ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΣΚΗΣΗ 1

Εισαγωγή στο περιβάλλον προγραμματισμού

Ομάδα oslab39
Δημήτριος Βασιλείου el19830
Ιωάννης Ρόκομος el19061
ΣΗΜΜΥ 6ο Εξάμηνο

ΑΣΚΗΣΗ 1.1

Ο κώδικας της main.c, η οποία καλεί την προκαθορισμένη συνάρτηση zing(), φαίνεται παρακάτω:

```
#include "zing.h"

int main (int argc, char **argv){
      zing();
      return 0;
}
```

όπου η επικεφαλίδα zing.h περιέχει τον κώδικα

```
#ifndef ZING_H__
#define ZING_H__

void zing(void);

#endif
```

Η διαδικασία μεταγλώττισης του προγράμματος γίνεται με την εκτέλεση της εντολής

gcc -Wall -c main.c

η οποία δημιουργεί το object file main.o.
Στη συνέχεια γίνεται η σύνδεση(link) των αρχείων main.o και zing.o και δημιουργείται το εκτελέσιμο αρχείο zing χρησιμοποιώντας την εντολή

gcc zing.o main.o -o zing.

Η έξοδος εκτέλεσης του προγράμματος φαίνεται παρακάτω:

```
oslaba39@os-node1:~/zing$ ./zing
Hello, oslaba39
```

Απαντήσεις στις ερωτήσεις

1). Το header file zing.h χρησιμοποιείται για την δήλωση της συνάρτησης zing() της οποίας η υλοποίηση δεν βρίσκεται εντός του αρχείου main.c, στο οποίο καλείται.

2). Για την διαδικασία μεταγλώττισης(compiling) και σύνδεσης(linking) χρησιμοποιείται αρχείο Makefile το περιεχόμενο του οποίου φαίνεται παρακάτω:

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

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

clean:
	rm main.o zing
```

Το πρώτο target είναι το εκτελέσιμο αρχείο zing, το οποίο εξαρτάται από τα αρχεία zing.o και main.o. Όταν ένα εκ των δύο αλλάζει, τότε εκτελείται η εντολή σύνδεσης

gcc zing.o main.o -o zing

η οποία δημιουργεί το εκτελέσιμο αρχείο zing.
Δεύτερο target είναι το αρχείο main.o το οποίο εξαρτάται από το αρχείο main.c. Όταν αλλάζει το main.c πρέπει να αλλάξει και το object file main.o οπότε εκτελείται η εντολή μεταγλώττισης

gcc -Wall -c main.c.

Τέλος υπάρχει το target clean που διαγράφει τα αρχεία zing και main.o.

Η διαδικασία μεταγλώττισης και σύνδεσης γίνονται με την χρήση της εντολής make.

```
oslaba39@os-node1:~/zing$ make
gcc -Wall -c main.c
gcc main.o zing.o -o zing
```

3). Το αρχείο zing2.c υλοποιεί μια νέα συνάρτηση zing(), η οποία καλεί την συνάρτηση getlogin(), η οποία επιστρέφει το όνομα του συνδεδεμένου χρήστη. Ο κώδικάς της φαίνεται παρακάτω:

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

void zing()
{
        printf("Goodbye, %s!!!\n" ,getlogin());
}
```

Για την διαδικασία μεταγλώττισης και την δημιουργία δύο εκτελέσιμων αρχείων zing και zing2, χρησιμοποιήθηκε το ακόλουθο Makefile:

```
all: zing2 zing

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

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

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

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

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

Για το κάθε εκτελέσιμο αρχείο, η main γνωρίζει ποια υλοποίηση της συνάρτησης zing() να καλέσει, με βάση το object file με το οποίο έχει γίνει link. Οι έξοδοι των δύο εκτελέσιμων φαίνονται παρακάτω:

```
oslaba39@orion:~/zing$ ./zing
Hello, oslaba39
oslaba39@orion:~/zing$ ./zing2
Goodbye, oslaba39!!!
```

4). Έχοντας 500 συναρτήσεις ορισμένες σε ένα μόνο αρχείο, με την αλλαγή έστω και μίας συνάρτησης καθίσταται αναγκαία η μεταγλώττιση όλων των συναρτήσεων. Η λύση στο πρόβλημα αυτό είναι η δημιουργία διαφορετικών αρχείων για την υλοποίηση κάθε συνάρτησης και ενός αρχείου επικεφαλίδας (header file) στο οποίο θα δηλώνονται όλες οι συναρτήσεις που χρησιμοποιούνται (το οποίο θα γίνεται include στο αρχείο της συνάρτησης main()). Με αυτόν τον τρόπο, κατά την αλλαγή μιας συνάρτησης στο αντίστοιχο αρχείο υλοποίησής της, απαιτείται η μεταγλώττιση μόνο αυτής της συνάρτησης.

5). Η εντολή

gcc -Wall -o foo foo.c

δημιουργεί το εκτελέσιμο αρχείο foo με όρισμα το source file foo.c.
Εκτελώντας την εντολή

gcc -Wall -o foo.c foo.c

το αρχείο foo.c που περιέχει κώδικα C, αντικαθίσταται από το εκτελέσιμο αρχείο με αποτέλεσμα ο πηγαίος κώδικας του foo.c να χαθεί. Ωστόσο, επειδή δημιουργείται εκτελέσιμο αρχείο, η εντολή

./foo.c

τρέχει το εκτελέσιμο και παράγει το επιθυμητό αποτέλεσμα.

ΑΣΚΗΣΗ 1.2

Ο πηγαίος κώδικας του αρχείου fconc.c φαίνεται παρακάτω:

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

/*writes the content of buff from stream f to the file with file descriptor fd*/
void doWrite(int fd, char *buff)
{
    size_t len, idx;
    ssize_t wcnt;
    idx = 0;
    len = strlen(buff);
    // writes buff into the output file

    do
    {
        wcnt = write(fd, buff + idx, len - idx);
            //error
            if (wcnt == -1)
            {
                  perror("write");
            }
        idx += wcnt;
    } while (idx < len);
}

/*writes the content of the file named infile to the file with file descriptor
fd*/
void write_file(int fd, const char *infile)
{
      int fdread = open(infile, O_RDONLY);
      char buff[1024];
      ssize_t rcnt;
      for (;;)
      {
```

```
            rcnt = read(fdread, buff, sizeof(buff) - 1);
            /* end-of-file */
            if (rcnt == 0)
                  break;

            /* error */
            if (rcnt == -1)
            {
                  perror("read");
                  break;
            }
            buff[rcnt] = '\0';
            doWrite(fd, buff);

      }

      close(fdread);
}

int main (int argc, char **argv)
{
      /*check if not enough or too many files given e.g ./fconc A or ./fconc A B
C D */
      if (argc != 3 && argc != 4)
      {
            printf("Usage: ./fconc2 infile1 infile2 [outfile
(default:fconc2.out)]\n");
            return 0;
      }

      int i;
      for (i = 1; i < 3; i++)
      {
        int fdread;
        fdread = open(argv[i], O_RDONLY);
        if (fdread == -1)
        {
            perror(argv[i]);
            exit(1);
        }

        close(fdread);
      }

      int fd, oflags, mode;
      oflags = O_CREAT | O_WRONLY | O_TRUNC;
      mode = S_IRUSR | S_IWUSR;

      /*if output file given, open output file for writing*/
      if (argv[3] != '\0')
      {
            fd = open(argv[3], oflags, mode);
      }
      /*if no ouptut file given,
        create default output file fconc.out*/
      else
            fd = open("fconc.out", oflags, mode);

      /*perform wtite(...)  */
      write_file(fd, argv[1]);
      write_file(fd, argv[2]);
      close(fd);
      return 0;
}
```

Απαντήσεις στις ερωτήσεις

1). Παραθέτουμε ένα παράδειγμα εκτέλεσης του fconc χρησιμοποιώντας την εντολή strace:

```
oslaba39@os-node1:~/fconc$ strace ./fconc A B C
execve("./fconc", ["./fconc", "A", "B", "C"], [/* 27 vars */]) = 0
brk(0)                                  = 0x70c000
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) = 0x7f669fb75000
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=32730, ...}) = 0
mmap(NULL, 32730, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f669fb6d000
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) = 0x7f669f5ac000
mprotect(0x7f669f74d000, 2097152, PROT_NONE) = 0
mmap(0x7f669f94d000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a1000) = 0x7f669f94d000
mmap(0x7f669f953000, 14880, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f669f953000
close(3)                                = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f669fb6c000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f669fb6b000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f669fb6a000
arch_prctl(ARCH_SET_FS, 0x7f669fb6b700) = 0
mprotect(0x7f669f94d000, 16384, PROT_READ) = 0
mprotect(0x7f669fb77000, 4096, PROT_READ) = 0
munmap(0x7f669fb6d000, 32730)           = 0
open("A", O_RDONLY)                     = 3
close(3)                                = 0
open("B", O_RDONLY)                     = 3
close(3)                                = 0
open("C", O_WRONLY|O_CREAT|O_TRUNC, 0600) = 3
open("A", O_RDONLY)                     = 4
read(4, "Hello\n", 1023)                = 6
write(3, "Hello\n", 6)                  = 6
read(4, "", 1023)                       = 0
close(4)                                = 0
open("B", O_RDONLY)                     = 4
read(4, "World!!!\n", 1023)             = 9
write(3, "World!!!\n", 9)               = 9
read(4, "", 1023)                       = 0
close(4)                                = 0
close(3)                                = 0
exit_group(0)                           = ?
+++ exited with 0 +++
```

Με την εκτέλεση της εντολής strace φαίνονται όλες οι κλήσεις συστήματος (system calls) που εκτελούνται.