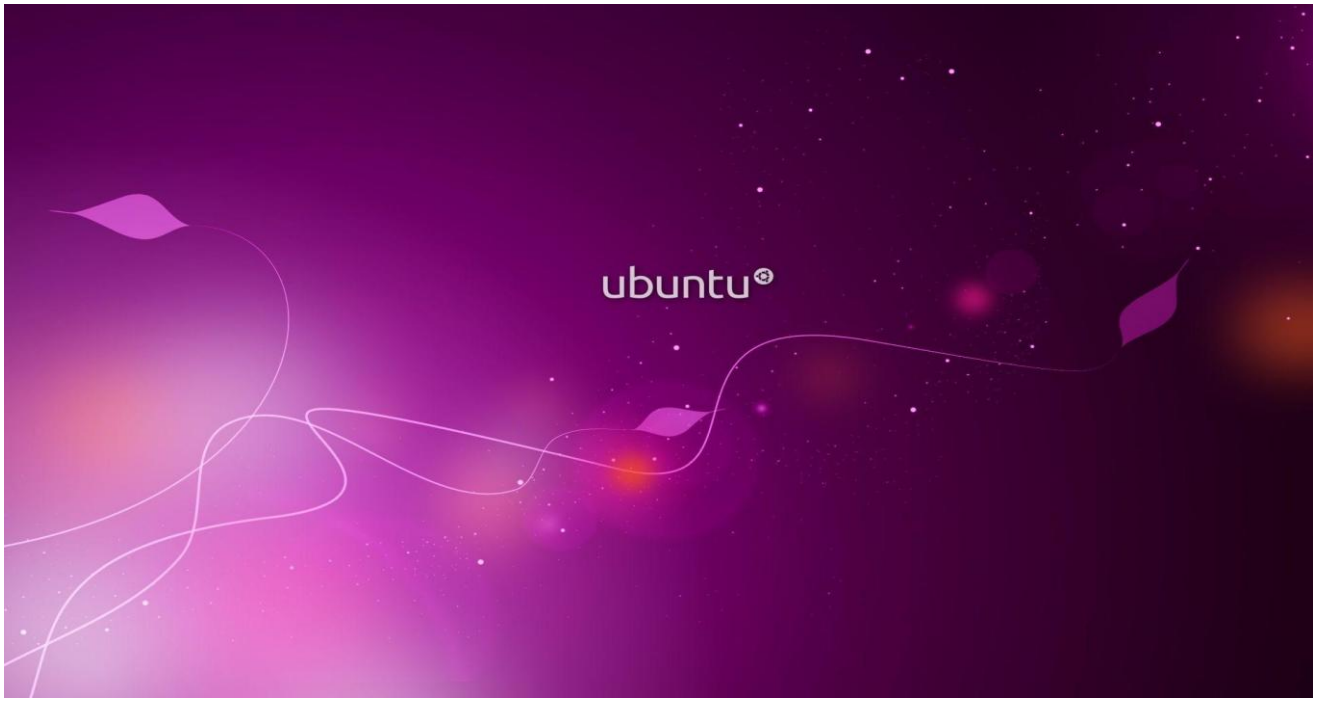
31 Μαρτίου 2020



ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ : 2019-2020

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

Άσκηση 1η - Εισαγωγή στο περιβάλλον προγραμματισμού

| Καπαρού Αλεξάνδρα (03117100)
Χλαπάνης Οδυσσέας (03117023)
Ομάδα : oslaba21

Άσκηση 1.1

Για να αντιγράψουμε τα αρχεία zing.h και zing.o στον κατάλογο εργασίας μας χρησιμοποιήσαμε την εντολή:

cp /home/oslab/code/zing/zing. .*

Δημιουργήσαμε την παρακάτω main.c:

```
#include <stdio.h>

#include"zing.h"

int main()

{

    zing();

    return 0;

}
```

Την οποία μετατρέψαμε σε main.o με την εντολή:

gcc -c main.c

Τέλος, συνδέσαμε τα δύο αρχεία αντικειμένων με την εντολή:

gcc main.o zing.o -o zing

Η έξοδος εκτέλεσης του προγράμματος (με χρήση της εντολής ./zing) είναι η:

Hello, oslaba21

➢ Ερωτήσεις

1. Το πηγαίο αρχείο τις περισσότερες φορές δεν γνωρίζει τι έχει οριστεί στα υπόλοιπα αρχεία και τα header files έρχονται να “γεμίσουν” αυτό το κενό εφοδιάζοντας το source file με δηλώσεις μεταβλητών, συναρτήσεων κλπ. Η επικεφαλίδα λοιπόν μας βοηθάει όχι μόνο να απομονώνουμε συναρτήσεις που μπορούμε να τις χρησιμοποιήσουμε πολλές φορές χωρίς να χρειάζεται να τις ξανακάνουμε compile κάθε φορά, αλλά και χωρίς να χρειάζεται να τις διαβάσουμε και να καταλάβουμε πως λειτουργούν. Ένας άλλος λόγος για τον οποίο χρησιμοποιούμε επικεφαλίδες είναι σε περιπτώσεις που δουλεύουμε συνεργατικά σε ένα μεγάλο project καθώς έτσι μπορεί κάποιος να αλλάξει την υλοποίηση μιας

συνάρτησης και εμείς το μόνο που θα χρειάζεται να κάνουμε μετά είναι να ξανακάνουμε linking τα αρχεία μεταξύ τους αφού ,όπως προαναφέραμε, κάθε αρχείο μεταγλωττίζεται μόνο του για να παράξει ένα object file και τέλος όλα τα αρχεία ενώνονται μαζί προκειμένου να δημιουργηθεί ένα εκτελέσιμο.

2. Για την δημιουργία του εκτελέσιμου της άσκησης δημιουργήσαμε το παρακάτω Makefile:

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

3. Παράξαμε το δικό μας zing2.o χρησιμοποιώντας την εντολή getlogin:

```
#include <unistd.h>
#include <stdio.h>
void zing()
{
char *lgn;
lgn=getlogin();
printf("This is %s\n",lgn);
}
```

Στην συνέχεια αλλάξαμε κατάλληλα το Makefile προκειμένου να παράγονται δύο εκτελέσιμα:

```
zing2: zing2.o main.o
    gcc -o zing2 zing2.o main.o
zing2.o: zing2.c
    gcc -Wall -c zing2.c
zing: zing.o main.o
    gcc -o zing zing.o main.o
```

main.o: main.c

gcc -Wall -c main.c

4. Ένα αρχείο το οποίο περιέχει 500 συναρτήσεις, σε περίπτωση που κάνουμε αλλαγές σε μία μόνο συνάρτηση, αναγκαστικά θα πρέπει να μεταγλωττίσει από την αρχή κάθε μία εκ των 500 συναρτήσεων και μετά να τις εκτελέσει. Όπως είναι προφανές σε ένα τέτοιο αρχείο ο χρόνος μεταγλώττισης είναι πολύ μεγάλος και πρέπει να ακολουθήσουμε άλλη τακτική προκειμένου να μειωθεί. Η τακτική αυτή είναι να χρησιμοποιήσουμε τις 499 συναρτήσεις, που δεν αλλάζουν, ως header files στο αρχείο που περιέχει την συνάρτηση που επιθυμούμε να αλλάζουμε. Με αυτόν τον τρόπο σε κάθε αλλαγή της 500ης συνάρτησης δεν χρειάζεται να μεταγλωττίζονται όλες οι υπόλοιπες, το οποίο συντελεί στην μείωση του χρόνου μεταγλώττισης.

5. Παρατηρούμε ότι η εντολή αυτή μεταγλωττίζει το αρχείο foo.c και δημιουργεί ένα εκτελέσιμο αρχείο με το ίδιο ακριβώς όνομα foo.c. Αυτό έχει σαν αποτέλεσμα να γίνεται overwrite στο αρχείο foo.c που μεταγλωττίσαμε, με αποτέλεσμα να εξαφανίζεται το αρχικό αρχείο και να αντικαθίσταται από το εκτελέσιμο. Έχουμε δηλαδή μετά το πέρας αυτής της εντολής ένα εκτελέσιμο αρχείο foo.c αλλά κανένα source code foo.c.

Άσκηση 1.2

Ο πηγαίος κώδικας που χρησιμοποιήσαμε, με βάση τον προτεινόμενο σκελετό υλοποίησης, είναι ο:

```
#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <stdlib.h>

#include <unistd.h>

#include <stdio.h>
```

```
#include <string.h>

void doWrite (int fd, const char *buff, int len)
{
	size_t  idx;
	ssize_t wcnt;
	idx = 0;
	do {
		wcnt = write(fd, buff+idx, len - idx);
		if (wcnt == -1){
			perror("write");
		}
		idx += wcnt;
	} while(idx <  len);
}

void write_file (int fd, const char *infile)
{
	int fd2, oflags;
	oflags = O_RDWR | O_APPEND;
	fd2 = open(infile, oflags);
	if (fd == -1){
		perror("open");
		exit(1);
	}

	char buff2[1024];
	ssize_t rcnt;
	rcnt = read (fd2, buff2, sizeof(buff2)-1);
	if (rcnt == 0) /* end-of-file */
```

```
        if (rcnt == -1) /* error */
        {
        perror("read");
        }
        buff2[rcnt] = '\0';
        doWrite(fd, buff2, strlen(buff2));
}
int main(int argc, char **argv)
{
int fd1,fd2,fd3;
size_t length;
char buffer[1024];
char *output;

        if ((argc<=2) || (argc>=5))             /*ola ta pithana sfalmata */
        {
        printf("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
        exit(1);
        }

fd1=open(argv[1],O_RDONLY);
        if (fd1==-1){
        printf("No such file or directory\n");
        exit(1);
        }

fd2=open(argv[2],O_RDONLY);
        if (fd2==-1){
        printf("No such file or directory\n");
        exit(1);
```

```
    }

    if (argc!=4) output="fconc.out";
    else output=argv[3];
fd3=open(output,O_WRONLY | O_CREAT |O_TRUNC , S_IRUSR | S_IWUSR);
write_file(fd3,argv[1]);
write_file(fd3,argv[2]);
}
```

Υλοποιήσαμε ωστόσο και έναν δεύτερο τρόπο επίλυσης χωρίς την χρήση της βοηθητικής συνάρτησης doWrite αλλά χρησιμοποιώντας την system call sendfile που επιτυγχάνει την ίδια ακριβώς λειτουργία αλλά ταχύτερα. Ο 2^{ος} τρόπος υλοποίησης παραδίδεται για λόγους πληρότητας:

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

void write_file(int fd, const char *infile)
{
int fd_infile;
struct stat stbuf;
fd_infile=open(infile,O_RDWR);
if (fd_infile == -1){
    perror("open");
    exit(1);
```

```
        }
fstat(fd_infile, &stbuf);
sendfile(fd,fd_infile,0,stbuf.st_size);
}
int main(int argc, char **argv)
{
int fd1,fd2,fd3;
size_t length;
char buffer[1024];
char *output;

if ((argc<=2)  || (argc>=5))              /*ola ta pithana sfalmata */
{
printf("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
exit(1);
}

fd1=open(argv[1],O_RDONLY);
if (fd1==-1){
        printf("No such file or directory\n");
        exit(1);
        }

fd2=open(argv[2],O_RDONLY);
if (fd2==-1){
        printf("No such file or directory\n");
        exit(1);
        }

if (argc!=4) output="fconc.out";
```

else output=argv[3];

fd3=open(output,O_WRONLY | O_CREAT |O_TRUNC , S_IRUSR | S_IWUSR);

write_file(fd3,argv[1]);

write_file(fd3,argv[2]);

}

➢ Ερωτήσεις

1. Χρησιμοποιήσαμε την εντολή strace ./fconc A B C (όπου A και B αρχεία που έχουν ήδη δημιουργηθεί και C το αρχείο εξόδου) και πήραμε τα κάτωθι αποτελέσματα:

```
alexandra@alexandra-Inspiron-5567:~/Desktop/1.2$ strace ./fconc A B C
execve("./fconc", ["./fconc", "A", "B", "C"], 0x7ffde38ecc88 /* 67 vars */) = 0
brk(NULL)                               = 0x560f63181000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=118000, ...}) = 0
mmap(NULL, 118000, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fa3f0ccd000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/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\0\260\34\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030544, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fa3f0ccb000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fa3f06d2000
mprotect(0x7fa3f08b9000, 2097152, PROT_NONE) = 0
mmap(0x7fa3f0ab9000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7fa3f0ab9000
mmap(0x7fa3f0abf000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fa3f0abf000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7fa3f0ccc4c0) = 0
mprotect(0x7fa3f0ab9000, 16384, PROT_READ) = 0
mprotect(0x560f62318000, 4096, PROT_READ) = 0
mprotect(0x7fa3f0cea000, 4096, PROT_READ) = 0
munmap(0x7fa3f0ccd000, 118000)          = 0
openat(AT_FDCWD, "A", O_RDONLY)         = 3
openat(AT_FDCWD, "B", O_RDONLY)         = 4
openat(AT_FDCWD, "C", O_WRONLY|O_CREAT|O_TRUNC, 0600) = 5
openat(AT_FDCWD, "A", O_RDWR|O_APPEND)  = 6
read(6, "Goodbye,\n", 1023)             = 9
write(5, "Goodbye,\n", 9)               = 9
openat(AT_FDCWD, "B", O_RDWR|O_APPEND)  = 7
read(7, "and thanks for all the fish!\n", 1023) = 29
write(5, "and thanks for all the fish!\n", 29) = 29
exit_group(0)                           = ?
+++ exited with 0 +++
```

Ωστόσο αυτά που μας ενδιαφέρουν είναι αυτά που έχουν να κάνουν αποκλειστικά με το συγκεκριμένο παράδειγμα, όσα δηλαδή περιλαμβάνουν τις εντολές open,read,write:

```
openat(AT_FDCWD, "A", O_RDONLY)         = 3
openat(AT_FDCWD, "B", O_RDONLY)         = 4
openat(AT_FDCWD, "C", O_WRONLY|O_CREAT|O_TRUNC, 0600) = 5
openat(AT_FDCWD, "A", O_RDWR|O_APPEND)  = 6
read(6, "Goodbye,\n", 1023)             = 9
write(5, "Goodbye,\n", 9)               = 9
openat(AT_FDCWD, "B", O_RDWR|O_APPEND)  = 7
read(7, "and thanks for all the fish!\n", 1023) = 29
write(5, "and thanks for all the fish!\n", 29) = 29
exit_group(0)                           = ?
+++ exited with 0 +++
```