

Λειτουργικά Συστήματα  
6ο Εξάμηνο  
**Άσκηση 1 - Αναφορά**

Χαρδούβελης Γεώργιος-Ορέστης  
el15100  
5ο Εξάμηνο

Κυριάκου Αθηνά  
el17405  
5ο Εξάμηνο

### **Άσκηση 1.1**

Το πρώτο βήμα της άσκησης είναι η αντιγραφή αρχείων `zing.h` και `zing.o` στον κατάλογο εργασίας μας. Για να γίνει αυτό έπρεπε πρώτα να εισέλθουμε από το `/home/oslab/oslabd14` στον επιθυμητό κατάλογο ως εξής:

```
$ cd /home/oslab/code/zing
```

Ύστερα αντιγράψαμε τα κατάλληλα αρχεία στον κατάλογο μας ως εξής:

```
$ cp zing.h zing.o /home/oslab/oslabd14
```

Με την εντολή αυτή αντιγράφουμε ταυτόχρονα και τα δύο αρχεία στο αρχείο με μονοπάτι `/home/oslab/oslabd14`.

Για την δημιουργία του αρχείου `main.o`, δημιουργήσαμε καταρχάς το αρχείο `main.c`. Ο **πηγαίος κώδικας** στο αρχείο αυτό είναι ο εξής:

```
#include "zing.h"
```

```
int main(int argc, char **argv){  
    zing();  
    return 0;  
}
```

Έχει γίνει `include` το header file `zing.h` δεδομένου ότι χρησιμοποιούμε την συνάρτηση `zing` που ορίζεται στο αντίστοιχο header. Έτσι, εξασφαλίζεται επιτυχής compilation ώστε να προκύψει το object file `main.o`. Συγκεκριμένα, η **μεταγλώττιση** γίνεται με την εντολή:

```
$ gcc -Wall -c main.c
```

Το **linking** των 2 αρχείων `main.o` και `zing.o` γίνεται με την εντολή:

```
$ gcc main.o zing.o -o zing
```

Έτσι προκύπτει το εκτελέσιμο αρχείο. Λόγω του flag `-o` το αρχείο αυτό έχει όνομα `zing`.

Η **έξοδος του προγράμματος** `zing` είναι:

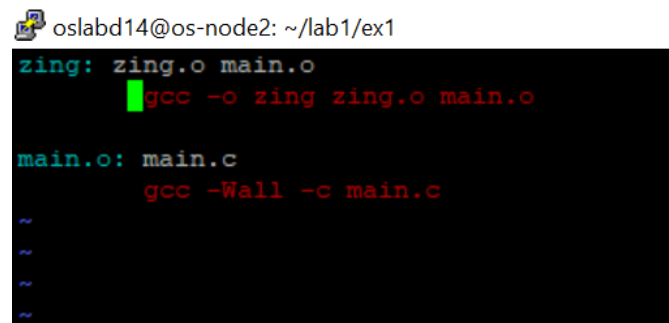
```
Hello, oslabd14
```

## Ερωτήσεις:

1) Στο παρόν πρόγραμμα η επικεφαλίδα `zing.h` συμπεριλαμβάνεται στο αρχείο `main.c` ώστε να πραγματοποιηθεί επιτυχώς το compilation και να προκύψει το αρχείο `main.o`.

Γενικότερα, με τις επικεφαλίδες ορίζουμε βιβλιοθήκες με συναρτήσεις που θέλουμε να χρησιμοποιήσουμε. Οι συναρτήσεις αυτές δηλώνονται στο header file (αρχείο με κατάληξη `.h`), ενώ ο κώδικάς τους βρίσκεται σε ένα ή περισσότερα αρχεία `.c`. Στο `.c` αρχείο του κύριου προγράμματος μας (πχ `main.c`), όπου χρησιμοποιούνται οι συναρτήσεις, συμπεριλαμβάνουμε το header file της βιβλιοθήκης με `#include "name.h"` για να μπορέσει να γίνει το compilation του σε object file (πχ `main.o`).

2) Το αρχείο Makefile έχει ως εξής:



```
oslabd14@os-node2: ~/lab1/ex1
zing: zing.o main.o
      gcc -o zing zing.o main.o
main.o: main.c
      gcc -Wall -c main.c
~
~
~
~
```

3) Ο κώδικας του αρχείου `zing2.c` έχει ως εξής:

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

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

```
void zing(){
    char *name;
    name=getlogin();
    if(!name)
        perror("getlogin() error");
    else
        printf("Answer me NOW %s\n",name);
}
```

Για τη συνάρτηση `zing` της οποίας ο κώδικας είναι στο αρχείο `zing2.c` δε δημιουργήθηκε καινούργιο header file. Στον κώδικα του κύριου προγράμματος (`main.c`) συμπεριλήφθηκε το ήδη υπάρχον `zing.h`, εφόσον η δήλωση της συνάρτησης `zing` είναι ίδια. Αλλάζει ωστόσο ο κώδικάς της.

Το καινούργιο Makefile γίνεται ως εξής:

```
oslabd14@os-node2: ~/lab1/ex1
all: zing zing2

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

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

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

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

~
~
```

4) Το παρόν πρόβλημα μπορεί να λυθεί αν ο κώδικας κάθε συνάρτησης είναι σε ξεχωριστό .c αρχείο ακόμα και αν είναι όλες έχουν δηλωθεί στο ίδιο header file. Έτσι, η μεταγλώττιση θα γίνεται κάθε φορά μόνο στο .c αρχείο της συνάρτησης που τροποποιήθηκε (για να προκύψει το νέο .o αρχείο της συνάρτησης) και θα γίνεται εκ νέου το linking με το .o αρχείο του κυρίου προγράμματος ώστε τελικά να προκύψει το νέο εκτελέσιμο. Έτσι μειώνεται ο χρόνος μεταγλώττισης, εφόσον η μεταγλώττιση δε γίνεται σε όλο των κώδικα.

Για παράδειγμα, τροποιώντας τον κώδικα στο zing2.c αρχείο και χρησιμοποιώντας την εντολή make, έχουμε:

```
oslabd14@os-node2: ~/lab1/ex1$ ls
main.c  main.o  Makefile  zing  zing2  zing2.c  zing2.o  zing.h  zing.o
oslabd14@os-node2:~/lab1/ex1$ vim zing2.c
oslabd14@os-node2:~/lab1/ex1$ make
make: Warning: File 'zing2.c' has modification time 43 s in the future
gcc -Wall -c zing2.c
gcc -o zing2 zing2.o main.o
make: warning: Clock skew detected. Your build may be incomplete.
oslabd14@os-node2:~/lab1/ex1$
```

5) Με την εντολή αυτή το αρχείο foo.c μετατρέπεται σε εκτελέσιμο αρχείο. Όμως, λόγω της σημαίας -o, ο binary κώδικας του εκτελέσιμου αποθηκεύεται στο ήδη υπάρχον αρχείο foo.c, με αποτέλεσμα να χάνεται ο πηγαίος κώδικας της συνάρτησης.

## Άσκηση 1.2

Ο παρακάτω κώδικας στο κύριο αρχείο της άσκησης fconv.c έχει ως εξής:

```
oslabd14@os-node2: ~/lab1/ex2
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include "rwfile.h" //function to read and write

int main(int argc, char *argv[]){

    int ifd1,ifd2,ofd;
    mode_t fileperms=S_IRUSR|S_IWUSR;

    //inputs
    if(argc>4 || argc<3){
        printf("Usage: ./fconv infile1 infile2 [outfile (default:fconv.out)]\n");
        exit(1);
    }

    ifd1=open(argv[1],O_RDONLY);
    if(ifd1==-1){
        perror("A");
        exit(1);
    }

    ifd2=open(argv[2],O_RDONLY);
    if(ifd2==-1){
        perror("B");
        if(close(ifd1)==-1)
            perror("A");
        exit(1);
    }

    ofd=open("fconv.out",O_RDWR|O_CREAT|O_TRUNC,fileperms);

    if(ofd==-1){
        perror("Output file");
        if(close(ifd1)==-1)
            perror("A");
        if(close(ifd2)==-1)
            perror("B");
        exit(1);
    }

    //input1 to output
    rwfile(ifd1,ofd);

    //input2 to output
    rwfile(ifd2,ofd);

    //closing and return(0)
    int Eflag=0;
    if(close(ifd1)==-1){
        perror("close");
        Eflag=1;
    }
    if(close(ifd2)==-1){
        perror("close");
        Eflag=1;
    }

    if(close(ofd)==-1){
        perror("close");
        Eflag=1;
    }

    //argc==4
    if(argc==4 && Eflag!=1){
        ifd1=open("fconv.out",O_RDONLY);
        if(ifd1==-1){
            perror("open");
            exit(1);
        }
        ofd=open(argv[3],O_RDWR|O_TRUNC|O_CREAT,fileperms);
        if(ofd==-1){
            perror("open");
            exit(1);
        }

        rwfile(ifd1,ofd);
    }
}
```

```
oslabd14@os-node2: ~/lab1/ex2
        perror("B");
    }

    //input1 to output
    rwfile(ifd1,ofd);

    //input2 to output
    rwfile(ifd2,ofd);

    //closing and return(0)
    int Eflag=0;
    if(close(ifd1)==-1){
        perror("close");
        Eflag=1;
    }
    if(close(ifd2)==-1){
        perror("close");
        Eflag=1;
    }

    if(close(ofd)==-1){
        perror("close");
        Eflag=1;
    }

    //argc==4
    if(argc==4 && Eflag!=1){
        ifd1=open("fconv.out",O_RDONLY);
        if(ifd1==-1){
            perror("open");
            exit(1);
        }
        ofd=open(argv[3],O_RDWR|O_TRUNC|O_CREAT,fileperms);
        if(ofd==-1){
            perror("open");
            exit(1);
        }

        rwfile(ifd1,ofd);
    }
}
```

```

        if(remove("fconc.out")==-1){
            perror("delete file");
            Eflag=1;
        }

        if(close(ifd1)==-1){
            perror("close");
            Eflag=1;
        }

        if(close(ofd)==-1){
            perror("close");
            Eflag=1;
        }
    }

    if(Eflag==1)
        exit(1);

    return 0;
}

```

Η συνάρτησή μας δηλώνεται στο header file `rwfile.h` με τον εξής κώδικα:

```
void rwfile(int inputfd,int outputfd);
```

Παράλληλα, ο κώδικας της συνάρτησης είναι στο αρχείο `rwfile.c` και έχει ως εξής:

oslabd14@os-node2: ~/lab1/ex2

```

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

#ifdef BUF_SIZE
#define BUF_SIZE 1024
#endif

void rwfile(int inputfd,int outputfd){
    char buf[BUF_SIZE];
    ssize_t rcnt,wcnt;
    size_t len,idx;
    while((rcnt=read(inputfd,buf,sizeof(buf)))!=0){
        if(rcnt==-1){
            perror("read");
            exit(1);
        }
        idx=0;
        len=rcnt;
        do{
            wcnt=write(outputfd,buf+idx,len-idx);
            if(wcnt==-1){
                perror("write");
                exit(1);
            }
            idx+=wcnt;
        }while(idx<len);
    }
}

```

## Ερωτήσεις:

1) Εκτελώντας την εντολή `$ strace ./fconc A B C`, έχουμε το παρακάτω αποτέλεσμα:

```
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) = 0x7fb455923000
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\3\0-\0\1\0\0\0P\34\2\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) = 0x7fb455362000
mprotect(0x7fb455362000, 2097152, PROT_NONE) = 0
mmap(0x7fb455703000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a1000) = 0x7fb455703000
mmap(0x7fb455709000, 14880, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fb455709000
close(3) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fb455922000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fb455921000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fb455920000
arch_prctl(ARCH_SET_FS, 0x7fb455921700) = 0
mprotect(0x7fb455703000, 16384, PROT_READ) = 0
mprotect(0x7fb45592d000, 4096, PROT_READ) = 0
munmap(0x7fb455923000, 30952) = 0
open("A", O_RDONLY) = 3
open("B", O_RDONLY) = 4
open("fconc.out", O_RDWR|O_CREAT|O_TRUNC, 0600) = 5
read(3, "Goodbye\n", 1024) = 12
write(5, "Goodbye\n", 12) = 12
read(3, "", 1024) = 0
read(4, "and thanks for all the fish!\n", 1024) = 29
write(5, "and thanks for all the fish!\n", 29) = 29
read(4, "", 1024) = 0
close(3) = 0
close(4) = 0
close(5) = 0
open("fconc.out", O_RDONLY) = 3
open("C", O_RDWR|O_CREAT|O_TRUNC, 0600) = 4
read(3, "Goodbye\nand thanks for all t...", 1024) = 41
write(4, "Goodbye\nand thanks for all t...", 41) = 41
read(3, "", 1024) = 0
unlink("fconc.out") = 0
close(3) = 0
close(4) = 0
exit_group(0) = ?
+++ exited with 0 +++
oslabd1@os-node2:~/lab1/ex2$
```

Η `strace` τρέχοντας την εντολή `./fconc A B C`, εμφανίζει στη οθόνη τα signals και τα system calls που καλούνται κατά την εκτέλεσή της. Πιο συγκεκριμένα, για κάθε system call εμφανίζεται το όνομά του, τα arguments του και οι επιστρεφόμενες τιμές του, απεικόνιση που βοηθάει πολύ στην αποσφαλμάτωση του κώδικα και στον εντοπισμό σφαλμάτων που προκύπτουν κατά την εκτέλεση. Στην παραπάνω εικόνα, έχουν πλαισιωθεί τα system calls που κλήθηκαν κατά την εκτέλεση της εντολής με σειρά που προσδιορίζεται στα αρχεία `.c` του προγράμματος.