1η Άσκηση στα Λειτουργικά Συστήματα

6ο Εξάμηνο, Ακαδημαϊκή περίοδος 2020-2021

Ονοματεπώνυμο	Αριθμός Μητρώου
Στεφανάκης Γεώργιος	el18436
Τζανακάκης Αλέξανδρος	el18431

Σύνδεση με αρχείο αντικειμένων

1) Τα αρχεία επικεφαλίδας εξυπηρετούν δύο κύριους σκοπούς.
   - Τα αρχεία επικεφαλίδων συστήματος δηλώνουν τις διεπαφές σε τμήματα του λειτουργικού συστήματος. Μπορούμε να τα συμπεριλάβουμε στο πρόγραμμά μας για να δώσουμε τους ορισμούς και τις δηλώσεις που χρειαζόμαστε για να πραγματοποιήσουμε κλήσεις συστήματος και να καλέσουμε βιβλιοθήκες.
   - Τα δικά μας αρχεία επικεφαλίδας περιέχουν δηλώσεις για διεπαφές μεταξύ των source files και του προγράμματος. Κάθε φορά που έχουμε μια ομάδα σχετικών δηλώσεων και ορισμών μακροεντολών (macros) και όλα ή τα περισσότερα από αυτά χρειάζονται σε πολλά διαφορετικά source files, είναι καλή ιδέα να δημιουργήσουμε ένα αρχείο επικεφαλίδας για αυτά.

   Όταν συμπεριλαμβάνουμε στον κώδικά μας ένα αρχείο επικεφαλίδας, ουσιαστικά αντιγράφουμε το περιεχόμενο του στην αρχή του κώδικά μας, γεγονός που μας επιτρέπει να αποφεύγουμε χρονοβόρες αντιγραφές και λάθη. Επιπλέον, εάν χρειαστεί να γίνει κάποια αλλαγή στο περιεχόμενο των αρχείων επικεφαλίδας, αυτή η αλλαγή θα γίνει ορατή και στα υπόλοιπα αρχεία που τις χρησιμοποιούν. Με την χρήση αρχείων επικεφαλίδας δεν απαιτείται η συνεχής αναζήτηση και αλλαγή όλων των αντιγράφων στα διάφορα σημεία που αυτή απαιτείται ενώ εξαλείφεται και ο κίνδυνος ότι η αποτυχία εύρεσης ενός αντιγράφου θα έχει ως αποτέλεσμα «ασυνέπειες» εντός ενός προγράμματος.

2) *"Makefile"*

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

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

clean:
    rm main.o zing
```

Παραπάνω βλέπουμε το αρχείο Makefile που γράψαμε προκειμένου να κατασκευάσουμε το εκτελέσιμο αρχείο zing, με χρήση του object file zing.o και του source code αρχείου main.c. Στην πράξη ενώνουμε τα object files main.o και zing.o για να τρέξουμε το επιθυμητό εκτελέσιμο αρχείο αφού πρώτα, ασφαλώς, μεταγλωττίσουμε το source file μας main.c.

3) Στην παρούσα φάση μας ζητήθηκε να παράξουμε ένα νέο object file, το zing2.o προκειμένου εν τέλει να τρέξουμε δύο εκτελέσιμα αρχεία επαναχρησιμοποιώντας το κοινό object file main.o, στο οποίο άλλωστε γίνεται η κλήση της συνάρτησης zing().

"zing2.c"

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

void zing(void) {
    printf("o user %s einai megalo tsakali\n", getlogin());
}
```

Κατασκευάσαμε το αρχείο πηγαίου κώδικα zing2.c που φαίνεται παραπάνω, προκειμένου να παράξουμε το object file zing2.o, το οποίο και θα συνενώσουμε με το ήδη υπάρχον object file main.o για να φτιάξουμε και το δεύτερο εκτελέσιμο zing2. Οι εντολές του νέου Makefile είναι οι εξής:

"Makefile"

```
all: zing zing2

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

zing2: main.o zing2.o
    gcc -Wall main.o zing2.o -o zing2

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

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

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

Με βάση τα νέα ζητούμενα και την ύπαρξη των δύο εκτελέσιμων τροποποιούμε το προηγούμενο Makefile έτσι ώστε κάνοντας κάποια αλλαγή σε οποιαδήποτε από τις συναρτήσεις (zing, main, zing2) να μπορούμε πιο συστηματοποιημένα να μεταγλωττίζουμε και να τρέχουμε τα αρχεία μας.

```
oslaba35@orion:~/exc_1/a/1.3$ make clean
rm main.o zing2.o zing zing2
oslaba35@orion:~/exc_1/a/1.3$ make
gcc -Wall -c main.c -o main.o
gcc -Wall main.o zing.o -o zing
gcc -Wall -c zing2.c -o zing2.o
gcc -Wall main.o zing2.o -o zing2
oslaba35@orion:~/exc_1/a/1.3$ ./zing
Hello, oslaba35
oslaba35@orion:~/exc_1/a/1.3$ ./zing2
o user oslaba35 einai megalo tsakali
oslaba35@orion:~/exc_1/a/1.3$
```

4) Ο κυριότερος λόγος χρήσης του Makefile έγκειται ακριβώς σε αυτήν την διευκόλυνση που μας προσφέρει καθότι αποφεύγουμε την τυποποιημένη διαδικασία με τις αλλαγές στον κώδικα (σ.σ μεταγλώττιση και εκτέλεση). Ουσιαστικά κατασκευάζοντας ένα Makefile δύναμαι να κάνω εκεί την μεταγλώττιση και εκτέλεση, όπως παραπάνω και απλά να αλλάζω τον κώδικα και να πατάω make κάθε φορά που επιθυμώ να εκτελέσω κάποιο ή κάποια από τα τα εκτελέσιμα αρχεία του. Ακόμα, στο Makefile όταν κάνουμε αλλαγές σε μία συνάρτηση και κάνουμε make θα μεταγλωττιστούν τα αρχεία που έχουνε ως προαπαιτούμενο αυτή την συνάρτηση και όχι όλο το αρχείο.
5) Το πρόβλημα που προέκυψε οφείλεται στο γεγονός ότι στην εντολή της εκφώνησης, ύστερα από την παράμετρο -o όπου και ορίζεται το όνομα του output αρχείου, ο χρήστης έχει πληκτρολογήσει foo.c το οποίο είναι το ίδιο με το όνομα του αρχείου πηγαίου κώδικα. Συνεπώς αυτό που συνέβη είναι ότι το τελικό αρχείο ονομάστηκε foo.c και αντικατέστησε το ήδη υπάρχον αρχείο κώδικα. Μια σωστή σύνταξη για την εντολή που επιθυμούμε να εκτελέσουμε είναι `gcc -Wall -o foo.out foo.c`.

Συνένωση δύο αρχείων σε τρίτο

“fconc.c”

```c
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>

extern int errno;

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

    do {
        if ((wcnt = write(fd, buff + i, len - i)) < 0) {
            // perror("fconc: Error while writing on output file.");
            printf("%s\n", strerror(errno));
            exit(errno);
        }
        i += wcnt;
    } while (i < len);
}

void writeFile(int fd, const char *infile) {
    size_t inLength;
    ssize_t rcnt;
    char buffer[1024];
    int input_fd = open(infile, O_RDONLY);
    if (input_fd < 0) {
        // perror("fconc: Error while opening file.");
        printf("%s\n", strerror(errno));
        exit(errno);
    }

    while (1) {
        rcnt = read(input_fd, buffer, sizeof(buffer));
        if (rcnt == 0) break;
        if (rcnt == -1) {
            // perror("fconc: Error while reading file.");
            printf("%s\n", strerror(errno));
            exit(errno);
        }

        inLength = rcnt;
        doWrite(fd, buffer, inLength);
    }
    close(input_fd);
}
```

```c
int main(int argc, char **argv) {
    int output_fd, oflags = O_WRONLY | O_CREAT | O_TRUNC;
    char *infile1Name = argv[1], *infile2Name = argv[2];

    if (argc < 3 || argc > 4) {
        printf("Usage: %s infile1 infile2 [outfile (default:fconc.out)]\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    if (argv[3] != NULL) {
        if (strcmp(argv[1], argv[3]) == 0 || strcmp(argv[2], argv[3]) == 0) {
            printf("Usage: %s infile1 infile2 [outfile (default:fconc.out)]\n", argv[0]);
            exit(EXIT_FAILURE);
        }
        output_fd = open(argv[3], oflags, 0644);
        writeFile(output_fd, infile1Name);
        writeFile(output_fd, infile2Name);
    } else {
        output_fd = open("fconc.out", oflags, 0644);
        writeFile(output_fd, infile1Name);
        writeFile(output_fd, infile2Name);
    }

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

1) Παρακάτω φαίνεται η διαδικασία κατασκευής δύο αρχείων κειμένου A και B τα οποία δίνουμε ως είσοδο στο fconc. Με βάση τα arguments που περνάμε κατά την κλήση του αναμένουμε να γράψει σε ένα τρίτο αρχείο C την συνένωση των A και B. Η strace μας δείχνει τις κλήσεις συστήματος που καλούνται κατά την εκτέλεση του προγράμματος.

```
oslaba35@orion:~/exc_1/b$ echo "Καληνυχτα Κεμαλ," > A
oslaba35@orion:~/exc_1/b$ echo "αυτος ο κοσμος δεν θα αλλαξει ποτε. Καληνυχτα." > B
oslaba35@orion:~/exc_1/b$ strace ./fconc A B C
execve("./fconc", ["./fconc", "A", "B", "C"], [/* 20 vars */]) = 0
brk(0)                                  = 0x1feb000
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) = 0x7f70dd385000
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) = 0x7f70dd37d000
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) = 0x7f70dcdbc000
mprotect(0x7f70dcf5d000, 2097152, PROT_NONE) = 0
mmap(0x7f70dd15d000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a1000)
= 0x7f70dd15d000
mmap(0x7f70dd163000, 14880, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) =
0x7f70dd163000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f70dd37c000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f70dd37b000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f70dd37a000
arch_prctl(ARCH_SET_FS, 0x7f70dd37b700) = 0
mprotect(0x7f70dd15d000, 16384, PROT_READ) = 0
mprotect(0x7f70dd387000, 4096, PROT_READ) = 0
munmap(0x7f70dd37d000, 29766)           = 0
open("C", O_WRONLY|O_CREAT|O_TRUNC, 0644) = 3
open("A", O_RDONLY)                     = 4
read(4, "\316\232\316\261\316\273\316\267\316\275\317\205\317\207\317\204\316\261
\316\232\316\265\316\274\316\261\316\273,\n", 1024) = 31
write(3, "\316\232\316\261\316\273\316\267\316\275\317\205\317\207\317\204\316\261
\316\232\316\265\316\274\316\261\316\273,\n", 31) = 31
read(4, "", 1024)                       = 0
close(4)                                = 0
open("B", O_RDONLY)                     = 4
read(4, "\316\261\317\205\317\204\316\277\317\202 \316\277
\316\272\316\277\317\203\316\274\316\277\317\202 \316\264\316\265\316"..., 1024) = 84
write(3, "\316\261\317\205\317\204\316\277\317\202 \316\277
\316\272\316\277\317\203\316\274\316\277\317\202 \316\264\316\265\316"..., 84) = 84
read(4, "", 1024)                       = 0
close(4)                                = 0
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
oslaba35@orion:~/exc_1/b$ cat C
Καληνυχτα Κεμαλ,
αυτος ο κοσμος δεν θα αλλαξει ποτε. Καληνυχτα.
oslaba35@orion:~/exc_1/b$
```