# Λειτουργικά Συστήματα

## Άσκηση 1: Εισαγωγή στο περιβάλλον προγραμματισμού


**Ομάδα:** oslabd43
**Ο/Ε:** Αντώνης Παπαοικονόμου 03115140
Γιάννης Πιτόσκας 03115077


### 1.1 Σύνδεση με αρχείο αντικειμένων

*Πηγαίος κώδικας `main.c`:*

```
#include "zing.h"
#include "zing2.h"
#include <stdio.h>

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

*Διαδικασία μεταγλώττισης και σύνδεσης:*

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

*Έξοδος εκτέλεσης του προγράμματος:*

```
oslabd43@orion:~/Askisi_1$ ./zing
Hello, oslabd43
```

**Ερωτήσεις:**

1. ***Ποιο σκοπό εξυπηρετεί η επικεφαλίδα;***

Η επικεφαλίδα (header) αρχικά είναι ένα αρχείο το οποίο περιέχει δηλώσεις συναρτήσεων (function declaration) δίχως να εμπεριέχεται η υλοποίησή τους. Πιο συγκεκριμένα ένα header file εξυπηρετεί στο να γνωστοποιεί την ύπαρξη μιας συνάρτησης, το όνομά της, πόσες παραμέτρους δέχεται, τι τύπου είναι η εκάστοτε παράμετρος καθώς και τι τύπο επιστρέφει. Έτσι, επιτυγχάνεται μάλιστα ο διαχωρισμός της διεπαφής (interface) από την υλοποίηση (implementation) . Ακόμη, οι headers μειώνουν το χρόνο μεταγλώττισης (compilation time), αφού προκειμένου να γίνει compile ένα αρχείο που χρησιμοποιεί μια συνάρτηση, απαιτείται μόνο η δήλωση της (η οποία έχει γίνει στην επικεφαλίδα).

2. ***Ζητείται κατάλληλο Makefile για τη δημιουργία του εκτελέσιμου της άσκησης.***

Το `Makefile` για την δημιουργία του εκτελέσιμου `zing` είναι το παρακάτω και μπορεί να εκτελεστεί χρησιμοποιώντας την εντολή `make`:

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

3. ***Παράξτε το δικό σας zing2.o, το οποίο θα περιέχει zing() που θα εμφανίζει διαφορετικό αλλά παρόμοιο μήνυμα με τη zing() του zing.o. Συμβουλευτείτε το manual page της getlogin(3). Αλλάξτε 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 `zing2.c` περιέχει τα εξής:

```
#include <stdio.h>
#include <unistd.h>
#include "zing2.h"

void zing(){
	printf("We are the team: %s. TRELO!\n", getlogin());
}
```

όπου το header file `zing2.h` απλά περιέχει τη δήλωση της `zing()` ως εξής:

```
void zing();
```

Τέλος, αλλάζουμε και το `Makefile` ώστε να παράγονται τα δύο εκτελέσιμα `zing` και `zing2`, οπού το πρώτο παράγεται με το `zing.o` και το δεύτερο με το `zing2.o`, χρησιμοποιώντας το κοινό object file `main.o` όπως και ζητείται. Η νέα `Makefile` φαίνεται παρακάτω :

```
all: zing zing2

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

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

zing2.o: zing2.c
	gcc -Wall -c zing2.c

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

clean:
	rm zing
	rm zing2
```

4. ***Έστω ότι έχετε γράψει το πρόγραμμά σας σε ένα αρχείο που περιέχει 500 συναρτήσεις. Αυτή τη στιγμή κάνετε αλλαγές μόνο σε μία συνάρτηση. Ο κύκλος εργασίας είναι: αλλαγές στον κώδικα, μεταγλώττιση, εκτέλεση, αλλαγές στον κώδικα, κ.ο.κ. Ο χρόνος μεταγλώττισης είναι μεγάλος, γεγονός που σας καθυστερεί. Πώς μπορεί να αντιμετωπισθεί το πρόβλημα αυτό;***

Προκειμένου να αντιμετωπίσουμε αυτό το πρόβλημα πρέπει να εκμεταλλευτούμε το γεγονός ότι οι αλλαγές γίνονται μόνο σε μία συνάρτηση και το περιεχόμενο των υπολοίπων μένει αναλλοίωτο. Μπορούμε λοιπόν να δημιουργήσουμε ένα αρχείο που θα περιέχει τις δηλώσεις των συναρτήσεων στις οποίες δε θέλουμε να επέμβουμε, καθώς και ένα αρχείο που θα περιέχει την υλοποίηση της εκάστοτε συνάρτησης που δε θέλουμε να πειράξουμε. Κάνουμε λοιπόν compile το αρχείο με τις συναρτήσεις και χρησιμοποιούμε την επικεφαλίδα για να μπορούμε εργαστούμε με την συνάρτηση στην οποία θέλουμε να επέμβουμε, και έτσι κερδίζουμε πάρα πολύ σε compilation time (γλιτώνουμε όλες τις υπόλοιπες συναρτήσεις).

5. ***Ο συνεργάτης σας και εσείς δουλεύατε στο πρόγραμμα foo.c όλη την προηγούμενη εβδομάδα. Καθώς κάνατε ένα διάλειμμα και ο συνεργάτης σας δούλευε στον κώδικα, ακούτε μια απελπισμένη κραυγή. Ρωτάτε τι συνέβει και ο συνεργάτης σας λέει ότι το αρχείο foo.c χάθηκε! Κοιτάτε το history του φλοιού και η τελευταία εντολή ήταν η: gcc -Wall -o foo.c foo.c Τι συνέβη;***

Η δομή της παραπάνω εντολής είναι: `gcc -Wall -o output_file input_file`
Ουσιαστικά, δόθηκε η εντολή στον gcc να γράψει το αποτέλεσμα στο αρχείο με όνομα foo.c το οποίο όμως περιέχει τον πηγαίο κώδικα. Έτσι, το αρχείο που περιείχε τον πηγαίο κώδικα αντικαταστάθηκε από το εκτελέσιμο μετά την εκτέλεση της εντολής.

## 1.2 Συνένωση δύο αρχείων σε τρίτο

*Πηγαίος κώδικας fconc.c:*

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

const int buff_size = 1024;

void doWrite(int fd, const char *buff, int len){
        size_t idx = 0;
        ssize_t wcnt;
        do {
                wcnt = write(fd,buff + idx, len - idx);
                if (wcnt == -1){ /* error */
                        perror("write");
                        exit(1);
                }
                idx += wcnt;
        } while (idx < len);
}

void write_file(int fd, const char *infile, int fd_in){
        char buff[buff_size];
        ssize_t rcnt;
        for (;;){
                rcnt = read(fd_in,buff,sizeof(buff)-1);
                if (rcnt == 0) /* end-of-file */
                        break;
                if (rcnt == -1){ /* error */
                        perror("read");
                        exit(1);
                }
                doWrite(fd, buff, rcnt);
        }
        close(fd_in);
}

int main(int argc, char **argv){
        char* output = "fconc.out"; /* default output file name */
        if(argc == 1 || argc == 2 || argc >=5){ /* incorrect call of program message */
                printf("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
        }
        else if (argc >= 3){
                if (argc == 4){ /* optional output file name as third input */
                        output = argv[3];
                }
                int flags1 = O_RDONLY;
                int fd_infile1 = open(argv[1], O_RDONLY);
                int fd_infile2 = open(argv[2], O_RDONLY);
                if (fd_infile1 == -1 || fd_infile2 == -1){
                        perror("open");
                        exit(1);
                }
                if (fd_infile1 != -1 && fd_infile2 != -1){ /* both files can be opened */
                        int fd_outfile, oflags, mode, new_fd;
                        int done = 0;
                        int input = 1;
                        oflags = O_CREAT | O_WRONLY | O_TRUNC;
                        if ((strcmp(argv[1], output) == 0) || (strcmp(argv[2], output) == 0)){ /* exception if input
file(s) == output file */
                                done = 1;
                                flags1 = O_RDWR;
                                if (strcmp(argv[1], output) == 0){
                                        input = 1;
                                }
                                if (strcmp(argv[2], output) == 0){
                                        input = 2;
                                }
```

```
                                printf("Warning: input file: %s has been used as an output file and is overwitten!\n",
output);
                                output = "dummy.in";
                                oflags = O_CREAT | O_RDWR | O_TRUNC;
                        }
                        mode = S_IRUSR | S_IWUSR;
                        fd_outfile = open(output, oflags, mode);
                        if (fd_outfile == -1){
                                perror("open");
                                exit(1);
                        }
                        else { /* output file can also be opened */
                                write_file(fd_outfile, argv[1], fd_infile1);
                                write_file(fd_outfile, argv[2], fd_infile2);
                                close(fd_outfile);
                                if (done == 1){
                                        fd_outfile = open(output, O_RDONLY, mode);
                                        new_fd = open(argv[input], flags1, mode);
                                        write_file(new_fd, output, fd_outfile);
                                        close(new_fd);
                                }
                        }
                }
        }
        return 0;
}
```

**Ερωτήσεις:**

1. ***Εκτελέστε ένα παράδειγμα του fconc χρησιμοποιώντας την εντολή strace. Αντιγράψτε το κομμάτι της εξόδου της strace που προκύπτει από τον κώδικα που γράψατε.***

Η strace μας δείχνει όλα τα system calls που γίνονται κατά την εκτέλεση του κώδικα μας. Έτσι εκτελώντας την παρακάτω εντολή παίρνουμε ως έξοδο:

```
oslabd43@orion:~/Askisi_1$ strace ./fconc A B C
execve("./fconc", ["./fconc", "A", "B", "C"], [/* 20 vars */]) = 0
brk(0)                                  = 0xcb5000
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) = 0x7f38ba698000
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) = 0x7f38ba690000
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) = 0x7f38ba0cf000
mprotect(0x7f38ba270000, 2097152, PROT_NONE) = 0
mmap(0x7f38ba470000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a1000) = 0x7f38ba470000
mmap(0x7f38ba476000, 14880, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f38ba476000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f38ba68f000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f38ba68e000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f38ba68d000
arch_prctl(ARCH_SET_FS, 0x7f38ba68e700) = 0
mprotect(0x7f38ba470000, 16384, PROT_READ) = 0
mprotect(0x7f38ba69a000, 4096, PROT_READ) = 0
munmap(0x7f38ba690000, 29766)           = 0
open("A", O_RDONLY)                     = 3
open("B", O_RDONLY)                     = 4
open("C", O_WRONLY|O_CREAT|O_TRUNC, 0600) = 5
read(3, "Goodbye,\n", 1023)             = 9
write(5, "Goodbye,\n", 9)               = 9
read(3, "", 1023)                       = 0
close(3)                                = 0
read(4, "and thanks for all the fish!\n", 1023) = 29
write(5, "and thanks for all the fish!\n", 29) = 29
read(4, "", 1023)                       = 0
close(4)                                = 0
close(5)                                = 0
```

```
exit_group(0)                           = ?
+++ exited with 0 +++
```