**Εθνικό Μετσόβιο Πολυτεχνείο**

**Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών**

**Λειτουργικά Συστήματα**

**6ο εξάμηνο, Ακαδημαϊκή περίοδος 2019-2020**

# Άσκηση 1: Εισαγωγή στο περιβάλλον προγραμματισμού


Στοιχεία σπουδαστών:

Όνομα: Μαντζούτας Ανδρέας  Α.Μ. : 03117108

Όνομα: Τσιτσής Αντώνιος  Α.Μ. : 03117045

Ομάδα εργαστηρίου: Oslabc23


Εξάμηνο:  $6^{ο}$

Ημερομηνία: 02/06/2020

## Σύντομη Περίληψη του Περιεχόμενου του Εργαστηρίου

Στην πρώτη εργαστηριακή άσκηση, κάναμε μία εισαγωγή στο λειτουργικό σύστημα Linux. Αναλύσαμε τις βασικές λειτουργίες, καθώς και την πλοήγηση σε αυτό μέσω του bash shell. Πιο συγκεκριμένα, μάθαμε να διαχειριζόμαστε καταλόγους, αρχεία, καθώς και να βρίσκουμε την τεκμηρίωση που χρειαζόμαστε, μέσω των κατάλληλων εντολών. Έπειτα, είδαμε κάποιες λειτουργίες της γλώσσας **C**, όπως η μεταγλώττιση και η σύνδεση, και ασχοληθήκαμε με τη χρήση του **Makefile**. Τέλος, ασχοληθήκαμε με τις βασικές κλήσεις συστήματος**, open**, **read**, **write** και **close**, με την εντολή **strace** για καταγραφή των κλήσεων συστήματος και με την αυτόματη στοίχιση του κώδικα στο vim.

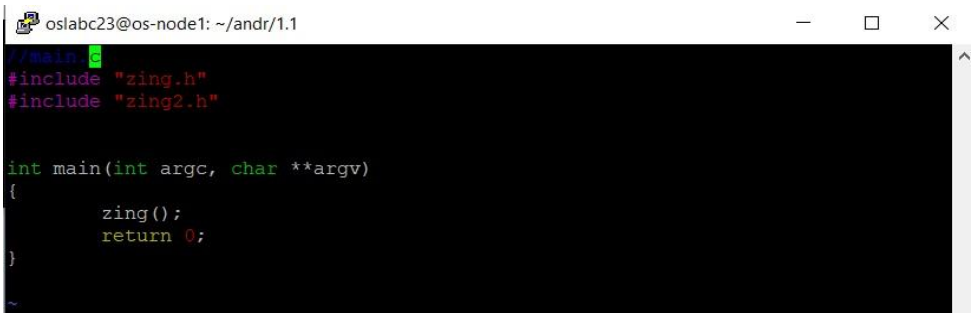
## Ασκήσεις

### 1.1 Σύνδεση με αρχείο αντικειμένων

Για την υλοποίηση αυτής της άσκησης, αρχικά αντιγράψαμε τα αρχεία zing.h και zing.o στον κατάλογο 1.1 που δημιουργήσαμε, από το path που μας δόθηκε. Αυτό επιτεύχθηκε με χρήση της εντολής:

**cp /home/oslab/code/zing/zing* 1.1**

Έπειτα, δημιουργήσαμε το κατάλληλο πρόγραμμα που καλεί τη συνάρτηση.
Πηγαίος Κώδικας Άσκησης:




Διαδικασία Μεταγλώττισης, σύνδεσης και έξοδος εκτέλεσης του προγράμματος:

## Ερωτήσεις

1. **Ποιο σκοπό εξυπηρετεί η επικεφαλίδα;**
   Οι επικεφαλίδες χρησιμοποιούνται για τη δήλωση μίας συνάρτησης που προορίζεται για χρήση σε περισσότερα αρχεία, που μεταγλωττίζονται ξεχωριστά. Επιτρέπουν στον μεταγλωττιστή να «καλέσει» τον κώδικα, έπειτα ο Linker βρίσκει το όρισμα και συνδέει το πρόγραμμα με το κατάλληλο κάθε φορά όρισμα.
   Σε περίπτωση που δεν ήταν γραμμένη ξεχωριστά η επικεφαλίδα, κατά τις διάφορες μεταγλωττίσεις θα δημιουργούνταν πολλαπλά ορίσματα, πράγμα μη επιθυμητό.

2. **Ζητείται κατάλληλο Makefile για τη δημιουργία του εκτελέσιμου της άσκησης.**
   Κατάλληλο Makefile για τη δημιουργία του εκτελέσιμου είναι το ακόλουθο:

```
zing: zing.o main.o
	gcc -o zing zing.o main.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.**

   Για την υλοποίηση αυτού του ερωτήματος, δημιουργήσαμε ένα νέο αρχείο zing2.c που περιείχε μια νέα συνάρτηση zing(), καθώς και ένα header file. Το εισάγαμε στο αρχείο main και ανανεώσαμε το Makefile, ώστε να το κάνει compile και link. Έπειτα, ανάλογα με ποιο εκτελέσιμο καλούταν εμφανιζόταν το κατάλληλο μήνυμα. Παρακάτω παρατίθενται οι κώδικες σε φωτογραφία.
   Στην τελευταία εικόνα φαίνεται η διαδικασία make, καθώς και η έξοδος της εκτέλεσης των δύο προγραμμάτων.

oslabc23@os-node1: ~/andr/1.1

```
//zing2.h
void zing(void);
~
```

oslabc23@os-node1: ~/andr/1.1

```
//zing2.c
#include <stdio.h>
#include <unistd.h>

void zing(void)
{
        printf("What's up, %s?\n", getlogin());
}
```

oslabc23@os-node1: ~/andr/1.1

```
//main.c
#include "zing.h"
#include "zing2.h"


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

~
```

oslabc23@os-node1: ~/andr/1.1

```
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
```

```
oslabc23@os-node1:~/andr/1.1$ make
gcc -Wall -c zing2.c
gcc -o zing2 zing2.o main.o
oslabc23@os-node1:~/andr/1.1$ ./zing
Hello, oslabc23
oslabc23@os-node1:~/andr/1.1$ ./zing2
What's up, oslabc23?
```

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;**

   Για την αντιμετώπιση αυτού του προβλήματος, μπορούμε να δημιουργήσουμε ξεχωριστά αρχεία για κάθε συνάρτηση, και με χρήση του Makefile να τα συνδέσουμε κατά την εκτέλεση. Με αυτόν τον τρόπο, κάθε φορά θα γίνεται compile μόνο ότι έχει αλλάξει, κι όχι όλο το αρχείο.

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;**

   Με αυτήν την εντολή ο συνεργάτης δημιούργησε ένα εκτελέσιμο αρχείο με όνομα foo.c, το οποίο αντικατέστησε το προηγούμενο αρχείο με το ίδιο όνομα, που περιείχε τον κώδικα. Ουσιαστικά, πανώγραψε το προηγούμενο αρχείο με το ίδιο όνομα. Αυτό είχε ως αποτέλεσμα να χαθεί ο κώδικας.

## 1.2 Συνένωση δύο αρχείων σε τρίτο

Για την επίτευξη του ζητούμενο της άσκησης, δημιουργήσαμε 4 βοηθητικές συναρτήσεις:

- Μία για το άνοιγμα των αρχείων, η οποία εμφάνιζει error αν υπάρξει σφάλμα κατά το άνοιγμα,
- Μία, η οποία μας «διαβεβαιώνει» ότι τα arguments είναι σωστά, αλλιώς εμφανίζει επεξηγηματικό μήνυμα,
- Και τέλος 2 ακόμα συναρτήσεις που είναι υπεύθυνες για τη σωστή εγγραφή του αρχείου.

Τέλος, σημειώνουμε ότι αφού δεν μας δόθηκαν σαφείς οδηγίες από την εκφώνηση για το πως να χειριστούμε ζητήματα που ταυτίζονται ένα από τα αρχεία εισόδου με το αρχείο εξόδου, εμείς το θεωρήσαμε ως μη επιτρεπτό και σε τέτοια περίπτωση το πρόγραμμά μας επιστρέφει μήνυμα σφάλματος.

Παρακάτω παρατίθεται ο πηγαίος κώδικας της άσκησης, καθώς και μερικά ενδεικτικά παραδείγματα εκτέλεσης.

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

int open_file(const char *file1, int oflags){  //anoigma arxeiou
      int fd;
      fd = open(file1,  oflags , S_IRUSR|S_IWUSR);
      if (fd == -1){ //error
            return -1;
      }
      return fd;//epistrofi pointer sto arxeio
}

int check_args (int argc){
      if (argc < 3 || argc > 4) {
            fprintf(stderr, "Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
            return -1;}
      return 0;
}

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

```
            idx += wcnt;
        } while (idx < len) ;
}

void write_file(int fd, const char *infile){
        int infd, rcnt;
        char buff[1024];
        infd = open_file(infile, O_RDONLY);
        if (infd == -1) {
                perror(infile);
                exit(1);
        }
        for (;;) {
                rcnt = read(infd, buff, sizeof(buff));
                if (rcnt == 0) break;
                if (rcnt == -1) {
                        perror("write");
                        exit(1);
                }

                doWrite(fd, buff, rcnt);
        }
        close(infd);
}

int main(int argc, char **argv)
{
        char  *outfile;
        int fileout;

        if ((strcmp(argv[1], argv[3]) == 0) || (strcmp(argv[2], argv[3]) == 0)){
                printf("Input and output have to be different files!\n");
                exit(1);
        }

        if (check_args(argc) != 0) exit(1);

        if (argc == 3){
                outfile = "fconc.out";
                fileout = open_file(outfile,  O_CREAT | O_WRONLY | O_TRUNC);
                if (fileout == -1) exit(1);

                write_file(fileout, argv[1]);
                write_file(fileout, argv[2]);
                close(fileout);
}
        if (argc == 4) {

                outfile = argv[3];
                fileout = open_file(outfile,  O_CREAT | O_WRONLY | O_TRUNC);
                if (fileout == -1) exit(1);
                write_file(fileout, argv[1]);
                write_file(fileout, argv[2]);
                close(fileout);
        }
        return 0;
}
```

```
oslabc23@os-node1:~/andr/1.2$ echo "hello" > A
oslabc23@os-node1:~/andr/1.2$ echo "world" > B
oslabc23@os-node1:~/andr/1.2$ ./fconc A B
oslabc23@os-node1:~/andr/1.2$ cat fconc.out
hello
world
oslabc23@os-node1:~/andr/1.2$ ./fconc A B C
oslabc23@os-node1:~/andr/1.2$ cat C
hello
world
oslabc23@os-node1:~/andr/1.2$ ./fconc A
Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]
oslabc23@os-node1:~/andr/1.2$ ./fconc C D
D: No such file or directory
oslabc23@os-node1:~/andr/1.2$
```

## 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.**
   Για την επίτευξη αυτού του ερωτήματος, χρησιμοποιήσαμε την εντολή strace για να μελετήσουμε τις κλήσεις συστήματος που πραγματοποιούνται κατά τη συνένωση δύο αρχείων σε τρίτο. Τέλος, κάναμε redirect την έξοδο σε ένα καινούριο αρχείο.

```
oslabc23@os-node1:~/andr/1.2$ strace ./fconc A B &> Mystracefile
oslabc23@os-node1:~/andr/1.2$
```

```
execve("./fconc", ["./fconc", "A", "B"], [/* 18 vars */]) = 0
brk(0)                          = 0x1ec3000
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) =
0x7fb58bb35000
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) = 0x7fb58bb2d000
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"..., 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) =
0x7fb58b56c000
mprotect(0x7fb58b70d000, 2097152, PROT_NONE) = 0
mmap(0x7fb58b90d000, 24576, PROT_READ|PROT_WRITE,
MAP_P0RIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a1000) = 0x7fb58b90d000
mmap(0x7fb58b913000, 14880, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fb58b913000
close(3)                        = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fb58bb2c000
```

```
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fb58bb2b000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fb58bb2a000
arch_prctl(ARCH_SET_FS, 0x7fb58bb2b700) = 0
mprotect(0x7fb58b90d000, 16384, PROT_READ) = 0
mprotect(0x7fb58bb37000, 4096, PROT_READ) = 0
munmap(0x7fb58bb2d000, 30952)          = 0
open("fconc.out", O_WRONLY|O_CREAT|O_TRUNC, 0600) = 3
open("A", O_RDONLY)                    = 4
read(4, "Hello,\n", 1024)              = 7
write(3, "Hello,\n", 7)                = 7
read(4, "", 1024)                      = 0
close(4)                               = 0
open("B", O_RDONLY)                    = 4
read(4, "and thanks for all the fish!\n", 1024) = 29
write(3, "and thanks for all the fish!\n", 29) = 29
read(4, "", 1024)                      = 0
close(4)                               = 0
close(3)                               = 0
exit_group(0)                          = ?
+++ exited with 0 +++
```

## Πηγές Πληροφόρησης

- https://stackoverflow.com/questions/2184646/what-is-the-point-of-header-files-in-c
- http://www.cslab.ece.ntua.gr/courses/os/files/2019-20/os-lab01-intro.pdf
- https://www.linuxquestions.org/questions/linux-general-1/strace-redirecting-to-file-456212/