1η Ομάδα Ασκήσεων

στα Λειτουργικά Συστήματα

Ιωακειμίδη Αθηνά
Α.Μ.: 03114758
Μαυρομμάτης Ιάσων
Α.Μ.: 03114771

Εξάμηνο 7ο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Εθνικό Μετσόβιο Πολυτεχνείο

Άσκηση 1.1

Πηγαίος κώδικας:

main.c :

```
#include "zing.h"

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

zing2.c :

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

void zing(void) {
        char *A = getlogin();
        printf("Bye bye, %s\n", A);
        return ;
}
```

Διαδικασία μεταγλώττισης και σύνδεσης:

makefile :

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

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

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

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

Έξοδοι εκτελέσεων:

```
Hello, oslaba38


Bye bye, oslaba38
```

Απαντήσεις στις ερωτήσεις:

1. Οι επικεφαλίδες περιέχουν ορισμούς και δηλώσεις των συναρτήσεων και καθολικών μεταβλητών που μοιράζονται πολλαπλά αρχεία πηγαίου κώδικα. Στην άσκηση αυτή η επικεφαλίδα `zing.h` περιέχει τη δήλωση της συνάρτησης `zing`, της οποίας εμείς έχουμε μόνο το αρχείο αντικειμένου. Η `main` αναγνωρίζει την `zing` μέσω αυτής της επικεφαλίδας και μετά μέσω του linking με το αρχείο αντικειμένου της τη βρίσκει για να την εκτελέσει.
2. Δίνεται παραπάνω.

3. Δίνεται παραπάνω.
4. Έχουμε ένα αρχείο με 500 συναρτήσεις εκ των οποίων επεξεργαζόμαστε μόνο τη μία ή και παραπανω. Αυτο που κανουμε λοιπον ειναι σβηνουμε την συναρτηση ή τις συναρτησεις που θελουμε να ανακατασκευασουμε απο το αρχικο αρχειο που ειναι ολες γραμμενες και τις ξαναδημιουργουμε σε ξεχωριστο αρχειο . Επειτα φτιαχνουμε τα object files και τα συνδέουμε (linking) μεταξύ τους. Με αυτόν τον τρόπο και η επεξεργασία και το debugging είναι πολύ πιο εύκολο, αλλά και παύουμε πια να κάνουμε compile 500 συναρτήσεις κάθε φορά, αφού κάνουμε compile μόνο τη μία που επεξεργαζόμαστε, και τελικά ο χρόνος μεταγλώττισης είναι αισθητά πολύ μικρότερος.
5. Η εντολή `gcc -Wall -o foo.c foo.c` έκανε compile το αρχείο `foo.c` και το object file που δημιουργήθηκε το ονόμασε `foo.c`. Έτσι έκανε overwrite το αρχείο που υπήρχε πριν με αυτό το όνομα και το αρχείο πηγαίου κώδικα χάθηκε. Για να αποφύγουμε τέτοια προβλήματα κρατάμε πάντα αντίγραφο των αρχείων μας και χρησιμοποιούμε το makefile το οποίο γράφουμε μια φορά και δεν χρειάζεται να ανησυχούμε κάθε φορά μήπως γράψουμε κάτι λάθος.

Άσκηση 1.2

Πηγαίος κώδικας:

main.c :

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

void readwrite(int fdread, int fdwrite);

int main(int argc, char **argv) {
        int f, g, fdwrite;
        int oflags, mode;
        oflags = O_CREAT | O_WRONLY | O_TRUNC;
        mode = S_IRUSR | S_IWUSR;

        if (argc==4) {          //3 files given
                if (strcmp(argv[1],argv[3])==0 ||
strcmp(argv[2],argv[3])==0){
                        printf("Try again with a different file names.\n");
                        exit(1);
                }
                fdwrite = open(argv[3], oflags, mode);
        }
        else if (argc==3) {     //2 files given
                if ( (strcmp(argv[1],"fconc.out")==0) ||
(strcmp(argv[2],"fconc.out")==0) ){
                        printf("Try again with different file names.\n");
                        exit(1);
                }
                fdwrite = open("fconc.out", oflags, mode);
        }
        else {                  //wrong number of files given
                printf("Usage: ./fconc infile1 infile2 [outfile
(default:fconc.out)]\n");
                exit(1);
        }
        if (fdwrite == -1){
                perror("open");
                exit(1);
        }

        f = open(argv[1], O_RDONLY);
        if (f == -1){
                perror(argv[1]);
                exit(1);
        }
        g = open(argv[2], O_RDONLY);
        if (g == -1){
                perror(argv[2]);
                exit(1);
        }
```

```
 readwrite(f, fdwrite);
 readwrite(g, fdwrite);

 close(f);
 close(g);
 close(fdwrite);
}
```

readwrite.c :

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

void writefile(void *buff, int fdwrite);

void readwrite(int fdread, int fdwrite){
        char buff[1024];
        ssize_t rcnt;
        for (;;){
                rcnt = read(fdread,buff,sizeof(buff)-1);
                if (rcnt == 0) /* end-of-file */
                        return ;
                if (rcnt == -1){ /* error */
                        perror("read");
                        exit(1);
                }
                buff[rcnt] = '\0';
                writefile(buff, fdwrite);
        }
        return ;
}
```

writefile.c :

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

void writefile(char *buff, int fdwrite) {
        size_t len, idx;
        ssize_t wcnt;
        idx = 0;
        len = strlen(buff);
        do {
                wcnt = write(fdwrite,buff + idx, len - idx);
                if (wcnt == -1){ /* error */
```

```
                        perror("write");
                        exit(1) ;
                }
                idx += wcnt;
        } while (idx < len);
        return ;
}
```

Έξοδος της strace:

```
execve("./fconc", ["./fconc", "A", "B", "C"], [/* 20 vars */]) = 0
brk(0)                                  = 0x182e000
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) =
0x7f6791644000
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) = 0x7f679163c000
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\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) =
0x7f679107b000
mprotect(0x7f679121c000, 2097152, PROT_NONE) = 0
mmap(0x7f679141c000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x1a1000) = 0x7f679141c000
mmap(0x7f6791422000, 14880, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x7f6791422000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f679163b000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f679163a000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f6791639000
arch_prctl(ARCH_SET_FS, 0x7f679163a700) = 0
mprotect(0x7f679141c000, 16384, PROT_READ) = 0
mprotect(0x7f6791646000, 4096, PROT_READ) = 0
munmap(0x7f679163c000, 29766)           = 0
open("C", O_WRONLY|O_CREAT|O_TRUNC, 0600) = 3
open("A", O_RDONLY)                     = 4
open("B", O_RDONLY)                     = 5
read(4, "Goodbye,\n", 1023)             = 9
write(3, "Goodbye,\n", 9)               = 9
read(4, "", 1023)                       = 0
read(5, "and thanks for all the fish!\n", 1023) = 29
write(3, "and thanks for all the fish!\n", 29) = 29
read(5, "", 1023)                       = 0
close(4)                                = 0
close(5)                                = 0
close(3)                                = 0
exit_group(0)                           = ?
```

```
+++ exited with 0 +++
```