ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
http://www.cslab.ece.ntua.gr

Πέτρου Νικόλας oslabb13 03117714
Πεγειώτη Νάταλυ 03117707

Λειτουργικά Συστήματα

6ο εξάμηνο, Ακαδημαϊκή περίοδος 2019-2020

Άσκηση 1
Εισαγωγή στο περιβάλλον προγραμματισμού

1 Ασκήσεις

1.1 Σύνδεση με αρχείο αντικειμένων

Βήματα

1. Για την αντιγραφή των αρχείων ***zing.h*** και ***zing.o*** στον κατάλογο εργασίας χρησιμοποιήσαμε τις εντολές

cp /home/oslab/code/zing/zing.h

και

cp /home/oslab/code/zing/zing.o αντίστοιχα.

2. Δημιουργήσαμε το αρχείο ***main.c*** ο κώδικας του οποίου φαίνεται πιο κάτω:

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

Με τη χρήση της εντολής ***gcc -Wall main.c -o main*** δημιουργήσαμε το αρχείο αντικειμένων ***main.o*** για τη συνάρτηση ***main()***.

3. Με χρήση της εντολής ***gcc main.o zing.o -o exec*** επυτήχαμε τη σύνδεση των δύο αρχείων αντικειμένων.

Ερωτήσεις

1.Ποιο σκοπό εξυπηρετεί η επικεφαλίδα;

Μια επικεφαλίδα περιέχει πρότυπα και δηλώσεις. Χρησιμοποιείται για τον ορισμό δικών μας συναρτήσεων ή και για τη χρηση συναρτήσεων και μεταβλητών που περιλαμβάνονται σε προϋπάρχουσες βιβλιοθήκες. Επιπλέον μπορεί να χρησιμοποιηθεί για τη διεπαφή προς άλλα κομμάτια κώδικα, δηλαδή για συναρτήσεις που είναι ορισμένες σε διαφορετικά αρχεία και τις οποίες θέλουμε να καλέσουμε σε κάποιο πρόγραμμα, κάνοντας #include το header file που τις περιέχει.

2. Ζητείται κατάλληλο Makefile για τη δημιουργία του εκτελέσιμου της άσκησης.

Δημιουργήσαμε το αρχείο ***Makefile*** το περιεχόμενο του οποίου φαίνεται παρακάτω:

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

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

Στη συνέχεια με την εντολή ***make*** δημιουργήσαμε το εκτελέσιμο της άσκησης, το οποίο και τρέξαμε. Πιο κάτω φαίνεται η διαδικασία και η έξοδος του προγράμματος:

```
oslabb13@os-node1:~/backup$ make
gcc -Wall -c main.c
gcc -o exec main.o zing.o
oslabb13@os-node1:~/backup$ ./exec
Hello, oslabb13
```

3. Παράξτε το δικό σας zing2.o το οποίο θα περιέχει zing() που θα εμφανίζει διαφορετικό αλλά παρόμοιο μήνυμα με τη zing() του zing.o. Αλλάξτε το Makefile ώστε να παράγονται δύο εκτελέσιμα, ένα με το zing.o και ένα με το zing2.o, επαναχρησιμοποιώντας το κοινό object file main.o.

Δημιουργήσαμε το δικό μας αρχείο ***zing2.c*** ο κώδικας του οποίου παρουσιάζεται πιο κάτω:

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

void zing(void)
{
        printf("You are the best, %s \n",  getlogin());
}
```

Η μεταγλώττιση του ***zing2.c*** μας έδωσε το νέο αρχείο αντικειμένων ***zing2.o***.

Επίσης τροποποιήσαμε το αρχείο ***Makefile*** ώστε να παράγεται και δεύτερο εκτελέσιμο. Ο ολοκληρωμένος κώδικας του ***Makefile*** είναι ο ακόλουθος:

```
all: exec exec2

exec: main.o zing.o
	gcc -o exec main.o zing.o

exec2: main.o zing2.o
	gcc -o exec2 main.o zing2.o

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

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

Η δημιουργία του εκτελέσιμου και η έξοδος του προγράμματος φαίνονται πιο κάτω:

```
oslabb13@os-node1:~/lab1.1$ make
gcc -Wall -c main.c
gcc -o exec main.o zing.o
gcc -Wall -c zing2.c
gcc -o exec2 main.o zing2.o
oslabb13@os-node1:~/lab1.1$ ./exec
Hello, oslabb13
oslabb13@os-node1:~/lab1.1$ ./exec2
You are the best, oslabb13
```

4. Έστω ότι έχετε γράψει το πρόγραμμά σας σε ένα αρχείο που περιέχει 500 συναρτήσεις. Αυτή τη στιγμή κάνετε αλλαγές μόνο σε μία συνάρτηση. Ο κύκλος εργασίας είναι: αλλαγές στον κώδικα, μεταγλώττιση, εκτέλεση, αλλαγές στον κώδικα, κ.ο.κ. Ο χρόνος μεταγλώττισης είναι μεγάλος, γεγονός που σας καθυστερεί, Πώς μπορεί να αντιμετωπισθεί το πρόβλημα αυτό;
Το συγκεκριμένο πρόβλημα μπορεί να αντιμετωπιστεί εύκολα, δημιουργώντας ένα νέο αρχείο στο οποίο θα μεταφέρουμε την υπό επεξεργασία συνάρτηση. Έτσι μετά τις αλλαγές στον κώδικα η μεταγλώττιση του νέου αρχείου το οποίο θα περιέχει μόνο μία συνάρτηση θα γίνεται σε πολύ μικρό χρονικό διάστημα. Στη συνέχεια θα συνδέσουμε (***link***) τα δύο αρχεία αντικειμένων (του κύριου αρχείου και του καινούριου) και θα δημιουργήσουμε έτσι το νέο εκτελέσιμο, χωρίς χρονικές καθυστερήσεις.

5. Ο συνεργάτης σας και εσείς δουλεύατε στο πρόγραμμα foo.c όλη την προηγούμενη εβδομάδα. Καθώς κάνατε ένα διάλειμμα και ο συνεργάτης σας δούλευε στον κώδικα, ακούτε μια απελπισμένη κραυγή. Ρωτάτε τι συνέβει και ο συνεργάτης σας λέει ότι το αρχείο foo.c χάθηκε! Κοιτάται το history του φλοιού και η τελευταία εντολή ήταν η:
gcc -Wall -o foo.c foo.c
Τι συνέβη;

Για τη σύνταξη της συγκεκριμένης εντολής χρειάζονται δύο ορίσματα. Το πρώτο πρεπεί να είναι το όνομα του εκτελέσιμου που θέλουμε να δημιουργήσουμε με τη μεταγλώττιση του αρχείου που θέτουμε ως δεύτερο όρισμα. Στη συγκεκριμένη περίπτωση ο συνεργάτης μας έχει κάνει το λάθος να χρησιμοποιήσει το ίδιο αρχείο και για τα δύο ορίσματα, δηλαδή έχει κάνει overwrite στο αρχείο του source code (***foo.c***) και το έχει μετατρέψει σε εκτελέσιμο. Το πρόγραμμα μας μπορεί να εκτελεστεί, αλλά ο αρχικός κώδικας έχει χαθεί με αποτέλεσμα να μην μπορούν να γίνουν οποιεσδήποτε τροποποιήσεις.

1.2 Σύνδεση δύο αρχείων σε τρίτο

Πιο κάτω παρουσιάζεται ο πηγαίος κώδικας της άσκησης:

```c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.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) /*error*/
                {
                        perror("write");
                        exit(1);
                }
                idx += wcnt;
        } while (idx < len);
}


void write_file(int fd, const char *infile)
{
        char buff[1024];
        ssize_t rcnt;
        int fd_in;
        fd_in = open(infile, O_RDONLY);
        if (fd_in == -1)
        {
                perror(infile);
                exit(1);
        }
        do{
                rcnt = read(fd_in, buff, sizeof(buff)-1);
                if (rcnt == -1){
                        perror("read");
                        exit(1);
                }
                doWrite(fd, buff, rcnt);
        }while (rcnt !=0);
        close(fd_in);
}
```

```c
int main(int argc, char **argv)
{
        char *out = "fconc.out";
        int fd_out, oflags, mode;

        if (argc<3 || argc>4)
        {
                printf("Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]\n");
                return -1;
        }

        if (argc == 4)
        {
                out = argv[3];
        }

        oflags = O_CREAT | O_WRONLY | O_TRUNC;
        mode = S_IRUSR |S_IWUSR;
        fd_out = open(out, oflags, mode);
        if (fd_out == -1)
        {
                perror("open");
                exit(1);
        }

        write_file(fd_out, argv[1]);
        write_file(fd_out, argv[2]);

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

Ακολουθεί ένα παράδειγμα εκτέλεσης του προγράμματος:

```
oslabb13@os-node1:~$ ./fconc A
Usage: ./fconc infile1 infile2 [outfile (default:fconc.out)]
oslabb13@os-node1:~$ ./fconc A B
A: No such file or directory
oslabb13@os-node1:~$ echo 'Goodbye,' > A
oslabb13@os-node1:~$ echo 'and thanks for all the fish!' > B
oslabb13@os-node1:~$ ./fconc A B
oslabb13@os-node1:~$ cat fconc.out
Goodbye,
and thanks for all the fish!
oslabb13@os-node1:~$ ./fconc A B C
oslabb13@os-node1:~$ cat C
Goodbye,
and thanks for all the fish!
```

Ερωτήσεις:

1. Εκτελέστε ένα παράδειγμα του fconc χρησιμοποιώντας την εντολή strace. Αντιγράψτε το κομμάτι της εξόδου της strace που προκύπτει από τον κώδικα που γράψατε.

Από την εκτέλεση της εντολής ***strace ./fconc A B*** προκύπτει η ακόλουθει έξοδος:

```
open("fconc.out", O_WRONLY|O_CREAT|O_TRUNC, 0600) = 3
open("A", O_RDONLY)                               = 4
read(4, "Goodbye,\n", 1023)                       = 9
write(3, "Goodbye,\n", 9)                         = 9
read(4, "", 1023)                                 = 0
write(3, "", 0)                                   = 0
close(4)                                          = 0
open("B", O_RDONLY)                               = 4
read(4, "and thanks for all the fish!\n", 1023) = 29
write(3, "and thanks for all the fish!\n", 29) = 29
read(4, "", 1023)                                 = 0
write(3, "", 0)                                   = 0
close(4)                                          = 0
close(3)                                          = 0
exit_group(0)                                     = ?
+++ exited with 0 +++
```