ΑΝΑΦΟΡΑ ΑΣΚΗΣΗΣ 1

Λειτουργικά Συστήματα 6ο εξάμηνο
Ακαδημαϊκή περίοδος 2019-2020
Ομάδα: oslabc18
Φοιτητές:Αθανασίου Ιωάννης Α.Μ.:03117041
Καραβαγγέλης Αθανάσιος Α.Μ.:03117022

ΑΣΚΗΣΗ 1.1

Source code:

zing.h:

```
#ifndef ZING_H__
#define ZING_H__
void zing(void);
#endif
```

main.c:

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

zing2.h:

```
#ifndef ZING2_H__
#define ZING2_H__
void zing(void);
#endif
```

zing2.c:

```
#include <stdio.h>
#include <unistd.h>
void zing(void) {
char* name = getlogin();
printf ("Hi, %s\n" , name );
}
```

makefile:

```
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
```

Διαδικασία μεταγλώττισης και σύνδεσης

- Αρχικά, αντιγράψαμε τα αρχεία “zing.h” και “zing.o” στο directory μας.
  Για να το κάνουμε αυτό, μπήκαμε στο directory με path /home/oslab/code/zing και εκτελέσαμε τις εντολές
  cp zing.h ../../oslabc18/ask1.1/zing.h και
  cp zing.o ../../oslabc18/ask1.1/zing.o
- Κατόπιν, γράφουμε τον κώδικα του αρχείου main.c και το αρχείο makefile, που κάνει compile και linking τα αρχεία που έχουμε, με αποτέλεσμα την παραγωγή του εκτελέσιμου **zing.**
- Για τα παρακάτω τμήματα της άσκησης, αρχικά γράφουμε το αρχείο zing2.c και στη συνέχεια τροποποιούμε το προηγούμενο makefile ώστε να μπορεί να ενημερώνει και τα δύο εκτελέσιμα που πλέον έχουμε.

Έξοδος του προγράμματος

Επιλέξαμε, όπως ζητείται, η εκτέλεση του **zing2** να εμφανίζει παρόμοιο μήνυμα με την εκτέλεση του **zing:**
Δίνοντας την εντολή “./zing”, εμφανίζεται στην οθόνη το μήνυμα “Hello, oslabc18” (με αλλαγή σειράς.
Ομοίως, δίνοντας την εντολή “./zing2”, εμφανίζεται στην οθόνη το μήνυμα
“Hi, oslabc18” (με αλλαγή σειράς και πάλι).

Ερωτήσεις

1. Το αρχείο με την επικεφαλίδα “zing.h” χρησιμοποιείται έτσι ώστε να αποκτήσουμε πρόσβαση στον κώδικα όπου ορίζεται η συνάρτηση zing. Υποθέτουμε ότι υπάρχει ένα τέτοιο αρχείο zing.c, το οποίο δεν μας δίνεται. Επομένως, για να μπορούμε στην main.c να χρησιμοποιήσουμε την συνάρτηση zing(), είναι αναγκαία η δήλωση #include “zing.h”.

2. Αρχικά το makefile για τη δημιουργία του εκτελέσιμου ***zing*** της άσκησης είναι :

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

3.Στο ερώτημα αυτό διαφοροποιήσαμε το makefile ,όπως φαίνεται στο κομμάτι *makefile* στον πηγαίο κώδικα παραπάνω, με σκοπό την παραγωγή ενός δεύτερου εκτελέσιμου ***zing2***. Το εκτελέσιμο **zing** παράγεται από τη σύνδεση των zing.o και main.o, ενώ το εκτελέσιμο **zing2** παράγεται από τη σύνδεση των zing2.o και main.o. Εφόσον στο σώμα της main.c καλούμε τη συνάρτηση zing(), ανάλογα με το εκτελέσιμο που επιλέγεται, γίνεται το αντίστοιχο linking και χρησιμοποιείται η σωστή υλοποίηση της συνάρτησης zing(). Για το εκτελέσιμο **zing** χρησιμοποιείται η

υλοποίηση της zing() που βρίσκεται στο zing.o, ενώ για το **zing2** χρησιμοποιείται η υλοποίηση στο zing2.o.

4.Το πρόβλημα που περιγράφεται στην ερώτηση είναι η χρονική επιβάρυνση σε ένα μεγάλο (σε κώδικα) πρόγραμμα όταν θέλουμε να κάνουμε μερική διόρθωση σε κάποια κομμάτια του. Το πρόβλημα αυτό θα μπορούσε να αντιμετωπιστεί με τη δημιουργία ενός νέου αρχείου ,για παράδειγμα *"filename.h"* , το οποίο θα περιέχει τη συνάρτηση στην οποία θέλουμε να πραγματοποιήσουμε αλλαγές .Στη συνέχεια, θα πρέπει να παράξουμε κατάλληλο *makefile* το οποίο θα υλοποιεί τόσο το linking για τα αρχεία μας σε ένα εκτελέσιμο όσο και το compiling για το καθένα ξεχωριστά. Έτσι , το υπόλοιπο μας πρόγραμμα τώρα δε θα έχει νέες αλλαγές άρα θα είναι ήδη μεταγλωττισμένο ,με αποτέλεσμα να κερδίζουμε σημαντικό χρόνο στο κομμάτι της μεταγλώττισης και να έχουμε πολύ μικρότερη καθυστέρηση.

5.Για το ερώτημα αυτό δημιουργήσαμε ένα δικό μας αρχείο ***foo.c*** που απλά εκτυπώνει κείμενο . Αφού κάναμε το compiling και παράξαμε το εκτελέσιμο foo, τρέξαμε στο terminal την εντολή που δίνεται στην εκφώνηση. Κάνοντας *ls* παρατηρούμε ότι το αρχείο foo.c εμφανίζεται πλέον ως εκτελέσιμο. Εκτελώντας την εντολή *vim foo.c* βλέπουμε ότι μέσα στο αρχείο δεν υπάρχει πλέον ο κώδικας που γράψαμε ,αλλά μη κατανοητοί από τον άνθρωπο χαρακτήρες .Ωστόσο ,εκτελώντας τόσο το ***foo*** όσο και το ***foo.c*** παρατηρούμε ότι "τρέχουν" κανονικά έχοντας μάλιστα και την ίδια έξοδο. Έτσι, συμπεραίνουμε ότι πιθανόν έχει γίνει κάποια επανεγγραφή της γλώσσας μηχανής που υπάρχει στο εκτελέσιμο αρχείο ***foo*** μέσα στο ***foo.c***.

ΑΣΚΗΣΗ 1.2

Source code:

doWrite.c

```
#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
void doWrite( int fd, const char *buff, int len) {
  ssize_t wcnt;
  size_t idx;
  idx = 0;
  do {
    wcnt = write(fd, buff+idx, len - idx);
    if ( wcnt == -1){
      perror("write");
      exit(1);
    }
    idx += wcnt;
  } while (idx < len);
}
```

doWrite.h

```
void doWrite( int , const char *, int );
```

write_file.c

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include "doWrite.h"
void write_file (int fd_out, const char *infile) {
  int fd_in = open (infile, O_RDONLY);
  char buff[4096];
  ssize_t  rcnt;
  for (;;) {
  rcnt = read (fd_in, buff, sizeof(buff)-1);
  if (rcnt==0)
    return;
  if (rcnt==-1) {
    perror ("reading file error");
    exit (1);
  }
  doWrite(fd_out, buff, rcnt);
  }
  close (fd_in);
}
```

write_file.h

```
void write_file (int,const char*);
```

fconc.c

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include "write_file.h"
int main (int argc, char ** argv) {
  char default_output []  = "fconc.out";
  if ((argc<3) || (argc>4)) {  //only one input file or more than 3 files given...
    fprintf(stderr, "Usage: %s infile1 infile2 [outfile (default:%s)]\n", argv[0], default_output);
    exit(1);
  }
  else {  //legal input...
    if (open(argv[1], O_RDONLY, S_IRUSR)<0) {
      perror(argv[1]);
      exit(1);
    }
    if (open(argv[2], O_RDONLY, S_IRUSR)<0) {
      perror(argv[2]);
      exit(1);
    }
    char* outfile;
    if (argc==4)
      outfile = argv[3];  //htan fconc.out anti gia defout...
    else
      outfile = default_output;
    int fd_out = open(outfile, O_CREAT|O_RDWR|O_TRUNC, S_IRUSR|S_IWUSR);
    write_file(fd_out, argv[1]);
    write_file(fd_out, argv[2]);
    close(fd_out);
    return 0;
  }
}
```

makefile

```
fconc: fconc.o write_file.o doWrite.o
	gcc -o fconc fconc.o write_file.o doWrite.o

fconc.o: fconc.c
	gcc -Wall -c fconc.c

write_file.o: write_file.c
	gcc -Wall -c write_file.c

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

Ερωτήσεις:

- Εκτελούμε με τη σειρά τις εντολές:
  $echo ‘Hello’ > A;
  $echo ‘World’ > B;
  $./fconc A B C
  $cat C
- και στη συνέχεια δοκιμάζουμε την εντολή: $strace cat C
  και εμφανίζονται στην οθόνη τα εξής:

```
execve("/bin/cat", ["cat", "C"], [/* 18 vars */]) = 0
brk(0)                                  = 0x1abc000
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) =
0x7f61acb91000
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) = 0x7f61acb89000
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) =
0x7f61ac5c8000
mprotect(0x7f61ac769000, 2097152, PROT_NONE) = 0
mmap(0x7f61ac969000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1a1000) = 0x7f61ac969000
mmap(0x7f61ac96f000, 14880, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f61ac96f000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f61acb88000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f61acb87000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f61acb86000
arch_prctl(ARCH_SET_FS, 0x7f61acb87700) = 0
mprotect(0x7f61ac969000, 16384, PROT_READ) = 0
mprotect(0x60b000, 4096, PROT_READ)     = 0
mprotect(0x7f61acb93000, 4096, PROT_READ) = 0
munmap(0x7f61acb89000, 30952)           = 0
brk(0)                                  = 0x1abc000
brk(0x1add000)                          = 0x1add000
open("/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=1859120, ...}) = 0
mmap(NULL, 1859120, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f61ac9c0000
close(3)                                = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1), ...}) = 0
open("C", O_RDONLY)                     = 3
fstat(3, {st_mode=S_IFREG|0600, st_size=12, ...}) = 0
fadvise64(3, 0, 0, POSIX_FADV_SEQUENTIAL) = 0
```

```
mmap(NULL, 270336, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f61ac586000
read(3, "Hello\nworld\n", 262144)       = 12
write(1, "Hello\nworld\n", 12Hello
world
)         = 12
read(3, "", 262144)                     = 0
munmap(0x7f61ac586000, 270336)          = 0
close(3)                                = 0
close(1)                                = 0
close(2)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

- αν εκτελέσουμε και την εντολή: strace ./fconc A B C
  στην οθόνη εμφανίζονται τα εξής:

```
execve("./fconc", ["./fconc", "A", "B", "C"], [/* 18 vars */]) = 0
brk(0)                                  = 0x1999000
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) =
0x7ff476d10000
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) = 0x7ff476d08000
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) =
0x7ff476747000
mprotect(0x7ff4768e8000, 2097152, PROT_NONE) = 0
mmap(0x7ff476ae8000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE,
3, 0x1a1000) = 0x7ff476ae8000
mmap(0x7ff476aee000, 14880, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7ff476aee000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7ff476d07000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7ff476d06000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7ff476d05000
arch_prctl(ARCH_SET_FS, 0x7ff476d06700) = 0
mprotect(0x7ff476ae8000, 16384, PROT_READ) = 0
mprotect(0x7ff476d12000, 4096, PROT_READ) = 0
munmap(0x7ff476d08000, 30952)           = 0
open("A", O_RDONLY)                     = 3
open("B", O_RDONLY)                     = 4
open("C", O_RDWR|O_CREAT|O_TRUNC, 0600) = 5
open("A", O_RDONLY)                     = 6
read(6, "Hello\n", 4095)                = 6
write(5, "Hello\n", 6)                  = 6
```

```
read(6, "", 4095)                       = 0
open("B", O_RDONLY)                             = 7
read(7, "world\n", 4095)                = 6
write(5, "world\n", 6)                  = 6
read(7, "", 4095)                       = 0
close(5)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```