



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

17 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 int main(int argc, char ** argv)
4 {
5     zing();
6     return 0;
7 }
```

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 κά-νουμε 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 : Sun 13 Nov 2011 05:31:09 PM EET
6
7  * Created By : Greg Liras <gregliras@gmail.com>
8
9  * Created By : Vasilis Gerakaris <vgerak@gmail.com>
10
11  .....*/
12
13  #ifndef FCONC_H
14  #define FCONC_H
15
16  #ifndef BUFFER_SIZE
17  #define BUFFER_SIZE 1024
18  #endif //BUFFER_SIZE
19
20  #include <unistd.h>
21  #include <fcntl.h>
22  #include <stdlib.h>
23
24  void doWrite(int fd, const char *buff, int len);
25  void write_file(int fd, const char *infile);
26  void print_err(const char *p);
27  #endif //FCONC_H

```

```
1  /* .....
2
3  * File Name : fconc.c
4
5  * Last Modified : Thu 17 Nov 2011 03:42:32 AM EET
6
7  * Created By : Greg Liras <gregliras@gmail.com>
8
9  * Created By : Vasilis Gerakaris <vgerak@gmail.com>
10
11  .....*/
12
13  #include "fconc.h"
14
15  int main(int argc, char ** argv)
16  {
17      int OUT;
18      int TMP;
19      int W_FLAGS = O_CREAT | O_WRONLY | O_TRUNC;
20      int C_PERMS = S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH ;
21      struct flock lock;
22      if (argc < 3)
23      {
24          print_err("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
25      }
26      TMP = open("/tmp/fconc.out.tmp",W_FLAGS,C_PERMS);
27      if (TMP < 0)
28      {
29          print_err("Error handling tmp file, is another instance running?\n");
30      }
31      fcntl(TMP,F_GETLK,lock); //get lock info on fd
32      lock.l_type = F_WRLCK; //set lock to write lock
33      fcntl(TMP,F_SETLK,lock); //set the lock on fd
34      write_file(TMP,argv[1]); //write on fd
35      write_file(TMP,argv[2]);
36      lock.l_type = F_UNLCK; //set lock to unlock
37      fcntl(TMP,F_SETLK,lock); //set the lock on fd
38      close(TMP); //close fd
39      if (argc > 3)
40      {
41          OUT = open(argv[3],W_FLAGS,C_PERMS);
42      }
43      else
44      {
```

```

45     OUT = open("fconc.out",W_FLAGS,C_PERMS);
46 }
47 if (OUT < 0)
48 {
49     print_err("Error handling output file\n");
50 }
51 fcntl(OUT,F_GETLK,lock);
52 lock.l_type = F_WRLCK;
53 fcntl(OUT,F_SETLK,lock);
54 write_file(OUT,"/tmp/fconc.out.tmp");
55 lock.l_type = F_UNLCK;
56 fcntl(OUT,F_SETLK,lock);
57 close(OUT);
58 if (unlink("/tmp/fconc.out.tmp") != 0)
59 {
60     print_err("Error deleting temporary file, please remove /tmp/fconc.out.tmp\n");
61 }
62 exit(EXIT_SUCCESS);
63 }
64
65 void doWrite(int fd,const char *buff,int len)
66 {
67     int written;
68     do
69     {
70         if ( (written = write(fd,buff,len)) < 0 )
71         {
72             print_err("Error in writing\n");
73         }
74     } while(written < len );
75 }
76
77
78 void write_file(int fd,const char *infile)
79 {
80     int A;
81     char buffer[BUFFER_SIZE];
82     int chars_read=0;
83     struct flock lock;
84     A = open(infile,O_RDONLY);
85     if (A ==-1)
86     {
87         print_err("No such file or directory\n");
88     }
89     fcntl(A,F_GETLK,lock); //get lock info on A
90     lock.l_type = F_RDLCK; //set lock to read lock
91     fcntl(A,F_SETLK,lock); //set lock on A
92     //time to read
93     while( (chars_read = read(A,buffer,BUFFER_SIZE)) > 0)
94     {
95         //and write
96         doWrite(fd,buffer,chars_read);
97     }
98     if ( chars_read == -1 )
99     {
100         print_err("Read Error\n");
101     }
102     lock.l_type = F_UNLCK; //set lock to unlock
103     fcntl(A,F_SETLK,lock); //set lock on A
104     //ok close
105     if ( close(A) == - 1 )
106     {
107         print_err("Close Error\n");
108     }
109 }
110
111 void print_err(const char *p)
112 {
113     int len = 0;
114     const char *b = p;
115     while( *b++ != '\0' ) len++;
116     doWrite(2,p,len); //doWrite to stderr
117     exit(-1);
118 }

```

```

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 strace
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"], [/* 47 vars */]) = 0
2  brk(0)                                = 0x8365000
3  mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7841000
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=102531, ...}) = 0
7  mmap2(NULL, 102531, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb7827000
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\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) = 0xb76c7000
13 mprotect(0xb7820000, 4096, PROT_NONE) = 0
14 mmap2(0xb7821000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) = 0
   xb7821000
15 mmap2(0xb7824000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
   xb7824000
16 close(3)                              = 0
17 mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb76c6000
18 set_thread_area({entry_number:-1 -> 6, base_addr:0xb76c66c0, limit:1048575, seg_32bit:1, contents:0,
   read_exec_only:0, limit_in_pages:1, seg_not_present:0, useable:1}) = 0
19 mprotect(0xb7821000, 8192, PROT_READ) = 0
20 mprotect(0x8049000, 4096, PROT_READ)  = 0
21 mprotect(0xb785f000, 4096, PROT_READ)  = 0
22 munmap(0xb7827000, 102531)             = 0
23 open("/tmp/fconc.out.tmp", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
24 fcntl64(3, F_GETLK, {type=0xffffffff /* F_??? */, whence=0xffffffff /* SEEK_??? */, start
   =1958774271, len=139823908, pid=404042597}) = -1 EINVAL (Invalid argument)
25 fcntl64(3, F_SETLK, {type=0x4589 /* F_??? */, whence=0xffffe9b8 /* SEEK_??? */, start=-1006, len
   =-1007971443}) = -1 EINVAL (Invalid argument)
26 open("A", O_RDONLY)                   = 4
27 fcntl64(4, F_GETLK, {...})             = -1 EFAULT (Bad address)
28 fcntl64(4, F_SETLK, {...})             = -1 EFAULT (Bad address)
29 read(4, "test\ntest\ntest\ntest\ntest\ntest\nte"... , 1024) = 40
30 write(3, "test\ntest\ntest\ntest\ntest\ntest\nte"... , 40) = 40
31 read(4, "", 1024)                      = 0
32 fcntl64(4, F_SETLK, {...})             = -1 EFAULT (Bad address)
33 close(4)                              = 0
34 open("B", O_RDONLY)                   = 4
35 fcntl64(4, F_GETLK, {...})             = -1 EFAULT (Bad address)
36 fcntl64(4, F_SETLK, {...})             = -1 EFAULT (Bad address)
37 read(4, "lkjh\n", 1024)                = 5
38 write(3, "lkjh\n", 5)                  = 5
39 read(4, "", 1024)                      = 0
40 fcntl64(4, F_SETLK, {...})             = -1 EFAULT (Bad address)
41 close(4)                              = 0
42 fcntl64(3, F_SETLK, {type=0xffffb845 /* F_??? */, whence=0x12e9 /* SEEK_??? */, start=-1912602628,
   len=-37491821}) = -1 EINVAL (Invalid argument)
43 close(3)                              = 0
44 open("C", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
45 fcntl64(3, F_GETLK, {type=0xffffb845 /* F_??? */, whence=0x12e9 /* SEEK_??? */, start=-1912602628,
   len=-37491821, pid=3296037375}) = -1 EINVAL (Invalid argument)
46 fcntl64(3, F_SETLK, {type=0x4589 /* F_??? */, whence=0xffffe9b8 /* SEEK_??? */, start=-1006, len
   =-1007971443}) = -1 EINVAL (Invalid argument)
47 open("/tmp/fconc.out.tmp", O_RDONLY)   = 4
48 fcntl64(4, F_GETLK, {...})             = -1 EFAULT (Bad address)
49 fcntl64(4, F_SETLK, {...})             = -1 EFAULT (Bad address)
50 read(4, "test\ntest\ntest\ntest\ntest\ntest\nte"... , 1024) = 45
51 write(3, "test\ntest\ntest\ntest\ntest\ntest\nte"... , 45) = 45
52 read(4, "", 1024)                      = 0

```

```

53 fcntl64(4, F_SETLK, {...})           = -1 EFAULT (Bad address)
54 close(4)                             = 0
55 fcntl64(3, F_SETLK, {type=0xffffb845 /* F_??? */, whence=0x12e9 /* SEEK_??? */, start=-1912602628,
    len=-37491821}) = -1 EINVAL (Invalid argument)
56 close(3)                             = 0
57 unlink("/tmp/fconc.out.tmp")          = 0
58 exit_group(0)                         = ?

```

1.3 Bonus

1. Η εντολή `strace 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
10 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
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) = 0xb768c000
13 mprotect(0xb77e5000, 4096, PROT_NONE) = 0
14 mmap2(0xb77e6000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) = 0xb77e6000
15 mmap2(0xb77e9000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xb77e9000
16 close(3) = 0
17 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
19 mprotect(0xb77e6000, 8192, PROT_READ) = 0
20 mprotect(0x8082000, 4096, PROT_READ) = 0
21 mprotect(0xb7827000, 4096, PROT_READ) = 0
22 munmap(0xb77ec000, 118009) = 0
23 brk(0) = 0x94ed000
24 brk(0x950e000) = 0x950e000
25 write(2, "usage: strace [-CdDffhiqrstTvVx"... , 1731) = 1731
26 exit_group(1) = ?

```

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

```

1  Dump of assembler code for function main:
2      0x00000000 <+0>:      push    %ebp
3      0x00000001 <+1>:      mov     %esp,%ebp
4      0x00000003 <+3>:      and     $0xffffffff,%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.
```

```

1  Dump of assembler code for function main:
2      0x08048424 <+0>:      push    %ebp
3      0x08048425 <+1>:      mov     %esp,%ebp
4      0x08048427 <+3>:      and     $0xffffffff0,%esp
5      0x0804842a <+6>:      call    0x8048438 <zing>
6      0x0804842f <+11>:     mov     $0x0,%eax
7      0x08048434 <+16>:     mov     %ebp,%esp
8      0x08048436 <+18>:     pop     %ebp
9      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

```



```

54     fcntl(OUT,F_GETLK,lock);
55     lock.l_type = F_WRLCK;
56     fcntl(OUT,F_SETLK,lock);
57     write_file(OUT,"/tmp/fconc.out.tmp");
58     lock.l_type = F_UNLCK;
59     fcntl(OUT,F_SETLK,lock);
60     close(OUT);
61     if (unlink("/tmp/fconc.out.tmp") != 0)
62     {
63         print_err("Error deleting temporary file, please remove /tmp/fconc.out.tmp\n");
64     }
65     exit(EXIT_SUCCESS);
66 }
67
68 void doWrite(int fd,const char *buff,int len)
69 {
70     int written;
71     do
72     {
73         if ( (written = write(fd,buff,len)) < 0 )
74         {
75             print_err("Error in writing\n");
76         }
77     } while(written < len );
78 }
79
80
81 void write_file(int fd,const char *infile)
82 {
83     int A;
84     char buffer[BUFFER_SIZE];
85     int chars_read=0;
86     struct flock lock;
87     A = open(infile,O_RDONLY);
88     if (A ==-1)
89     {
90         print_err("No such file or directory\n");
91     }
92     fcntl(A,F_GETLK,lock); //get lock info on A
93     lock.l_type = F_RDLCK; //set lock to read lock
94     fcntl(A,F_SETLK,lock); //set lock on A
95     //time to read
96     while( (chars_read = read(A,buffer,BUFFER_SIZE)) > 0)
97     {
98         //and write
99         doWrite(fd,buffer,chars_read);
100     }
101     if ( chars_read == -1 )
102     {
103         print_err("Read Error\n");
104     }
105     lock.l_type = F_UNLCK; //set lock to unlock
106     fcntl(A,F_SETLK,lock); //set lock on A
107     //ok close
108     if ( close(A) == - 1 )
109     {
110         print_err("Close Error\n");
111     }
112 }
113
114 void print_err(const char *p)
115 {
116     int len = 0;
117     const char *b = p;
118     while( *b++ != '\0' ) len++;
119     doWrite(2,p,len); //doWrite to stderr
120     exit(-1);
121 }

```

```

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"], [/* 47 vars */]) = 0
2  brk(0)                                = 0x9f07000
3  mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb78d8000
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=102531, ...}) = 0
7  mmap2(NULL, 102531, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb78be000
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\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) = 0xb775e000
13 mprotect(0xb78b7000, 4096, PROT_NONE) = 0
14 mmap2(0xb78b8000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) =
    0xb78b8000
15 mmap2(0xb78bb000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
    xb78bb000
16 close(3)                              = 0
17 mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb775d000
18 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)   = 0
21 mprotect(0xb78f6000, 4096, PROT_READ) = 0
22 munmap(0xb78be000, 102531)            = 0
23 open("/tmp/fconc.out.tmp", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
24 fcntl64(3, F_GETLK, {type=0xffff838d /* F_??? */, whence=0xffffffff18 /* SEEK_??? */, start
    =-953548801, len=-2063401023, pid=824800511}) = -1 EINVAL (Invalid argument)
25 fcntl64(3, F_SETLK, {...})            = -1 EFAULT (Bad address)
26 open("A", O_RDONLY)                   = 4
27 fcntl64(4, F_GETLK, {...})            = -1 EFAULT (Bad address)
28 fcntl64(4, F_SETLK, {...})            = -1 EFAULT (Bad address)
29 read(4, "asdf\n", 1024)               = 5
30 write(3, "asdf\n", 5)                 = 5
31 read(4, "", 1024)                     = 0
32 fcntl64(4, F_SETLK, {...})            = -1 EFAULT (Bad address)
33 close(4)                              = 0
34 open("B", O_RDONLY)                   = 4
35 fcntl64(4, F_GETLK, {...})            = -1 EFAULT (Bad address)
36 fcntl64(4, F_SETLK, {...})            = -1 EFAULT (Bad address)
37 read(4, "lkjh\n", 1024)               = 5
38 write(3, "lkjh\n", 5)                 = 5
39 read(4, "", 1024)                     = 0
40 fcntl64(4, F_SETLK, {...})            = -1 EFAULT (Bad address)
41 close(4)                              = 0
42 open("C", O_RDONLY)                   = 4
43 fcntl64(4, F_GETLK, {...})            = -1 EFAULT (Bad address)
44 fcntl64(4, F_SETLK, {...})            = -1 EFAULT (Bad address)
45 read(4, "test\n", 1024)               = 5
46 write(3, "test\n", 5)                 = 5
47 read(4, "", 1024)                     = 0
48 fcntl64(4, F_SETLK, {...})            = -1 EFAULT (Bad address)
49 close(4)                              = 0
50 open("D", O_RDONLY)                   = 4
51 fcntl64(4, F_GETLK, {...})            = -1 EFAULT (Bad address)
52 fcntl64(4, F_SETLK, {...})            = -1 EFAULT (Bad address)
53 read(4, "test2\n", 1024)              = 6
54 write(3, "test2\n", 6)                 = 6
55 read(4, "", 1024)                     = 0
56 fcntl64(4, F_SETLK, {...})            = -1 EFAULT (Bad address)
57 close(4)                              = 0
58 open("E", O_RDONLY)                   = 4
59 fcntl64(4, F_GETLK, {...})            = -1 EFAULT (Bad address)
60 fcntl64(4, F_SETLK, {...})            = -1 EFAULT (Bad address)
61 read(4, "test3\ntest4\n", 1024)       = 12
62 write(3, "test3\ntest4\n", 12)        = 12
63 read(4, "", 1024)                     = 0

```

```

64 fcntl64(4, F_SETLK, {...}) = -1 EFAULT (Bad address)
65 close(4) = 0
66 fcntl64(3, F_SETLK, {...}) = -1 EFAULT (Bad address)
67 close(3) = 0
68 open("F", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
69 fcntl64(3, F_GETLK, {...}) = -1 EFAULT (Bad address)
70 fcntl64(3, F_SETLK, {...}) = -1 EFAULT (Bad address)
71 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)
74 read(4, "asdf\nlkjh\ntest\ntest2\ntest3\ntest4"... , 1024) = 33
75 write(3, "asdf\nlkjh\ntest\ntest2\ntest3\ntest4"... , 33) = 33
76 read(4, "", 1024) = 0
77 fcntl64(4, F_SETLK, {...}) = -1 EFAULT (Bad address)
78 close(4) = 0
79 fcntl64(3, F_SETLK, {...}) = -1 EFAULT (Bad address)
80 close(3) = 0
81 unlink("/tmp/fconc.out.tmp") = 0
82 exit_group(0) = ?

```

4. Όντως τρέχοντας το εκτελέσιμο whoops η έξοδος ήταν αυτή:

```

$ /home/oslab/oslab03/code/whoops/whoops
Problem!

```

Η έξοδος της strace είναι η παρακάτω:

```

1  execve("./whoops", ["/whoops"], [/* 45 vars */]) = 0
2  brk(0) = 0x92d3000
3  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
6  fstat64(3, {st_mode=S_IFREG|0644, st_size=118009, ...}) = 0
7  mmap2(NULL, 118009, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb7810000
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\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) = 0xb76b0000
13 mprotect(0xb7809000, 4096, PROT_NONE) = 0
14 mmap2(0xb780a000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) = 0xb780a000
15 mmap2(0xb780d000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0xb780d000
16 close(3) = 0
17 mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb76af000
18 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
19 mprotect(0xb780a000, 8192, PROT_READ) = 0
20 mprotect(0xb784b000, 4096, PROT_READ) = 0
21 munmap(0xb7810000, 118009) = 0
22 open("/etc/shadow", O_RDONLY) = -1 EACCES (Permission denied)
23 write(2, "Problem!\n", 9) = 9
24 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.