



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

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

Τμήμα Β, Ομάδα 3^η

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:

```
1  #include "zing.h"
2
3
4  int main(int argc, char ** argv)
5  {
6      zing();
7      return 0;
8  }
```

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

```
1  all:    zing
2  zing:   main.o
3          gcc main.o zing.o -o zing -Wall -m32
4  main.o: main.c
5          gcc -c main.c -o main.o -Wall -m32
6  clean:
7          rm main.o zing
```

Τρέχοντας στο shell την εντολή make έχουμε την παρακάτω έξοδο

```
1  gcc -c main.c -o main.o -Wall -m32
2  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:

```
1  oslab03 ~/code/zing $ ./main
2  Hello oslab03!
```

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 μεταγλώττιση 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 μεταγλώττιση 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.o. Τώρα πλέον το foo.o είναι το εκτελέσιμο και ο πηγαίος κώδικας χάθηκε.

1.2 Συνένωση δύο αρχείων σε τρίτο

Ο παρακάτω κώδικας που χρησιμοποιήσαμε αρχικά ήταν ο εξής:

```
1  /* .....
2
3  * File Name : fconc.h
4
5  * Last Modified : Fri 11 Nov 2011 06:21:15 PM EET
6
7  * Created By : Greg Liras <gregliras@gmail.com>
8
9  * 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);
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
```

```
1  /* .....
2
3  * File Name : fconc.c
4
5  * Last Modified : Fri 11 Nov 2011 09:17:03 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     if (argc < 3)
29     {
30         print_err("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
31     }
32     if (argc > 3)
33     {
34         OUT = open(argv[3],W_FLAGS,C_PERMS);
35     }
36     else
37     {
38         OUT = open("fconc.out",W_FLAGS,C_PERMS);
39     }
40     if (OUT < 0)
41     {
42         print_err("Error handling output file\n");
43     }
44
45
46     write_file(OUT,argv[1]);
47     write_file(OUT,argv[2]);
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,buffer,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];
70     int chars_read=0;
71     A = open(infile,O_RDONLY);
72     if (A ==-1)
73     {
74         print_err("No such file or directory\n");
75     }
76     //time to read
77     while( (chars_read = read(A,buffer,BUFFER_SIZE)) > 0)
78     {
79         //and write
80         doWrite(fd,buffer,chars_read);
81     }
82     if ( chars_read == -1 )
83     {
84         print_err("Read Error\n");
85     }
86
87     //ok close
88     if ( close(A) == - 1 )
89     {
90         print_err("Close Error\n");
91     }
92
93 }
94
95
96 void print_err(const char *p)
97 {
98     int len = 0;
99     const char *b = p;
100     while( *b++ != '\0' ) len++;
101     doWrite(2,p,len); //doWrite to stderr
102     exit(-1);
103 }

```

```

1 all:                fconc
2 fconc:              fconc.o
3                     gcc fconc.o -o fconc
4 fconc.o:            fconc.c fconc.h
5                     gcc -c fconc.c -o fconc.o -Wall
6 .PHONY: clean test
7 clean:
8                     rm fconc.o fconc C
9 test:
10                    ./fconc A B C
11 strace:
12                    strace -o strace_outfile ./fconc A B C
13

```

Η έξοδος της strace είναι η παρακάτω:

```

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

```

```

13 mprotect(0xb76cb000, 4096, PROT_NONE) = 0
14 mmap2(0xb76cc000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) = 0
    x76cc000
15 mmap2(0xb76cf000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
    x76cf000
16 close(3) = 0
17 mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7571000
18 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
19 mprotect(0xb76cc000, 8192, PROT_READ) = 0
20 mprotect(0x8049000, 4096, PROT_READ) = 0
21 mprotect(0xb770d000, 4096, PROT_READ) = 0
22 munmap(0xb76d2000, 118009) = 0
23 open("C", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
24 open("A", O_RDONLY) = 4
25 read(4, "asdf\n", 1024) = 5
26 write(3, "asdf\n", 5) = 5
27 read(4, "", 1024) = 0
28 close(4) = 0
29 open("B", O_RDONLY) = 4
30 read(4, "lkjh\n", 1024) = 5
31 write(3, "lkjh\n", 5) = 5
32 read(4, "", 1024) = 0
33 close(4) = 0
34 exit_group(0) = ?

1 execve("./fconc", ["/fconc", "A", "B", "C"], [/* 48 vars */]) = 0
2 brk(0) = 0x9525000
3 mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb76ef000
4 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
5 open("/etc/ld.so.cache", O_RDONLY) = 3
6 fstat64(3, {st_mode=S_IFREG|0644, st_size=118009, ...}) = 0
7 mmap2(NULL, 118009, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb76d2000
8 close(3) = 0
9 open("/lib/libc.so.6", O_RDONLY) = 3
10 read(3, "\177ELF\1\1\0\0\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
11 fstat64(3, {st_mode=S_IFREG|0755, st_size=1429996, ...}) = 0
12 mmap2(NULL, 1440296, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb7572000
13 mprotect(0xb76cb000, 4096, PROT_NONE) = 0
14 mmap2(0xb76cc000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) = 0xb76cc000
15 mmap2(0xb76cf000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xb76cf000
16 close(3) = 0
17 mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7571000
18 set_thread_area({entry_number:-1 -> 6, base_addr:0xb75716c0, limit:1048575, seg_32bit:1, contents:0, read_exec_only:0, limit_in_
19 mprotect(0xb76cc000, 8192, PROT_READ) = 0
20 mprotect(0x8049000, 4096, PROT_READ) = 0
21 mprotect(0xb770d000, 4096, PROT_READ) = 0
22 munmap(0xb76d2000, 118009) = 0
23 open("C", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
24 open("A", O_RDONLY) = 4
25 read(4, "asdf\n", 1024) = 5
26 write(3, "asdf\n", 5) = 5
27 read(4, "", 1024) = 0
28 close(4) = 0
29 open("B", O_RDONLY) = 4
30 read(4, "lkjh\n", 1024) = 5
31 write(3, "lkjh\n", 5) = 5
32 read(4, "", 1024) = 0
33 close(4) = 0
34 exit_group(0) = ?

```

1.3 Bonus

1. Η εντολή `strace` μας έδωσε την ακόλουθη έξοδο:

```

1 execve("/usr/bin/strace", ["/strace"], [/* 45 vars */]) = 0
2 brk(0) = 0x94ed000
3 mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7809000
4 access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
5 open("/etc/ld.so.cache", O_RDONLY) = 3
6 fstat64(3, {st_mode=S_IFREG|0644, st_size=118009, ...}) = 0
7 mmap2(NULL, 118009, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb77ec000
8 close(3) = 0
9 open("/lib/libc.so.6", O_RDONLY) = 3

```

```

9 read(3, "\177ELF\1\1\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
10 fstat64(3, {st_mode=S_IFREG|0755, st_size=1429996, ...}) = 0
11 mmap2(NULL, 1440296, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb768c000
12 mprotect(0xb77e5000, 4096, PROT_NONE) = 0
13 mmap2(0xb77e6000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) =
    0xb77e6000
14 mmap2(0xb77e9000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
    xb77e9000
15 close(3) = 0
16 mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb768b000
17 set_thread_area({entry_number:-1 -> 6, base_addr:0xb768b6c0, limit:1048575, seg_32bit:1,
    contents:0, read_exec_only:0, limit_in_pages:1, seg_not_present:0, useable:1}) = 0
18 mprotect(0xb77e6000, 8192, PROT_READ) = 0
19 mprotect(0x8082000, 4096, PROT_READ) = 0
20 mprotect(0xb7827000, 4096, PROT_READ) = 0
21 munmap(0xb77ec000, 118009) = 0
22 brk(0) = 0x94ed000
23 brk(0x950e000) = 0x950e000
24 write(2, "usage: strace [-CdffhiqrsttvVx"... , 1731) = 1731
25 exit_group(1) = ?

```

```

1  Dump of assembler code for function main:
2      0x00000000 <+0>:      push    %ebp
3      0x00000001 <+1>:      mov     %esp,%ebp
4      0x00000003 <+3>:      and     0xffffffff0,%esp
5      0x00000006 <+6>:      call    0x7 <main+7>
6      0x0000000b <+11>:     mov     $0x0,%eax
7      0x00000010 <+16>:     mov     %ebp,%esp
8      0x00000012 <+18>:     pop     %ebp
9      0x00000013 <+19>:     ret
10 End of assembler dump.
```

3. Ο πηγαίος κώδικας που χρησιμοποιήσαμε τελικά ήταν ο εξής:

```

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];
70      int chars_read=0;
71      A = open(infile,O_RDONLY);
72      if (A ==-1)
73      {
74          print_err("No such file or directory\n");
75      }
76      //time to read
77      while( (chars_read = read(A,buffer,BUFFER_SIZE)) > 0)
78      {
79          //and write
80          doWrite(fd,buffer,chars_read);
81      }
82      if ( chars_read == -1 )
83      {
84          print_err("Read Error\n");
85      }
86
87      //ok close

```

```

88     if ( close(A) == - 1 )
89     {
90         print_err("Close Error\n");
91     }
92
93
94 }
95
96 void print_err(const char *p)
97 {
98     int len = 0;
99     const char *b = p;
100     while( *b++ != '\0' ) len++;
101     doWrite(2,p,len); //doWrite to stderr
102     exit(-1);
103 }

```

```

1  all:                fconc
2  fconc:              fconc.o
3                      gcc fconc.o -o fconc
4  fconc.o:            fconc.c fconc.h
5                      gcc -c fconc.c -o fconc.o -Wall
6  .PHONY: clean test
7  clean:
8                      rm fconc.o fconc C
9  test:
10                     ./fconc A B C D E F
11  strace:
12                     strace -o strace_outfile ./fconc A B C D E F
13

```

Η έξοδος της strace είναι η παρακάτω:

```

1  execve("./fconc", ["/fconc", "A", "B", "C", "D", "E", "F"], [/* 48 vars */]) = 0
2  brk(0) = 0x970b000
3  mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb774e000
4  access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
5  open("/etc/ld.so.cache", O_RDONLY) = 3
6  fstat64(3, {st_mode=S_IFREG|0644, st_size=118009, ...}) = 0
7  mmap2(NULL, 118009, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb7731000
8  close(3) = 0
9  open("/lib/libc.so.6", O_RDONLY) = 3
10 read(3, "\177ELF\1\1\1\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
11 fstat64(3, {st_mode=S_IFREG|0755, st_size=1429996, ...}) = 0
12 mmap2(NULL, 1440296, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb75d1000
13 mprotect(0xb772a000, 4096, PROT_NONE) = 0
14 mmap2(0xb772b000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) =
    0xb772b000
15 mmap2(0xb772e000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
    xb772e000
16 close(3) = 0
17 mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb75d0000
18 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
19 mprotect(0xb772b000, 8192, PROT_READ) = 0
20 mprotect(0x8049000, 4096, PROT_READ) = 0
21 mprotect(0xb776c000, 4096, PROT_READ) = 0
22 munmap(0xb7731000, 118009) = 0
23 open("F", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
24 open("A", O_RDONLY) = 4
25 read(4, "asdf\n", 1024) = 5
26 write(3, "asdf\n", 5) = 5
27 read(4, "", 1024) = 0
28 close(4) = 0
29 open("B", O_RDONLY) = 4
30 read(4, "lkjh\n", 1024) = 5
31 write(3, "lkjh\n", 5) = 5
32 read(4, "", 1024) = 0
33 close(4) = 0
34 open("C", O_RDONLY) = 4
35 read(4, "test\n", 1024) = 5
36 write(3, "test\n", 5) = 5
37 read(4, "", 1024) = 0
38 close(4) = 0
39 open("D", O_RDONLY) = 4
40 read(4, "test2\n", 1024) = 6

```


41	<code>write(3, "test2\n", 6)</code>	<code>= 6</code>
42	<code>read(4, "", 1024)</code>	<code>= 0</code>
43	<code>close(4)</code>	<code>= 0</code>
44	<code>open("E", O_RDONLY)</code>	<code>= 4</code>
45	<code>read(4, "test3\ntest4\n", 1024)</code>	<code>= 12</code>
46	<code>write(3, "test3\ntest4\n", 12)</code>	<code>= 12</code>
47	<code>read(4, "", 1024)</code>	<code>= 0</code>
48	<code>close(4)</code>	<code>= 0</code>
49	<code>exit_group(0)</code>	<code>= ?</code>