ΕΘΝΙΚΟ ΜΕΤΣΟΒΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ: ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΜΑΘΗΜΑ: Λειτουργικά Συστήματα

1η Εργαστηριακή Άσκηση

ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΜΕΛΩΝ :

ΣΤΑΜΑΤΗΣ ΑΛΕΞΑΝΔΡΟΠΟΥΛΟΣ (03117060)

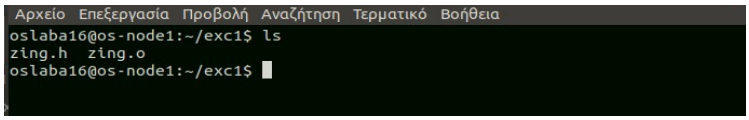
ΠΟΛΥΤΙΜΗ-ΑΝΝΑ ΓΚΟΤΣΗ (03117201)

ΑΘΗΝΑ

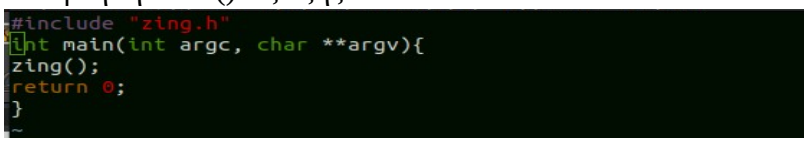
2020

Άσκηση 1.1

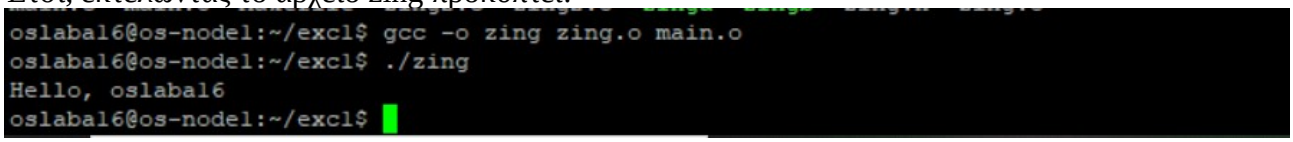
Στόχος του ερωτήματος αυτού ήταν η δημιουργία ενός εκτελέσιμου αρχείου (με όνομα zing) που καλεί τη συνάρτηση zing(). Η συνάρτηση αυτή δηλώνεται στο αρχείο zing.h. Επίσης, είναι διαθέσιμο το αρχείο αντικειμένων (object file) zing.o. Η διαδικασία που ακολουθήσαμε ήταν:
Τα αρχεία αυτά βρίσκονται στον κατάλογο /home/oslab/code/zing και αντιγράφηκαν στον κατάλογο home/oslab/oslaba16/exc1, χρησιμοποιώντας την εντολή: **cp /home/oslab/code/zing/ /home/oslab/oslaba16/exc1** .



Στην συνέχεια δημιουργήσαμε ένα αρχείο main.c (εντολή: vim main.c) όπου υλοποιήσαμε μια συνάρτηση main() ως εξής :



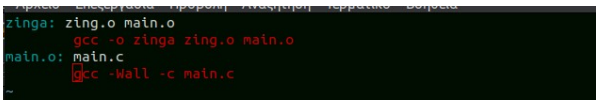
μέσω της οποίας με την εντολή ***gcc -Wall -c main.c*** δημιουργήσαμε το αρχείο αντικειμένων main.o. Η σύνδεση των αρχείων γίνεται τέλος με χρήση της εντολής *gcc -o zing zing.o main.o.*
Έτσι, εκτελώντας το αρχείο zing προκύπτει:



Όσον αφορά τον πηγαίο κώδικα,τη διαδικασία μεταγλώττισης και σύνδεσης καθώς και την έξοδο εκτέλεσης του προγράμματος του ερωτήματος 3, απαντώνται στο σχετικό ερώτημα.

1. Οι επικεφαλίδες (header files) αποτελούν αρχεία με την κατάληξη .h τα οποία ορίζουν την διεπαφή του προγράμματος στο οποίο δηλώνονται με άλλα κομμάτια κώδικα (API). (Στην c δηλώνονται (συνήθως στην αρχή του κώδικα) με την μορφή #include <headerfile.h> (για επικεφαλίδες του συστήματος που ανήκουν στην στάνταρ λίστα των directories του συστήματος) ή #include “headerfile.h” (για επικεφαλίδες που έχουν γραφτεί από τον χρήστη και βρίσκονται στο directory όπου περιέχεται το παρόν πρόγραμμα που εκτελείται.)) Οι επικεφαλίδες περιέχουν πρότυπα και δηλώσεις από συναρτήσεις και καθολικές μεταβλητές, χωρίς συνήθως να περιέχουν την υλοποίηση των συναρτήσεων, οι οποίες περιέχονται σε αντίστοιχο της επικεφαλίδας αρχείο με την κατάληξη .c. Έτσι, όταν στο main πρόγραμμά μας υπάρχει η δήλωση μιας επικεφαλίδας, ο preprocessor (προηγείται του compiler) συμπεριλαμβάνει στον κώδικα την πληροφορία (τα κομμάτια κώδικα) που ορίζεται μέσω των επικεφαλίδων.

2. Προκειμένου να επιτύχουμε την σύνδεση των δύο αρχείων αντικειμένων φτιάξαμε ένα αρχείο Makefile χρησιμοποιώντας την εντολή **vim Makefile** με περιεχόμενο:



Γράφουμε στο command line την εντολή make, με την εκτέλεση της οποίας εκτελούνται οι εντολές που περιέχονται στο αρχείο Makefile, με την σειρά με την οποία εμφανίζονται και στο command line:

```
Αρχείο  Επεξεργασία  Προβολή  Αναζήτηση  Τερματικό  Βοήθεια
oslaba16@os-node1:~/exc1$ make
gcc -Wall -c main.c
gcc -o zinga zing.o main.o
oslaba16@os-node1:~/exc1$
```

Έπειτα από την εντολή αυτή έχει παραχθεί το αρχείο αντικειμένων main.o και έχουν συνδεθεί τα δύο object files main.o και zing.o ενώ έχει δημιουργηθεί και το εκτελέσιμο αρχείο ***zinga.*** *Ύστερα από την εκτέλεση του αρχείου προκύπτει:*

```
Αρχείο  Επεξεργασία  Προβολή  Αναζήτηση  Τερματικό  Βοήθεια
oslaba16@os-node1:~/exc1$ make
gcc -Wall -c main.c
gcc -o zinga zing.o main.o
oslaba16@os-node1:~/exc1$ ./zinga
Hello, oslaba16
oslaba16@os-node1:~/exc1$
```

3. Για το 3ο ερώτημα του μέρους αυτού παράξαμε ένα δικό μας zing2.o, το οποίο θα περιέχει zing() που εμφανίζει διαφορετικό αλλά παρόμοιο μήνυμα με τη zing() του zing.o και συγκεκριμένα το "***Hi, oslaba16"***. Για το σκοπό αυτό, συμβουλευτήκαμε το manual page της getlogin(3), χρησιμοποιόντας την εντολή ***man 3 getlogin****, σύμφωνα με το οποίο η εντολή επιστρέφει ένα δείκτη σε ένα string που περιέχει ένα username που σχετίζεται με το user ID (εδώ oslaba16) της διαδικασίας*. Επίσης η συνάρτηση getlogin() περιέχεται στη βιβλιοθήκη <unistd.h>. Το zing2.c που δημιουργήσαμε είναι το εξής:

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

void zing(){
        char *name;
        name=getlogin();
        printf("Hi %s\n", name);
}
```

Επιπλέον, δημιουργήσαμε το header file zing2.h:

```
void zing();
```

Τέλος, τροποποιήσαμε την main ώστε να συμπεριλάβει και την επικεφαλίδα zing2.h. Νέα main:

```
#include "zing.h"
#include "zing2.h"
int main(int argc, char **argv){
zing();
return 0;
}
```

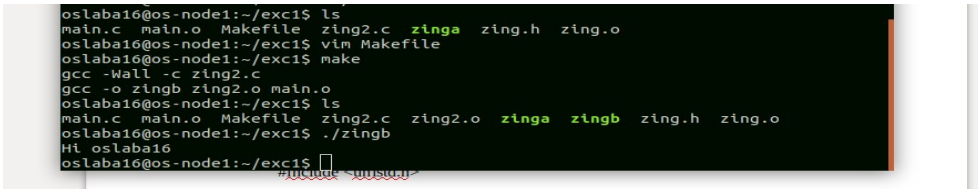
Έπειτα, αλλάξαμε το Makefile ώστε να παράγονται δύο εκτελέσιμα, ένα με το zing.o, ένα με το zing2.o, επαναχρησιμοποιώντας το κοινό object file main.o. Το τροποποιημένο Makefile είναι:

```
zingb:zing2.o main.o
        gcc -o zingb zing2.o main.o
zinga: zing.o main.o
        gcc -o zinga zing.o main.o
main.o: main.c
        gcc -Wall -c main.c
zing2.o: zing2.c
        gcc -Wall -c zing2.c
```

Γράφουμε στο command line την εντολή make και προέκυψαν τα ακόλουθα:

```
oslaba16@os-node1:~/exc1$ make
gcc -Wall -c zing2.c
gcc -Wall -c main.c
gcc -o zingb zing2.o main.o
oslaba16@os-node1:~/exc1$
```

Έπειτα από την εντολή αυτή έχει πραγματοποιηθεί μεταγλώττιση των zing2.c, main.c έχουν παραχθεί τα δύο object files main.o και zing.o έχει γίνει το linking και δημιουργήθηκε επιπλέον το εκτελέσιμο αρχείο ***zingb.*** *Ύστερα από την εκτέλεση του αρχείου προκύπτει:*



*Παρατηρούμε ότι τρέχοντας το zingb εκτυπώνεται στην οθόνη το μήνυμα "**Hi oslaba16"***

4. Ο λόγος που ο χρόνος μεταγλώττισης είναι μεγάλος είναι ότι το πρόγραμμα μας περιέχει πολύ μεγάλο αριθμό συναρτήσεων και άρα κάθε φορά που συμβαίνει μεταγλώττιση το μέγεθος του κώδικα που πρέπει να μεταγλωττιστεί είναι σημαντικό. Παρατηρούμε ωστόσο ότι, εφόσον πραγματοποιούμε αλλαγές μόνο σε μία συνάρτηση, όλος ο υπόλοιπος κώδικας του προγράμματος παραμένει ο ίδιος μετά από κάθε πραγματοποιούμενη τροποποίηση και άρα είναι περιττό να μεταγλωττίζεται κάθε φορά εκ νέου. Μπορούμε επομένως για να αντιμετωπίσουμε το πρόβλημα, αντί να κρατάμε των κώδικα όλων τον συναρτήσεων στο κύριο πρόγραμμα μας, να δημιουργήσουμε ξεχωριστά αρχεία .h και .c για κάθε συνάρτηση, ορίζοντας την δήλωση της συνάρτησης στο αντίστοιχο αρχείο-επικεφαλίδα και την υλοποίηση της στο σχετικό .h αρχείο. Έτσι, παράγοντας ένα Makefile αρχείο το οποίο να περιέχει τις εντολές μεταγλώττισης καθενός από τα αρχεία που περιέχουν τις υλοποιήσεις των συναρτήσεων και την μεταγλώττισης του main προγράμματος και την σύνδεση όλων των αρχείων αντικειμένων, και εκτελώντας την εντολή make, θα εκτελούνται κάθε φορά μόνο η μεταγλώττιση του αντίστοιχου αρχείου που περιέχει την συνάρτηση που τροποποιήσαμε και η εντολή για το linking του object file του κυρίου προγράμματος με τα object files όλων των συναρτήσεων. Έτσι ο χρόνος μεταγλώττισης είναι πλέον μικρός.

5. Ο συνεργάτης πληκτρολόγησε την εξής εντολή: gcc –Wall –o foo.c foo.c. Η εντολή αυτή έχει σε αυτή την περίπτωση την μορφή gcc –Wall –o [output file] [input file]. Επομένως, δίνεται ουσιαστικά εντολή να γίνει μεταγλώττιση και παραγωγή του object file του αρχείου εισόδου foo.c, ενώ ως output file ορίζεται το ίδιο αρχείο, άρα ο αρχικός κώδικας του foo.c αντικαθίσταται τώρα από την εκτελέσιμη μορφή του αρχείου, οπότε και περιέχει τον εκτελέσιμο κώδικα άρα κώδικα μηχανής σε μορφή κατανοητή από το λειτουργικό σύστημα που χρησιμοποιείται, αλλά μη αναγνώσιμη για εμάς.

Άσκηση 1.2

Στην άσκηση ζητήθηκε πρόγραμμα fconc το οποίο δέχεται δύο ή τρία ορίσματα και δημιουργεί ένα αρχείο (αρχείο εξόδου), τα περιεχόμενα του οποίου προκύπτουν από την συνένωση των περιεχομένων δύο αρχείων εισόδου.
Το πρόγραμμα γράφτηκε με την προτεινόμενη μορφή, επομένως με την χρήση δύο συναρτήσεων, doWrite και write_file από τις οποίες η πρώτη γράφει το περιεχόμενο ενός δοσμένου πίνακα (buff) στο αρχείο εξόδου , λαμβάνοντας ως ορίσματα τον file descriptor του αρχείου εξόδου, έναν δείτκη στον πίνακα από characters buff και το μήκος του περιεχομένου που πρέπει να γραφεί, ενώ η δεύτερη συνάρτηση δέχεται ως ορίσματα τον file descriptor του αρχείου εξόδου και έναν δείκτη στην αρχή του string του ονόματος του αρχείου εισόδου και διαβάζει το αρχείο εισόδου και αντιγράφει το περιεχόμενο του σε έναν buffer και καλεί την doWrite . Σημειώνεται ότι η write_file εκτελεί διαδοχικές αναγνώσεις μέχρι το τέλος του αρχείου διαβάζοντας κάθε φορά το μέγιστο όσα bytes της επιτρέπει το μέγεθος του buffer, αντιγράφοντας το περιεχόμενο του μέσω της doWrite και επαναλαμβάνοντας την ίδια διαδικασία.

Ο κώδικας, ο οποίος έχει συμπεριληφθεί στο παραδοθέν αρχείο που γράφηκε είναι ο εξής:

```c
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

//function to write the content of a given buffer to a given output file
void doWrite(int fd, const char *buff, int len) {
        int writevalue;
        int idx=0;
        do {
                writevalue=write(fd, buff+idx, len-idx);
                if (writevalue==-1) {   //an error has occurred
                        perror("write");    //print descripitve error message
                        exit(1);
                }
                idx+=writevalue;
        }
        while (idx<len);        //loop until the end of the buffer's content
        return;
}
//function to read given input file using a buffer and call doWrite to write to the given output file:
void write_file(int fd, const char *infile) {
        int inf;        //file descriptor of input file will be saved here
        inf=open(infile, O_RDONLY);     //input file opened for read only
        if (inf==-1) {          //if an error occurs, open returns -1
                perror(infile);         //print descriptive error message
                exit(1);
        }
        ssize_t value;
        char buff[1024];
        for (;;) {      //loop until the whole file is read
                value=read(inf, buff, sizeof(buff)-1);  //read fron input file, as many bytes as the buffer size -1 allows
                if (value==0) break;    //the whole file is read/EOF is encountered
                if (value==-1) {        //an erroe has occurred
                        perror("read");
                        exit(1);
                }
                buff[value]='\0';       //end of string character (marks end of words to be coppied from thw buffer)
                int len=strlen(buff);
                doWrite(fd, buff, len);         //write content of buffer to output file
        }
        close(inf);
        return;
}
```

```c
int main (int argc, char **argv) {
        if (argc<3 || argc>4) {
                printf("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)\n");
                return 1;
        }
        char **infile=argv;

        //open (or create and open) output file:
        char *outfile;          //outfile will keep the name of the output file
        if (argc==4) {                  //outfile is the given output file
                outfile=argv[3];
        }
        else outfile="fconc.out";       //if no output file is given, outfile becomes fconc.out
        int outf, oflags, mode;         //create variable of file descriptor of output file, create and set mode and flags for fconc.out
        oflags=O_APPEND | O_CREAT | O_WRONLY | O_TRUNC;
        mode= S_IRUSR | S_IWUSR;
        outf=open(outfile, oflags, mode);       //if no output file is given outfile="fconc.out" is created (O_CREAT)

        write_file(outf, infile[1]);    //call write_file function to copy the first and the second file to the output file
        write_file(outf, infile[2]);
        close(outf);
        return 0;
}
```

Εκτελώντας τον κώδικα:

```
oslabal6@os-node1:~/excl/ex1_2$ ./fconc
Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)
oslabal6@os-node1:~/excl/ex1_2$ ./fconc A
Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)
oslabal6@os-node1:~/excl/ex1_2$ ./fconc A B
oslabal6@os-node1:~/excl/ex1_2$ cat fconc.out
This is team
oslabal6!
oslabal6@os-node1:~/excl/ex1_2$ ./fconc A B C
oslabal6@os-node1:~/excl/ex1_2$ cat C
This is team
oslabal6!
oslabal6@os-node1:~/excl/ex1_2$ ./fconc A B C D
Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)
```

Εκτελώντας τώρα ένα παράδειγμα του fconc χρησιμοποιώντας την εντολή strace προκύπτει:

```
oslabal6@os-node1:~/excl/ex1_2$ strace ./fconc A B C
execve("./fconc", ["./fconc", "A", "B", "C"], [/* 18 vars */]) = 0
brk(0)                                  = 0x1641000
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) = 0x7f7c9cbd2000
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=30952, ...}) = 0
mmap(NULL, 30952, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f7c9cbca000
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) = 0x7f7c9c609000
mprotect(0x7f7c9c7aa000, 2097152, PROT_NONE) = 0
mmap(0x7f7c9c9aa000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a1000) = 0x7f7c9c9aa000
mmap(0x7f7c9c9b0000, 14880, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f7c9c9b0000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f7c9cbc9000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f7c9cbc8000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f7c9cbc7000
arch_prctl(ARCH_SET_FS, 0x7f7c9cbc8700) = 0
mprotect(0x7f7c9c9aa000, 16384, PROT_READ) = 0
mprotect(0x7f7c9cbd4000, 4096, PROT_READ) = 0
munmap(0x7f7c9cbca000, 30952)           = 0
open("C", O_WRONLY|O_CREAT|O_TRUNC|O_APPEND, 0600) = 3
open("A", O_RDONLY)                     = 4
read(4, "This is team\n", 1023)         = 13
write(3, "This is team\n", 13)          = 13
read(4, "", 1023)                       = 0
close(4)                                = 0
open("B", O_RDONLY)                     = 4
read(4, "oslabal6!\n", 1023)            = 10
write(3, "oslabal6!\n", 10)             = 10
read(4, "", 1023)                       = 0
close(4)                                = 0
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
oslabal6@os-node1:~/excl/ex1_2$
```

(Η εντολή strace καταγράφει και τυπώνει όλα τα system calls που πραγματοποιούνται από μία διεργασία καθώς και τα signals που δημιουργούνται από αυτή).