Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Λειτουργικά Συστήματα – Άσκηση 1

Των Προπτυχιακών Φοιτητών
Αρώνη Παναγιώτη Α.Μ. 03109083
Δανασή Παναγιώτη Α.Μ. 03109004
e-mail: el09004@mail.ntua.gr
panaro32@hotmail.com

1 Ασκήσεις:

1.1 Σύνδεση με αρχείο αντικειμένων:

main.c

```
#include "zing.h"

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

Makefile

```
CC = gcc -std=c99
CF = -Wall -O3

zing : main.o zing.o
	$(CC) main.o zing.o -o zing

zing2 : main.o zing2.o
	$(CC) main.o zing2.o -o zing2

zing2.o : zing2.c
	$(CC) $(CF) -c zing2.c

main.o : main.c
	$(CC) $(CF) -c main.c

clean :
	rm -f main.o zing2.o
```

zing2.c

```
#include <unistd.h>
#include <string.h>

void zing()
{
	char *str=getlogin();
	int size=strlen(str);
	write(STDOUT_FILENO,"Don't burn yourself out ",24);
	write(STDOUT_FILENO,str,size);
	write(STDOUT_FILENO,"! :P\n",5);
	return;
}
```

Η διαδικασία μεταγλώττισης γίνεται με την εντολή:

gcc –Wall –c main.c

ενώ η διαδικασία σύνδεσης με την εντολή:

gcc –Wall main.o zing.o -o zing

Η έξοδος εκτέλεσης του προγράμματος zing είναι:

Hello oslabc02!

ενώ για το πρόγραμμα zing2 είναι:

Don't burn yourself out oslabc02! :P

Απάντηση Ερωτήσεων 1.1:

1. Η επικεφαλίδα περιέχει την δήλωση της συνάρτησης zing() που χρησιμοποιείται στην main και συνεπώς πρέπει να δηλωθεί.
2. Ο κώδικας του Makefile παρατίθεται στην αρχή της άσκησης.
3. Ο κώδικας του zing2 παρατίθεται στην αρχή της άσκησης.
4. Για να αντιμετωπίσουμε το πρόβλημα θα χωρίσουμε τις συναρτήσεις μας σε περισσότερα αρχεία, το καθένα εκ των οποίων θα περιέχει λιγότερες συναρτήσεις. Έτσι κάθε φορά που αλλάζουμε κάτι σε μία συνάρτηση, θα χρειάζεται να κάνουμε ξανά compile μόνο το αρχείο που περιείχε αυτήν την συνάρτηση, μειώνοντας έτσι τον χρόνο μεταγλώττισης. Επίσης θα ήταν χρήσιμο να φτιάχναμε και ένα Makefile, το οποίο θα αναλάμβανε κάθε φορά να κάνει compile μόνο τα αρχεία που χρειάζεται να ξαναμεταγλωττίσουμε.
5. Αυτό που συνέβη με την εντολή “gcc –Wall –o foo.c foo.c” είναι ότι ζητήσαμε από τον compiler να μεταγλωττίσει το foo.c και το αρχείο που θα παραχθεί να το ονομάσει πάλι foo.c. Έτσι χάσαμε το αρχικό μας αρχείο, μια και είχε το ίδιο όνομα με το output.

1.2 Συνένωση δύο αρχείων σε τρίτο:

fconc.c

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>

#define LEN 1000

void doWrite(int,const char*,int);
void write_file(int,const char*);

int main(int argc,char *argv[])
{
	if(argc<3)
	{
		write(STDERR_FILENO,"Usage: ",7);
		write(STDERR_FILENO,argv[0],strlen(argv[0]));
		write(STDERR_FILENO," infile1 infile2 [outfile (default:fconc.out)]\n",47);
		return 1;
	}
	int fdout;
	if(argc>3)
	{
	fdout=open(argv[3],O_WRONLY|O_TRUNC|O_CREAT,S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH);
		if(fdout<0)
		{
			perror("Could not open output file");
			exit(1);
		}
	}
	else
	{
		argv[argc++]="fconc.out";
		fdout=open(argv[3],O_CREAT|O_WRONLY,S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH);
		if(fdout<0)
		{
			perror("Could not create output file");
			exit(1);
		}
	}
	if(strcmp(argv[3],argv[1])) write_file(fdout,argv[1]);
	else write(STDERR_FILENO,"Output file is also an input file.Data on this file lost.\n",58);
	if(strcmp(argv[3],argv[2])) write_file(fdout,argv[2]);
	else write(STDERR_FILENO,"Output file is also an input file.Data on this file lost.\n",58);
	if(close(fdout)<0)
	{
		perror("Could not close output file");
		exit(1);
	}
	return 0;
}
```

```
void doWrite(int fd,const char *buff,int len)
{
	int idx=0,wcnt;
	while(idx<len)
	{
		wcnt=write(fd,buff+idx,len-idx);
		if(wcnt<0)
		{
			perror("Could not write output file");
			if(close(fd)<0)
			{
				perror("Could not close output file");
				exit(1);
			}
			exit(1);
		}
		idx+=wcnt;
	}
	return;
}
```

```
void write_file(int fd,const char *infile)
{
	int fdin=open(infile,O_RDONLY);
	if(fdin<0)
	{
		perror("Could not open input file");
		if(close(fd)<0)
		{
			perror("Could not close output file");
			exit(1);
		}
		exit(1);
	}
	int len;
	char *buff=(char*)malloc(LEN);
	if(!buff)
	{
		write(STDERR_FILENO,"Could not allocate memory\n",26);
		if(close(fd)<0)
		{
			perror("Could not close output file");
			exit(1);
		}
		exit(1);
	}
	while((len=read(fdin,buff,LEN)))
	{
		if(len<0)
		{
			perror("Could not read input file");
			if(close(fd)<0)
			{
				perror("Could not close output file");
				exit(1);
			}
			exit(1);
		}
		doWrite(fd,buff,len);
	}
	if(close(fdin)<0)
	{
		perror("Could not close input file");
		if(close(fd)<0)
		{
			perror("Could not close output file");
			exit(1);
		}
		exit(1);
	}
	return;
}
```

Απάντηση Ερωτήσεων 1.2:

```
execve("./fconc", ["./fconc", "a.txt", "b.txt", "myout.txt"], [/* 40 vars */]) = 0
brk(0)                          = 0x9608000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb76f1000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=65574, ...}) = 0
mmap2(NULL, 65574, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb76e0000
close(3)                        = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/i386-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0000\226\1\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1730024, ...}) = 0
mmap2(NULL, 1743580, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0xb7536000
mprotect(0xb76d9000, 4096, PROT_NONE)   = 0
mmap2(0xb76da000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1a3) = 0xb76da000
mmap2(0xb76dd000, 10972, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS,
-1, 0) = 0xb76dd000
close(3)                        = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xb7535000
set_thread_area({entry_number:-1 -> 6, base_addr:0xb7535900, limit:1048575, seg_32bit:1, contents:0,
read_exec_only:0, limit_in_pages:1, seg_not_present:0, useable:1}) = 0
mprotect(0xb76da000, 8192, PROT_READ)   = 0
mprotect(0x8049000, 4096, PROT_READ)    = 0
mprotect(0xb7714000, 4096, PROT_READ)   = 0
munmap(0xb76e0000, 65574)               = 0
open("myout.txt", O_WRONLY|O_CREAT|O_TRUNC, 0644) = 3
open("a.txt", O_RDONLY)                 = 4
brk(0)                          = 0x9608000
brk(0x9629000)                  = 0x9629000
read(4, "This is file A.\n", 1000)      = 16
write(3, "This is file A.\n", 16)       = 16
read(4, "", 1000)                   = 0
close(4)                        = 0
open("b.txt", O_RDONLY)                 = 4
read(4, "This is file B.\n", 1000)      = 16
write(3, "This is file B.\n", 16)       = 16
read(4, "", 1000)                   = 0
close(4)                        = 0
close(3)                        = 0
exit_group(0)                       = ?
```

Προαιρετικές Ερωτήσεις:

1. Εκτελώντας την εντολή ‘ strace –c strace echo “hello” ‘ παίρνουμε τα εξής αποτελέσματα:

% time	seconds	usecs/call	calls	errors	syscall
62.43	0.000211	3	78		wait4
37.57	0.000127	0	544		ptrace
0.00	0.000000	0	1		read
0.00	0.000000	0	75		write
0.00	0.000000	0	2		open
0.00	0.000000	0	2		close
0.00	0.000000	0	1		execve
0.00	0.000000	0	3	3	access
0.00	0.000000	0	3		brk
0.00	0.000000	0	1		munmap
0.00	0.000000	0	1		clone
0.00	0.000000	0	4		mprotect
0.00	0.000000	0	8		rt_sigaction
0.00	0.000000	0	156		rt_sigprocmask
0.00	0.000000	0	6		mmap2
0.00	0.000000	0	8	6	stat64
0.00	0.000000	0	2		fstat64
0.00	0.000000	0	1		getuid32
0.00	0.000000	0	1		getgid32
0.00	0.000000	0	1		set_thread_area
100.00	0.000338		898	9 total	

Από τον παραπάνω πίνακα λοιπόν βλέπουμε ότι η κλήση συστήματος με την οποία υλοποιείται η strace είναι η ptrace.

2. Την αλλαγή αυτή την έκανε ο linker, καθώς πριν από το linking η main δεν γνώριζε την συνάρτηση που καλούσε παρά μόνο είχε την δήλωσή της από την επικεφαλίδα zing.h. Αντίθετα, στο εκτελέσιμο zing όπου έχει προηγηθεί το στάδιο του linking των δύο object files (main.o και zing.o) πλέον η main γνωρίζει τον ορισμό της συνάρτησης που θα εκτελέσει.

3. Παρατίθεται ο κώδικας για την υποστήριξη αόριστου αριθμού αρχείων εισόδου:

fconc2.c

```
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <fcntl.h>

#define LEN 1000

void doWrite(int,const char*,int);
void write_file(int,const char*);

int main(int argc,char *argv[])
{
	if(argc<3)
	{
		write(STDERR_FILENO,"Usage: ",7);
		write(STDERR_FILENO,argv[0],strlen(argv[0]));
		write(STDERR_FILENO," infile1 infile2 [...] [outfile (default:fconc2.out)]\n",54);
		return 1;
	}
	int fdout;
	if(argc>3)
	{
		fdout=open(argv[argc-
1],O_WRONLY|O_TRUNC|O_CREAT,S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH);
		if(fdout<0)
		{
			perror("Could not open output file");
			exit(1);
		}
	}
	else
	{
		argv[argc++]="fconc2.out";
		fdout=open(argv[argc-1],O_CREAT|O_WRONLY,S_IRUSR|S_IWUSR|S_IRGRP|S_IROTH);
		if(fdout<0)
		{
			perror("Could not create output file");
			exit(1);
		}
	}
	for(int cnt=1;cnt<argc-1;cnt++)
		if(strcmp(argv[argc-1],argv[cnt])) write_file(fdout,argv[cnt]);
		else write(STDERR_FILENO,"Output file is also an input file.Data on this file lost.\n",58);
	if(close(fdout)<0)
	{
		perror("Could not close output file");
		exit(1);
	}
	return 0;
}
```

```
void doWrite(int fd,const char *buff,int len)
{
	int idx=0,wcnt;
	while(idx<len)
	{
		wcnt=write(fd,buff+idx,len-idx);
		if(wcnt<0)
		{
			perror("Could not write output file");
			if(close(fd)<0)
			{
				perror("Could not close output file");
				exit(1);
			}
			exit(1);
		}
		idx+=wcnt;
	}
	return;
}
```

```
void write_file(int fd,const char *infile)
{
	int fdin=open(infile,O_RDONLY);
	if(fdin<0)
	{
		perror("Could not open input file");
		if(close(fd)<0)
		{
			perror("Could not close output file");
			exit(1);
		}
		exit(1);
	}
	int len;
	char *buff=(char*)malloc(LEN);
	if(!buff)
	{
		write(STDERR_FILENO,"Could not allocate memory\n",26);
		if(close(fd)<0)
		{
			perror("Could not close output file");
			exit(1);
		}
		exit(1);
	}
	while((len=read(fdin,buff,LEN)))
	{
		if(len<0)
		{
			perror("Could not read input file");
			if(close(fd)<0)
			{
				perror("Could not close output file");
				exit(1);
			}
			exit(1);
		}
		doWrite(fd,buff,len);
	}
	if(close(fdin)<0)
	{
		perror("Could not close input file");
		if(close(fd)<0)
		{
			perror("Could not close output file");
			exit(1);
		}
		exit(1);
	}
	return;
}
```

4. Τρέχοντας ένα strace για το εκτελέσιμο whoops παίρνουμε τα εξής αποτελέσματα:

```
execve("./whoops", ["./whoops"], [/* 17 vars */]) = 0
[ Process PID=22041 runs in 32 bit mode. ]
brk(0)                          = 0x9762000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0xffffffffff7704000
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=38829, ...}) = 0
mmap2(NULL, 38829, PROT_READ, MAP_PRIVATE, 3, 0) = 0xffffffffff76fa000
close(3)                        = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib32/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\0n\1\0004\0\0\0\374"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1327556, ...}) = 0
mmap2(NULL, 1337704, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0xffffffffff75b3000
mprotect(0xf76f3000, 4096, PROT_NONE)   = 0
mmap2(0xf76f4000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x140) = 0xffffffffff76f4000
mmap2(0xf76f7000, 10600, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -
1, 0) = 0xffffffffff76f7000
close(3)                        = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0xffffffffff75b2000
set_thread_area(0xffae66d8)           = 0
open("/dev/urandom", O_RDONLY)        = 3
read(3, "\262\226\273"..., 3)         = 3
close(3)                        = 0
mprotect(0xf76f4000, 8192, PROT_READ)   = 0
mprotect(0xf7723000, 4096, PROT_READ)   = 0
munmap(0xf76fa000, 38829)             = 0
open("/etc/shadow", O_RDONLY)         = -1 EACCES (Permission denied)
write(2, "Problem!\n"..., 9Problem!
)          = 9
exit_group(1)                   = ?
```

Στην γραμμή 28 παρατηρούμε ότι το εκτελέσιμο προσπαθεί να ανοίξει το αρχείο "/etc/shadow" για ανάγνωση, όμως του απαγορεύεται η πρόσβαση (η κλήση open επιστρέφει -1 EACCES (Permission denied)). Συνεπώς εκεί δημιουργείται το πρόβλημα, μια και δεν μπορεί να ανοίξει το αρχείο και στην επόμενη κλήση συστήματος γράφει το μήνυμα "Problem!\n" που βλέπουμε στην οθόνη.