ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

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

Τμήμα Β, Ομάδα 3η

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

17 Νοεμβρίου 2011

1.1 Σύνδεση με αρχείο αντικειμένων

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

```
#include "zing.h"

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

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

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

Τρέχοντας στο shell την εντολή make έχουμε την παρακάτω έξοδο

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

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

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

Απαντήσεις στις θεωρητικές ερωτήσεις

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

2. Απαντήθηκε παραπάνω.

3. Αντί να έχουμε όλες τις συναρτήσεις σε ένα αρχείο θα μπορούσαμε να χρησιμοποιούμε ένα αρχείο για κάθε συνάρτηση με το αντίστοιχο αρχείο επικεφαλίδας. Έτσι η μεταγλώτιση 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:

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

* 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 <vgerak@gmail.com>

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

#ifndef FCONC_H
#define FCONC_H

#ifndef BUFFER_SIZE
#define BUFFER_SIZE 1024
#endif //BUFFER_SIZE

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

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

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

* 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 <vgerak@gmail.com>

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

#include "fconc.h"

int main(int argc, char ** argv)
{
  int OUT;
  int TMP;
  int W_FLAGS = O_CREAT | O_WRONLY | O_TRUNC;
  int C_PERMS = S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH ;
  struct flock lock;
  if (argc < 3)
  {
    print_err("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
  }
  TMP = open("/tmp/fconc.out.tmp",W_FLAGS,C_PERMS);
  if (TMP < 0)
  {
    print_err("Error handling tmp file, is another instance running?\n");
  }
  fcntl(TMP,F_GETLK,lock);  //get lock info on fd
  lock.l_type = F_WRLCK;    //set lock to write lock
  fcntl(TMP,F_SETLK,lock);  //set the lock on fd
  write_file(TMP,argv[1]);  //write on fd
  write_file(TMP,argv[2]);
  lock.l_type = F_UNLCK;    //set lock to unlock
  fcntl(TMP,F_SETLK,lock);  //set the lock on fd
  close(TMP);               //close fd
  if (argc > 3)
  {
    OUT = open(argv[3],W_FLAGS,C_PERMS);
  }
  else
  {
    OUT = open("fconc.out",W_FLAGS,C_PERMS);
  }
  if (OUT < 0)
  {
    print_err("Error handling output file\n");
  }
  fcntl(OUT,F_GETLK,lock);
  lock.l_type = F_WRLCK;
  fcntl(OUT,F_SETLK,lock);
  write_file(OUT,"/tmp/fconc.out.tmp");
  lock.l_type = F_UNLCK;
  fcntl(OUT,F_SETLK,lock);
  close(OUT);
  if (unlink("/tmp/fconc.out.tmp") != 0)
  {
    print_err("Error deleting temporary file, please remove /tmp/fconc.out.tmp\n");
  }
  exit(EXIT_SUCCESS);
}

void doWrite(int fd,const char *buff,int len)
{
  int written;
  do
  {
    if ( (written = write(fd,buff,len)) < 0 )
    {
      print_err("Error in writing\n");
    }
  } while(written < len );
}


void write_file(int fd,const char *infile)
{
  int A;
  char buffer[BUFFER_SIZE];
  int chars_read=0;
  struct flock lock;
  A = open(infile,O_RDONLY);
  if (A ==-1)
  {
    print_err("No such file or directory\n");
  }
  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
  //time to read
  while( (chars_read = read(A,buffer,BUFFER_SIZE)) > 0)
  {
    //and write
    doWrite(fd,buffer,chars_read);
  }
  if ( chars_read == -1 )
  {
    print_err("Read Error\n");
  }
  lock.l_type = F_UNLCK;  //set lock to unlock
  fcntl(A,F_SETLK,lock);  //set lock on A
  //ok close
  if ( close(A) == - 1 )
  {
    print_err("Close Error\n");
  }
}

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

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

```

Η έξοδος της strace είναι η παρακάτω:

```
execve("./fconc", ["./fconc", "A", "B", "C"], [/* 47 vars */]) = 0
brk(0)                                  = 0x8365000
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7841000
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)                                = 0
open("/lib/libc.so.6", O_RDONLY)        = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\0\244\1\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1429996, ...}) = 0
mmap2(NULL, 1440296, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb76c7000
mprotect(0xb7820000, 4096, PROT_NONE)   = 0
mmap2(0xb7821000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) = 0
    xb7821000
mmap2(0xb7824000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
    xb7824000
close(3)                                = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb76c6000
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
mprotect(0xb7821000, 8192, PROT_READ)   = 0
mprotect(0x8049000, 4096, PROT_READ)    = 0
mprotect(0xb785f000, 4096, PROT_READ)   = 0
munmap(0xb7827000, 102531)              = 0
open("/tmp/fconc.out.tmp", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
fcntl64(3, F_GETLK, {type=0xffffffba /* F_??? */, whence=0xffffffff /* SEEK_??? */, start
    =1958774271, len=139823908, pid=404042597}) = -1 EINVAL (Invalid argument)
fcntl64(3, F_SETLK, {type=0x4589 /* F_??? */, whence=0xfffe9b8 /* SEEK_??? */, start=-1006, len
    =-1007971443}) = -1 EINVAL (Invalid argument)
open("A", O_RDONLY)                     = 4
fcntl64(4, F_GETLK, {...})              = -1 EFAULT (Bad address)
fcntl64(4, F_SETLK, {...})              = -1 EFAULT (Bad address)
read(4, "test\ntest\ntest\ntest\ntest\ntest\nte"..., 1024) = 40
write(3, "test\ntest\ntest\ntest\ntest\ntest\nte"..., 40) = 40
read(4, "", 1024)                       = 0
fcntl64(4, F_SETLK, {...})              = -1 EFAULT (Bad address)
close(4)                                = 0
open("B", O_RDONLY)                     = 4
fcntl64(4, F_GETLK, {...})              = -1 EFAULT (Bad address)
fcntl64(4, F_SETLK, {...})              = -1 EFAULT (Bad address)
read(4, "lkjh\n", 1024)                 = 5
write(3, "lkjh\n", 5)                   = 5
read(4, "", 1024)                       = 0
fcntl64(4, F_SETLK, {...})              = -1 EFAULT (Bad address)
close(4)                                = 0
fcntl64(3, F_SETLK, {type=0xffffb845 /* F_??? */, whence=0x12e9 /* SEEK_??? */, start=-1912602628,
    len=-37491821}) = -1 EINVAL (Invalid argument)
close(3)                                = 0
open("C", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
fcntl64(3, F_GETLK, {type=0xffffb845 /* F_??? */, whence=0x12e9 /* SEEK_??? */, start=-1912602628,
    len=-37491821, pid=3296037375}) = -1 EINVAL (Invalid argument)
fcntl64(3, F_SETLK, {type=0x4589 /* F_??? */, whence=0xfffe9b8 /* SEEK_??? */, start=-1006, len
    =-1007971443}) = -1 EINVAL (Invalid argument)
open("/tmp/fconc.out.tmp", O_RDONLY)    = 4
fcntl64(4, F_GETLK, {...})              = -1 EFAULT (Bad address)
fcntl64(4, F_SETLK, {...})              = -1 EFAULT (Bad address)
read(4, "test\ntest\ntest\ntest\ntest\ntest\nte"..., 1024) = 45
write(3, "test\ntest\ntest\ntest\ntest\ntest\nte"..., 45) = 45
read(4, "", 1024)                       = 0
```

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

1.3 Bonus

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

```
execve("/usr/bin/strace", ["strace"], [/* 45 vars */]) = 0
brk(0)                                  = 0x94ed000
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7809000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=118009, ...}) = 0
mmap2(NULL, 118009, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb77ec000
close(3)                                = 0
open("/lib/libc.so.6", O_RDONLY)        = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\0\244\1\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1429996, ...}) = 0
mmap2(NULL, 1440296, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb768c000
mprotect(0xb77e5000, 4096, PROT_NONE)   = 0
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
    xb77e9000
close(3)                                = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb768b000
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
mprotect(0x8082000, 4096, PROT_READ)    = 0
mprotect(0xb7827000, 4096, PROT_READ)   = 0
munmap(0xb77ec000, 118009)              = 0
brk(0)                                  = 0x94ed000
brk(0x950e000)                          = 0x950e000
write(2, "usage: strace [-CdDffhiqrtttTvVx"..., 1731) = 1731
exit_group(1)                           = ?
```

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

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

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

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

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

* File Name : fconc.h

* Last Modified : Sun 13 Nov 2011 05:31:09 PM EET
```

```c

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

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

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

#ifndef FCONC_H
#define FCONC_H

#ifndef BUFFER_SIZE
#define BUFFER_SIZE 1024
#endif //BUFFER_SIZE

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

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

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

* 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 <vgerak@gmail.com>

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

#include "fconc.h"

int main(int argc, char ** argv)
{
  int OUT;
  int TMP;
  int W_FLAGS = O_CREAT | O_WRONLY | O_TRUNC;
  int C_PERMS = S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH ;
  int counter=0;
  struct flock lock;
  if (argc < 3)
  {
    print_err("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
  }
  TMP = open("/tmp/fconc.out.tmp",W_FLAGS,C_PERMS);
  if (TMP < 0)
  {
    print_err("Error handling tmp file, is another instance running?\n");
  }
  fcntl(TMP,F_GETLK,lock);  //get lock info on fd
  lock.l_type = F_WRLCK;    //set lock to write lock
  fcntl(TMP,F_SETLK,lock);  //set the lock on fd
  for(counter = 1 ; counter < argc-1 ; counter++ )
  {
    write_file(TMP,argv[counter]);
  }
  lock.l_type = F_UNLCK;    //set lock to unlock
  fcntl(TMP,F_SETLK,lock);  //set the lock on fd
  close(TMP);               //close fd
  if (argc > 3)
  {
    OUT = open(argv[argc-1],W_FLAGS,C_PERMS);
  }
  else
  {
    OUT = open("fconc.out",W_FLAGS,C_PERMS);
  }
  if (OUT < 0)
  {
    print_err("Error handling output file\n");
  }
  fcntl(OUT,F_GETLK,lock);
  lock.l_type = F_WRLCK;
  fcntl(OUT,F_SETLK,lock);
  write_file(OUT,"/tmp/fconc.out.tmp");
  lock.l_type = F_UNLCK;
  fcntl(OUT,F_SETLK,lock);
  close(OUT);
  if (unlink("/tmp/fconc.out.tmp") != 0)
  {
    print_err("Error deleting temporary file, please remove /tmp/fconc.out.tmp\n");
  }
  exit(EXIT_SUCCESS);
}

void doWrite(int fd,const char *buff,int len)
{
  int written;
  do
  {
    if ( (written = write(fd,buff,len)) < 0 )
    {
      print_err("Error in writing\n");
    }
  } while(written < len );
}


void write_file(int fd,const char *infile)
{
  int A;
  char buffer[BUFFER_SIZE];
  int chars_read=0;
  struct flock lock;
  A = open(infile,O_RDONLY);
  if (A ==-1)
  {
    print_err("No such file or directory\n");
  }
  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
  //time to read
  while( (chars_read = read(A,buffer,BUFFER_SIZE)) > 0)
  {
    //and write
    doWrite(fd,buffer,chars_read);
  }
  if ( chars_read == -1 )
  {
    print_err("Read Error\n");
  }
  lock.l_type = F_UNLCK;  //set lock to unlock
  fcntl(A,F_SETLK,lock);  //set lock on A
  //ok close
  if ( close(A) == - 1 )
  {
    print_err("Close Error\n");
  }
}

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

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

```

Η έξοδος της strace είναι η παρακάτω:

```
execve("./fconc", ["./fconc", "A", "B", "C", "D", "E", "F"], [/* 47 vars */]) = 0
brk(0)                                  = 0x9f07000
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb78d8000
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) = 0xb78be000
close(3)                                = 0
open("/lib/libc.so.6", O_RDONLY)        = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\0\244\1\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1429996, ...}) = 0
mmap2(NULL, 1440296, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 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
close(3)                                = 0
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
mprotect(0xb78b8000, 8192, PROT_READ)   = 0
mprotect(0x8049000, 4096, PROT_READ)    = 0
mprotect(0xb78f6000, 4096, PROT_READ)   = 0
munmap(0xb78be000, 102531)              = 0
open("/tmp/fconc.out.tmp", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
fcntl64(3, F_GETLK, {type=0xffff838d /* F_??? */, whence=0xffffff18 /* SEEK_??? */, start
    =-953548801, len=-2063401023, pid=824800511}) = -1 EINVAL (Invalid argument)
fcntl64(3, F_SETLK, {...})              = -1 EFAULT (Bad address)
open("A", O_RDONLY)                     = 4
fcntl64(4, F_GETLK, {...})              = -1 EFAULT (Bad address)
fcntl64(4, F_SETLK, {...})              = -1 EFAULT (Bad address)
read(4, "asdf\n", 1024)                 = 5
write(3, "asdf\n", 5)                   = 5
read(4, "", 1024)                       = 0
fcntl64(4, F_SETLK, {...})              = -1 EFAULT (Bad address)
close(4)                                = 0
open("B", O_RDONLY)                     = 4
fcntl64(4, F_GETLK, {...})              = -1 EFAULT (Bad address)
fcntl64(4, F_SETLK, {...})              = -1 EFAULT (Bad address)
read(4, "lkjh\n", 1024)                 = 5
write(3, "lkjh\n", 5)                   = 5
read(4, "", 1024)                       = 0
fcntl64(4, F_SETLK, {...})              = -1 EFAULT (Bad address)
close(4)                                = 0
open("C", O_RDONLY)                     = 4
fcntl64(4, F_GETLK, {...})              = -1 EFAULT (Bad address)
fcntl64(4, F_SETLK, {...})              = -1 EFAULT (Bad address)
read(4, "test\n", 1024)                 = 5
write(3, "test\n", 5)                   = 5
read(4, "", 1024)                       = 0
fcntl64(4, F_SETLK, {...})              = -1 EFAULT (Bad address)
close(4)                                = 0
open("D", O_RDONLY)                     = 4
fcntl64(4, F_GETLK, {...})              = -1 EFAULT (Bad address)
fcntl64(4, F_SETLK, {...})              = -1 EFAULT (Bad address)
read(4, "test2\n", 1024)                = 6
write(3, "test2\n", 6)                  = 6
read(4, "", 1024)                       = 0
fcntl64(4, F_SETLK, {...})              = -1 EFAULT (Bad address)
close(4)                                = 0
open("E", O_RDONLY)                     = 4
fcntl64(4, F_GETLK, {...})              = -1 EFAULT (Bad address)
fcntl64(4, F_SETLK, {...})              = -1 EFAULT (Bad address)
read(4, "test3\ntest4\n", 1024)         = 12
write(3, "test3\ntest4\n", 12)          = 12
read(4, "", 1024)                       = 0
```

```
fcntl64(4, F_SETLK, {...})             = -1 EFAULT (Bad address)
close(4)                               = 0
fcntl64(3, F_SETLK, {...})             = -1 EFAULT (Bad address)
close(3)                               = 0
open("F", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
fcntl64(3, F_GETLK, {...})             = -1 EFAULT (Bad address)
fcntl64(3, F_SETLK, {...})             = -1 EFAULT (Bad address)
open("/tmp/fconc.out.tmp", O_RDONLY)   = 4
fcntl64(4, F_GETLK, {...})             = -1 EFAULT (Bad address)
fcntl64(4, F_SETLK, {...})             = -1 EFAULT (Bad address)
read(4, "asdf\nlkjh\ntest\ntest2\ntest3\ntest4"..., 1024) = 33
write(3, "asdf\nlkjh\ntest\ntest2\ntest3\ntest4"..., 33) = 33
read(4, "", 1024)                      = 0
fcntl64(4, F_SETLK, {...})             = -1 EFAULT (Bad address)
close(4)                               = 0
fcntl64(3, F_SETLK, {...})             = -1 EFAULT (Bad address)
close(3)                               = 0
unlink("/tmp/fconc.out.tmp")           = 0
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:

```
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) = 0
       xb780d000
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.