

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

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

Τμήμα Β, Ομάδα 3η

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

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:

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. ο 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 : Sun 13 Nov 2011 05:31:09 PM EET
    * Created By : Greg Liras <gregliras@gmail.com>
    * Created By : Vasilis Gerakaris <ugerak@gmail.com>
10
11
   _._._._.*/
12
   #ifndef FCONC H
13
14
   #define FCONC_H
15
   #ifndef BUFFER_SIZE
16
   #define BUFFER_SIZE 1024
17
   #endif //BUFFER SIZE
18
19
   #include <unistd.h>
   #include <fcntl.h>
21
   #include <stdlib.h>
22
   void doWrite(int fd, const char *buff, int len);
24
25
   void write_file(int fd, const char *infile);
   void print_err(const char *p);
26
27
   #endif //FCONC_H
   /* -.-.-.-.-.-.
    * File Name : fconc.c
   * Last Modified : Thu 17 Nov 2011 03:42:32 AM EET
    * Created By : Greg Liras <gregliras@gmail.com>
   * Created By : Vasilis Gerakaris <ugerak@gmail.com>
10
    _._....*/
11
12
13
   #include "fconc.h"
14
   int main(int argc, char ** argv)
15
16
     int OUT;
17
18
     int TMP:
     int W_FLAGS = O_CREAT | O_WRONLY | O_TRUNC;
19
     int C_PERMS = S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH ;
20
     struct flock lock;
     if (argc < 3)
22
23
     {
      print_err("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
24
25
     TMP = open("/tmp/fconc.out.tmp", W_FLAGS, C_PERMS);
26
27
     {
28
      print_err("Error handling tmp file, is another instance running?\n");
29
30
31
     fcntl(TMP,F_GETLK,lock); //get lock info on fd
     lock.l_type = F_WRLCK;
                              //set lock to write lock
32
     fcntl(TMP,F_SETLK,lock); //set the lock on fd
33
     write_file(TMP,argv[1]); //write on fd
     write_file(TMP,argv[2]);
35
     lock.l_type = F_UNLCK;
                              //set lock to unlock
36
     fcntl(TMP,F_SETLK,lock);
                              //set the lock on fd
     close(TMP);
                              //close fd
38
     if (argc > 3)
39
       OUT = open(argv[3], W_FLAGS, C_PERMS);
41
42
     else
43
     {
```

```
OUT = open("fconc.out", W_FLAGS, C_PERMS);
45
46
       if (OUT < 0)
47
 48
         print_err("Error handling output file\n");
49
       fcntl(OUT,F_GETLK,lock);
51
       lock.l_type = F_WRLCK;
52
       fcntl(OUT,F_SETLK,lock);
53
       write_file(OUT,"/tmp/fconc.out.tmp");
54
 55
       lock.l_type = F_UNLCK;
       fcntl(OUT,F_SETLK,lock);
 56
       close(OUT);
57
        if (unlink("/tmp/fconc.out.tmp") != 0)
 58
59
          print_err("Error deleting temporary file, please remove /tmp/fconc.out.tmp\n");
 60
61
       exit(EXIT_SUCCESS);
62
 63
64
     void doWrite(int fd,const char *buff,int len)
65
67
       int written;
68
       do
 69
          if ( (written = write(fd,buff,len)) < 0 )</pre>
70
 71
            print_err("Error in writing\n");
72
73
 74
       } while(written < len );</pre>
75
76
 77
     void write_file(int fd,const char *infile)
78
 79
       int A;
 80
       char buffer[BUFFER_SIZE];
81
       int chars_read=0;
       struct flock lock;
 83
       A = open(infile,O_RDONLY);
84
 85
       if (A ==-1)
       {
86
         print_err("No such file or directory\n");
87
 88
       fcntl(A,F_GETLK,lock); //get lock info on A
lock.l_type = F_RDLCK; //set lock to read lock
fcntl(A,F_SETLK,lock); //set lock on A
89
 90
 91
       //time\ to\ read
92
 93
       while( (chars_read = read(A,buffer,BUFFER_SIZE)) > 0)
94
95
          //and write
          doWrite(fd,buffer,chars_read);
 96
97
 98
       if ( chars\_read == -1 )
99
       {
         print_err("Read Error\n");
100
101
       lock.l_type = F_UNLCK; //set lock to unlock
102
       fcntl(A,F_SETLK,lock); //set lock on A
103
        //ok close
104
       if (close(A) == -1)
105
106
          print_err("Close Error\n");
107
108
     }
109
110
111
     void print_err(const char *p)
112
       int len = 0;
113
114
        const char *b = p;
       while( *b++ != '\0' ) len++;
115
       {\tt doWrite(2,p,len);} \ /\!/ {\tt doWrite to stderr}
116
       exit(-1);
117
118
```

```
all:
 1
           fconc:
                                                         fconc.o
2
                                  gcc fconc.o -o fconc
                                          fconc.c fconc.h
                                  gcc -c fconc.c -o fconc.o -Wall
             .PHONY: clean test strace
           clean:
 7
                                   rm fconc.o fconc C
                                   ./fconc A B C
10
11
            strace:
                                   strace -o strace_outfile ./fconc A B C
12
13
                      Η έξοδος της strace είναι η παρακάτω:
            execve("./fconc", ["./fconc", "A", "B", "C"], [/* 47 \text{ vars } */]) = 0
           brk(0)
                                                                                                                                                 = 0x8365000
  2
           mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7841000
  3
            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=102531, ...}) = 0
           mmap2(NULL, 102531, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb7827000
           close(3)
           open("/lib/libc.so.6", O_RDONLY)
           read(3, "177ELF11110000000000000030301000000244100040000"..., 512) = 512
           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) = 0xb76c7000
            mprotect(0xb7820000, 4096, PROT_NONE)
                                                                                                                                                 = 0
           mmap2(0xb7821000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) = 0
 14
                          xb7821000
           mmap2(0xb7824000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
 15
                          xb7824000
            close(3)
                                                                                                                                                  = 0
           mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb76c6000
 17
           set_thread_area({entry_number:-1 -> 6, base_addr:0xb76c66c0, limit:1048575, seg_32bit:1, contents:0,
 18
                             read_exec_only:0, limit_in_pages:1, seg_not_present:0, useable:1}) = 0
           mprotect(0xb7821000, 8192, PROT_READ) = 0
 19
           mprotect(0x8049000, 4096, PROT_READ)
 20
                                                                                                                                                 = 0
21
           mprotect(0xb785f000, 4096, PROT_READ)
           munmap(0xb7827000, 102531)
                                                                                                                                                  = 0
 22
            open("/tmp/fconc.out.tmp", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
 23
           fcntl64(3, F_GETLK, {type=0xffffffba /* F_??? */, whence=0xffffffff /* SEEK_??? */, start
24
                          =1958774271, len=139823908, pid=404042597}) = -1 EINVAL (Invalid argument)
           fcntl64(3, F_SETLK, {type=0x4589 /* F_??? */, whence=0xffffe9b8 /* SEEK_??? */, start=-1006, len
                         =-1007971443}) = -1 EINVAL (Invalid argument)
           open("A", O_RDONLY)
                                                                                                                                                = 4
 26
            fcnt164(4, F_GETLK, {...})
                                                                                                                                                 = -1 EFAULT (Bad address)
 27
           fcnt164(4, F_SETLK, {...})
                                                                                                                                                 = -1 EFAULT (Bad address)
 28
           read(4, "test\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest
           write(3, "test\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntes
 30
           read(4, "", 1024)
 31
                                                                                                                                                 = 0
           fcnt164(4, F_SETLK, {...})
                                                                                                                                                 = -1 EFAULT (Bad address)
 32
           close(4)
                                                                                                                                                 = 0
 33
           open("B", O_RDONLY)
                                                                                                                                                 = 4
 34
          fcnt164(4, F_GETLK, {...})
                                                                                                                                                 = -1 EFAULT (Bad address)
 35
          fcnt164(4, F_SETLK, {...})
                                                                                                                                                 = -1 EFAULT (Bad address)
 36
 37
           read(4, "lkjh\n", 1024)
          write(3, "lkjh\n", 5)
 38
          read(4, "", 1024)
 39
           fcnt164(4, F_SETLK, {...})
                                                                                                                                                 = -1 EFAULT (Bad address)
           close(4)
 41
           fcntl64(3, F_SETLK, {type=0xffffb845 /* F_??? */, whence=0x12e9 /* SEEK_??? */, start=-1912602628,
 42
                          len=-37491821}) = -1 EINVAL (Invalid argument)
           close(3)
 43
            open("C", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
            fcntl64(3, F_GETLK, {type=0xffffb845 /* F_??? */, whence=0x12e9 /* SEEK_??? */, start=-1912602628,
 45
                         len=-37491821, pid=3296037375}) = -1 EINVAL (Invalid argument)
            fcntl64(3, F_SETLK, {type=0x4589 /* F_??? */, whence=0xffffe9b8 /* SEEK_??? */, start=-1006, len
                          =-1007971443}) = -1 EINVAL (Invalid argument)
           open("/tmp/fconc.out.tmp", O_RDONLY)
 47
                                                                                                                                               = 4
          fcnt164(4, F_GETLK, {...})
                                                                                                                                                 = -1 EFAULT (Bad address)
           fcnt164(4, F_SETLK, {...})
                                                                                                                                                = -1 EFAULT (Bad address)
 49
           read(4, "test\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest
          write(3, "test\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntest\ntes
 51
 s2 read(4, "", 1024)
```

1.3 Bonus

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

```
execve("/usr/bin/strace", ["strace"], [/* 45 \text{ vars } */]) = 0
                                           = 0x94ed000
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)
   open("/etc/ld.so.cache", O_RDONLY)
                                          = 3
6 fstat64(3, {st_mode=S_IFREG|0644, st_size=118009, ...}) = 0
mmap2(NULL, 118009, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb77ec000
                                          = 0
8 close(3)
   open("/lib/libc.so.6", O_RDONLY)
                                           = 3
10 read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\0\244\1\0004\0\0\0"..., 512) = 512
II fstat64(3, {st_mode=S_IFREG|0755, st_size=1429996, ...}) = 0
   \verb|mmap2(NULL, 1440296, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb768c000|
mprotect(0xb77e5000, 4096, PROT_NONE)
mmap2(0xb77e6000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) =
        0xb77e6000
  mmap2(0xb77e9000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
15
       xb77e9000
  close(3)
                                           = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb768b000
18 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
mprotect(0xb77e6000, 8192, PROT_READ) = 0
20 mprotect(0x8082000, 4096, PROT_READ)
                                           = 0
21 mprotect(0xb7827000, 4096, PROT_READ)
                                           = 0
                                           = 0
22 munmap(0xb77ec000, 118009)
                                           = 0x94ed000
23 brk(0)
24 brk(0x950e000)
                                           = 0x950e000
write(2, "usage: strace [-CdDffhiqrtttTvVx"..., 1731) = 1731
26 exit_group(1)
```

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

```
Dump of assembler code for function main:
      0x00000000 <+0>: push
                                  %ebp
      0x00000001 <+1>:
                                  %esp,%ebp
                          mov
      0x00000003 <+3>:
                                  $0xffffffff0,%esp
                          and
      0x00000006 <+6>:
                                  0x7 <main+7>
                          call
                                  $0x0, %eax
      0x0000000b <+11>:
                           mov
      0x00000010 <+16>:
                           mov
                                  %ebp,%esp
      0x00000012 <+18>:
                           pop
                                  %ebp
      0x00000013 <+19>:
                           ret
  End of assembler dump.
10
  Dump of assembler code for function main:
      0x08048424 <+0>: push
                                  %ebp
                                  %esp,%ebp
      0x08048425 <+1>:
                          mov
      0x08048427 <+3>:
                                  $0xffffffff0, %esp
                           and
      0x0804842a <+6>:
                          call
                                  0x8048438 <zing>
                                  $0x0,%eax
      0x0804842f <+11>:
                          mov
      0x08048434 <+16>:
                           mov
                                  %ebp,%esp
      0x08048436 <+18>:
                                  %ebp
                           pop
      0x08048437 <+19>:
                           ret
10 End of assembler dump.
```

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

```
1  /* -.-.-.
2
3  * File Name : fconc.h
4
5  * Last Modified : Sun 13 Nov 2011 05:31:09 PM EET
```

```
* Created By : Greg Liras <gregliras@gmail.com>
    * Created By : Vasilis Gerakaris <ugerak@gmail.com>
10
11
   _._._._.*/
12
   #ifndef FCONC_H
13
   #define FCONC_H
14
15
16
   #ifndef BUFFER_SIZE
   #define BUFFER_SIZE 1024
17
   #endif //BUFFER_SIZE
18
   #include <unistd.h>
20
   #include <fcntl.h>
21
   #include <stdlib.h>
23
void doWrite(int fd, const char *buff, int len);
   void write_file(int fd, const char *infile);
   void print_err(const char *p);
26
27 #endif //FCONC_H
   /* -.-.-.-.-.
    * File Name : fconc.c
   * Last Modified : Thu 17 Nov 2011 04:07:43 AM EET
   * Created By : Greg Liras <gregliras@gmail.com>
    * Created By : Vasilis Gerakaris <ugerak@gmail.com>
10
   _------*/
12
   #include "fconc.h"
13
   int main(int argc, char ** argv)
15
16
    int OUT;
17
    int TMP;
18
     int W_FLAGS = O_CREAT | O_WRONLY | O_TRUNC;
19
    int C_PERMS = S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH ;
20
21
     int counter=0;
     struct flock lock;
22
     if (argc < 3)
23
24
     {
      print_err("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
25
26
27
     TMP = open("/tmp/fconc.out.tmp", W_FLAGS, C_PERMS);
     if (TMP < 0)
28
     {
29
       print_err("Error handling tmp file, is another instance running?\n");
30
31
     \verb| fcntl(TMP,F_GETLK,lock)|; // get lock info on fd|
32
     33
34
     for(counter = 1 ; counter < argc-1 ; counter++ )</pre>
35
36
       write_file(TMP,argv[counter]);
37
38
     }
     lock.l_type = F_UNLCK;
                             //set lock to unlock
39
     fcntl(TMP,F_SETLK,lock); //set the lock on fd
40
     close(TMP);
                              //close fd
41
     if (argc > 3)
42
43
       OUT = open(argv[argc-1], W_FLAGS, C_PERMS);
44
     }
45
47
       OUT = open("fconc.out", W_FLAGS, C_PERMS);
48
     if (OUT < 0)
50
     {
51
       print_err("Error handling output file\n");
52
53
```

```
fcntl(OUT,F_GETLK,lock);
54
       lock.l_type = F_WRLCK;
55
       fcntl(OUT,F_SETLK,lock);
       write_file(OUT,"/tmp/fconc.out.tmp");
57
       lock.l_type = F_UNLCK;
58
       fcntl(OUT,F_SETLK,lock);
       close(OUT);
60
       if (unlink("/tmp/fconc.out.tmp") != 0)
61
62
         print_err("Error deleting temporary file, please remove /tmp/fconc.out.tmp\n");
63
64
       exit(EXIT_SUCCESS);
65
66
     void doWrite(int fd,const char *buff,int len)
68
69
70
       int written;
71
       do
72
         if ( (written = write(fd,buff,len)) < 0 )</pre>
73
74
           print_err("Error in writing\n");
76
      } while(written < len );</pre>
77
    }
79
80
     void write_file(int fd,const char *infile)
81
    {
82
83
      char buffer[BUFFER_SIZE];
84
85
      int chars_read=0;
       struct flock lock;
86
       A = open(infile,O_RDONLY);
87
       if (A ==-1)
88
        print_err("No such file or directory\n");
90
       }
91
       fcntl(A,F_GETLK,lock); //get lock info on A
lock.l_type = F_RDLCK; //set lock to read lock
92
93
       fcntl(A,F_SETLK,lock); //set lock on A
       //time to read
95
       while( (chars_read = read(A,buffer,BUFFER_SIZE)) > 0)
96
         //and write
98
99
         doWrite(fd,buffer,chars_read);
100
       if ( chars\_read == -1 )
101
102
         print_err("Read Error\n");
103
104
       lock.l_type = F_UNLCK; //set lock to unlock
105
       fcntl(A,F_SETLK,lock); //set lock on A
106
107
       //ok close
       if (close(A) == -1)
108
109
       {
         print_err("Close Error\n");
110
       }
111
    7
112
     void print_err(const char *p)
114
115
      int len = 0;
116
       const char *b = p;
while( *b++ != '\0' ) len++;
117
118
       doWrite(2,p,len); //doWrite to stderr
119
120
       exit(-1);
121
    all:
 1
                      fconc
 2
    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:
9
           ./fconc A B C D E F
11
   strace:
          strace -o strace_outfile ./fconc A B C D E F \,
12
   Η έξοδος της strace είναι η παρακάτω:
execve("./fconc", ["./fconc", "A", "B", "C", "D", "E", "F"], [/* 47 vars */]) = 0
2 brk(0)
                                            = 0x9f07000
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb78d8000
                                        = -1 ENOENT (No such file or directory)
4 access("/etc/ld.so.preload", R_OK)
open("/etc/ld.so.cache", O RDONLY)
6 fstat64(3, {st_mode=S_IFREG|0644, st_size=102531, ...}) = 0
   mmap2(NULL, 102531, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb78be000
  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\0\3\0\1\0\0\0\0\244\1\0004\0\0\0"..., 512) = 512
ii fstat64(3, {st_mode=S_IFREG|0755, st_size=1429996, ...}) = 0
mmap2(NULL, 1440296, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb775e000
mprotect(0xb78b7000, 4096, PROT_NONE)
                                           = 0
mmap2(0xb78b8000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) =
         0xb78b8000
mmap2(0xb78bb000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
       xb78bb000
16 close(3)
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb775d000
set_thread_area({entry_number:-1 -> 6, base_addr:0xb775d6c0, limit:1048575, seg_32bit:1, contents:0, read_exec_only:0, limit_in_pages:1, seg_not_present:0, useable:1}) = 0
_{19} mprotect(0xb78b8000, 8192, PROT_READ) = 0
20 mprotect(0x8049000, 4096, PROT_READ)
21 mprotect(0xb78f6000, 4096, PROT_READ)
                                            = 0
                                            = 0
22 munmap(0xb78be000, 102531)
open("/tmp/fconc.out.tmp", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
_{24} fcntl64(3, F_GETLK, {type=0xffff838d /* F_??? */, whence=0xffffff18 /* SEEK_??? */, start
        =-953548801, len=-2063401023, pid=824800511}) = -1 EINVAL (Invalid argument)
25 fcntl64(3, F_SETLK, {...})
                                            = -1 EFAULT (Bad address)
open("A", O_RDONLY)
_{27} fcntl64(4, F_GETLK, {...})
                                            = -1 EFAULT (Bad address)
28 fcnt164(4, F_SETLK, {...})
                                            = -1 EFAULT (Bad address)
   read(4, "asdf\n", 1024)
                                            = 5
30 write(3, "asdf\n", 5)
31 read(4, "", 1024)
                                            = 5
                                            = 0
32 fcnt164(4, F_SETLK, {...})
                                            = -1 EFAULT (Bad address)
33 close(4)
                                            = 0
open("B", O_RDONLY)
                                            = 4
35 fcnt164(4, F_GETLK, {...})
                                            = -1 EFAULT (Bad address)
36 fcnt164(4, F_SETLK, {...})
                                            = -1 EFAULT (Bad address)
                                            = 5
read(4, "lkjhn", 1024)
38 write(3, "lkjh\n", 5)
39 read(4, "", 1024)
                                            = 5
                                            = 0
40 fcnt164(4, F_SETLK, {...})
                                            = -1 EFAULT (Bad address)
41 close(4)
                                            = 0
open("C", O_RDONLY)
                                            = 4
_{43} fcntl64(4, F_GETLK, {...})
                                            = -1 EFAULT (Bad address)
44 fcnt164(4, F_SETLK, {...})
                                            = -1 EFAULT (Bad address)
read(4, "testn", 1024)
46 write(3, "test\n", 5)
47 read(4, "", 1024)
                                            = 0
48 fcnt164(4, F_SETLK, {...})
                                            = -1 EFAULT (Bad address)
49 close(4)
                                            = 0
open("D", O_RDONLY)
                                            = 4
  fcnt164(4, F_GETLK, {...})
                                            = -1 EFAULT (Bad address)
52 fcnt164(4, F_SETLK, {...})
                                            = -1 EFAULT (Bad address)
                                            = 6
read(4, "test2\n", 1024)
54 write(3, "test2\n", 6)
55 read(4, "", 1024)
                                            = 0
56 fcnt164(4, F_SETLK, {...})
                                            = -1 EFAULT (Bad address)
57 close(4)
                                            = 0
open("E", O_RDONLY)
                                            = 4
59 fcnt164(4, F_GETLK, {...})
                                            = -1 EFAULT (Bad address)
60 fcnt164(4, F_SETLK, {...})
                                            = -1 EFAULT (Bad address)
                                            = 12
read(4, "test3\ntest4\n", 1024)
write(3, "test3\ntest4\n", 12)
                                            = 12
63 read(4, "", 1024)
                                            = 0
```

```
64 fcnt164(4, F_SETLK, {...})
                                            = -1 EFAULT (Bad address)
65 close(4)
66 fcnt164(3, F_SETLK, {...})
                                            = -1 EFAULT (Bad address)
  close(3)
                                            = 0
open("F", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
69 fcnt164(3, F_GETLK, {...}) = -1 EFAULT (Bad address)
70 fcnt164(3, F_SETLK, {...}) = -1 EFAULT (Bad address)
open("/tmp/fconc.out.tmp", O_RDONLY)
                                            = 4
72 fcntl64(4, F_GETLK, {...})
                                            = -1 EFAULT (Bad address)
73 fcntl64(4, F_SETLK, {...})
                                            = -1 EFAULT (Bad address)
   read(4, "asdf\nlkjh\ntest\ntest2\ntest3\ntest4"..., 1024) = 33
vrite(3, "asdf\nlkjh\ntest\ntest3\ntest4"..., 33) = 33
76 read(4, "", 1024)
                                            = 0
   fcnt164(4, F_SETLK, {...})
                                            = -1 EFAULT (Bad address)
                                            = 0
  close(4)
79 fcnt164(3, F_SETLK, {...})
                                            = -1 EFAULT (Bad address)
   close(3)
                                            = 0
                                            = 0
unlink("/tmp/fconc.out.tmp")
82 exit_group(0)
```

- 4. Όντως τρέχοντας το εκτελέσιμο whoops η έξοδος ήταν αυτή:
 - \$ /home/oslab/oslabb03/code/whoops/whoops
 Problem!

Η έξοδος της strace είναι η παρακάτω:

```
execve("./whoops", ["./whoops"], [/* 45 vars */]) = 0
2 brk(0)
                                         = 0x92d3000
  mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb782d000
4 access("/etc/ld.so.preload", R_OK)
                                      = -1 ENOENT (No such file or directory)
5 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) = 0xb7810000
                                         = 0
  open("/lib/libc.so.6", O_RDONLY)
                                         = 3
  read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\0\244\1\0004\0\0\0"..., 512) = 512
ii fstat64(3, {st_mode=S_IFREG|0755, st_size=1429996, ...}) = 0
mprotect(0xb7809000, 4096, PROT_NONE)
                                        = 0
14 mmap2(0xb780a000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) =
       0xb780a000
  mmap2(0xb780d000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
15
      xb780d000
16
  close(3)
                                         = 0
  mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb76af000
17
  set_thread_area({entry_number:-1 -> 6, base_addr:0xb76af6c0, limit:1048575, seg_32bit:1,
       contents:0, read_exec_only:0, limit_in_pages:1, seg_not_present:0, useable:1}) = 0
  mprotect(0xb780a000, 8192, PROT_READ)
mprotect(0xb784b000, 4096, PROT_READ)
19
                                        = 0
21 munmap(0xb7810000, 118009)
                                         = 0
open("/etc/shadow", O_RDONLY)
                                         = -1 EACCES (Permission denied)
  write(2, "Problem!\n", 9)
                                         = 9
  exit_group(1)
```

Όπως βλέπουμε στη γραμμή 22 το πρόγραμμά μας προσπαθεί να διαβάσει το αρχείο /etc/shadow. Όμως ο χρήστης που τρέχει το πρόγραμμα whoops δεν έχει δικαίωμα να διαβάσει το συγκεκριμένο αρχείο οπότε το λειτουργικό σύστημα δεν επιστρέφει κάποιο file descriptor στην εφαρμογή για να διαβάσει. Από εκεί προκύπτει το πρόβλημα το οποίο μας γράφει το πρόγραμμά μας στο stderr όπως φαίνεται στη γραμμή 23.