεγάλο σύνολο αρχείων
- σύνθετες εξαρτήσεις μεταγλώττισης μεταξύ των αρχείων

η διαδικασία μπορεί να αποδειχτεί επίπονη.

Χρειάζεται κάθε στιγμή να ξέρουμε:

- ποιά αρχεία είναι ενημερωμένα (ο αντικείμενος κώδικάς τους είναι έτοιμος για σύνδεση) αν συνδέσουμε ένα μη-ενημερωμένο αντικείμενο κώδικα θα προκύψει σφάλμα
- ονόματα αρχείων, επιλογές, ονόματα διαδρομής βιβλιοθηκών, κ.τ.λ
  - οι μακρές και σύνθετες εντολές μεταγλώττισης οδηγούν σε σφάλματα πληκτρολόγησης
  - η επαναληπτική διαδικασία κουράζει, λάθη που οδηγούν σε άσκοπο debugging

Μεταγλώττιση Πολλών Αρχείων

makefiles

Τα περισσότερα περιβάλλοντα προγραμματισμού παρέχουν ένα εργαλείο που λέγεται `make` το οποίο βοηθάει στην μεταγλώττιση προγραμμάτων που αποτελούνται από πολλά αρχεία.

Το πρόγραμμα `make` πέρνει οδηγίες μέσω ενός αρχείου με όνομα `makefile` (το οποίο το γράφει ο προγραμματιστής). Το αρχείο αυτό περιέχει κανόνες μεταγλώττισης.

Οι λεπτομέρειες του προγράμματος `make` εξαρτώνται από τον κατασκευαστή του προγράμματος.

- Μια ακολουθία από εντολές που περιγράφουν τον τρόπο που μεταγλωττίζεται το πρόγραμμά μας από τα επιμέρους αρχεία του
- Η ακολουθία μεταγλώττισης περιγράφεται και αποθηκεύεται σε ένα **makefile** και αποτελείται από δύο ειδών κανόνες
  - κανόνες εξάρτησης (dependency rules)
  - κανόνες οικοδόμησης (construction rules)
- Αν το αρχείο που ορίζουμε προς μεταγλώττιση δεν είναι ενημερωμένο (out of date), δηλαδή κάποια από τις εξαρτήσεις του έχει αλλάξει τότε εφαρμόζονται οι κανόνες μεταγλώττισης που ακολουθούν τον κανόνα εξάρτησης

Ένας κανόνας εξάρτησης έχει δύο μέρη (το αριστερό και το δεξί) τα οποία χωρίζονται με : .

- Το αριστερό μέρος (left side) δίνει το όνομα του στόχου (target), δηλαδή το όνομα του αρχείου προς οικοδόμηση
- Το δεξιό μέρος (right side) δίνει τα ονόματα των εξαρτήσεων (dependencies) τα ονόματα των αρχείων από τα οποία εξαρτάται ο στόχος π.χ., αρχεία πηγαίου κώδικα, αρχεία κεφαλής, αρχεία δεδομένων.

Εκτέλεση `makefile`

Έτσι για ένα τυπικό πρόγραμμα C, όταν η εφαρμογή make διατρέχει ένα makefile, εκτελούνται οι ακόλουθες εργασίες:

1. Η make διαβάζει το makefile
   - το makefile εξηγεί ποιά αρχεία αντικείμενου κώδικα και βιβλιοθήκες πρέπει να συνδεθούν
   - ποιά αρχεία πηγαίου κώδικα ή αρχεία κεφαλής πρέπει να μεταγλωττιστούν πριν τη σύνδεση
2. Για κάθε στόχο ελέγχονται η ημέρα και ώρα τροποποίησης και συγκρίνονται με τις ημέρες και ώρες τροποποίησης των εξαρτήσεων
   - αν κάποια εξάρτηση έχει μεταγενέστερη ημέρα και ώρα τροποποίησης τότε ο στόχος θεωρείται μη-ενημερωμένος
   - εφαρμόζονται οι αντίστοιχοι κανόνες οικοδόμησης.

Εκτέλεση `makefile`

1. Συγκρίνεται η ημέρα και ώρα τροποποίησης των εκτελεσίμων αρχείων σε σχέση με την ημέρα και ώρα τροποποίησης των στόχων
   - αν υπάρχουν στόχοι με μεταγενέστερη ημέρα και ώρα μεταγλωττίζονται ή συνδέονται για τη παραγωγή νέων εκτελεσίμων
2. Η εφαρμογή make υπακούει οποιαδήποτε εντολή που βρίσκεται στο makefile και αντιστοιχεί:
   - σε εντολή που συνδέεται με τη μεταγλώττιση προγραμμάτων
   - σε νόμιμη εντολή φλοιού UNIX/Linux ή DOS ανάλογα με το λειτουργικό

Δημιουργία `makefile`

GNU `make`

Για την δημιουργία ενός makefile αρκεί ένας απλός συντάκτης κειμένου.
Το makefile περιέχει απλά τους κανόνες εξάρτησης και οικοδόμησης.

```
main: main.o mystring.o
    gcc -o main main.o mystring.o

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

mystring.o: mystring.c
    gcc -c mystring.c
```

Προσοχή! Οι κανόνες οικοδόμησης σε νέα σειρά πρέπει να ξεκινούν με `TAB` και χωρίς άλλα κενά.

Η εφαρμογή make ερμηνεύει το makefile ως εξής:

- Ο στόχος main εξαρτάται από δύο αρχεία:
  - τα main.o και mystring.o
  - αν κάποιο από αυτά τα αρχεία αντικείμενου κώδικα έχουν τροποποιηθεί μετά τη σύνδεση του main, τότε απαιτείται επανασύνδεση
- Ο στόχος main.o εξαρτάται από δύο αρχεία:
  - τα header.h και main.c
  - αν κάποιο από αυτά τα αρχεία έχει τροποποιηθεί μετά τη μεταγλώττιση του main.o τότε απαιτείται επαναμεταγλώττιση
- Ο στόχος mystring.o εξαρτάται από ένα αρχείο:
  - το mystring.c
  - αν αυτό το αρχείο έχει τροποποιηθεί μετά τη μεταγλώττιση του mystring.o τότε απαιτείται επαναμεταγλώττιση

Άμεσοι και Έμμεσοι Κανόνες

Άμεσοι κανόνες (explicit rules) λέγονται οι κανόνες στους οποίους τα αρχεία εμφανίζονται με το πλήρες όνομα τους.

Μπορούμε να έχουμε και έμμεσους κανόνες (implicit rules) οι οποίοι γενικεύουν όμοιους κανόνες.

```
foo1.o: foo1.c
    gcc -c foo1.c

foo2.o: foo2.c
    gcc -c foo2.c

foo3.o: foo3.c
    gcc -c foo3.c
```

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

Με την χρήση του % μπορούμε να γράψουμε κανόνες που να ταιριάξουν περισσότερα από ένα αρχεία.

Το `$<` αντικαθίσταται με το όνομα του αρχείου πηγαίου κώδικα. Αντίστοιχα το `$@` αντικαθίσταται με το όνομα του στόχου.

Σχόλια και Άλλες Εντολές

Τα σχόλια σε ένα makefile ξεκινούν με τον χαρακτήρα `#`.

Η εφαρμογή make περιλαμβάνει

- ενσωματωμένες εντολές
- εντολές του φλοιού UNIX/Linux ή DOS
- δυνατότητες συντμήσεων
- κανονικών εκφράσεων
- δομών ελέγχου

Για τις λεπτομέρειες κοιτάξτε στο εγχειρίδιο της εφαρμογής. Προσοχή, ανάλογα με την υλοποίηση μπορεί να υπάρχουν μικροδιαφορές στην σύνταξη, π.χ τα makefiles της microsoft διαφέρουν από αυτά του GNU/Linux.

Μακροεντολές και Αντικατάσταση

Το περιβάλλον προγραμματισμού της make επιτρέπει και τον ορισμό μακροεντολών (ή μεταβλητών)

- χρησιμοποιούνται για την αποθήκευση ονομάτων αρχείων πηγαίου ή αντικείμενου κώδικα, διαδρομών καταλόγων βιβλιοθηκών, αρχείων κεφαλής, επιλογών μεταγλώττισης κλπ.

```
SOURCES = program.c foo1.c foo2.c
CFLAGS = –g –C
LIBS = –lm
PROGRAM = program
OBJECTS = (SOURCES: .c = .o)
```

Η τελευταία γραμμή είναι παράδειγμα αντικατάστασης όπου αλλάζουμε την κατάληξη των αρχείων της λίστας SOURCES από .c σε .o .

Χρήση Μακροεντολών

Η χρήση μακροεντολών γίνεται με την εντολή

```
$(macro name)
```

Υπάρχουν πολλές εσωτερικές μακροεντολές (δείτε το εγχειρίδιο χρήσης), π.χ:

- `$*` -- τό ονομα του αρχείου της τρέχουσας εξάρτησης, χωρίς τη κατάληξη (.c, .h, .o κλπ)
- `$@` -- το πλήρες όνομα αρχείου του τρέχοντος στόχου.
- `$<` -- αρχείο στόχου με κατάληξη .c.

Άλλο Παράδειγμα

```
OBJS =  main.o factorial.o hello.o
CFLAGS = -Wall -g
CC = gcc
INCLUDES =
LIBS = -lm

hello: ${OBJS}
    ${CC} ${CFLAGS} ${INCLUDES} -o $@ ${OBJS} ${LIBS}

clean:
    rm -f *.o core *.core

.c.o:
    ${CC} ${CFLAGS} ${INCLUDES} -c $<
```

Δημιουργία Βιβλιοθήκης

Μια βιβλιοθήκη είναι

- μια συλλογή από αρχεία αντικειμένων (object file) τα οποία έχουν ενωθεί σε ένα αρχείο
- ένα σύνολο από αρχεία επικεφαλίδας τα οποία παρέχουν τα ονόματα και τους τύπους που υπάρχουν στην βιβλιοθήκη αυτή.

Η ένωση πολλών αρχείων αντικειμένων σε ένα μπορεί να επιτευχθεί σε ένα σύστημα GNU/Linux με την χρήση του προγράμματος `ar`.

```
ar rcs  libfoo.a  foo1.o foo2.o foo3.o
```

Περισσότερες λεπτομέρειες εξαρτώνται από το λειτουργικό σύστημα.