Είσοδος/έξοδος χαμηλού επιπέδου

- Για απλές διαδικασίες εισόδου/εξόδου χρησιμοποιείται η προκαθορισμένη βιβλιοθήκη *stdio*, η οποία παρέχει σημαντικές ευκολίες, όπως ενδιάμεση μνήμη και μετατροπή δεδομένων
- Η *stdio* είναι ένα φιλικό εργαλείο για τον προγραμματιστή και έχει κατασκευασθεί μέσω ενός συνόλου από κλήσεις συστήματος που υποστηρίζουν είσοδο/έξοδο χαμηλού επιπέδου
- Οι κλήσεις συστήματος για είσοδο/έξοδο χαμηλού επιπέδου είναι αναγκαίες μόνο όταν οι ευκολίες που παρέχει η *stdio* για προσπέλαση αρχείων ή συσκευών δεν είναι επιθυμητές ή όταν προγραμματίζεται επικοινωνία μεταξύ διεργασιών μέσω σωλήνων και υποδοχών

- Ο προσδιορισμός αρχείων, συσκευών, άκρων σωλήνων και υποδοχών γίνεται στις κλήσεις συστήματος για είσοδο/έξοδο χαμηλού επιπέδου από περιγραφείς αρχείων που είναι ακέραιοι αριθμοί, σε αντίθεση με την *stdio* όπου χρησιμοποιούνται, για αρχεία και συσκευές, δείκτες σε αρχεία

- Προκαθορισμένοι περιγραφείς αρχείων

  0 : Προκαθορισμένη είσοδος

  1 : Προκαθορισμένη έξοδος

  2 : Προκαθορισμένη έξοδος για διαγνωστικά μηνύματα

- Οι προκαθορισμένοι περιγραφείς αρχείων 0, 1 και 2 αντιστοιχούν στους προκαθορισμένους δείκτες σε αρχεία *stdin*, *stdout* και *stderr* της *stdio*

- Οι περιγραφείς αρχείων κληρονομούνται από μία διεργασία-πατέρα στις διεργασίες-παιδιά που δημιουργεί

- Κλήση συστήματος **open**
  - int **open**(char **fileName*, int *mode* [, int *permissions*])
  - Ανοίγει ή δημιουργεί το αρχείο με απόλυτο ή σχετικό όνομα *fileName* για διάβασμα και/ή γράψιμο
  - Το *mode* είναι ένας ακέραιος που παριστάνει τον τρόπο προσπέλασης του αρχείου και δίνεται σαν διάζευξη συμβολικών ονομάτων, όπως:

    `O_RDONLY:` Ανοιγμα για διάβασμα μόνο
    `O_WRONLY:` Ανοιγμα για γράψιμο μόνο
    `O_RDWR:` Ανοιγμα για διάβασμα και γράψιμο
    `O_APPEND:` Γράψιμο στο τέλος του αρχείου
    `O_CREAT:` Δημιουργία αρχείου, εφ' όσον δεν υπάρχει
    `O_TRUNC:` Διαγραφή περιεχομένων αρχείου, εφ' όσον υπάρχει

    Απαίτηση: `#include <sys/file.h>`
  - Το προαιρετικό όρισμα *permissions* είναι ένας ακέραιος του οποίου η τιμή πρέπει να αντιστοιχεί στα επιθυμητά δικαιώματα προστασίας ενός αρχείου κατά τη δημιουργία του (δικαιώματα που αποκλείονται από την τρέχουσα τιμή του *umask* δεν δίνονται στο αρχείο)
  - Η **open** επιστρέφει έναν περιγραφέα αρχείου σε επιτυχία ή -1 σε αποτυχία

- Κλήση συστήματος **read**
  - int **read**(int *fd*, char **buf*, int *count*)
  - Διαβάζει το πολύ *count* bytes από την οντότητα (αρχείο, συσκευή, άκρο σωλήνα ή υποδοχή) που αντιστοιχεί στον περιγραφέα *fd* και τα τοποθετεί στο *buf*
  - Επιστρέφει τον αριθμό των bytes που διαβάστηκαν, 0 αν έγινε απόπειρα διαβάσματος αφού έχει διαβαστεί και το τελευταίο byte ή -1 σε αποτυχία
- Κλήση συστήματος **write**
  - int **write**(int *fd*, char **buf*, int *count*)
  - Γράφει το πολύ *count* bytes από το *buf* στην οντότητα που αντιστοιχεί στον περιγραφέα *fd*
  - Επιστρέφει τον αριθμό των bytes που γράφτηκαν ή -1 σε αποτυχία
- Κλήση συστήματος **close**
  - int **close**(int *fd*)
  - Ελευθερώνει τον περιγραφέα *fd*
  - Επιστρέφει 0 σε επιτυχία ή -1 σε αποτυχία

- Χρήση των κλήσεων **open**, **read**, **write** και **close**

```
/* File: io_demo.c */
#include <stdio.h>      /* For printf */
#include <sys/file.h>   /* For O_RDONLY, O_WRONLY,
                           O_CREAT, O_APPEND */
main()
{  int fd, bytes;
   char buf[50];
   if ((fd = open("t", O_WRONLY | O_CREAT, 0600)) == -1) {
      perror("open"); exit(1); }
   bytes = write(fd, "First write. ", 13);   /* Data out */
   printf("%d bytes were written\n", bytes);
   close(fd);
   if ((fd = open("t", O_WRONLY | O_APPEND)) == -1) {
      perror("open"); exit(1); }
   bytes = write(fd, "Second write.\n", 14); /* Data out */
   printf("%d bytes were written\n", bytes);
   close(fd);
   if ((fd = open("t", O_RDONLY)) == -1) {
      perror("open"); exit(1); }
   bytes = read(fd, buf, sizeof(buf));       /* Data in */
   printf("%d bytes were read\n", bytes);
   close(fd);
   buf[bytes] = '\0';
   printf("%s", buf); }
```

```
% io_demo
13 bytes were written
14 bytes were written
27 bytes were read
First write. Second write.
% ls -l t
-rw-------  1 takis          27 Dec  9 14:45 t
% rm t
%
```

- Κλήσεις συστήματος **dup** και **dup2**
  - int **dup**(int *oldFd*)
  - int **dup2**(int *oldFd*, int *newFd*)
  - Η **dup** βρίσκει το μικρότερο ελεύθερο περιγραφέα αρχείου και τον αντιστοιχεί στην ίδια οντότητα με τον περιγραφέα *oldFd*
  - Η **dup2** ελευθερώνει τον περιγραφέα *newFd*, εφ' όσον είναι δεσμευμένος, και τον αντιστοιχεί στην ίδια οντότητα με τον περιγραφέα *oldFd*
  - Σε επιτυχία επιστρέφουν το νέο περιγραφέα, δηλαδή η **dup2** το *newFd*, ή -1 σε αποτυχία

- Χρήση των κλήσεων **dup** και **dup2**

```
/* File: duplicate_fd.c */
#include <stdio.h>      /* For printf */
#include <sys/file.h>   /* For O_RDWR, O_CREAT, O_TRUNC */
main()
{  int fd1, fd2, fd3;
   if ((fd1 = open("r", O_RDWR | O_CREAT | O_TRUNC,
                   0644)) == -1) {
      perror("open");
      exit(1); }
   printf("fd1 = %d\n", fd1);
   write(fd1, "What ", 5);     /* Write data to fd1 */
   fd2 = dup(fd1);             /* Make a copy of fd1 */
   printf("fd2 = %d\n", fd2);
   write(fd2, "time", 4);      /* Write data to fd2 */
   close(0);                   /* Close standard input */
   fd3 = dup(fd1);             /* Another copy of fd1 */
   printf("fd3 = %d\n", fd3);
   write(fd3, " is it", 6);    /* Write data to fd3 */
   dup2(fd1, 2);               /* Duplicate fd1 to 2 */
   write(2, "?\n", 2);         /* Write data to 2 */
   close(fd1);
   close(fd2);
   close(fd3); }
```

```
% duplicate_fd
fd1 = 3
fd2 = 4
fd3 = 0
% ls -l r
-rw-r--r--  1 takis          17 Dec  9 15:40 r
% cat r
What time is it?
% rm r
%
```

- *Να γραφεί ένα πρόγραμμα C που να αντιγράφει ένα αρχείο στο τέλος ενός άλλου.*

```
/* File: append_file.c */
#include <sys/file.h>   /* For O_RDONLY. O_WRONLY,
                           O_CREAT, O_APPEND */
main(int argc, char *argv[])
{  int n, from, to;
   char buf[1024];
   if (argc != 3) {         /* Check for proper usage */
      write(2, "Usage: ", 7);
      write(2, *argv, strlen(*argv));
      write(2, " from-file to-file\n", 19);
      exit(1); }
   if ((from = open(argv[1], O_RDONLY)) < 0) {
                                 /* Open from-file */
      perror("open");
      exit(1); }
   if ((to = open(argv[2], O_WRONLY | O_CREAT | O_APPEND,
                0660)) < 0) {       /* Open to-file */
      perror("open");
      exit(1); }
   while ((n = read(from, buf, sizeof(buf))) > 0)
      write(to, buf, n);                 /* Copy data */
   close(from);                    /* Close from-file */
   close(to); }                     /* Close to-file */
```

```
% append_file
Usage: append_file from-file to-file
% ls -l file*
-rw-r--r--  1 takis          39 Dec  9 16:07 file1
-rw-r--r--  1 takis          26 Dec  9 16:07 file2
% cat file1
file1 line 1
file1 line 2
file1 line 3
% cat file2
file2 line 1
file2 line 2
% append_file file2 file1
% cat file1
file1 line 1
file1 line 2
file1 line 3
file2 line 1
file2 line 2
% append_file file2 file3
% cat file3
file2 line 1
file2 line 2
% rm file1 file2 file3
%
```