1η Αναφορά στο μάθημα “Λειτουργικά Συστήματα”

Αντωνίου Αριστείδης, ΑΜ: 03114748
Λιαροκάπης Αλέξανδρος, ΑΜ: 03114860

1.1

ex1/main.c

```c
#include "zing.h"

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

ex1/zing2.c

```c
#include <unistd.h>

#include <stdio.h>

void zing() {
        char * l = getlogin();
        printf("Hello my friend, %s\n", l);
}
```

ex1/Makefile

```
.PHONY: clean, all

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
zing2.o: zing2.c
        gcc -c zing2.c
main.o: main.c zing.h
        gcc -c main.c
clean:
        rm main.o zing2.o zing zing2
```

Αρχικά κατασκευάζουμε τα object files που χρειάζονται και τα αναλαμβάνει ο linker και ως αποτέλεσμα έχουμε το εκτελέσιμο αρχείο. Με εκτέλεση του zing, εμφανίζεται το μήνυμα “Hello oslabb23”, ενώ με την εκτέλεση του zing2 που φτιάξαμε εμείς, εμφανίζεται το αλλαγμένο μήνυμα “Hellow my friend, oslabb23”

Η επικεφαλίδα περιέχει τις δηλώσεις των συναρτήσεων προς επαναχρησιμοποίηση. Αυτό συμβαίνει και στην δικιά μας περίπτωση, στην main, όπου χρησιμοποιείται η επικεφαλίδα της zing.
Στην ερώτηση 4, το πρόβλημα είναι πως η συνάρτηση αυτή, βρίσκεται σε αρχείο με όλες τις άλλες συναρτήσεις. Έτσι, κάθε αλλαγή που κάνουμε σε αυτήν, οδηγεί σε μεταγλώττιση ολόκληρου του αρχείου, γι'αυτό έχουμε τελικά τόσο μεγάλη καθυστέρηση. Οπότε, θα ήταν πολύ καλύτερο να έχουμε μικρότερα αρχεία με λιγότερες συναρτήσεις, ώστε να αποφύγουμε αυτό το πρόβλημα.
Στην ερώτηση 5, ενώ δίνουμε το σωστό αρχείο στον μεταγλωττιστή, δίνουμε ως διευθύνση εξόδου το ίδιο αρχείο, και έτσι αντικαθίσταται το αρχείο μας με το εκτελέσιμο, με αποτέλεσμα να χάνεται τελικά ο κώδικας.

1.2

ex2/filelib.h

```h
#ifndef FILELIB_H
#define FILELIB_H

#include <unistd.h>

void do_write(int fd, const char *buff, ssize_t len);
int write_file(int fd, const char *infile);

#endif
```

ex2/filelib.c

```c
#include "filelib.h"
#include <unistd.h>
#include <limits.h>
#include <sys/types.h>
#include <fcntl.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>


void do_write(int fd, const char *buff, ssize_t len) {
        ssize_t write_count, pos = 0;
        while((write_count = write(fd, buff + pos, len - pos)) > 0) {
                pos += write_count;
        }
}

int write_file(int fd, const char *infile) {
        ssize_t read_count;
        char buff[1024];
        int ifd = open(infile, O_RDONLY);

        if (ifd == -1) {
                close(ifd);
```

```
                return -1;
        }

        while ((read_count = read(ifd, buff, 1024)) > 0) {
                do_write(fd, buff, read_count);
        }

        close(ifd);
}
```

ex2/fconc.c

```
#include "filelib.h"
#include <unistd.h>
#include <limits.h>
#include <sys/types.h>
#include <fcntl.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>


int main(int argc, char* argv[])
{
        if (argc < 3 || argc > 4)
        {
                printf("Usage: ./fconc infile1 infile2 [outfile
(default:fconc.out)]\n");
                return EXIT_FAILURE;
        }

        const char * out_name = "./fconc.out";

        if (argc == 4)
        {
                out_name = argv[3];
        }

        int fd3 = open(out_name, O_WRONLY | O_CREAT | O_TRUNC | O_APPEND, 0666);

        if (write_file(fd3, argv[1]) == -1) {
                close(fd3);
                printf("%s file not found\n", argv[1]);
                return EXIT_FAILURE;
        }

        if (write_file(fd3, argv[2]) == -1) {
                close(fd3);
                printf("%s file not found\n", argv[2]);
                return EXIT_FAILURE;
        }

        return EXIT_SUCCESS;
}
```

ex2/Makefile

```
.PHONY: all clean

all: fconc

fconc: fconc.o filelib.o
	gcc fconc.o filelib.o -o fconc
fconc.o: fconc.c filelib.h
	gcc -c fconc.c
filelib.o: filelib.c filelib.h
	gcc -c filelib.c
clean:
	rm fconc fconc.o filelib.o
```

Με την εκτέλεση της strace, εκτυπώνονται όλες οι κλήσεις συστήματος που έγιναν κατά την εκτέλεση του κώδικα μας. Το αποτέλεσμα της strace φαίνεται παρακάτω.

```
execve("./fconc", ["./fconc", "A", "C"], [/* 27 vars */]) = 0
brk(0)                                  = 0xbef000
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) =
0x7f174a1de000
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=30868, ...}) = 0
mmap(NULL, 30868, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f174a1d6000
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) =
0x7f1749c15000
mprotect(0x7f1749db6000, 2097152, PROT_NONE) = 0
mmap(0x7f1749fb6000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1a1000) = 0x7f1749fb6000
mmap(0x7f1749fbc000, 14880, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f1749fbc000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f174a1d5000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f174a1d4000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f174a1d3000
arch_prctl(ARCH_SET_FS, 0x7f174a1d4700) = 0
mprotect(0x7f1749fb6000, 16384, PROT_READ) = 0
mprotect(0x7f174a1e0000, 4096, PROT_READ) = 0
munmap(0x7f174a1d6000, 30868)           = 0
open("./fconc.out", O_WRONLY|O_CREAT|O_TRUNC|O_APPEND, 0666) = 3
open("A", O_RDONLY)                     = 4
read(4, "This is file B\n", 1024)       = 15
write(3, "This is file B\n", 15)        = 15
write(3, "", 0)                         = 0
read(4, "", 1024)                       = 0
close(4)                                = 0
```

```
open("C", O_RDONLY)                     = 4
read(4, "This is file A\nThis is file B\n", 1024) = 30
write(3, "This is file A\nThis is file B\n", 30) = 30
write(3, "", 0)                         = 0
read(4, "", 1024)                       = 0
close(4)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Επίσης, να σημειωθούν κάποιες “ειδικές περιπτώσεις” εκτέλεσης της fconc. Όταν χρησιμοποιήσουμε ως αρχεία εισόδου τα A, B και ως αρχείο εξόδου το A, το αποτέλεσμα είναι να εμφανιστεί σε αυτό, μόνο το B, γιατί γίνεται μηδενισμός του περιεχομένου του A στην εγγραφή, και δεν υπάρχει τίποτα για ανάγνωση. Επίσης, όταν χρησιμοποιήσουμε ως αρχεία εισόδου τα A,B και ως αρχείο εξόδου το B, το πρόγραμμα δεν θα τερματίσει ποτέ.