ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ

# Λειτουργικά Συστήματα Υπολογιστών
# 1ο Εργαστήριο

Ομάδα 36
Αναστάσιος Λαγός - el13531
Κωνσταντίνος Βασιλάκης - el16504

## Ασκηση 1.1

**1.** Η επικεφαλίδα περιέχει δηλώσεις συναρτήσεων που υπάρχουν σε κάποιο αρχείο πηγαίου κώδικα και χρησιμοποιείται σαν διεπαφή μεταξύ άλλων αρχείων πηγαίου κώδικα που χρησιμοποιούν αυτές τις συναρτήσεις. Έτσι τα αρχεία μπορούν να γίνουν compile χωρίς να έχουν την πλήρη συνάρτηση η θέση της οποία συμπληρώνεται κατά την διαδικασία του linking.

**2.** Έχουμε τα παρακάτω αρχεία

**main.c**

```
1 #include "zing.h"
2
3 int main() {
4     zing();
5     return 0;
6 }
```

**zing.h**

```
1 #ifndef ZING_H__
2 #define ZING_H__
3
4 void zing(void);
5
6 #endif
```

**Makefile**

```
1 zing: zing.o main.o
2         gcc main.o zing.o -o zing
3
4 main.o: main.c
5         gcc -Wall -c main.c
```

Στο παραπάνω Makefile στις γραμμές 3,4 ορίζεται η εντολή για να γίνει compile το main.c σε main.o. Στις γραμμές 1,2 ορίζεται η εντολή για να γίνουν link τα zing.o που δίνεται και το main.o που έγινε compile στο προηγούμενο βήμα και να παραχθεί το εκτελέσιμο zing.

Η έξοδος του εκτελέσιμου είναι

**./zing**

```
1 Hello, oslaba36
```

**3.** Έχουμε τα παρακάτω αρχεία (το main.c και zing.h είναι ίδια με το προηγούμενο ερώτημα)

**main.c**

```c
#include "zing.h"

int main() {
    zing();
    return 0;
}
```

**zing.h**

```c
#ifndef ZING_H__
#define ZING_H__

void zing(void);

#endif
```

**zing2.c**

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

void zing () {
    printf("Goodbye, %s\n", getlogin());
    return;
}
```

**Makefile**

```
all: zing zing2

zing2: zing2.c main.o
        gcc main.o zing2.c -o zing2

zing: zing.o main.o
        gcc main.o zing.o -o zing

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

Στο παραπάνω Makefile οι γραμμές 6 εώς 10 είναι όπως το προηγούμενο ερώτημα και ορίζουν την δημιουργία του εκτελέσιμου zing. Στις γραμμές 3,4 ορίζεται η δημιουργία του zing2 που χρησιμοποιεί το main.o που έχει ήδη δημιουργηθεί μαζί με το καινούριο zing2.c που περιέχει τον διαφοροποιημένο κώδικα. Οι γραμμές 1,2 τρέχουν τις

επόμενες εντολές και δημιουργούν τα δύο εκτελέσιμα zing και zing2 που ζητούνται.

Η έξοδος των εκτελέσιμων είναι

**./zing**

```
Hello, oslaba36
```

**./zing2**

```
Goodbye, oslaba36
```

**4.** Μπορούμε να "σπάσουμε" τις συναρτήσεις σε μικρότερες ομάδες σε διαφορετικά αρχεία και να τις συμπεριλάβουμε με κατάλληλα header files. Χρησιμοποιώντας ένα make file τότε κατά την δημιουργία του εκτελέσιμου κάθε φορά θα ξαναμεταγλωττίζονται μόνο τα αρχεία που έχουν αλλαγές.

**5.** Με αυτήν την εντολή το foo.c μεταγλωττίστικε σε ένα αρχείο με το ίδιο όνομα οπότε έκανε overwrite το αρχικό.

## Ασκηση 1.2

**1.** Ο κώδικας της άσκησης δίνεται παρακάτω.

**fconc.c**

```c
#include "fileIO.h"
#include <stdio.h>
#include <stdlib.h>

//This is a driver program for the functions writeFile and iRead
//The goal is to concatenate 2 text files into an output file
//The idea is read all the files one by one into separate,
   dynamically allocated buffers in a buffer array
//then write those buffers to the output file.
//Works if given a file, both as input and output!

int main(int argc, char *argv[]) {
    if (argc != 3 && argc != 4) {   //Usage error
        printf("Usage: %s inputfile1 inputfile2 [outputfile](
           optional)\n", argv[0]);
        exit(-1);
    }
```

```
    char *outputFilename;
    if (argv[3] == NULL) {
        outputFilename = "fconc.out";   //Default outputFilename
        writeFile(outputFilename, argv, argc);
    }
    else {
        outputFilename = argv[3];
        writeFile(outputFilename, argv, argc - 1);
    }

    return 0;
}
```

### fileIO.c

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stddef.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>

void iRead (const char *infile, char *buff[], int idx) {
    int fds;
    off_t fileSize = 0;
    ssize_t rcnt;
    fds = open(infile, O_RDONLY);
    if (fds == -1) {        //Opening error
            perror("Error while opening: ");
            close(fds);
            exit(-1);
        }

    //Get file size
    fileSize = lseek(fds, 0L, SEEK_END);            //Set the file potitional indicator at the end of file with offset 0
    lseek(fds, 0L, SEEK_SET);                       //Set the indicator at the start of the file with offset 0
    //Allocate memory for buffer
    buff[idx] = (char*) malloc(fileSize * sizeof(char));  //Allocate memory equal to the file size
    if (buff[idx] == NULL) {perror("Allocation error: "); close(fds); exit(-1);}
    // Read the file
    do {
        rcnt = read(fds, (void*) buff[idx], fileSize);
        if (fds == -1) {perror("Error while reading: "); close(fds); exit(-1);}
    } while(rcnt != 0);
    close(fds);

    return;
```

```
}

void writeFile (const char *outputFilename, char *inputFilename[],
    int numberOfFiles) {
     char **buffer;
     buffer = (char**) malloc((numberOfFiles - 1) * sizeof(char*));
        //Array of strings for each input file
     ssize_t wcnt;

     //All the files are read in a separate buffer
     for (int i = 1; i <= numberOfFiles - 1; i++) {
         iRead(inputFilename[i], buffer, i - 1);
     }

     //A new file is opened to write the data. If it does not exist
        it is created. If it exists and contains
     //data the data are deleted.
     int fdd = open(outputFilename, O_WRONLY | O_CREAT | O_TRUNC,
        S_IRUSR | S_IWUSR);

     //We write the data of each buffer to the file. We also include
         a do while during each write in order to continue
     //from the same position in case a system call temporarily
        interrupts the write process.
     for (int i = 0; i < numberOfFiles - 1; i++) {
         int idx = 0;
         int len = strlen(buffer[i]);
         do {
             wcnt = write(fdd, buffer[i] + idx, len - idx);
             if (wcnt == -1) {perror("Error while writing: "); close
                (fdd); exit(-1);}
             idx += wcnt;
         } while (idx < len);
             free(buffer[i]);
     }
     close(fdd);
     free(buffer);

     return;
}
```

### fileIO.h

```
#ifndef FILEIO_H__
#define FILEIO_H__

void iRead (const char *infile, char *buff[], int idx);

void writeFile (const char *outputFilename, char *inputFilename[],
    int numberOfFiles);

#endif
```

### Makefile

```
fconc: fconc.c fileIO.o
        gcc -Wall -std=c11 fconc.c fileIO.o -o ../fconc

fileIO.o: fileIO.c
        gcc -Wall -c -std=c11 fileIO.c
```

### Έξοδος της εντολής strace

Η έξοδος της strace από το σημείο που αρχίζει η πρώτη open του κώδικα δίνεται παρακάτω. Παρατηρούμε ότι για κάθε αρχείο έχουμε ένα open μετά lseek για να βρούμε το μέγεθος του και να θέσουμε τον δείκτη πάλι στην αρχή του, το read για το διάβασμα του αρχείου και στο τέλος close για να το κλείσουμε. Στην γραμμή 13 ανοίγουμε το αρχείο το οποίο θα γράψουμε, απο κάτω ακολουθούν τα αντίστοιχα write και στο τέλος ένα close για να το κλείσουμε.

```
open("C", O_RDONLY)                     = 3
lseek(3, 0, SEEK_END)                   = 410
lseek(3, 0, SEEK_SET)                   = 0
read(3, "If you're visiting this page, yo"..., 410) = 410
read(3, "", 410)                        = 0
close(3)                                = 0
open("C", O_RDONLY)                     = 3
lseek(3, 0, SEEK_END)                   = 410
lseek(3, 0, SEEK_SET)                   = 0
read(3, "If you're visiting this page, yo"..., 410) = 410
read(3, "", 410)                        = 0
close(3)                                = 0
open("C", O_WRONLY|O_CREAT|O_TRUNC, 0600) = 3
write(3, "If you're visiting this page, yo"..., 410) = 410
write(3, "If you're visiting this page, yo"..., 410) = 410
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

## Bonus

**1.** Τρέχοντας την strace πάνω στην strace πχ με

```
strace strace ls
```

ή

```
strace strace echo test
```

βλέπουμε ότι η strace καλεί την ptrace που είναι system call. Αυτή είναι η κλήση συστήματος που επιτρέπει στην strace να ελέγξει και να παρακολουθήσει με την σειρά της η εκάστοτε εντολή ποιες κλήσεις συστήματος χρησιμοποιεί.

**2.** Αυτή η αλλαγή έγινε κατά την διαδικασία του linking όπου ο linker πλέον αντικατέστησε την πραγματική διεύθυνση την συνάρτησης zinq.

**3.** Το fileIO θα είναι το ίδιο με αυτό της Άσκησης 1.2 αφού αυτό μπορεί να διαχειριστεί αυθαίρετο αριθμό αρχείων αφού οι buffer δημιουργούνται δυναμικά και η write τρέχει για όλους τους διαθέσιμους buffers. Η μόνη διαφορά θα είναι στην συνάρτηση main του αρχείου fconc.c έτσι ώστε να περάσουμε αυθαίρετο αριθμό ορισμάτων που περιέχουν τα ζητούμενα ονόματα των αρχείων. Ο κώδικας δίνεται παρακάτω

**fconc.c**

```c
#include "fconc.h"
#include <stdio.h>
#include <stdlib.h>

//This is a driver program for the functions writeFile and iRead.
//The goal is to concatenate N text files into an output file.
//The idea is read all the files one by one into separate, dynamically allocated buffers in a buffer array
//then write those buffers to the output file.
//Works if given a file, both as input and output!
//Will always write to the last file given as argument.

int main(int argc, char *argv[]) {
    if (argc < 3) {   //Usage error
        printf("Usage: %s inputfile1 inputfile2 ... inputfileN [outputfile](required)\n", argv[0]);
        exit(-1);
    }

    char *outputFilename;
    outputFilename = argv[argc - 1];

    printf("Created/overwritten file: %s\n", outputFilename);
    writeFile(outputFilename, argv, argc - 1);

    return 0;
}
```

Η παραπάνω συνάρτηση περνάει όλα τα ορίσματα από το δεύτερο (το πρώτο αγνοείται αφού είναι το όνομα της συνάρτησης) εώς το προτελευταίο στην αντίστοιχη

συνάρτηση σαν ονόματα αρχείων που θα διαβαστούν και το τελευταίο το χρησιμοποιεί για όνομα του αρχείου εξόδου.

**4.** Τρέχοντας την strace για το whoops έχουμε τις παρακάτω εντολές. (Δίνεται το τέλος της εξόδου της strace που μας ενδιαφέρει).

```
open("/etc/shadow", O_RDONLY)            = -1 EACCES (Permission
    denied)
write(2, "Problem!\n", 9Problem!
)                = 9
exit_group(1)                            = ?
+++ exited with 1 +++
```

Παρατηρούμε ότι το πρόγραμμα προσπαθεί να ανοίξει το αρχείο /etc/shadow που περιέχει του κωδικούς των χρηστών του συστήματος και επειδή ο χρήστης που είμαστε συνδεδεμένοι δεν έχει τα απαραίτητα δικαιώματα το πρόγραμμα τυπώνει το μήνυμα λάθους.