ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΜ&ΜΥ
Λειτουργικά Συστήματα 1η Άσκηση
Ακ. έτος 2010-2011

Τμήμα Β, Ομάδα 3η

Γερακάρης Βασίλης Α.Μ.: 03108092
Λύρας Γρηγόρης Α.Μ.: 03109687

13 Νοεμβρίου 2011

1.1 Σύνδεση με αρχείο αντικειμένων

Ο πηγαίος κώδικας της main.c που κληθήκαμε να γράψουμε ήταν ο εξής:

```
#include "zing.h"


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

Στη συνέχεια δημιουργήσαμε το makefile για τη μεταγλώττιση του προγράμματος με τα εξής περιεχόμενα:

```
all:    zing
zing:   main.o
        gcc main.o zing.o -o zing -Wall -m32
main.o: main.c
        gcc -c main.c -o main.o -Wall -m32
clean:
        rm main.o zing
```

Τρέχοντας στο shell την εντολή make έχουμε την παρακάτω έξοδο

```
gcc -c main.c -o main.o -Wall -m32
gcc main.o zing.o -o main -Wall -m32
```

και τη δημιουργία των αρχείων main.o και του εκτελέσιμου main.
Εκτελώντας το main, το πρόγραμμα δίνει την παρακάτω έξοδο:

```
oslabb03 ~/code/zing $ ./main
Hello oslabb03!
```

Απαντήσεις στις θεωρητικές ερωτήσεις

1. Η επικεφαλίδα που χρησιμοποιήσαμε περιέχει τις απαραίτητες δηλώσεις για τη διεπαφή των αρχείων κώδικα του προγράμματος μας. Η άσκηση αυτή μας παρείχε το object file zing.o , αλλά η συνάρτηση zing( ) δηλώνεται στο zing.h, χωρίς τη χρήση του οποίου δε θα μπορούσαμε να την καλέσουμε επιτυχώς στη main.

2. Απαντήθηκε παραπάνω.

3. Αντί να έχουμε όλες τις συναρτήσεις σε ένα αρχείο θα μπορούσαμε να χρησιμοποιούμε ένα αρχείο για κάθε συνάρτηση με το αντίστοιχο αρχείο επικεφαλίδας. Έτσι η μεταγλώτισση θα γίνεται για κάθε αρχείο χωριστά. Συνεπώς αλλάζοντας ένα αρχείο ο χρόνος μεταγλώττισης θα είναι μικρότερος. Επίσης με αυτό τον τρόπο μπορούμε να κάνουμε παράλληλη μεταγώττιση αρχείων σε περίπτωση που το σύστημα μας δίνει αυτή τη δυνατότητα.

4. Στην περίπτωση αυτή βλέπουμε πως το αρχείο foo.c μεταγλωττίστηκε στο αρχείο foo.c. Τώρα πλέον το foo.c είναι το εκτελέσιμο και ο πηγαίος κώδικας χάθηκε.

1.2 Συνένωση δύο αρχείων σε τρίτο

Ο πηγαίος κώδικας που χρησιμοποιήσαμε αρχικά ήταν ο εξής:

```
/* -.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.

* File Name : fconc.h

* Last Modified : Fri 11 Nov 2011 06:21:15 PM EET

* Created By : Greg Liras <gregliras@gmail.com>

* Created By : Vasilis Gerakaris <vgerak@gmail.com>

_.-._._._._._._._._._._._._._._._._._._._._._._._._._._._._.*/

void doWrite(int fd, const char *buff, int len);
void write_file(int fd, const char *infile);
void print_err(const char *p);
```

```
/* -.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.

* File Name : fconc.c

* Last Modified : Fri 11 Nov 2011 09:17:03 PM EET

* Created By : Greg Liras <gregliras@gmail.com>

* Created By : Vasilis Gerakaris <vgerak@gmail.com>

_.-._._._._._._._._._._._._._._._._._._._._._._._._._._._._.*/

#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>

#include "fconc.h"

#ifndef BUFFER_SIZE
#define BUFFER_SIZE 1024
#endif

int main(int argc, char ** argv)
{
  int OUT;
  int W_FLAGS = O_CREAT | O_WRONLY | O_TRUNC;
  int C_PERMS = S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH ;
  if (argc < 3)
  {
    print_err("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
  }
  if (argc > 3)
  {
    OUT = open(argv[3],W_FLAGS,C_PERMS);
  }
  else
  {
    OUT = open("fconc.out",W_FLAGS,C_PERMS);
  }
  if (OUT < 0)
  {
    print_err("Error handling output file\n");
  }


  write_file(OUT,argv[1]);
  write_file(OUT,argv[2]);


  exit(EXIT_SUCCESS);
}

void doWrite(int fd,const char *buff,int len)
{
  int written;
  do
```

```
  {
    if ( (written = write(fd,buff,len)) < 0 )
    {
      print_err("Error in writing\n");
    }
  } while(written < len );
}


void write_file(int fd,const char *infile)
{
  int A;
  char buffer[BUFFER_SIZE];
  int chars_read=0;
  A = open(infile,O_RDONLY);
  if (A ==-1)
  {
    print_err("No such file or directory\n");
  }
  //time to read
  while( (chars_read = read(A,buffer,BUFFER_SIZE)) > 0)
  {
    //and write
    doWrite(fd,buffer,chars_read);
  }
  if ( chars_read == -1 )
  {
    print_err("Read Error\n");
  }

  //ok close
  if ( close(A) == - 1 )
  {
    print_err("Close Error\n");
  }


}

void print_err(const char *p)
{
  int len = 0;
  const char *b = p;
  while( *b++ != '\0' ) len++;
  doWrite(2,p,len); //doWrite to stderr
  exit(-1);
}
```

```
all:                fconc
fconc:               fconc.o
        gcc fconc.o -o fconc
fconc.o:         fconc.c fconc.h
        gcc -c fconc.c -o fconc.o -Wall
.PHONY: clean test
clean:
        rm fconc.o fconc C
test:
        ./fconc A B C
strace:
        strace -o strace_outfile ./fconc A B C

```

Η έξοδος της strace είναι η παρακάτω:

```
execve("./fconc", ["./fconc", "A", "B", "C"], [/* 48 vars */]) = 0
brk(0)                                  = 0x9525000
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb76ef000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=118009, ...}) = 0
mmap2(NULL, 118009, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb76d2000
close(3)                                = 0
open("/lib/libc.so.6", O_RDONLY)        = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\0\244\1\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1429996, ...}) = 0
mmap2(NULL, 1440296, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb7572000
```

```
mprotect(0xb76cb000, 4096, PROT_NONE)   = 0
mmap2(0xb76cc000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) = 0
     xb76cc000
mmap2(0xb76cf000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
     xb76cf000
close(3)                                = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7571000
set_thread_area({entry_number:-1 -> 6, base_addr:0xb75716c0, limit:1048575, seg_32bit:1, contents:0,
      read_exec_only:0, limit_in_pages:1, seg_not_present:0, useable:1}) = 0
mprotect(0xb76cc000, 8192, PROT_READ)   = 0
mprotect(0x8049000, 4096, PROT_READ)    = 0
mprotect(0xb770d000, 4096, PROT_READ)   = 0
munmap(0xb76d2000, 118009)              = 0
open("C", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
open("A", O_RDONLY)                     = 4
read(4, "asdf\n", 1024)                 = 5
write(3, "asdf\n", 5)                   = 5
read(4, "", 1024)                       = 0
close(4)                                = 0
open("B", O_RDONLY)                     = 4
read(4, "lkjh\n", 1024)                 = 5
write(3, "lkjh\n", 5)                   = 5
read(4, "", 1024)                       = 0
close(4)                                = 0
exit_group(0)                           = ?
```

```
execve("./fconc", ["./fconc", "A", "B", "C"], [/* 48 vars */]) = 0
brk(0)                                  = 0x9525000
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb76ef000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=118009, ...}) = 0
mmap2(NULL, 118009, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb76d2000
close(3)                                = 0
open("/lib/libc.so.6", O_RDONLY)        = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\0\244\1\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1429996, ...}) = 0
mmap2(NULL, 1440296, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb7572000
mprotect(0xb76cb000, 4096, PROT_NONE)   = 0
mmap2(0xb76cc000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) = 0xb76cc000
mmap2(0xb76cf000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xb76cf000
close(3)                                = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7571000
set_thread_area({entry_number:-1 -> 6, base_addr:0xb75716c0, limit:1048575, seg_32bit:1, contents:0, read_exec_only:0, limit_in
mprotect(0xb76cc000, 8192, PROT_READ)   = 0
mprotect(0x8049000, 4096, PROT_READ)    = 0
mprotect(0xb770d000, 4096, PROT_READ)   = 0
munmap(0xb76d2000, 118009)              = 0
open("C", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
open("A", O_RDONLY)                     = 4
read(4, "asdf\n", 1024)                 = 5
write(3, "asdf\n", 5)                   = 5
read(4, "", 1024)                       = 0
close(4)                                = 0
open("B", O_RDONLY)                     = 4
read(4, "lkjh\n", 1024)                 = 5
write(3, "lkjh\n", 5)                   = 5
read(4, "", 1024)                       = 0
close(4)                                = 0
exit_group(0)                           = ?
```

1.3 Bonus

Ο πηγαίος κώδικας που χρησιμοποιήσαμε τελικά ήταν ο εξής:

```
/* -.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.

* File Name : fconc.h

* Last Modified : Fri 11 Nov 2011 06:21:35 PM EET

* Created By : Greg Liras <gregliras@gmail.com>

* Created By : Vasilis Gerakaris <vgerak@gmail.com>
```

```
10
11  _.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.*/
12
13  void doWrite(int fd, const char *buff, int len);
14  void write_file(int fd, const char *infile);
15  void print_err(const char *p);

1   /* -.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.

2
3   * File Name : fconc.c
4
5   * Last Modified : Fri 11 Nov 2011 09:19:52 PM EET
6
7   * Created By : Greg Liras <gregliras@gmail.com>
8
9   * Created By : Vasilis Gerakaris <vgerak@gmail.com>
10
11  _.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.-._.*/
12
13  #include <unistd.h>
14  #include <fcntl.h>
15  #include <stdlib.h>
16
17  #include "fconc.h"
18
19  #ifndef BUFFER_SIZE
20  #define BUFFER_SIZE 1024
21  #endif
22
23  int main(int argc, char ** argv)
24  {
25    int OUT;
26    int W_FLAGS = O_CREAT | O_WRONLY | O_TRUNC;
27    int C_PERMS = S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH ;
28    int counter=0;
29    if (argc < 3)
30    {
31      print_err("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
32    }
33    if (argc > 3)
34    {
35      OUT = open(argv[argc-1],W_FLAGS,C_PERMS);
36    }
37    else
38    {
39      OUT = open("fconc.out",W_FLAGS,C_PERMS);
40    }
41    if (OUT < 0)
42    {
43      print_err("Error handling output file\n");
44    }
45    for(counter = 1 ; counter < argc-1 ; counter++ )
46    {
47      write_file(OUT,argv[counter]);
48    }
49
50    exit(EXIT_SUCCESS);
51  }
52
53  void doWrite(int fd,const char *buff,int len)
54  {
55    int written;
56    do
57    {
58      if ( (written = write(fd,buff,len)) < 0 )
59      {
60        print_err("Error in writing\n");
61      }
62    } while(written < len );
63  }
64
65
66  void write_file(int fd,const char *infile)
67  {
68    int A;
69    char buffer[BUFFER_SIZE];
```

```
  int chars_read=0;
  A = open(infile,O_RDONLY);
  if (A ==-1)
  {
    print_err("No such file or directory\n");
  }
  //time to read
  while( (chars_read = read(A,buffer,BUFFER_SIZE)) > 0)
  {
    //and write
    doWrite(fd,buffer,chars_read);
  }
  if ( chars_read == -1 )
  {
    print_err("Read Error\n");
  }

  //ok close
  if ( close(A) == - 1 )
  {
    print_err("Close Error\n");
  }


}

void print_err(const char *p)
{
  int len = 0;
  const char *b = p;
  while( *b++ != '\0' ) len++;
  doWrite(2,p,len); //doWrite to stderr
  exit(-1);
}
```

```
all:                fconc
fconc:               fconc.o
        gcc fconc.o -o fconc
fconc.o:         fconc.c fconc.h
        gcc -c fconc.c -o fconc.o -Wall
.PHONY: clean test
clean:
        rm fconc.o fconc C
test:
        ./fconc A B C D E F
strace:
        strace -o strace_outfile ./fconc A B C D E F

```

Η έξοδος της strace είναι η παρακάτω:

```
execve("./fconc", ["./fconc", "A", "B", "C", "D", "E", "F"], [/* 48 vars */]) = 0
brk(0)                                  = 0x970b000
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb774e000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=118009, ...}) = 0
mmap2(NULL, 118009, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb7731000
close(3)                                = 0
open("/lib/libc.so.6", O_RDONLY)        = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\0\0\244\1\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1429996, ...}) = 0
mmap2(NULL, 1440296, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb75d1000
mprotect(0xb772a000, 4096, PROT_NONE)   = 0
mmap2(0xb772b000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) = 0
    xb772b000
mmap2(0xb772e000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
    xb772e000
close(3)                                = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb75d0000
set_thread_area({entry_number:-1 -> 6, base_addr:0xb75d06c0, limit:1048575, seg_32bit:1, contents:0,
     read_exec_only:0, limit_in_pages:1, seg_not_present:0, useable:1}) = 0
mprotect(0xb772b000, 8192, PROT_READ)   = 0
mprotect(0x8049000, 4096, PROT_READ)    = 0
mprotect(0xb776c000, 4096, PROT_READ)   = 0
munmap(0xb7731000, 118009)              = 0
```

```
open("F", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
open("A", O_RDONLY)                       = 4
read(4, "asdf\n", 1024)                   = 5
write(3, "asdf\n", 5)                     = 5
read(4, "", 1024)                         = 0
close(4)                                  = 0
open("B", O_RDONLY)                       = 4
read(4, "lkjh\n", 1024)                   = 5
write(3, "lkjh\n", 5)                     = 5
read(4, "", 1024)                         = 0
close(4)                                  = 0
open("C", O_RDONLY)                       = 4
read(4, "test\n", 1024)                   = 5
write(3, "test\n", 5)                     = 5
read(4, "", 1024)                         = 0
close(4)                                  = 0
open("D", O_RDONLY)                       = 4
read(4, "test2\n", 1024)                  = 6
write(3, "test2\n", 6)                    = 6
read(4, "", 1024)                         = 0
close(4)                                  = 0
open("E", O_RDONLY)                       = 4
read(4, "test3\ntest4\n", 1024)           = 12
write(3, "test3\ntest4\n", 12)            = 12
read(4, "", 1024)                         = 0
close(4)                                  = 0
exit_group(0)                             = ?
```

Με τη χρήση του gdb στα αρχεία main.o και zing είχαμε την παρακάτω έξοδο.

```
Dump of assembler code for function main:
   0x00000000 <+0>:     push   %ebp
   0x00000001 <+1>:     mov    %esp,%ebp
   0x00000003 <+3>:     and    $0xfffffff0,%esp
   0x00000006 <+6>:     call   0x7 <main+7>
   0x0000000b <+11>:    mov    $0x0,%eax
   0x00000010 <+16>:    mov    %ebp,%esp
   0x00000012 <+18>:    pop    %ebp
   0x00000013 <+19>:    ret
End of assembler dump.
```

```
Dump of assembler code for function main:
   0x08048424 <+0>:     push   %ebp
   0x08048425 <+1>:     mov    %esp,%ebp
   0x08048427 <+3>:     and    $0xfffffff0,%esp
   0x0804842a <+6>:     call   0x8048438 <zing>
   0x0804842f <+11>:    mov    $0x0,%eax
   0x08048434 <+16>:    mov    %ebp,%esp
   0x08048436 <+18>:    pop    %ebp
   0x08048437 <+19>:    ret
End of assembler dump.
```