# Λειτουργικά Συστήματα Υπολογιστών

## 1η Εργαστηριακή Αναφορά

Θοδωρής Παπαρρηγόπουλος el18040
Ορφέας Φιλιππόπουλος el18082

### Άσκηση 1

**Πηγαίος Κώδικας**

main.c:

```
#include “zing.h”
int main(int argc, char **argv) {
   zing();
   return 0;
}
```

**Ερωτήσεις**

1) Γενικά, η χρήση επικεφαλίδας χρησιμοποιείται προκειμένου να δηλώσουμε συναρτήσεις χωρίς την υλοποίηση τους. Οι επικεφαλίδες μας εξυπηρετούν για πολλούς σκοπούς. Αρχικά, όταν γίνεται compile ένα αρχείο που χρησιμοποιεί μία συνάρτηση, ο compiler χρειάζεται μόνο το declaration της συνάρτησης. Έτσι, μειώνεται ο χρόνος το compile. Επιπλέον, σε μεγάλα project, το να έχεις χωρισμένο τα αρχεία σε declaration και implementation βολεύει από άποψη ότι εύκολα ανατρέχεις να δεις τις διαθέσιμες συναρτήσεις σε κάθε αρχείο χωρίς να χρειάζεται να περνάς και τις υλοποιήσεις τις κάθε συνάρτησης.

2) Makefile:

```
all: main mainy

main: main.o zing.o
	gcc -o main main.o zing.o
mainy: main.o zing2.o
	gcc -o mainy main.o zing2.o

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

zing2.o: zing2.c
	gcc -Wall -c zing2.c

clean:
	rm -f main.o *~
```

3) Κάνουμε implement την zing() στο zing2.c ως
#include "zing2.h"

```
void zing(){
   char *s;
   if( ( s = getlogin() ) == NULL ) {
      printf( "cannot find login name\n" );
   } else {
      printf( "Hello,your login name is %s\n", s );
   }
}
```

4) Γενικά είναι good practice να σπάμε τα αρχεία μας σε υπο αρχεία. Στη συγκεκριμένη περίπτωση, μπορούμε να κάνουμε ένα νέο αρχείο που θα βάλουμε την συνάρτηση που δουλεύουμε, έτσι στο compile δεν θα ανιχνεύει αλλαγή στις υπόλοιπες 499 συναρτήσεις του άλλου αρχείο. Συνεπώς το compile θα γίνεται πολύ πιο γρήγορα σε σχέση με πριν.

5) Αυτό που έγινε ήταν ότι έδωσε για εκτελέσιμο αρχείο το αρχείο που είχε τον πηγαίο κώδικα. Έτσι, ο κώδικας που έγραφε έγινε overwrite από το αποτέλεσμα του compile (το binary file). Για πειραματικούς σκοπούς έφτιαξα ένα απλό αρχείο test.c

```
#include <stdio.h>
int main() {
        printf(“Hello world\n”);
        return 0;
}
```

Τρέξαμε gcc -Wall -o test.c test.c
Και μετά χρησιμοποιώντας το νέο test.c τρέξαμε ./test.c και έτρεξε κανονικά τυπώνοντας Hello world !

Ωστόσο, τρέχοντας την ίδια εντολή σε native linux environment με gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0 μου πέταξε το εξής error ο compiler:
gcc: fatal error: input file ‘test.c’ is the same as output file
compilation terminated.
Συνεπώς, ο συνεργάτης καλό είναι να δουλεύει locally πρώτα.

## Άσκηση 2

**Πηγαίος Κώδικας**
Έχουμε κάνει 2 διαφορετικές υλοποιήσεις.

Η πρώτη υλοποίηση είναι με άμεση χρήση των read και write και χρήση ενός δυναμικού buffer προκειμένου να κάνουμε store τα δεδομένα που διαβάζουμε. Η δεύτερη λύση χρησιμοποιεί higher level functions προκειμένου να ελαχιστοποιήσει τα system calls και είναι υλοποιημένη με την νοοτροπία του get Char από το ένα αρχείο και put Char στο αρχείο εξόδου.

1)
==========
----------fconc.c----------

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include "write_file.h"
//i#include "write_fun.h"
#include <unistd.h>
#include <stdbool.h>

struct all{

        int fd;
        char* copy;
        bool boo;
};


int main(int argc, char **argv){

    if(argc>4){
        printf("Too many arguments");
        return 0;
    }

    struct all array[2];


    if(argc > 4 || argc < 3){
        fprintf(stderr, "You need to only add 2 or 3 arguments\n");

        return 0;
    }

    int oflags = O_CREAT | O_WRONLY | O_TRUNC;

    int mode = S_IRUSR | S_IWUSR;

    int fd3;
    int i;
```

```
   for(i = 0; i < argc-1; i++){
        //printf("comparing ready\n");
        //printf("comparing = %d\n", strcmp(argv[i+1], "fconc.out"));
        if(argc==4 && i == argc-1) break;
        if((*argv[i+1] == *argv[argc-1] && argc == 4) || (0==strcmp(argv[i+1], "fconc.out") &&
argc == 3)){
            array[i].boo = true;
            printf("Waring: %s is already passed as an argument and it will be overwritten, are you
sure?\n", argv[argc-1]);
            printf("Press enter to continue or any other button and enter to do not");

            char c;

            scanf("%c",&c);

            if(c == 10){
                continue;
            }
            else return 0;

      }
   }

   FILE* files[argc-2];
   int counter = 0;
   for(i = 0; i<2; i++){
      if(array[i].boo == true){
            printf("i am bored\n");
            counter = counter+1;
            array[i].fd = open(argv[i+1], O_RDONLY);
            if(array[i].fd == -1){
            printf("%s: No such file or directory \n", argv[i+1]);
            exit(1);
            }


            files[i] = fdopen(array[i].fd, "r");

            fseek(files[i], 0L, SEEK_END);
            long int res = ftell(files[i]); // it affects array[i].fd too
            //fseek(fp, 0L, SEEK_SET); it doesnt affect array[i].fd
            fclose(files[i]);

            array[i].fd = open(argv[i+1], O_RDONLY);

            array[i].copy = (char *)calloc(res+1, sizeof(char));

            ssize_t rcnt1 = 0;

            for(;;){
                rcnt1 = read(array[i].fd, array[i].copy, res);
```

```
                if(rcnt1 == 0){
                        break;
                }

                if(rcnt1 == -1){
                        perror("read");
                        exit(1);
                 }

            }
            printf("this is the one: %s\n", array[i].copy);
        }
     }

   if(argc == 4) fd3 = open(argv[argc-1], oflags, mode);
   else fd3 = open("fconc.out", oflags, mode);



   for(i = 0; i<argc-1; i++){
       if(argc==4 && i == argc-2) break;
       write_file(fd3, argv[i+1], array[i].fd, array[i].copy, array[i].boo);
   }


//   printf("counter: %d\n", counter);
     for(i = 0; i<counter; i++){
       if(0 == strcmp(array[i].copy, "")) continue;
       free(array[i].copy);
       //printf("hallo there\n");
   }

   return 0;
}
```

----------end of main----------

----------WRITE_FILE----------

----------write_file.h----------

```
#ifndef WRITE_FILE_H
#define WRITE_FILE_H
#include "write_fun.h"
#include <unistd.h>
#include <sys/types.h>
```

```
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

void write_file(int fd, const char *infile, int fd_inn,  char* buff, bool boo);

#endif


----------------------------------write_file.c----------------------------------------

#include "write_file.h"


void  write_file(int fd, const char *infile, int fd_inn, char* buff, bool boo){

	if(boo == true){
		doWrite(fd, buff, strlen(buff));

	}
	else{

		int fd1;
		fd1 = open(infile, O_RDONLY);


		if(fd1 == -1){
			printf("%s: No such file or directory \n", infile);
			exit(1);
		}

		FILE * fp = fdopen(fd1, "r");

		fseek(fp, 0L, SEEK_END);
		long int res = ftell(fp); // it affects fd1 too
	//	fseek(fp, 0L, SEEK_SET); it doesnt affect fd1

		fd1 = open(infile, O_RDONLY);

		char * con;
		con = (char *)calloc(res+1, sizeof(char));



		ssize_t rcnt1 = 0;

		for(;;){

			rcnt1 = read(fd1, con, res);
			if(rcnt1 == 0){
```

```
                        break;
                }

                if(rcnt1 == -1){
                        perror("read");
                        exit(1);
                }

        }

        //con[res-1] = 0; without \n between the files

        doWrite(fd, con, strlen(con));

        if(close(fd1) == -1){
                printf("error closing file %s", infile);
                exit(1);
        }

        free(con);
        fclose(fp);
    }
}
```

----------WRITE_FUN----------

----------write_fun.h----------

```
#ifndef WRITE_FUN_H
#define WRITE_FUN_H

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

void  doWrite(int fd, char *  buff, int len);

#endif
```

----------write_fun.c----------

```
#include "write_fun.h"


void  doWrite(int fd, char *  buff, int len){
        int idx = 0;
        idx = 0;
        int wcnt;
        do {
```

```
            wcnt = write(fd, buff + idx, len - idx);
            if(wcnt == -1){
               perror("write");
               exit(1);
            }
            else{
            idx += wcnt;
            }
        } while (idx < len);

}
```

---

==========END_OF_CODE==========

2)

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define defaultOutput "fconc.out"
#define defaultOutput2 "fconc_2.out"
#define defaultOutput3 "fconc_3.out"

int main(int argc, char **argv) {
   FILE *inputFile1, *inputFile2;
   FILE *outputFile;

   /* Check if number of arguments is correct, if not show message to standard error */
   if (argc != 3 && argc != 4) {
      fprintf(stderr, "Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
      return 0;
   }

   /* Open the 2 input files */
   inputFile1 = fopen(argv[1],"r");
   if (inputFile1 == NULL) {
         fprintf(stderr, "%s: No such file or directory\n", argv[1]);
         exit(EXIT_FAILURE);
   }

   inputFile2 = fopen(argv[2], "r");
   if (inputFile2 == NULL) {
         fclose(inputFile1);
         fprintf(stderr, "%s: No such file or directory\n", argv[2]);
         exit(EXIT_FAILURE);
   }

   /* Open the output file */
   if (argc == 3) {
      if ( !((strcmp(argv[1], defaultOutput) == 0) | (strcmp(argv[2], defaultOutput) == 0) )){
```

```
            outputFile = fopen(defaultOutput, "w+");
            printf("Output file is at: %s\n", defaultOutput);
        }
        else if(! ((strcmp(argv[1], defaultOutput2) == 0) | (strcmp(argv[2], defaultOutput2) == 0) ) ) {
            outputFile = fopen(defaultOutput2, "w+");
            printf("Output file is at: %s\n", defaultOutput2);
        }
        else {
            outputFile = fopen(defaultOutput3, "w+");
            printf("Output file is at: %s\n", defaultOutput3 );
        }
    }
    else {    /* argc == 4 */
        if (! ((strcmp(argv[1], argv[3]) == 0) | (strcmp(argv[2], argv[3]) == 0) )) {
            /* Print to argv[3] */
            outputFile = fopen(argv[3], "w+");
            printf("Output file is: %s\n", argv[3]);
        }
        else {
            /* Output file is the same with input -> It will be overwritten */
            if ( !((strcmp(argv[1], defaultOutput) == 0) | (strcmp(argv[2], defaultOutput) == 0) )){
                outputFile = fopen(defaultOutput, "w+");
                printf("Output file is at: %s\n", defaultOutput);
            }
            else if(! ((strcmp(argv[1], defaultOutput2) == 0) | (strcmp(argv[2], defaultOutput2) == 0) ) )
{
                outputFile = fopen(defaultOutput2, "w+");
                printf("Output file is at: %s\n", defaultOutput2);
            }
            else {
                outputFile = fopen(defaultOutput3, "w+");
                printf("Output file is at: %s\n", defaultOutput3 );
            }
        }
    }

    /* Read from files character by character and immediately
        write to the output file (char by char)*/
    char ch;
    while((ch = fgetc(inputFile1)) != EOF) {
        fputc(ch,outputFile);
    }

    while((ch = fgetc(inputFile2)) != EOF) {
        fputc(ch,outputFile);
    }

    fclose(inputFile1);
    fclose(inputFile2);
    fclose(outputFile);

    return 0;
```

}

**Ερωτήσεις**

Τρέξαμε την εντολή strace ./fconc A B C στα 2 προγράμματα και αντίστοιχα λάβαμε τις εξής απαντήσεις. Τα δεδομένα που μας ενδιαφέρουν τα έχουμε βάλει σε μεγαλύτερη γραμματοσειρά. Βλέπουμε πως στην δεύτερη λύση τα system calls όντως είναι αρκετά μειωμένα!

- Στον πρώτο πηγαίο κώδικα αντιστοιχεί :

```
thodpap@thodpap:~/Documents/Σχολή/University/ComputerFlow/OperationalSystems/Labab/2nd Exercise(Orfeas)$ strace ./fconc A B C
execve("./fconc", ["./fconc", "A", "B", "C"], 0x7ffd186d0268 /* 70 vars */) = 0
brk(NULL)                               = 0x55e52ef8e000
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=209156, ...}) = 0
mmap(NULL, 209156, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f9cf2249000
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\20\35\2\0\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030928, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f9cf2247000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f9cf1c63000
mprotect(0x7f9cf1e4a000, 2097152, PROT_NONE) = 0
mmap(0x7f9cf204a000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) = 0x7f9cf204a000
mmap(0x7f9cf2050000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f9cf2050000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f9cf2248500) = 0
mprotect(0x7f9cf204a000, 16384, PROT_READ) = 0
mprotect(0x55e52e209000, 4096, PROT_READ) = 0
mprotect(0x7f9cf227d000, 4096, PROT_READ) = 0
munmap(0x7f9cf2249000, 209156)          = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
brk(NULL)                               = 0x55e52ef8e000
brk(0x55e52efaf000)                     = 0x55e52efaf000
write(1, "Nubmer of arguments: 4 \n", 24Nubmer of arguments: 4
) = 24
write(1, "we go a true one\n", 17we go a true one
)      = 17
openat(AT_FDCWD, "A", O_RDONLY)         = 3
fcntl(3, F_GETFL)                       = 0x8000 (flags O_RDONLY|O_LARGEFILE)
fstat(3, {st_mode=S_IFREG|0644, st_size=34, ...}) = 0
fstat(3, {st_mode=S_IFREG|0644, st_size=34, ...}) = 0
lseek(3, 0, SEEK_SET)                   = 0
read(3, "This is a test1.\nThis is a test1"..., 34) = 34
close(3)                                = 0
openat(AT_FDCWD, "A", O_RDONLY)         = 3
write(1, "3\n", 23
)                 = 2
write(1, "hallo there\n", 12hallo there
)          = 12
read(3, "This is a test1.\nThis is a test1"..., 34) = 34
write(1, "hallo there\n", 12hallo there
)          = 12
read(3, "", 34)                         = 0
```

```
openat(AT_FDCWD, "C", O_WRONLY|O_CREAT|O_TRUNC, 0600) = 4
write(1, "4\n", 24
)                       = 2
write(4, "This is a test1.\nThis is a test1"..., 34) = 34
openat(AT_FDCWD, "B", O_RDONLY)         = 5
fcntl(5, F_GETFL)                       = 0x8000 (flags O_RDONLY|O_LARGEFILE)
fstat(5, {st_mode=S_IFREG|0644, st_size=42, ...}) = 0
fstat(5, {st_mode=S_IFREG|0644, st_size=42, ...}) = 0
lseek(5, 0, SEEK_SET)                   = 0
read(5, "This is a test1.\nThis is the sec"..., 42) = 42
openat(AT_FDCWD, "B", O_RDONLY)         = 6
read(6, "This is a test1.\nThis is the sec"..., 42) = 42
read(6, "", 42)                         = 0
write(4, "This is a test1.\nThis is the sec"..., 42) = 42
close(6)                                = 0
close(5)                                = 0
exit_group(0)                           = ?
```

- Στον δεύτερο πηγαίο κώδικα αντιστοιχεί :

```
thodpap@thodpap:~/Documents/Σχολή/University/ComputerFlow/OperationalSystems/Lab/1stLab/2nd Exercise(Theo)$ strace
./fconc A B C
execve("./fconc", ["./fconc", "A", "B", "C"], 0x7fff64f52148 /* 71 vars */) = 0
brk(NULL)                               = 0x55696dd1e000
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=209156, ...}) = 0
mmap(NULL, 209156, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f31fd2c6000
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\20\35\2\0\0\0\0"..., 832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=2030928, ...}) = 0
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f31fd2c4000
mmap(NULL, 4131552, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7f31fcce0000
mprotect(0x7f31fcec7000, 2097152, PROT_NONE) = 0
mmap(0x7f31fd0c7000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1e7000) =
0x7f31fd0c7000
mmap(0x7f31fd0cd000, 15072, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7f31fd0cd000
close(3)                                = 0
arch_prctl(ARCH_SET_FS, 0x7f31fd2c54c0) = 0
mprotect(0x7f31fd0c7000, 16384, PROT_READ) = 0
mprotect(0x55696dcba000, 4096, PROT_READ) = 0
mprotect(0x7f31fd2fa000, 4096, PROT_READ) = 0
munmap(0x7f31fd2c6000, 209156)          = 0
brk(NULL)                               = 0x55696dd1e000
brk(0x55696dd3f000)                     = 0x55696dd3f000
openat(AT_FDCWD, "A", O_RDONLY)         = 3
openat(AT_FDCWD, "B", O_RDONLY)         = 4
openat(AT_FDCWD, "C", O_RDWR|O_CREAT|O_TRUNC, 0666) = 5
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 0), ...}) = 0
write(1, "Output file is: C\n", 18Output file is: C
)     = 18
fstat(3, {st_mode=S_IFREG|0644, st_size=17, ...}) = 0
read(3, "This is a test1.\n", 4096)     = 17
fstat(5, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
read(3, "", 4096)                       = 0
fstat(4, {st_mode=S_IFREG|0644, st_size=25, ...}) = 0
read(4, "This is the second test.\n", 4096) = 25
```

```
read(4, "", 4096)                       = 0
close(3)                                = 0
close(4)                                = 0
write(5, "This is a test1.\nThis is the sec"..., 42) = 42
close(5)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```