National Technical University of Athens

School of Electrical and Computer Engineering

Λειτουργικά Συστήματα Υπολογιστών (Τμήμα 1)

Άσκηση 1: Εισαγωγή στο περιβάλλον προγραμματισμού

ΧΡΙΣΤΟΦΟΡΟΣ ΒΑΡΔΑΚΗΣ (03118883)

ΓΕΩΡΓΙΟΣ ΛΥΜΠΕΡΑΚΗΣ (03118881)

oslaba69

24 Μαρτίου 2021

Άσκηση : Σύνδεση με αρχείο αντικειμένων

1.

Η κεφαλίδα (header file) χρησιμοποιείτε για να δηλωθεί η συνάρτηση και τα ορίσματα της στον κώδικα της *main*. Χωρίς αυτήν θα έπρεπε να δηλώσουμε χειροκίνητα τη συνάρτηση ή να έχουμε πρόσβαση στον πηγαίο κώδικά της. Στην συγκεκριμένη όμως περίπτωση, όπου χωρίς την επικεφαλίδα δε γνωρίζουμε το όνομα και τα ορίσματα και δεν έχουμε πρόσβαση στον πηγαίο κώδικα θα ήταν αδύνατο να χρησιμοποιήσουμε την συγκεκριμένη συνάρτηση στο πρόγραμμα μας.

2.

Αρχικά, δημιουργούμαι το παρακάτω Makefile έτσι ώστε να παράγεται το ζητούμενο εκτελέσιμο.

Κώδικας του Makefile

```
#Compiler
cc = gcc

#Compiler Flags
CFLAGS = -Wall

#Targets
TARGET = zing clean

all: $(TARGET)

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

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

zing.o:
	touch  zing.o

clean :
	rm main.o
```

Για να μπορέσουμε να ανακαλύψουμε το περιεχόμενο του αρχείου zing.o δημιουργούμε μια πρόχειρη main η οποία την καλεί. Οπότε,

Κώδικας της main.c

```
#include "zing.h"


int main(int argc, char * argv[]) {
  zing();

  return 0;
}
```

Με τις παρακάτω εντολές μεταγλωττίζουμε τα αρχεία και βλέπουμε και την **έξοδο**.

```
oslaba69@os-node1:~/Ex_1/ex11$ make
gcc -Wall main.c -c
gcc -Wall main.o zing.o -o test
oslaba69@os-node1:~/Ex_1/ex11$ ./test
Hello, oslaba69
```

3.

Στο τρέχων ερώτημα υλοποιούμαι την δική μας συνάρτηση zing() και μετατρέπουμε το Makefile έτσι ώστε να παράγονται δύο εκτελέσιμα αρχεία.

Επομένως , έχουμε τον κάτωθι κώδικα

Κώδικας της main.c

```c
#include "zing.h"


int main(int argc, char * argv[]) {
  zing();

  return 0;
}
```

Κώδικας της zing.h

```c
#ifndef ZING_H__
#define ZING_H__

void zing(void);

#endif
```

Κώδικας της zing2.c

```c
#include "stdio.h"
#include <unistd.h>

void zing(void){
    char* username = getlogin();
    printf("Hello, %s !! I am the second version\n",username);
}
```

Κώδικας του Makefile

```
#Compiler
cc = gcc

#Compiler Flags
CFLAGS = -Wall

#Targets
TARGET = zing zing2 clean

all: $(TARGET)

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

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

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

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

zing.o:
	touch  zing.o

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

Με τις παρακάτω εντολές μεταγλωττίζουμε τα αρχεία

```
oslaba69@os-node1:~/Ex_1$ make
gcc -Wall main.c -c
gcc -Wall main.o zing.o -o zing
gcc -Wall zing2.c -c
gcc -Wall main.o zing2.o -o zing2
oslaba69@os-node1:~/Ex_1$
```

Η **έξοδος** είναι η :

```
oslaba69@os-node1:~/Ex_1$ ./zing
Hello, oslaba69
oslaba69@os-node1:~/Ex_1$ ./zing2
Hello, oslaba69 !! I am the second version
```

4.

Μπορούμε να γράψουμε την κάθε συνάρτηση σε ξεχωριστό αρχείο και να τις εισάγουμε στον κώδικα μας μέσω header file(s). Έπειτα με την δημιουργία κατάλληλου ***makefile*** κατά την δημιουργία του εκτελέσιμου θα μεταγλωττίζεται μόνο η συνάρτηση που έχει τροποποιηθεί αποφεύγοντας με τον τρόπο αυτό την μεταγλώττιση ολοκλήρου του προγράμματος.

5.

Το αρχείο *foo.c* χάθηκε μετά την εντολή *"gcc -Wall -o foo.c foo.c"* για τον λόγο ότι ο συνεργάτης μας με τη χρήση της εντολή "-o" ,η οποία μας προσφέρει την ευκαιρία να ονομάσουμε το object file όπως επιθυμούμε, ζητά να ονομάσει το object file που παράγεται κατά τη διαδικασία της μεταγλώττισης του αρχείο foo.c με το όνομα του αρχείου που εμπεριέχει τον πηγαίο κώδικα ,δηλαδή foo.c .Οπότε, το object file δε χρειάζεται να δημιουργήσει νέο αρχείο και να το ονομάσει foo.c μιας και ήδη υπάρχει έτοιμο και άρα πάει στο αρχείο και αντιγράφει εκεί τα δεδομένα του τα οποία είναι σε γλώσσα μηχανής (πράγμα που κάνει αδύνατη την κατανόηση του από έναν άνθρωπο).

Άσκηση : Συνένωση δύο αρχείων σε τρίτο

Σε πρώτη φάση παρουσιάζουμε τον κώδικα που γράφτηκε ώστε να υλοποιηθεί η ζητούμενη εργασία . Πιο συγκεκριμένα, έχουμε τα κάτωθι

Κώδικας του writefile.c

```c
#include"headers.h"

void write_file(int fd, const char *infile)
{
	int fdi;
	fdi = open(infile, O_RDONLY);
	char buff[1024];
	ssize_t rcnt;

	for (;;)
	{
		rcnt = read(fdi, buff, sizeof(buff) - 1);
		if (rcnt == 0) break;
		if (rcnt == -1)
		{
			perror("ERROR READ\n");
			exit(1);
		}

		buff[rcnt] = '\0';
		doWrite(fd, buff, strlen(buff));
	}

	close(fdi);
}
```

Κώδικας του doWrite.c

```c
#include "headers.h"

void doWrite(int fd, const char *buff, int len)
{
     size_t idx = 0;
     ssize_t wcnt;

     do {
          wcnt = write(fd, buff + idx, len - idx);
          if (wcnt == -1)
          {
               perror("ERROR WRITE\n");
               exit(1);
          }

          idx += wcnt;
     } while (idx < len);
  }
```

Κώδικας του fconc.c

```c
#include "headers.h"

int main(int argc, char **argv)
{
	char *outfile, *infile1, *infile2;
	if (argc < 3 || argc > 4)
	{
		printf("./fconc infile1 infile2[outfile (default:fconc.out)]\n");
		return -1;
	}
	else if (argc == 4)
	{
		outfile = argv[3];
	}
	else outfile = "fconc.out";

	infile1 = argv[1];
	infile2 = argv[2];

	int fd;
	fd = open(outfile, O_WRONLY | O_CREAT, S_IRUSR | S_IWUSR);

	if (fd == -1)
	{
		perror("ERROR OPEN OUTFILE\n");
		exit(1);
	}

	write_file(fd, infile1);
	write_file(fd, infile2);

	close(fd);

	return 0;
}
```

Κώδικας του headers.h

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

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

void write_file(int , const char *);

#endif
```

Περιεχόμενα του Makefile

```
#Compiler
cc = gcc

#Compiler Flags
CFLAGS = -Wall

#Targets
TARGET =  fconc clean

all: $(TARGET)

fconc: fconc.o doWrite.o writefile.o
	$(CC) $(CFLAGS) fconc.o doWrite.o writefile.o -o fconc

fconc.o: fconc.c
	$(CC) $(CFLAGS) fconc.c -c

doWrite.o: doWrite.c
	$(CC) $(CFLAGS) doWrite.c -c

writefile.o: writefile.c
	$(CC) $(CFLAGS) writefile.c -c

clean:
	rm fconc.o doWrite.o writefile.o
```

1.

Αρχικά, δημιουργήσαμε 3 αρχεία τα οποία θα δοθούν ως ορίσματα στη συνάρτηση fconc.c οπότε έχουμε τα παρακάτω

Περιεχόμενο Αρχείου A

```
An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs.

Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting
software for cost allocation of processor time, mass storage, printing, and other resources.
```

Περιεχόμενο Αρχείου B

```
For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between programs and the computer hardware,[1][2] although the application code is usually executed directly by the hardware and frequently makes system calls to an OS function or is interrupted by it. Operating systems are found on many devices that contain a computer β from cellular phones and video game consoles to web servers and supercomputers.
```

Και μεταγλωττίζοντας το πρόγραμμα έχουμε το εξής αποτέλεσμα

Περιεχόμενο Αρχείου C

```
An operating system (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs.

Time-sharing operating systems schedule tasks for efficient use of the system and may also include accounting
software for cost allocation of processor time, mass storage, printing, and other resources.


For hardware functions such as input and output and memory allocation, the operating system acts as an intermediary between programs and the computer hardware,[1][2] although the application code is usually executed directly by the hardware and frequently makes system calls to an OS function or is interrupted by it. Operating systems are found on many devices that contain a computer β from cellular phones and video game consoles to web servers and supercomputers.
```

Επομένως , τώρα είμαστε σε θέση να εκτελέσουμε την εντολή ***strace*** το οποίο δίνει την εξής έξοδος :

```
execve("./fconc", ["./fconc", "file1", "file2"], [/* 28 vars */]) = 0
brk(0)                                  = 0x1784000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0e0caaa000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=29766, ...}) = 0
mmap(NULL, 29766, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f0e0caa2000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\34\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1738176, ...}) = 0
mmap(NULL, 3844640, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f0e0c4e1000
mprotect(0x7f0e0c682000, 2097152, PROT_NONE) = 0
mmap(0x7f0e0c882000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a1000) =
0x7f0e0c882000
mmap(0x7f0e0c888000, 14880, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7f0e0c888000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0e0caa1000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0e0caa0000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0e0ca9f000
arch_prctl(ARCH_SET_FS, 0x7f0e0caa0700) = 0
mprotect(0x7f0e0c882000, 16384, PROT_READ) = 0
mprotect(0x7f0e0caac000, 4096, PROT_READ) = 0
munmap(0x7f0e0caa2000, 29766)           = 0
open("fconc.out", O_WRONLY|O_CREAT, 0600) = 3
open("file1", O_RDONLY)                 = 4
read(4, "An operating system (OS) is syst"..., 1023) = 353
write(3, "An operating system (OS) is syst"..., 353) = 353
read(4, "", 1023)                       = 0
close(4)                                = 0
open("file2", O_RDONLY)                 = 4
read(4, "\n\nFor hardware functions such as"..., 1023) = 476
write(3, "\n\nFor hardware functions such as"..., 476) = 476
read(4, "", 1023)                       = 0
open("file2", O_RDONLY)                 = 4
read(4, "\n\nFor hardware functions such as"..., 1023) = 476
write(3, "\n\nFor hardware functions such as"..., 476) = 476
read(4, "", 1023)                       = 0
close(4)                                = 0
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Προαιρετικές Ερωτήσεις:

1.

Στο τρέχων ερώτημα προσπαθούμε να βρούμε με ποια κλήση συστήματος υλοποιείται η εντολή ***strace***. Για αυτό το λόγο τρέχουμε μια ενδεικτική εντολή και αποθηκεύουμε τα αποτελέσματα της strace για την strace στο φάκελο file.

```
oslaba69@os-node1:~/G/pr4$ strace -o file strace echo "Hello Planet Earth!"
```

Στο αρχείο ***file*** επομένως παρατηρούμε τα αποτελέσματα και διακρίνουμε την συνεχή εμφάνιση της κλήσης συστήματος ***ptrace*** .

Επομένως παρατηρούμε ότι η ***strace*** υλοποιείται με την κλήση συστήματος ***ptrace*** (Process Trace) .

2.

Η αλλαγή που παρατηρείται στο όρισμα της εντολής ***call*** οφείλεται στον ***Linker*** . Το αρχείο main.o δεν περιέχει την υλοποίηση της ***zing()*** και για αυτό το λόγο η εντολή call δεν μπορεί να πάρει ως όρισμα της διεύθυνση που βρίσκεται η συνάρτηση ***zing()*** .Στο εκτελέσιμο zing όμως ο ***Linker*** έχει συνδέσει το περιεχόμενο της main.o με την zing.o οπότε η call έχει τώρα ως όρισμα την διεύθυνση που θα βρει την ***zing()***.Μάλιστα, κάνοντας disassemble την zing του εκτελέσιμου zing παρατηρούμε ότι η διεύθυνση της πρώτης της εντολής ταυτίζεται με το όρισμα της call.

3.

Σαν συνέχεια της άσκησης ***1.2*** τροποποιούμε τη συνάρτηση ώστε να δέχεται έναν αόριστο αριθμό αρχείων εισόδου γράφοντας την έξοδο της στο τελευταίο όρισμα. Επομένως, ο κώδικας έγινε ο κάτωθι

Κώδικας του fconc.c

```
#include "Write.h"

int main(int argc, char **argv)
{
	char *outfile;
	if (argc < 3)
	{
		printf("./fconc infile1 infile2[outfile (default:fconc.out)]\n");
		return 1;
	}
	else
	{
		outfile = argv[argc - 1];
	}

	int fd;
	fd = open(outfile, O_WRONLY | O_CREAT, S_IRUSR | S_IWUSR);
	if (fd == -1)
	{
		perror("ERROR OPEN OUTFILE\n");
		exit(1);
	}

	int i;
	for (i = 1; i < argc - 1; i++)
	{
		write_file(fd, argv[i]);
	}

	close(fd);
	return 0;
}
}
```

*Να σημειωθεί ότι ο κώδικας των άλλων συναρτήσεων και αρχείων που συνδέονται με την **fconc.c** παρέμεινε αμετάβλητος για αυτό και δεν επαναλαμβάνεται.*

4.

Για να καταλάβουμε τι λάθος υπάρχει εκτελέσαμε την εντολή ***strace*** η οποία μας έδωσε το κάτωθι αποτέλεσμα

```
execve("./whoops", ["./whoops"], [/* 26 vars */]) = 0
brk(0)                                  = 0x9fea000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xfffffffff76f3000
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=30952, ...}) = 0
mmap2(NULL, 30952, PROT_READ, MAP_PRIVATE, 3, 0) = 0xfffffffff76eb000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib32/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\1\1\1\3\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0\300\233\1\0004\0\0\0"..., 512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1750708, ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xfffffffff76ea000
mmap2(NULL, 1755772, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) =
0xfffffffff753d000
mmap2(0xf76e4000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3,
0x1a7000) = 0xfffffffff76e4000
mmap2(0xf76e7000, 10876, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0xfffffffff76e7000
close(3)                                = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0xfffffffff753c000
set_thread_area({entry_number:-1, base_addr:0xf753c700, limit:1048575, seg_32bit:1,
contents:0, read_exec_only:0, limit_in_pages:1, seg_not_present:0, useable:1}) = 0 ($
mprotect(0xf76e4000, 8192, PROT_READ)   = 0
mprotect(0xf7717000, 4096, PROT_READ)   = 0
munmap(0xf76eb000, 30952)               = 0
open("/etc/shadow", O_RDONLY)           = -1 EACCES (Permission denied)
write(2, "Problem!\n", 9)               = 9
exit_group(1)                           = ?
+++ exited with 1 +++
```

Παρατηρούμε ότι ζητήθηκε η κλήση συστήματος ***open(…)*** ώστε να διαβαστεί το περιεχόμενο ενός αρχείου που δεν έχει δοθεί άδεια στον χρήστη ώστε να το προσπελάσει. Για αυτό το λόγο η συνάρτηση επέστρεψε αρνητικό αριθμό το οποίο υποδηλώνει ότι κάτι δεν πήγε καλά (υπάρχει πρόβλημα).

Για λόγους εποπτικούς στο παραπάνω πλαίσιο η κλήση συστήματος που αναφερθήκαμε έχει υπογραμμιστεί και χρωματιστεί με το κόκκινο χρώμα.