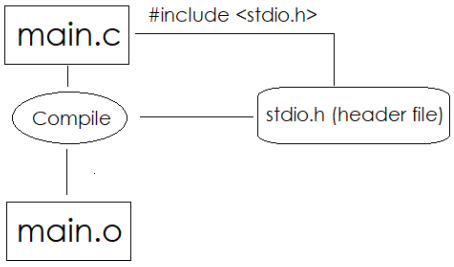
# Λειτουργικά Συστήματα: Εργαστήριο

*Μέρος Α:*

1.

Τα header files μας επιτρέπουν να γράφουμε δηλώσεις σε μια τοποθεσία και να τις μεταφέρουμε εκεί που θέλουμε με την βοήθεια των επικεφαλίδων. Συνήθως τα αρχεία επικεφαλίδων περιλαμβάνονται στο κύριο αρχείο μας μέσω του #include - της επικεφαλίδας δηλαδή. Όταν γίνεται μεταγλώττιση, όταν ο μεταγλωττιστής διαβάσει την επικεφαλίδα, γνωρίζει πως πρέπει να συμπεριλάβει και το περιεχόμενο του αρχείου της συγκεκριμένης επικεφαλίδας.

Τα αρχεία επικεφαλίδας καλό είναι, όταν συνδυάζονται με ένα αρχείο .c να έχουν το ίδιο όνομα. Έχουν suffix .h και ότι αφορά το περιεχόμενο τους καλό είναι αυτό να μην περιλαμβάνει ορίσματα μεταβλητών και συναρτήσεις.




Παραπάνω φαίνεται ο τρόπος με τον οποίο λειτουργεί ένα αρχείο main.c μαζί με ένα αρχείο επικεφαλίδας, μέσω του οποίου προστίθεται η βασική βιβλιοθήκη της C, όταν μεταγλωττίζεται και φτιάχνει το τελικό αρχείο main.o.

2.

Έγινε αντιγραφή αρχείων στον προσωπικό κατάλογο εργασίας μέσω της εντολής: cp zing.o zing.h directory destination(δηλαδή $HOME). Δημιουργήθηκε ένα

αρχείο main.c,όπως αυτό που παρουσιάζεται παρακάτω και ύστερα πραγματοποιήθηκε η σύνδεση των δύο αντικειμένων μέσα στο αρχείο Makefile.

```
zing: zing.o main.o
	gcc -o zing zing.o main.o
main.o: main.c
	gcc -Wall -c main.c
```

Στιγμιότυπο 1: Αρχείο Makefile

oslaba46@os-node1: ~

```
void zing(void);

int main(){

	zing();

  return 0;

}
```

Στιγμιότυπο 2: Αρχείο main.c

3.

Αρχείο Makefile (περιέχει μέσα όλες τις απαραίτητες συνδέσεις μεταξύ αρχείων, αλλά και τις μεταγλωττίσεις που απαιτούνται, το εκτελούμε μέσω της εντολής make):

oslaba46@os-node1: ~

```
all: zing zing2

zing: zing.o main.o
	gcc -o zing zing.o main.o
zing2: zing2.o main.o
	gcc -o zing2 zing2.o main.o
main.o: main.c
	gcc -Wall -c main.c
zing2.o: zing2.c
	gcc -Wall -c zing2.c
~
```

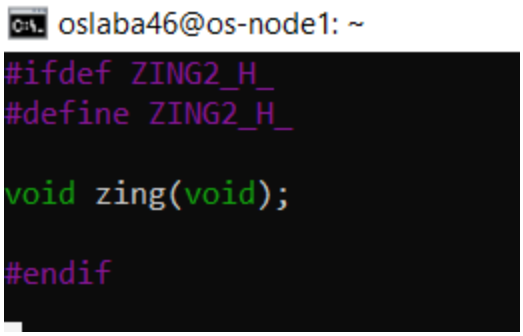
Στιγμιότυπο 3: Άρχείο Makefile

Νέο αρχείο με όνομα zing2.c, το οποίο χρησιμοποιεί την συνάρτηση zing στο εσωτερικό του και εκτυπώνει μήνυμα “Welcome to the World,oslaba46!”. Το παρόν αρχείο διαθέτει επικεφαλίδα, η οποία επισυνάπτεται παρακάτω. Το όνομα της επικεφαλίδας είναι zing2.h, δηλαδή διαθέτει το ίδιο όνομα με το αρχείο με suffix .h και καλεί στο εσωτερικό της την συνάρτηση zing, για να μπορεί να την χρησιμοποιήσει.

oslaba46@os-node1: ~

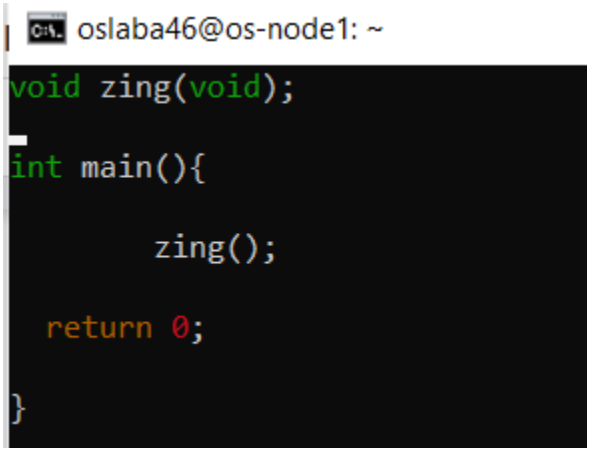
```c
#include "stdio.h"
#include "zing2.h"
#include "unistd.h"

void zing() {
        char *name;
        name=getlogin();
        if (!name)
                perror("getlogin() error");
        else
                printf("Welcome to the World, %s!\n",name);

}
```

Στιγμιότυπο 4:Αρχείο zing2.c



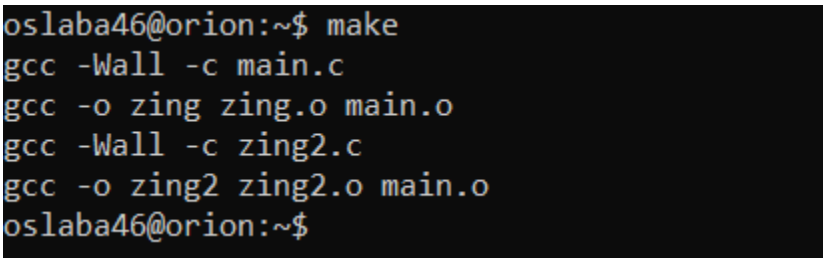

Στιγμιότυπο 5: Αρχείο επικεφαλίδας zing2.h

Το αρχείο της main παραμένει ίδιο με αυτό που είχαμε και στην δεύτερη άσκηση.




Στιγμιότυπο 6: Αρχείο main.c

Με την εντολή make γίνονται τα παρακάτω σύμφωνα με το Makefile:




Στιγμιότυπο 7:Αποτέλεσμα της εντολής make

4.

Η πιο γρήγορη και αποτελεσματική αντιμετώπιση για το πρόβλημα αυτό είναι η δημιουργία ενός Makefile, το οποίο θα περιέχει όλες τις εντολές που χρειαζόμαστε για την μεταγλώττιση. Το Makefile έχει την ιδιότητα να

εντοπίζει σε ποιο αρχείο έχουν πραγματοποιηθεί αλλαγές και να μεταγλωττίζει μόνο το συγκεκριμένο αρχείο και όσα συνδέονται με αυτό. Επομένως, σε περίπλοκα προγράμματα, όπου απαιτούνται πολλές κλήσεις συναρτήσεων κρίνεται απαραίτητη η δημιουργία ενός Makefile, ώστε ο χρόνος μεταγλώττισης να είναι μικρότερος και να μην παρουσιάζονται καθυστερήσεις.

5.

Αν θέλει κάποιος να συνδυάσει την μεταγλώττιση με την δημιουργία του εκτελέσιμου αρχείου μπορεί να μια γραμμή να εκτελέσει την εξής εντολή:

```
$ gcc [options] -o [output file]  [source file]

$ gcc   -Wall     -o      foo.c              foo.c
```

Επομένως, ο χρήστης σε αυτή την περίπτωση εκτέλεσε την δεύτερη εντολή συνδυάζοντας μεταγλώττιση με δημιουργία εκτελέσιμου αρχείου. Ο προγραμματιστής επεξεργαζόταν και έγραφε κώδικα στο αρχείο foo.c. Όταν έλαβε την απόφαση να μεταγλωττίσει και να φτιάξει το εκτελέσιμο αρχείο, ονόμασε το εκτελέσιμο αρχείο “foo.c”, όπως δηλαδή ονομάζεται το αρχείο source.Οπότε ο μεταγλωττιστής, αντικατέστησε εκείνο το αρχείο με το εκτελέσιμο, οπότε το αρχικό αρχείο foo.c καταστράφηκε, και για αυτόν τον λόγο δεν μπορεί να το βρει το αρχείο ξανά.

*Μέρος Β:*

Ο πηγαίος κώδικας της άσκησης είναι ο παρακάτω:

oslaba46@orion: ~

```c
#include <string.h>
#include <stdbool.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int read_open(char * file){
        int fr;
        fr = open(file, O_RDONLY);
        if (fr == -1){
                perror("open");
                exit(1);
        }
        return fr;
}

int write_open(char* file){
        int oflags = O_CREAT | O_WRONLY | O_TRUNC;
        int mode= S_IRUSR | S_IWUSR;
        int fd=open(file, oflags, mode);
        if(fd==-1){
                perror("open");
                exit(1);
        }
        return fd;
}

void read_write(int fr,int fr2,int fd){
        char buff[1024];
        ssize_t wcnt, rcnt;
        size_t len, idx;
        int c=1;
        for (;;){
                rcnt = read(fr,buff,sizeof(buff)-1);
                if (rcnt == 0){ /* end-of-file */
                        if((c++==1)&&(fr2!=-2)) {fr=fr2;}// condition in case of one read file
                        else break ;
                }
                if (rcnt == -1){ /* error */
                        perror("read");
                        exit(1);
                }
                buff[rcnt] = '\0';
                idx=0;
```

```
                len = strlen(buff);
                do {
                        wcnt = write(fd,buff + idx, len - idx);
                        if (wcnt == -1){ /* error */
                                perror("write");
                                exit(1);
                        }
                        idx += wcnt;
                } while (idx < len);

        }
}

int main(int argc, char **argv){
        if(argc<2||argc>4){
                perror("You have given none or too many arguments.");
                exit(1);
        }
        else if(argc==2){
                perror("You have only given one argument.");
                exit(1);
        }
        //In case of ABA ABB
        int fd;
        int ABA=0;
        int ABB=0;
        if (argc==4){
                if((strcmp(argv[1], argv[3]))==0) ABA=1;
                if((strcmp(argv[2], argv[3]))==0) ABB=1;
        }
        //This is the file to write to: identification fd
        //There are three cases, in which I will be using the fconc.out file
        if(argc==3||(ABA)||(ABB)) {fd=write_open("fconc.out");}
        else fd=write_open(argv[3]);

        //This is the file to read from:
        int fr=read_open(argv[1]);

        //This is the 2nd file to read from
        int fr2=read_open(argv[2]);

        //Read from the two(?) file and write in the third
        read_write(fr, fr2, fd);

        close(fd);
        close(fr);
        close(fr2);
```

```
        if((argc==4)&&ABA){ //printf("Caught3\n");
                fr=read_open("fconc.out");
                fd=write_open(argv[1]);
                read_write(fr,-2,fd);
                close(fr);
                close(fd);
        }
        if((argc==4)&&ABB){ //printf("Caught4\n");
                fr=read_open("fconc.out");
                fd=write_open(argv[2]);
                read_write(fr,-2,fd);
                close(fr);
                close(fd);
        }

        return 0;
}
```

Ένα παράδειγμα εκτέλεσης του παραπάνω προγράμματος, όπως δίνεται στην εκφώνηση:

```
oslaba46@orion:~$ ./fconc A
You have only given one argument.: Success
oslaba46@orion:~$ ./fconc A B
open: No such file or directory
oslaba46@orion:~$ echo "Goodbye," > A
oslaba46@orion:~$ echo "and thanks for all the fish!" > B
oslaba46@orion:~$ ./fconc A B
oslaba46@orion:~$ cat fconc.out
Goodbye,
and thanks for all the fish!
oslaba46@orion:~$ ./fconc A B C
oslaba46@orion:~$ cat C
Goodbye,
and thanks for all the fish!
oslaba46@orion:~$
```

Όταν εκτελείται η εντολή strace ./fconc A B C, το αποτέλεσμα είναι το παρακάτω:

```
oslaba46@orion:~$ strace ./fconc A B C
execve("./fconc", ["./fconc", "A", "B", "C"], [/* 19 vars */]) = 0
brk(0)                                  = 0x1518000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f64c1216000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=29766, ...}) = 0
mmap(NULL, 29766, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f64c120e000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\34\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1738176, ...}) = 0
mmap(NULL, 3844640, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f64c0c4d000
mprotect(0x7f64c0dee000, 2097152, PROT_NONE) = 0
mmap(0x7f64c0fee000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a1000) = 0x7f64c0fee000
mmap(0x7f64c0ff4000, 14880, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f64c0ff4000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f64c120d000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f64c120c000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f64c120b000
arch_prctl(ARCH_SET_FS, 0x7f64c120c700) = 0
mprotect(0x7f64c0fee000, 16384, PROT_READ) = 0
mprotect(0x7f64c1218000, 4096, PROT_READ) = 0
munmap(0x7f64c120e000, 29766)           = 0
open("C", O_WRONLY|O_CREAT|O_TRUNC, 0600) = 3
open("A", O_RDONLY)                     = 4
open("B", O_RDONLY)                     = 5
read(4, "Goodbye,\n", 1023)             = 9
write(3, "Goodbye,\n", 9)               = 9
read(4, "", 1023)                       = 0
write(3, "", 0)                         = 0
read(5, "and thanks for all the fish!\n", 1023) = 29
write(3, "and thanks for all the fish!\n", 29) = 29
read(5, "", 1023)                       = 0
close(3)                                = 0
close(4)                                = 0
close(5)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Στην παραπάνω φωτογραφία μας δίνεται η δυνατότητα να παρατηρήσουμε τις διάφορες κλήσεις συστήματος, που πραγματοποιούνται, όταν εκτελείται το πρόγραμμα μας και μπορούμε μέσω της συνάρτησης strace ./A B C να δούμε την ροή του προγράμματος μας. Πιο συγκεκριμένα, παρατηρούμε αρχικά δίνονται τρεις διαφορετικοί file descriptors για κάθε μια από τις κλήσεις συστήματος για το άνοιγμα του αρχείου C (με file descriptor 3), A (με file descriptor 4) και B (με file descriptor 5). Έπειτα παρατηρούμε τις κλήσεις συστήματος read και write, οι οποίες εξισώνονται με το πλήθος των χαρακτήρων κάθε φορά του εκάστοτε αρχείου. Επίσης, ως όρισμα σε αυτές τις συναρτήσεις χρησιμοποιείται ο file descriptor του εκάστοτε αρχείου που

χρειαζόμαστε. Τέλος φαίνεται πως κλείνουμε όλα τα αρχεία, που είχαμε ανοίξει στην αρχή του προγράμματος μας.

**Προαιρετικές ερωτήσεις:**

2.

Η εντολή callq δέχεται ως όρισμα την διεύθυνση της συνάρτησης, που καλεί. Επιπλέον, κάνει push στον stack, την διεύθυνση επιστροφής και κάνει jump στην διεύθυνση της συνάρτησης που κάλεσε. Συνοδεύεται αυτή η εντολή με το retq, το οποίο επαναφέρει την σωσμένη διεύθυνση. Άρα, σε αυτό το παράδειγμα, η main, όταν εκτελείται, καλεί την συνάρτηση zing, επομένως για αυτό αλλάζει το όρισμα του callq στο αρχείο zing.

4.

Αυτό που παρατηρούμε στην παρούσα άσκηση είναι ότι το directory έχει το ίδιο όνομα με το αρχείο. Όμως στα λειτουργικά συστήματα Unix, ισχύει πως “Everything is a file”, το οποίο συνεπάγεται στο ότι και οι διευθύνσεις είναι απλά ένας ειδικός τύπου φακέλου, το οποίο από την προοπτική του χρήστη περιέχει απλά και άλλους φακέλους. Επομένως, αν τόσο η διεύθυνση όσο και το αρχείο έχουν το ίδιο όνομα, το σύστημα, δεν θα μπορεί να ξεχωρίσει ποιο από τα δύο θέλουμε.