



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

18 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 : Thu 17 Nov 2011 10:07:16 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  #include <stdio.h>
24  #include <sysxits.h>
25
26  void doWrite(int fd, const char *buff, int len);
27  void write_file(int fd, const char *infile);
28  #endif //FCONC_H

```

```
1  /* .....
2
3  * File Name : fconc.c
4
5  * Last Modified : Fri 18 Nov 2011 09:25:21 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 "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          perror("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
25          exit(EX_USAGE);
26      }
27      TMP = open("/tmp/fconc.out.tmp",W_FLAGS,C_PERMS);
28      if (TMP < 0)
29      {
30          perror("Error opening tmp file, is another instance running?\n");
31          exit(EX_TEMPFAIL);
32      }
33      fcntl(TMP,F_GETLK,lock); //get lock info on fd
34      lock.l_type = F_WRLCK; //set lock to write lock
35      fcntl(TMP,F_SETLK,lock); //set the lock on fd
36      write_file(TMP,argv[1]); //write on fd
37      write_file(TMP,argv[2]);
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[3],W_FLAGS,C_PERMS);
```

```

44     }
45     else
46     {
47         OUT = open("fconc.out",W_FLAGS,C_PERMS);
48     }
49     if (OUT < 0)
50     {
51         perror("Error handling output file\n");
52         exit(EX_IOERR);
53     }
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         perror("Error deleting temporary file, please remove /tmp/fconc.out.tmp\n");
64         exit(EX_BASE);
65     }
66     exit(EXIT_SUCCESS);
67 }
68
69 void doWrite(int fd,const char *buff,int len)
70 {
71     int written = 0;
72     int current = 0;
73     do
74     {
75         if ( (current = write(fd,buff+written,len-written)) < 0 )
76         {
77             perror("Error in writing\n");
78             exit(EX_IOERR);
79         }
80         written+=current;
81     } while(written < len );
82 }
83
84
85 void write_file(int fd,const char *infile)
86 {
87     int A;
88     char buffer[BUFFER_SIZE];
89     int chars_read=0;
90     struct flock lock;
91     A = open(infile,O_RDONLY);
92     if (A ==-1)
93     {
94         char error_message[BUFFER_SIZE];
95         sprintf(error_message,"%s",infile);
96         perror(error_message);
97         exit(EX_NOINPUT);
98     }
99     fcntl(A,F_GETLK,lock); //get lock info on A
100     lock.l_type = F_RDLCK; //set lock to read lock
101     fcntl(A,F_SETLK,lock); //set lock on A
102     //time to read
103     while( (chars_read = read(A,buffer,BUFFER_SIZE)) > 0)
104     {
105         //and write
106         doWrite(fd,buffer,chars_read);
107     }
108     if ( chars_read == -1 )
109     {
110         perror("Read Error\n");
111         exit(EX_IOERR);
112     }
113     lock.l_type = F_UNLCK; //set lock to unlock
114     fcntl(A,F_SETLK,lock); //set lock on A
115     //ok close
116     if ( close(A) == - 1 )
117     {
118         perror("Close Error\n");

```

```

119     exit(EX_IOERR);
120 }
121 }

1 all:          fconc
2 fconc:        fconc.o
3              gcc fconc.o -o fconc -m32
4 fconc.o:      fconc.c fconc.h
5              gcc -c fconc.c -o fconc.o -Wall -m32
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

```

Η έξοδος της strace είναι η παρακάτω:

```

1 execve("./fconc", ["/fconc", "A", "B", "C"], [/* 47 vars */]) = 0
2 brk(0)                                = 0x9466000
3 mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb785a000
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|0755, st_size=1429996, ...}) = 0
7 mmap2(NULL, 103261, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb7840000
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) = 0xb76e0000
13 mprotect(0xb7839000, 4096, PROT_NONE) = 0
14 mmap2(0xb783a000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) = 0
   xb783a000
15 mmap2(0xb783d000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
   xb783d000
16 close(3)                               = 0
17 mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb76df000
18 set_thread_area({entry_number:-1 -> 6, base_addr:0xb76df6c0, limit:1048575, seg_32bit:1, contents:0,
   read_exec_only:0, limit_in_pages:1, seg_not_present:0, useable:1}) = 0
19 mprotect(0xb783a000, 8192, PROT_READ)  = 0
20 mprotect(0x8049000, 4096, PROT_READ)   = 0
21 mprotect(0xb7878000, 4096, PROT_READ)  = 0
22 munmap(0xb7840000, 103261)             = 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=0xffff93e9 /* F_??? */, whence=0xffffffff /* SEEK_??? */, start
   =-1869573889, len=-1869574000}) = -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\nte"... , 1024) = 40
30 write(3, "test\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=0xffff93 /* F_??? */, whence=0xffffffff /* SEEK_??? */, start
   =-1869574000, len=-1869574000}) = -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=0xffff93 /* F_??? */, whence=0xffffffff /* SEEK_??? */, start
   =-1869574000, len=-1869574000, pid=2304086160}) = -1 EINVAL (Invalid argument)
46 fcntl64(3, F_SETLK, {type=0xffff93e9 /* F_??? */, whence=0xffffffff /* SEEK_??? */, start
   =-1869573889, len=-1869574000}) = -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)

```



```

23  #include <stdio.h>
24  #include <sysextits.h>
25
26  void doWrite(int fd, const char *buff, int len);
27  void write_file(int fd, const char *infile);
28  #endif //FCONC_H

```

```

1  /* .....
2
3  * File Name : fconc.c
4
5  * Last Modified : Fri 18 Nov 2011 09:26:16 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 "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      int counter=0;
22      struct flock lock;
23      if (argc < 3)
24      {
25          perror("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
26          exit(EX_USAGE);
27      }
28      TMP = open("/tmp/fconc.out.tmp",W_FLAGS,C_PERMS);
29      if (TMP < 0)
30      {
31          perror("Error opening tmp file, is another instance running?\n");
32          exit(EX_TEMPFAIL);
33      }
34      fcntl(TMP,F_GETLK,lock); //get lock info on fd
35      lock.l_type = F_WRLCK; //set lock to write lock
36      fcntl(TMP,F_SETLK,lock); //set the lock on fd
37      for(counter = 1 ; counter < argc-1 ; counter++ )
38      {
39          write_file(TMP,argv[counter]);
40      }
41      lock.l_type = F_UNLCK; //set lock to unlock
42      fcntl(TMP,F_SETLK,lock); //set the lock on fd
43      close(TMP); //close fd
44      if (argc > 3)
45      {
46          OUT = open(argv[3],W_FLAGS,C_PERMS);
47      }
48      else
49      {
50          OUT = open("fconc.out",W_FLAGS,C_PERMS);
51      }
52      if (OUT < 0)
53      {
54          perror("Error handling output file\n");
55          exit(EX_IOERR);
56      }
57      fcntl(OUT,F_GETLK,lock);
58      lock.l_type = F_WRLCK;
59      fcntl(OUT,F_SETLK,lock);
60      write_file(OUT,"/tmp/fconc.out.tmp");
61      lock.l_type = F_UNLCK;
62      fcntl(OUT,F_SETLK,lock);
63      close(OUT);
64      if (unlink("/tmp/fconc.out.tmp") != 0)
65      {
66          perror("Error deleting temporary file, please remove /tmp/fconc.out.tmp\n");
67          exit(EX__BASE);
68      }
69      exit(EXIT_SUCCESS);

```

```

70 }
71
72 void doWrite(int fd,const char *buff,int len)
73 {
74     int written = 0;
75     int current = 0;
76     do
77     {
78         if ( (current = write(fd,buff+written,len-written)) < 0 )
79         {
80             perror("Error in writing\n");
81             exit(EX_IOERR);
82         }
83         written+=current;
84     } while(written < len );
85 }
86
87 void write_file(int fd,const char *infile)
88 {
89     int A;
90     char buffer[BUFFER_SIZE];
91     int chars_read=0;
92     struct flock lock;
93     A = open(infile,O_RDONLY);
94     if (A ==-1)
95     {
96         char error_message[BUFFER_SIZE];
97         sprintf(error_message,"%s",infile);
98         perror(error_message);
99         exit(EX_NOINPUT);
100     }
101     fcntl(A,F_GETLK,lock); //get lock info on A
102     lock.l_type = F_RDLCK; //set lock to read lock
103     fcntl(A,F_SETLK,lock); //set lock on A
104     //time to read
105     while( (chars_read = read(A,buffer,BUFFER_SIZE)) > 0)
106     {
107         //and write
108         doWrite(fd,buffer,chars_read);
109     }
110     if ( chars_read == -1 )
111     {
112         perror("Read Error\n");
113         exit(EX_IOERR);
114     }
115     lock.l_type = F_UNLCK; //set lock to unlock
116     fcntl(A,F_SETLK,lock); //set lock on A
117     //ok close
118     if ( close(A) == - 1 )
119     {
120         perror("Close Error\n");
121         exit(EX_IOERR);
122     }
123 }

```

```

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

```


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", ["/usr/bin/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.