ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΗΛΙΑΣ ΝΤΟΝΤΟΡΟΣ

ΘΟΔΩΡΗΣ- ΙΩΑΝΝΗΣ ΚΙΤΣΗΣ

ΑΝΑΦΟΡΑ 1ης ΣΕΙΡΑΣ ΑΣΚΗΣΕΩΝ

ΑΣΚΗΣΗ 1.1

1.1.1 Header

Η επικεφαλίδα (header) χρησιμοποιείται για να δηλωθούν οι συναρτήσεις και τα ορίσματα τους ώστε να βοηθηθεί ο compiler ώστε να βρει τις συναρτήσεις που έχουν χρησιμοποιηθεί μέσα στο πρόγραμμα. Επίσης η δήλωση των επικεφαλίδων βοηθάει στην δημιουργία συναρτήσεων που μπορούν να χρησιμοποιηθούν σε περισσότερα από ένα προγράμματα.

1.1.2 Makefile

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

1.1.3 Zing2

zing2.c

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

void zing() {
        char *user;
        user = getlogin();
        printf("Bye %s.\n",user);
}
```

Output zing2

```
Bey oslaba111
```

1.1.4 Πολλές Συναρτήσεις

Όταν ένα πρόγραμμα απαιτεί πολλές συναρτήσεις μια καλή τεχνική είναι να χωρίζουμε τις συναρτήσεις σε διαφορετικά αρχεία .c ώστε όταν απαιτείται να τροποποιήσουμε μια από αυτές να μην χρειάζεται να ξανακάνουμε compile σε όλες τις συναρτήσεις δηλαδή σε όλα τα διαφορετικά αρχεία παρά μόνο σε αυτές που έχουμε κάνει τροποποιήσεις. Με αυτόν τον τρόπο θα γλιτώνουμε πολύ χρόνο στο compilation αν για παράδειγμα

1.1.5 gcc -Wall -o foo.c foo.c

Η παραπάνω εντολή δημιουργεί ένα εκτελέσιμο, με όνομα το οποίο το παίρνει από το πρώτο όρισμα μετά το -o, από τον κώδικα που υπάρχει στα υπόλοιπα ορίσματα στην εντολή. Αφού καλέσουμε την παραπάνω εντολή θα δημιουργηθεί ένα εκτελέσιμο με όνομα foo.c καταστρέφοντας το αρχείο foo.c που υπήρχε ήδη και περιείχε το κώδικα

ΑΣΚΗΣΗ 1.2

1.2.1 fconc.c

```
1. #include <sys/types.h>
2. #include <sys/stat.h>
3. #include <fcntl.h>
4. #include <stdio.h>
5. #include <stdlib.h>
6. #include <unistd.h>
7. #include <string.h>
8.
9. // WRITE
10.  void doWrite(int fd, const char *buff, size_t len)
11.  {
12.          size_t idx;
13.          ssize_t wcnt;
14.          idx = 0;
15.          do
16.          {
17.                  wcnt = write(fd, buff + idx, len - idx);
18.                  if (wcnt == -1)
19.                  {
20.                          perror("write");
21.                          exit(1);
22.                  }
23.                  idx += wcnt;
```

```
24.         } while (idx < len);
25. }
26.
27. void write_file(int fd, const char *infile)
28. {
29.         ssize_t rcnt = 0;
30.         int idx_read;
31.         int fd2;
32.         fd2 = open(infile, O_RDONLY);
33.         char buff[1024];
34.         if (fd2 == -1)
35.         {
36.                 perror("open");
37.                 exit(1);
38.         }
39.         //  Read until end of file
40.
41.         for(;;)
42.         {
43.
44.                 rcnt = read(fd2, buff, sizeof(buff) - 1);
45.                 if (rcnt == 0)
46.                 {
47.                         break;
48.                 }
49.                 if (rcnt == -1)
50.                 {
51.                         perror("read");
52.         }
53.
54.                 int len = strlen(buff);
55.                 doWrite(fd, buff, rcnt);
56.         }
57.
58.         close(fd2);
59. }
60.
61. int main(int argc, char **argv)
62.
63. {
64.         int oflags, mode;
65.         int fd;
66.         oflags = O_CREAT | O_WRONLY | O_TRUNC;
67.         mode = S_IRUSR | S_IWUSR;
68.         if (argc == 2) // just one input file
69.         {
70.                 printf("Usage:./fconc infile1 infile2
    [outfiledefault:fconc.out)]");
71.         }
72.         else if (argc == 3) // two input files
73.         {
74.                 fd = open("fconc.out", oflags, mode);
75.         }
76.
```

```
        else if (argc == 4)
                // two input files one output
        {
                fd = open(argv[3], oflags, mode);
        }

        if ((argc == 3) || (argc == 4))
        {

                if (fd == -1)
                {
                        perror("open");
                        exit(1);
                }
                write_file(fd, argv[1]);
                write_file(fd, argv[2]);
                close(fd);
        }
        return 0;
}
```

Output fconc.out

```
1. Goodbey,
2. and thanks for the meat
```

Output strace fconc

```
1. execve("./fconc", ["./fconc", "A", "B", "C"], [/* 18 vars */])
   = 0
2. brk(0)                                       = 0x2074000
3. access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such
   file or directory)
4. mmap(NULL, 8192, PROT_READ|PROT_WRITE,
   MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fc6b16c6000
5. access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such
   file or directory)
6. open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
7. fstat(3, {st_mode=S_IFREG|0644, st_size=32730, ...}) = 0
8. mmap(NULL, 32730, PROT_READ, MAP_PRIVATE, 3, 0) =
   0x7fc6b16be000
9. close(3)                                     = 0
10.  access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such
   file or directory)
11.  open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) =
   3
12.  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
13.  fstat(3, {st_mode=S_IFREG|0755, st_size=1738176, ...}) = 0
```

```
14.  mmap(NULL, 3844640, PROT_READ|PROT_EXEC,
     MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fc6b10fd000
15.  mprotect(0x7fc6b129e000, 2097152, PROT_NONE) = 0
16.  mmap(0x7fc6b149e000, 24576, PROT_READ|PROT_WRITE,
     MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a1000) =
     0x7fc6b149e000
17.  mmap(0x7fc6b14a4000, 14880, PROT_READ|PROT_WRITE,
     MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fc6b14a4000
18.  close(3)                                   = 0
19.  mmap(NULL, 4096, PROT_READ|PROT_WRITE,
     MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fc6b16bd000
20.  mmap(NULL, 4096, PROT_READ|PROT_WRITE,
     MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fc6b16bc000
21.  mmap(NULL, 4096, PROT_READ|PROT_WRITE,
     MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fc6b16bb000
22.  arch_prctl(ARCH_SET_FS, 0x7fc6b16bc700) = 0
23.  mprotect(0x7fc6b149e000, 16384, PROT_READ) = 0
24.  mprotect(0x7fc6b16c8000, 4096, PROT_READ) = 0
25.  munmap(0x7fc6b16be000, 32730)               = 0
26.  open("C", O_WRONLY|O_CREAT|O_TRUNC, 0600) = 3
27.  open("A", O_RDONLY)                        = 4
28.  read(4, "goodbey mitsos, \n", 1023)        = 17
29.  write(3, "goodbey mitsos, \n", 17)         = 17
30.  read(4, "", 1023)                          = 0
31.  close(4)                                   = 0
32.  open("B", O_RDONLY)                        = 4
33.  read(4, "and thanks for the meat\n", 1023) = 24
34.  write(3, "and thanks for the meat\n", 24) = 24
35.  read(4, "", 1023)                          = 0
36.  close(4)                                   = 0
37.  close(3)                                   = 0
38.  exit_group(0)                              = ?
39.  +++ exited with 0 +++
```