ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Λειτουργικά Συστήματα

ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΝΑΦΟΡΑ 1Η

Από τους

Ιωάννης Μιχαήλ Καζελίδης
Α.Μ.: 03117885

Μάριος Κερασιώτης
Α.Μ.: 03117890

1.1 Σύνδεση με αρχείο αντικειμένων

Για το πρώτο μέρος της άσκησης ο φάκελος εργασίας μας περιέχει τα παρακάτω αρχεία μαζί το zing.o και για να τρέξουμε τα αρχεία κάνουμε compile τον κώδικα εκτελώντας στο terminal την εντολή *make* και αν όλα λειτουργούν σωστά και χωρίς λάθη τον τρέχουμε εκτελώντας την εντολή *./zing*. Ως έξοδο του προγράμματος λαμβάνουμε το κείμενο: *Hello, oslabb25* .

main.c

```c
#include "zing.h"

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

zing.h

```c
#ifndef ZING_H__
#define ZING_H__

void zing(void);

#endif
```

makefile

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

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

clean:
    rm -f main.o zing
```

Αντί της χρήση του makefile μπορούμε να μεταγλωττίσουμε και να συνδέσουμε το αρχείο ως εξής (για το πρώτο κομμάτι της άσκησης):

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

Για το δεύτερο μέρος της άσκησης ο φάκελος εργασίας μας περιέχει τα παρακάτω στοιχεία μαζί με το zing.o και όπως και πριν εκτελούμε στο terminal τις εντολές *make*, *./zing* και *./zing2* . Εάν τρέξουμε το zing έχουμε έξοδο όπως και πριν: *Hello, oslabb25* ενώ άμα τρέξουμε το zing2 έχουμε έξοδο: *Welcome, oslabb25!* .

main.c

```c
#include "zing.h"
#include "zing2.h"
int main(int argc, char **argv) {
  zing();
  return (0);
}
```

zing.h

```c
#ifndef ZING_H__
#define ZING_H__
```

```
void zing(void);

#endif
```

zing2.h

```
#ifndef ZING2
#define ZING2

void zing(void);

#endif  // ZING2
```

zing2.c

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

void zing(void) {
  char* username;
  username = getlogin();

  if (username == NULL) {
    printf("Error getting username\n");
  } else {
    printf("Welcome, %s!\n", username);
  }

  return;
}
```

makefile

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

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

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

clean:
    rm -f main.o zing2.o zing zing2
```

Αντί της χρήση του makefile μπορούμε να μεταγλωττίσουμε και να συνδέσουμε το αρχείο ως εξής (για το δεύτερο κομμάτι της άσκησης):

```
gcc -Wall -c main.c
gcc -Wall -c zing2.c
gcc -Wall main.o zing.o -o zing
gcc -Wall main.o zing2.o -o zing2
```

Ποιο σκοπό εξυπηρετεί η επικεφαλίδα;

Μια επικεφαλίδα είναι ένα αρχείο κώδικα που περιέχει δηλώσεις συναρτήσεων, δομών δεδομένων, σταθερών και άλλων στοιχείων που καλούμε σε ένα άλλο αρχείο για να χρησιμοποιήσουμε τις συναρτήσεις αυτές. Αυτό το αρχείο λειτουργεί σαν σύνδεσμος μεταξύ του κώδικα που θέλουμε να καλέσουμε τα στοιχεία αυτά και του κώδικα που περιέχει τους ορισμούς και τις υλοποιήσεις τους και

όταν γίνεται εισαγωγή (include) αυτού του αρχείου τότε λειτουργεί σαν να αντιγράφεται ο κώδικας στο αρχείο από το αρχείο με τους ορισμούς.

Ζητείται κατάλληλο Makefile για τη δημιουργία του εκτελέσιμου της άσκησης.
Βλέπε παραπάνω.

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.
Βλέπε παραπάνω.

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;
Θα μπορούσαμε να «χωρίσουμε» τις συναρτήσεις μας σε λογικές υποομάδες συναρτήσεων τις οποίες θα τις έχουμε σε διαφορετικά αρχεία, μέσα στα οποία θα τις ορίζουμε και θα δείχνουμε την υλοποίησή τους, και για να τις χρησιμοποιήσουμε θα έχουμε ξεχωριστά header files (ένα για κάθε τέτοιο αρχείο) τα οποία θα καλούμε στην αρχή του προγράμματός μας στο οποίο θα θέλουμε να χρησιμοποιήσουμε τις συναρτήσεις αυτές. Στη συνέχεια θα δημιουργήσουμε ένα makefile στο οποίο θα κάνουμε τα απαραίτητα links και με το οποίο θα κάνουμε compile. Έτσι θα παράγουμε εκτελέσιμα μόνο όταν αλλάζει ο κώδικάς μας και θα γλυτώνουμε τόσο χρόνο όσο και πόρους συστήματος (δηλαδή σε περίπτωση που μια μόνο συνάρτηση θέλει αλλαγή, θα μεταγλωττίσουμε μόνο το αρχείο που περιέχει την συνάρτηση αυτήν και όχι όλο το πρόγραμμα, όπως θα ήταν στην περίπτωση της εκφώνησης).

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;
Ο συνεργάτης μου έδωσε εντολή στο gcc να γράψει το αποτέλεσμα του στο foo.c. Έτσι έγινε overwrite και αντί να περιέχει τον κώδικα που θέλουμε περιέχει το αποτέλεσμα της μεταγλώττισης.

1.2 Συνένωση δύο αρχείων σε τρίτο

main.c

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

#include "functions.h"

int main(int argc, char const *argv[]) {
  if (argc < 3 || argc > 4) {
    printf("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
    return 0;
  }
```

```
  const char *infile1 = argv[1];
  const char *infile2 = argv[2];
  const char *outfile = (argc == 4) ? argv[3] : "fconc.out";

  // checks if both infiles exist, and if at least one doesn't then the
outfile
  // is not created
  if (check_file(infile1) != 0) {
    perror(infile1);
    exit(EXIT_FAILURE);
  }
  if (check_file(infile2) != 0) {
    perror(infile2);
    exit(EXIT_FAILURE);
  }

  int open_flags = O_CREAT | O_WRONLY | O_TRUNC;
  int open_mode = S_IRUSR | S_IWUSR;

  int outf = open(outfile, open_flags, open_mode);
  if (outf == -1) {
    perror(outfile);
    exit(EXIT_FAILURE);
  }

  write_file(outf, infile1);
  write_file(outf, infile2);

  if (close(outf) == -1) {
    perror("Close");
    exit(EXIT_FAILURE);
  }
  return 0;
}
```

functions.h

```
#if !defined(FUNCTIONS_H)
#define FUNCTIONS_H

// Writes buffer data into outfile
// int fd: file to write buffer into
// const char * buff: data to write into the file
// int len: size of buffer
void doWrite(int fd, const char *buff, int len);

// Opens infile and writes it's data into outfile using doWrite()
// int fd: file to write infile's data into
// const char * infile: filename
void write_file(int fd, const char *infile);

// Checks if a file exists.
// const char *infile: filename
int check_file(const char *infile);

#endif  // FUNCTIONS_H
```

functions.c

```
#include "functions.h"
```

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

#define BUFFER_SIZE 1024

void doWrite(int fd, const char *buff, int len) {
  ssize_t wcnt;
  ssize_t idx = 0;

  // using idx to verify that all the data is written to the outfile
  do {
    wcnt = write(fd, buff + idx, len - idx);
    if (wcnt == -1) {
      perror("Writing outfile");
      exit(EXIT_FAILURE);
    }
    idx += wcnt;
  } while (idx < len);
}

void write_file(int fd, const char *infile) {
  int open_flags = O_RDONLY;
  int open_mode = S_IRUSR;
  int inf = open(infile, open_flags, open_mode);
  if (inf == -1) {
    perror(infile);
    exit(EXIT_FAILURE);
  }

  char buff[BUFFER_SIZE];
  ssize_t rcnt;

  // write into buffer and call doWrite to write into the outfile.
  // If buffer is not enough (for reading infile), repeat
  for (;;) {
    rcnt = read(inf, buff, sizeof(buff) - 1);
    if (rcnt == 0) break;
    if (rcnt == -1) {
      perror(infile);
      exit(EXIT_FAILURE);
    }
    buff[rcnt] = '\0';

    doWrite(fd, buff, rcnt);
  }

  if (close(inf) == -1) {
    perror("Error closing file");
    exit(EXIT_FAILURE);
  }
  return;
}

int check_file(const char *infile) {
  int open_flags = O_RDONLY;
  int open_mode = S_IRUSR;
```

```
  int inf = open(infile, open_flags, open_mode);

  if (inf == -1) return 1;
  if (close(inf) == -1) return 1;
  return 0;
}
```

makefile

```
fconc: main.o functions.o
	gcc -Wall -Werror main.o functions.o -o fconc

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

functions.o: functions.c
	gcc -Wall -Werror -c functions.c

clean:
	rm -f *.o fconc *.out
```

Εκτελέστε ένα παράδειγμα του fconc χρησιμοποιώντας την εντολή strace. Αντιγράψτε το κομμάτι της εξόδου της strace που προκύπτει από τον κώδικα που γράψατε.

Τρέχουμε το πρόγραμμα με παραμέτρους infile1 infile2 και λαμβάνουμε ως έξοδο το fconc.out.

Με εκτέλεση της εντολής strace ως εξής: *strace -o output.txt ./fconc infile1 infile2* λαμβάνουμε ως έξοδο το αρχείο strace_output.txt (το οποίο φαίνεται παρακάτω) στο οποίο παρατηρούμε τις κλήσεις του συστήματος που έγιναν όπως για παράδειγμα τα read/ write με τα οποία διαβάζουμε τους χαρακτήρες των infile και τους αποθέτουμε στο fconc.out. Ουσιαστικά, το κομμάτι που μας ενδιαφέρει άμεσα και αναφέρεται στις κλήσεις συστήματος του κώδικά μας είναι από τη γραμμή `open("infile1", O_RDONLY) = 3` και κάτω.

strace_output.txt

```
execve("./fconc", ["./fconc", "infile1", "infile2"], [/* 27 vars */]) = 0
brk(0)                                  = 0x14ca000
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) =
0x7f224d555000
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) = 0x7f224d54d000
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) =
0x7f224cf8c000
mprotect(0x7f224d12d000, 2097152, PROT_NONE) = 0
mmap(0x7f224d32d000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a1000) = 0x7f224d32d000
mmap(0x7f224d333000, 14880, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f224d333000
```

```
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f224d54c000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f224d54b000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f224d54a000
arch_prctl(ARCH_SET_FS, 0x7f224d54b700) = 0
mprotect(0x7f224d32d000, 16384, PROT_READ) = 0
mprotect(0x7f224d557000, 4096, PROT_READ) = 0
munmap(0x7f224d54d000, 30952)           = 0
open("infile1", O_RDONLY)               = 3
close(3)                                = 0
open("infile2", O_RDONLY)               = 3
close(3)                                = 0
open("fconc.out", O_WRONLY|O_CREAT|O_TRUNC, 0600) = 3
open("infile1", O_RDONLY)               = 4
read(4, "Mouse somehow manage to catch a "..., 1023) = 1023
write(3, "Mouse somehow manage to catch a "..., 1023) = 1023
read(4, "om yourself 4 hours - checked, h"..., 1023) = 511
write(3, "om yourself 4 hours - checked, h"..., 511) = 511
read(4, "", 1023)                       = 0
close(4)                                = 0
open("infile2", O_RDONLY)               = 4
read(4, "I'm bored inside, let me out i'm"..., 1023) = 1023
write(3, "I'm bored inside, let me out i'm"..., 1023) = 1023
read(4, " or sleep nap. Sweet beast scoot"..., 1023) = 1023
write(3, " or sleep nap. Sweet beast scoot"..., 1023) = 1023
read(4, " human why take bird out i could"..., 1023) = 963
write(3, " human why take bird out i could"..., 963) = 963
read(4, "", 1023)                       = 0
close(4)                                = 0
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```