שימוש בקבצים ביוניקס

יוניקס מספקת פעולות בסיסיות כלפי כל קבצים:
פתיחה, קריאה, כתיבה, תנועה (lseek), וסגירה.

כל פעולת ק/כ מתייחסת ק.מ.
ולכן יש מכונות קלט פלט חציצה
unbuffered i/o
(שם מעל מערכת שכן מה בהחלט מתיקה
חוצצים עדיף פעולות הק/כ).

מבחינת הגרעין, כל פניה לקובץ פתוח נעשית
באמצעות מתאר הקובץ (file descriptor)
מספר טבעי המאפשר לתהליך את הוא יוצר או
פותח קובץ.

כלומר:

מתאר קובץ	0#	מתייחס	לקלט	הסטנדרטי
"	1#	"	לפלט	"
"	2#	"	לשגיאה	"

? - <unistd.h> מגדרים:
STDIN_FILENO, STDOUT_FILENO,
STDERR_FILENO
ואלו להשתמש בהם, ולא ב- 0,1,2.

1. הפעלות על קבצים

1.א. כדי לכתוב קוד נשתמש בפונ' open:

```
#include    <sys/types.h>
   "        <sys/stat.h>
   "        <fcntl.h>

int open(const char *pathname,
         int oflag,
         ...
         /* , mode_t mode */ );
```

הארגומנט השני לפונ' מתואר ע"י ...

לפי הקונבנציה ב- ANSI C

(ANSI = American National Standards Institute)

אין שאפרים וטיפוסים של יתר הפרמטרים אינו מוגדר, כלומר תלוי להשתנות.

בקישור של open יתכן רק ארגומנט שני, המעביר את יצרים קובץ. הוא מגדיר בהרשאות תיאור הארגומנט מופע בהערת התיעוד

הפונ' מחזירה: מספר של מתאר קובץ, או 1- בכישלון.

הארגומנט pathname מגדיר את שם הקובץ.
הארגומנט oflag מגדיר כיצד לפתוח הקובץ:

O_RDONLY (מוגדרים ב:
O_WRONLY <fcntl.h>)
O_RDWR יכולים אחד מהם

אפשר להם ניתן להוסיף ע"י or:

O_APPEND = הוסף בסוף
O_CREAT = צור אם לא קיים
(ואז יש לספק גם ארגומנט
שלישי, כמו שנראה)

O_EXCL = היכשל אם ביקשו לצור
והקובץ כבר קיים

O_TRUNC = אם קיים ונפתח (גם) לכתיבה
אז רוקן!

O_SYNC = כל כתיבה תמתין לסיום
הכתיבה הפיזית לדיסק,
ורק אז ה- write יתן תשובה.
(ולא תסתפק בכך שהנתונים בחו

כל הנ"ל נקראים דגלי הסטטוס של הקובץ

בקריאת open מבטיחה שיוקצה מתאר הקובץ
הבלתי הראשון, כפי שראינו בהקשר של צינור
(עת, לדוגמה, קישרנו את הפלט הסטנדרטי
לצינור, אחרי שסגרנו אותו).

1.ב. עת רוצים <u>לצור קובץ</u>, במקום להשתמש ב:

```
open( pathname,
      O_WRONLY | O_CREAT | O_TRUNC,
      mode)
```

ניתן להשתמש ב:

```
int creat ( const char *pathname,
            mode_t mode);
```

לוי ב כסוף

1.ג. <u>סגירת קובץ</u> תעשה בעזרת

```
int close(int filedes);
```

חוצר: 0 בהצלחה
-1 בכישלון.

3.1 כדי לשנות את המצביע לקובץ נשתמש בפעולת ה- lseek.

המיקום בקובץ = הפסטה בביתים מתחילת הקובץ.

פעולת קו'כ משנות את ערך המצביע.

```
off_t lseek(int filedes,
            off_t offset,
            int whence);
```

long בשביל

מוחצר: המיקום החדש, או 1- בכישלון.

משמעותו של whence:

SEEK_SET => קבע את המצביע להיות offset בתים מתחילת הקובץ

SEEK_CUR => צע יחסית למיקום הנוכחי: (הרפטה עשויה להיות שלילית).

SEEK_END => צע יחסית לסוף הקובץ (הרפטה עשויה להיות שלילית)

הצבה בשלב 0 יחסית לנוכחי תחזיר את מיקומנו בקובץ.

lseek רק משנה את ערכו של המצביע לקובץ
בזיכרון - שום פעולת קלט אינה מבוצעת.

בהמשך, כקובץ שנפתח לכתיבה, שווה להוות
מעבר לסוף הקובץ. במצב זה פעולת הכתיבה
הבאה תגדל את הקובץ ותייצר בו "חורי"
שיכיל אפסים.

לדוגמה:

```
int fd;
if ((fd = creat("hole.txt", 0600) < 0)
{        שגיאה        }

if (write(fd, "abc", 3) != 3)
{        שגיאה        }

if (lseek(fd, 10, SEEK_SET) == -1)
{        שגיאה        }

if (write(fd, "ABC", 3) != 3)
{        שגיאה        }

close(fd);
```

אם אחרי הרצת התכנית נקבל

```
ls -l hole.txt
```

כדי לראות את גודלו נראה שהקובץ מכיל 18 בתים.

הפקודה: `od -c hole.txt`

תציג את תכולת הקובץ c- = כתווים

הפלט:

```
0000000  a  b  c  0  0  ...  0  A
```

12 כאלה

```
0020  ...  0  B  C
```

ההסטה בבתים באוקטלי (בסיס 8)

```c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>        // EXIT_SUCCESS
#include <stdio.h>         // perror
#include <unistd.h>        // read/write/lseek

int main() {
   int fd ;

   if ((fd = creat("hole.txt", 0600)) < 0) {
       perror("create file hole.txt") ;
       return( EXIT_FAILURE ) ;
   }

   if (write(fd, "abc", 3) != 3) {
       perror("write file hole.txt") ;
       return( EXIT_FAILURE ) ;
   }

   if (lseek(fd, 15, SEEK_SET) == -1) {
       perror("lseek file hole.txt") ;
       return( EXIT_FAILURE ) ;
   }

   if (write(fd, "ABC", 3) != 3) {
       perror("write file hole.txt") ;
       return( EXIT_FAILURE ) ;
   }
   close(fd) ;

   return( EXIT_SUCCESS ) ;
}


//-----------------------------------------------------------------
<222|0>yoramb@inferno-04:~/os/os2> ls -l hole.txt
-rw-------  1 yoramb teach 18 Feb 18  2007 hole.txt

<225|0>yoramb@inferno-04:~/os/os2> od -c hole.txt
0000000   a   b   c  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
  \0   A
0000020   B   C
0000022
```

1.ה. כדי לקרוא נתונים מקובץ נשתמש בפונ' הספריה:

```
ssize_t read(int filedes,
             void *buff,
             size_t nbytes);
```

<u>הפונ' מחזירה:</u>

מספר הבתים שנקראו בפועל או אפס אם אין עוד EOF או -1 בכישלון.

מתי מספר הבתים שנקראו בפועל יהיה קטן מהמבוקש:

א. אם עד סוף הקובץ נותרו פחות בתים. (רק ה-read הבא יחזיר 0)

ב. עת קוראים מטרמינל נקראת בדיו לכל היותר שורה יחידה

ג. עת קוראים מהרשת, בשל בעיות חבילה והעברת נתונים יתכן שיקראו פחות בתים מהמבוקש.

```c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdlib.h>        // EXIT_SUCCESS
#include <stdio.h>         // perror
#include <unistd.h>        // read/write/lseek

#define          BUFF_SIZE        10

//-------------------------------------------------------------
int main() {
   int fd ;
   int nbytes ;
   char buff[BUFF_SIZE] ;

   if ((fd = open("hole.txt", O_RDONLY)) < 0) {
       perror("open for read file hole.txt") ;
       return( EXIT_FAILURE ) ;
   }

   if (read(fd, buff, 3) != 3) {
       perror("read file hole.txt") ;
       return( EXIT_FAILURE ) ;
   }
   buff[3] = '\0' ;
   puts(buff) ;

   if (lseek(fd, 15, SEEK_SET) == -1) {
       perror("lseek file hole.txt") ;
       return( EXIT_FAILURE ) ;
   }

   while ((nbytes = read(fd, buff, 2)) > 0) {
      buff[nbytes] = '\0' ;
      puts(buff) ;
   }

   close(fd) ;

   return( EXIT_SUCCESS ) ;
}


//-------------------------------------------------------------
```

1.1. פונ' ה write:

```
ssize_t write(int filedes,
              const void *buff,
              size_t nbytes);
```

<u>מחזיר</u> מספר הבתים שנכתבו, או
1- בכישלון.

סיבות לכישלון:
- דיסק מלא
- הקובץ עבר את הגודל המרבי המותר לו.

תפנית המשתמשת בפעולות
write/read תרוץ במהירות הגבוהה
ביותר (בשניות) עד גודל החוצץ
לתוכו/ממנו קוראים/כותבים יהיה בגודל
גוש כמו שמוגדר במה (או גדול יותר).
הגדר של גודל החוצץ מעבר לכך
לא תגדל יותר את מהירות הריצה.