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

26 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 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 : Thu Nov 24 11:37:08 2011

 * 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 <sys/stat.h>
#include <stdlib.h>
#include <stdio.h>
#include <sysexits.h>
#include <string.h>

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

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

 * File Name : fconc.c

 * Last Modified : Thu Nov 24 12:31:54 2011

 * 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 i;
  const char *output;
  int duplicate = 0;
  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)
  {
    output = "fconc.out";
  }
  else if (argc == 4)
  {
    output = argv[3];
  }
  else
  {
    perror("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
    exit(EX_USAGE);
  }

  for (i=1; i<3; i++)
  {
```

```
    if (strcmp (argv[i], output) == 0)
    {
      duplicate = 1;
      break;
    }
  }

  if (duplicate)               //if outfile matches an infile, work on a tempfile
  {
    TMP = open("fconc.out.tmp",W_FLAGS,C_PERMS);
    if (TMP < 0)
    {
      perror("Error opening tmp file, is another instance running?\n");
      exit(EX_TEMPFAIL);
    }

    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
    OUT = open(output,W_FLAGS,C_PERMS);

    if (OUT < 0)
    {
      perror("Error handling output file\n");
      exit(EX_IOERR);
    }
    fcntl(OUT,F_GETLK,&lock);
    lock.l_type = F_WRLCK;
    fcntl(OUT,F_SETLK,&lock);

    write_file(OUT,"fconc.out.tmp");
    lock.l_type = F_UNLCK;
    fcntl(OUT,F_SETLK,&lock);
    close (OUT);
    if (unlink("fconc.out.tmp") != 0)
    {
      perror("Error deleting temporary file, please remove fconc.out.tmp\n");
      exit(EX__BASE);
    }
  }

  else
  {
    OUT = open(output,W_FLAGS,C_PERMS);
    if (OUT < 0)
    {
      perror("Error handling output file\n");
      exit(EX_IOERR);
    }
    fcntl(OUT,F_GETLK,&lock);
    lock.l_type = F_WRLCK;
    fcntl(OUT,F_SETLK,&lock);

    write_file(OUT,argv[1]);
    write_file(OUT,argv[2]);

    lock.l_type = F_UNLCK;
    fcntl(OUT,F_SETLK,&lock);
    if ( close(OUT) != 0)
    {
      perror("Error in closing");
      exit(1);
    }
  }
  return 0;
  //exit(EXIT_SUCCESS);
}

void doWrite(int fd,const char *buff,int len)
{
```

```
  int written = 0;
  int current = 0;
  do
  {
    if ( (current = write(fd,buff+written,len-written)) < 0 )
    {
      perror("Error in writing\n");
      exit(EX_IOERR);
    }
    written+=current;
  } 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)
  {
    char error_message[BUFFER_SIZE];
    snprintf(error_message,BUFFER_SIZE,"%s",infile);
    perror(error_message);
    exit(EX_NOINPUT);
  }
  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 )
  {
    perror("Read Error\n");
    exit(EX_IOERR);
  }
  lock.l_type = F_UNLCK;  //set lock to unlock
  fcntl(A,F_SETLK,&lock);  //set lock on A
  //ok close
  if ( close(A) == - 1 )
  {
    perror("Close Error\n");
    exit(EX_IOERR);
  }
}
```

```
all:            fconc
fconc:          fconc.o
        gcc -g fconc.o -o fconc -m32
fconc.o:        fconc.c fconc.h
        gcc -g -c fconc.c -o fconc.o -Wall -Wextra -m32
.PHONY: clean test strace
clean:
        rm fconc.o fconc C
test:
        echo -n "Goodbye " > A;
        echo "and thanks for all the fish" > B;
        ./fconc A B C
        cat C
strace:
        strace -o strace_outfile ./fconc A B C
```

Η έξοδος της strace είναι η παρακάτω:

```
execve("./fconc", ["./fconc", "A", "B", "C"], [/* 46 vars */]) = 0
brk(0)                                  = 0x9746000
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb784f000
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=105890, ...}) = 0
mmap2(NULL, 105890, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb7835000
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) = 0xb76d5000
mprotect(0xb782e000, 4096, PROT_NONE)   = 0
mmap2(0xb782f000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x159) = 0
    xb782f000
mmap2(0xb7832000, 10792, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0
    xb7832000
close(3)                                = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb76d4000
set_thread_area({entry_number:-1 -> 6, base_addr:0xb76d46c0, limit:1048575, seg_32bit:1, contents:0,
     read_exec_only:0, limit_in_pages:1, seg_not_present:0, useable:1}) = 0
mprotect(0xb782f000, 8192, PROT_READ)   = 0
mprotect(0x8049000, 4096, PROT_READ)    = 0
mprotect(0xb786d000, 4096, PROT_READ)   = 0
munmap(0xb7835000, 105890)              = 0
open("C", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
fcntl64(3, F_GETLK, {type=0xffff8955 /* F_??? */, whence=0x57e5 /* SEEK_??? */, start=15225686, len
    =1526726656, pid=330941313}) = -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, "Goodbye ", 1024)               = 8
write(3, "Goodbye ", 8)                 = 8
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, "and thanks for all the fish\n", 1024) = 28
write(3, "and thanks for all the fish\n", 28) = 28
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
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. Την αλλαγή αυτή την κάνει ο linker σε στάδιο μετά τη μεταγλώττιση. Συγκεκριμένα, οφείλεται στο ότι ο linker θα αποτιμήσει την τιμή της διεύθυνσης που βρίσκεται η συνάρτηση, αφού πάρει το αρχείο zing.o, όπου και θα μας δώσει το τελικό εκτελέσιμο zing.

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

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

* File Name : fconc.h

* Last Modified : Thu Nov 24 11:33:17 2011

* 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 <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <sysexits.h>
#include <string.h>

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

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

 * File Name : fconc.c

```

```
 * Last Modified : Thu Nov 24 11:34:39 2011

 * 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 i;
  const char * output;
  int duplicate = 0;
  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 == 1)
  {
    perror("Use at least 1 file name when calling fconc");
    exit(EX_USAGE);
  }
  if (argc == 2)
  {
    //No need to chance anything
    exit(0);
  }
  else
  {
    output = argv[argc-1];
  }

  for (i=1; i<(argc-1); i++)
  {
    if (strcmp (argv[i], output) ==0 )
    {
      duplicate = 1;
      break;
    }
  }

  if (duplicate)
  {

    TMP = open("/tmp/fconc.out.tmp",W_FLAGS,C_PERMS);
    if (TMP < 0)
    {
      perror("Error opening tmp file, is another instance running?\n");
      exit(EX_TEMPFAIL);
    }
    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(i=1; i <(argc-1); i++)
    {
      write_file(TMP,argv[i]);
    }
    lock.l_type = F_UNLCK;    //set lock to unlock
    fcntl(TMP,F_SETLK,lock);  //set the lock on fd
    close(TMP);               //close fd
    OUT = open(output,W_FLAGS,C_PERMS);

    if (OUT < 0)
    {
      perror("Error handling output file\n");
      exit(EX_IOERR);
    }
    fcntl(OUT,F_GETLK,lock);
    lock.l_type = F_WRLCK;
    fcntl(OUT,F_SETLK,lock);

```

```
80      write_file(OUT,"/tmp/fconc.out.tmp");
81      lock.l_type = F_UNLCK;
82      fcntl(OUT,F_SETLK,lock);
83      close (OUT);
84      if (unlink("/tmp/fconc.out.tmp") != 0)
85      {
86        perror("Error deleting temporary file, please remove /tmp/fconc.out.tmp\n");
87        exit(EX__BASE);
88      }
89    }
90
91    else
92    {
93      OUT = open(output,W_FLAGS,C_PERMS);
94      if (OUT < 0)
95      {
96        perror("Error handling output file\n");
97        exit(EX_IOERR);
98      }
99      fcntl(OUT,F_GETLK,lock);
100     lock.l_type = F_WRLCK;
101     fcntl(OUT,F_SETLK,lock);
102     for (i=1;i<(argc-1);i++)
103     {
104       write_file(OUT,argv[i]);
105     }
106     lock.l_type = F_UNLCK;
107     fcntl(OUT,F_SETLK,lock);
108     close(OUT);
109   }
110
111   exit(EXIT_SUCCESS);
112 }
113
114
115 void doWrite(int fd,const char *buff,int len)
116 {
117   int written = 0;
118   int current = 0;
119
120   do
121   {
122     if ( (current = write(fd,buff+written,len-written)) < 0 )
123     {
124       perror("Error in writing\n");
125       exit(EX_IOERR);
126     }
127     written+=current;
128   } while(written < len );
129 }
130
131 void write_file(int fd,const char *infile)
132 {
133   int A;
134   char buffer[BUFFER_SIZE];
135   int chars_read=0;
136   struct flock lock;
137
138   A = open(infile,O_RDONLY);
139   if (A ==-1)
140   {
141     char error_message[BUFFER_SIZE];
142     sprintf(error_message,"%s",infile);
143     perror(error_message);
144     exit(EX_NOINPUT);
145   }
146   fcntl(A,F_GETLK,lock);  //get lock info on A
147   lock.l_type = F_RDLCK;  //set lock to read lock
148   fcntl(A,F_SETLK,lock);  //set lock on A
149   //time to read
150   while( (chars_read = read(A,buffer,BUFFER_SIZE)) > 0)
151   {
152     //and write
153     doWrite(fd,buffer,chars_read);
154   }
```

```
  if ( chars_read == -1 )
  {
    perror("Read Error\n");
    exit(EX_IOERR);
  }
  lock.l_type = F_UNLCK;  //set lock to unlock
  fcntl(A,F_SETLK,lock);  //set lock on A
  //ok close
  if ( close(A) == - 1 )
  {
    perror("Close Error\n");
    exit(EX_IOERR);
  }
}
```

```
all:            fconc
fconc:          fconc.o
        gcc fconc.o -o fconc -m32
fconc.o:        fconc.c fconc.h
        gcc -c fconc.c -o fconc.o -Wall -m32
.PHONY: clean test
clean:
        rm fconc.o fconc A B C D E F
test:
        echo "This is file A" > A
        echo "This is file B" > B
        echo "Right guess, file C"  > C
        echo "Yep, that's file D" > D
        echo "And that's file E" > E
        ./fconc A B C D E A F
        cat F
strace:
        strace -o strace_outfile ./fconc A B C D E F
```

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
brk(0)                                  = 0x92d3000
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb782d000
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) = 0xb7810000
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) = 0xb76b0000
mprotect(0xb7809000, 4096, PROT_NONE)   = 0
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
     xb780d000
close(3)                                = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb76af000
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)   = 0
mprotect(0xb784b000, 4096, PROT_READ)   = 0
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.