ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
& ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Εργαστήριο Λειτουργικών Συστημάτων

Αναφορά 1ης Εργαστηρικής Άσκησης

Ιάσων - Λάζαρος Παπαγεωργίου | Α.Μ: 03114034
Γρηγόρης Θανάσουλας | Α.Μ: 03114131

Άσκηση 1

Αντιγράψαμε τον directory zing με τα περιεχόμενα του στο path του workspace μας εκτελώντας:

```
cp -r /home/oslab/code/zing/ ~/ask1/
```

Στη συνέχεια δημιουργήσαμε το αρχείο **main.c**. Αρχικά κάναμε include το header file **zing.h,** το οποίο περιλαμβάνει το prototype της συνάρτησης zing(), ενώ στη συνέχεια καλέσαμε τη συνάρτηση zing() στο body της main. Ο κώδικας του αρχείου main.c είναι ο παρακάτω:

```c
// main.c

#include "zing.h"

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

Για την μεταγλώττιση, αρχικά χωρίς τη χρήση Makefile, τρέξαμε:

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

Τρέχοντας το εκτελέσιμο με την εντολή ./zing, η έξοδος που λάβαμε στο τερματικό είναι:

```
Hello, oslaba04
```

Ερωτήσεις

1) Η επικεφαλίδα περιέχει το prototype της συνάρτησης zing() και είναι απαραίτητη ώστε να γνωρίζει ο compiler ποιος είναι ο σωστός τρόπος κλήσης της συνάρτησης (ορίσματα και τύπος που επιστρέφει).

2) Το Makefile που χρησιμοποιήσαμε είναι το παρακάτω:

```
# Makefile-1
all: zing

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

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

3) Η δική μας έκδοση της zing είναι η παρακάτω:

```
// zing2.c
 1 #include <stdio.h>
 2 #include <unistd.h>
 3 #include <string.h>
 4
 5 void zing(void) {
 6     char *s;
 7     s= getlogin();
 8     printf("And the user is %s\n", s);
 9 }
```

Το Makefile που χρησιμοποιήσαμε για την παραγωγή των δύο εκτελέσιμων zing και zing2 είναι το παρακάτω:

```
# Makefile-2
all: zing zing2

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

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

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

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

4) Η σωστή πρακτική στην περίπτωση αυτή θα ήταν να έχει η συνάρτηση ένα ξεχωριστό αρχείο και να γίνεται μεταγλώττιση μόνο του δικού της κώδικα. Στη συνέχεια θα έπρεπε να γίνει link ο νέος object code της συνάρτησης με τον προυπάρχοντα object code των υπολοίπων συναρτήσεων.

5) Η εντολή **gcc -Wall -o foo.c foo.c** λέει στον compiler να μεταγλωττίσει το αρχείο foo.c και να παράξει ως έξοδο object code στο αρχείο foo.c. Έτσι, αν εκτελέσουμε αυτή την ενοτλή το αρχικό μας αρχείο foo.c γίνεται overwrite με τον object code και χάνουμε τον πηγαίο κώδικα που γράψαμε.

Άσκηση 2

Εκτελώντας την εντολή strace ./fconc A B, όπου A και B τα δύο δοκιμαστικά αρχεία εισόδου μας, λαμβάνουμε την εξής έξοδο στο τερματικό μας:

```
execve("./fconc", ["./fconc", "A", "B"], [/* 30 vars */]) = 0
```

```
brk(0)                                  = 0x6c5000
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) =
0x7fa482d02000
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) = 0x7fa482cfa000
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) =
0x7fa482739000
mprotect(0x7fa4828da000, 2097152, PROT_NONE) = 0
mmap(0x7fa482ada000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a1000) = 0x7fa482ada000
mmap(0x7fa482ae0000, 14880, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fa482ae0000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fa482cf9000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fa482cf8000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7fa482cf7000
arch_prctl(ARCH_SET_FS, 0x7fa482cf8700) = 0
mprotect(0x7fa482ada000, 16384, PROT_READ) = 0
mprotect(0x7fa482d04000, 4096, PROT_READ) = 0
munmap(0x7fa482cfa000, 29766)           = 0
open("A", O_RDONLY)                     = 3
open("B", O_RDONLY)                     = 4
open("fconc.out", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 5
read(3, "Hello beautifull world\n", 1023) = 23
write(5, "Hello beautifull world\n", 23) = 23
read(3, "", 1023)                       = 0
read(4, "What is this?\n", 1023)        = 14
write(5, "What is this?\n", 14)         = 14
read(4, "", 1023)                       = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Ο κώδικας της άσκησής μας είναι ο παρακάτω:

//Αρχείο doWrite.c

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

```c
#include <unistd.h>

// Writes the buffer from index 0 till length to the output file descriptor fd.

int doWrite(int fd, char buff[], int len){
    ssize_t wcnt;
    ssize_t idx = 0;
    do {
        wcnt = write(fd, buff + idx, len - idx);
        if (wcnt == -1) { //error
            perror("write");
            return 1;
        }
        idx += wcnt;
    } while (idx < len);
    return 0;
}
```

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include "func.h"

int main(int argc, char **argv) {
    int i;
    if (argc < 3 || argc > 4) {
        printf("Usage: .fconc inFile1 inFile2 [outFile (default:fconc.out)]\n");
        return 1;
    }

    //Open Input File 1
    int fd1 = open(argv[1], O_RDONLY);
    if (fd1 == -1) {
        perror("Error while opening inFile1");
        return 2;
    }

    //Open Input File 2
    int fd2 = open(argv[2], O_RDONLY);
    if (fd2 == -1) {
        perror("Error while opening inFile2");
        return 2;
    }

    int fd3;
    // Determine output file name
```

```
 31     if (argc == 4) {
 32         if (strcmp(argv[3], argv[1]) == 0
 33             || strcmp(argv[3], argv[2]) == 0) {
 34             printf("ERROR: OutFile name must be different than input file
name.\n");
 35             return 2;
 36         }
 37         else {
 38             fd3 = open(argv[3], O_WRONLY | O_CREAT | O_TRUNC, 0666);
 39         }
 40     }
 41     else {
 42         if (strcmp("fconc.out", argv[1]) == 0
 43             || strcmp("fconc.out", argv[2]) == 0) {
 44             printf("ERROR: File fconc.out must not be used as input.\n");
 45             return 2;
 46         }
 47         else {
 48             fd3 = open("fconc.out", O_WRONLY | O_CREAT | O_TRUNC, 0666);;q
 49         }
 50     }
 51
 52     if (fd3 == -1){
 53             perror("Error opening/creating the outFile");
 54             return 3;
 55     }
 56
 57     //Write inFile1 to OutFile
 58     i = write_file(fd3, fd1);
 59     if (i == 1) {
 60             perror("Write inFile1 to outFile Failed.");
 61             return 4;
 62     }
 63
 64     //Write inFile1 to OutFile
 65     i = write_file(fd3, fd2);
 66     if (i == 1){
 67             perror("Write inFile2 to outFile Failed.");
 68             return 4;
 69     }
 70
 71     return 0;
 72 }
 73
```

```
//Αρχείο write_file.c
```

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include "func.h"

// Tries to write the contents of in_fd to out_fd.
// Return 0 on success, 1 on failure.
int write_file(int out_fd, int in_fd) {
    char buff[BUFF_SIZE];
    ssize_t rcnt;
    for (;;){
        rcnt = read(in_fd, buff, BUFF_SIZE - 1);
        if (rcnt == 0) /* End-of-file */
            break;
        if (rcnt == -1){ /* error */
            perror("Error while reading input file");
            return 1;
        }
        doWrite(out_fd, buff, rcnt);
    }
    return 0;
}

```