מערכות הפעלה תרגול 1

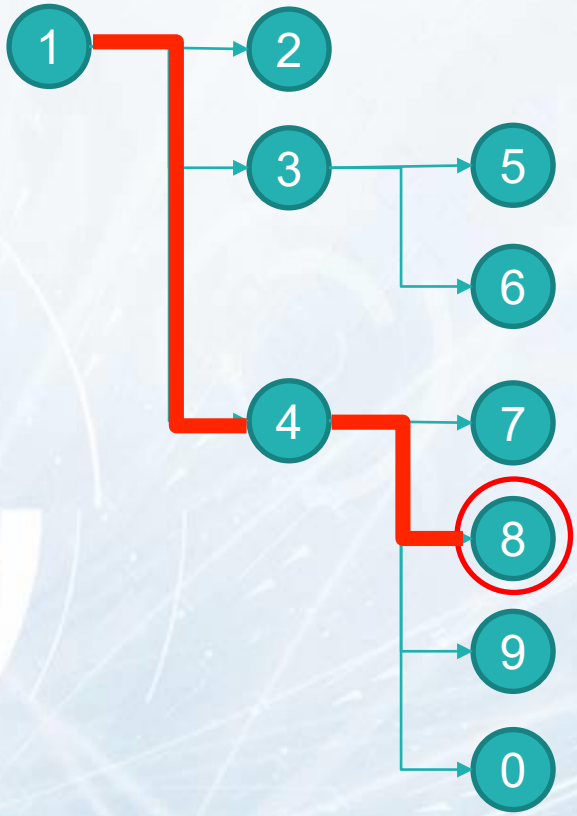
Files

מתרגל-יורם סגל
yoramse@colman.ac.il

Contents

INODE

לינוקס מכיר את התיקיות והקבצים לפי מספרים המכונים INODE



שם קובץ
בצורת INODE

1/4/8

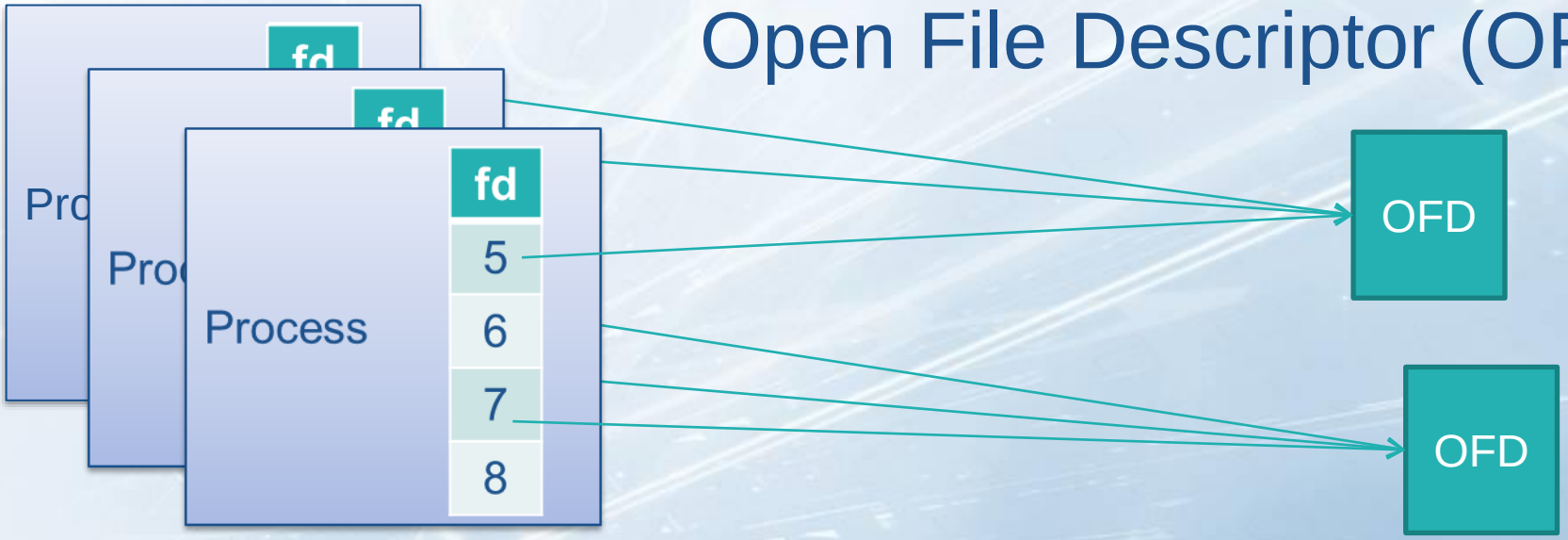
שם קובץ
בצורת טקסט

root/rmi/MyFile

שיטת הINODE אינה נוחה למשתמש בן אנוש ולכן אנחנו ממירים את המספרים לשמות שם קובץ או תיקייה הוא כל המסלול המלא

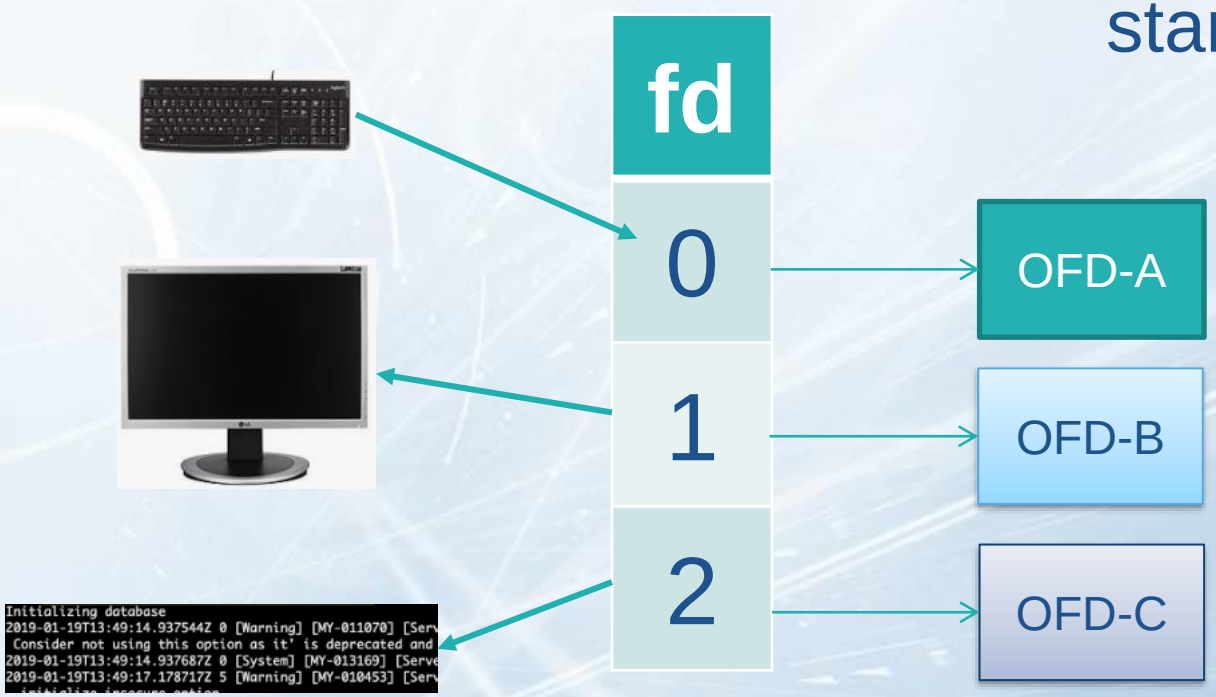
File descriptors

- כדי לגשת לקובץ (בלינוקס) משתמשים ב File Descriptors (fd).
- לכל תהליך (process) יש רשימה של fd-ים.
- ה fd הינו מספר המשויך למבנה (structure) הנקרא: Open File Descriptor (OFD)



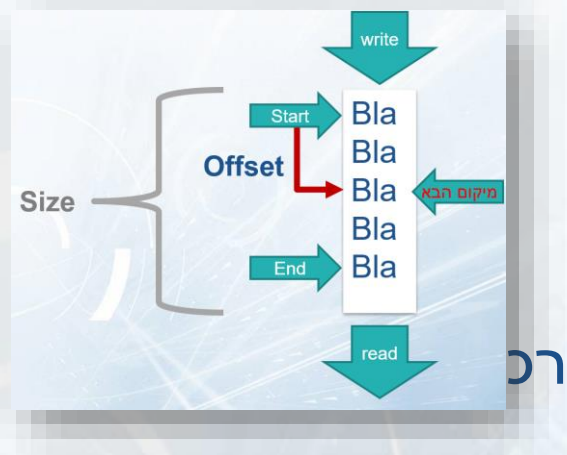
File descriptors

- המוסכמה היא ש file descriptors 0,1,2 כבר מוקצים:
  - standard input – 0
  - standard output – 1
  - standard error – 2



Open file descriptions (OFD)

- Open file description הינו מבנה המכיל מידע על קובץ.
- למשל:
  - Offset: איפה הכניסה הבאה לקובץ תהיה
  - האם הקובץ ניתן לקריאה/כתיבה (לפי ההיררכ
  - ועוד דגלים



- יכולים להיות מספר fd-ים המצביעים על ofd אחד

Open syscall

open(אופן הפתיחה,שם קובץ)

- int open(const char *pathname, int flags);
- **Flags:**
  - O_RDONLY - פתח לקריאה בלבד
  - O_WRONLY - פתח לכתיבה בלבד
  - O_RDWR - פתח לקריאה ולכתיבה
  - O_CREAT - פתח קובץ וצור אותו אם לא קיים
  - O_EXCL - גרום לפקודה פתח להכשל אם הקובץ קיים והדגל o-creat דולק
  - O_TRUNC - פתח את הקובץ וקצץ אותו לגודל 0
  - O_APPEND - כתיבה תתבצע לסוף הקובץ

Open syscall

open(שם קובץ, אופן הפתיחה, **הרשאות**)

int open(const char *pathname, int flags, mode_t mode);

Modes:

S_IRUSR - user (file owner) has read permission
S_IWUSR - user (file owner) has write permission
S_IXUSR - user (file owner) has execute permission
S_IRWXU - user (file owner) has read, write and execute permission
S_IRGRP - group has read permission
S_IWGRP - group has write permission
S_IXGRP - group has execute permission
S_IRWXG - group has read, write and execute permission
S_IROTH - others have read permission
S_IWOTH - others have write permission
S_IXOTH - others have execute permission
S_IRWXO - others have read, write and execute permission

Open syscall

- Examples:
  - open(“1.txt”,O_RDONLY);

    The file “1.txt” will be open only for reading.
  - open(“2.txt”,O_RDWR | O_TRUNC);

    The file “2.txt” will be open only for reading and writing, it’s size will be 0.
  - open(“3.txt”,O_WRONLY | O_CREAT | O_EXCL, S_IRWXU | S_IXGRP);

    The file “3.txt” will be open only for writing, will be created if not exist. owner has read, write and execute permission and group has execute permission

Access syscall

- int access(char* pathname, int mode);
- כדי לבדוק האם לתהליך הקורא יש גישה לקובץ.

- Mode:
  - R_OK – האם לתהליך הנוכחי יש גישת קריאה.
  - W_OK – האם לתהליך הנוכחי יש גישת כתיבה.
  - X_OK – האם לתהליך הנוכחי יש גישת הרצה.
  - F_OK – האם הקובץ קיים.

מידע על הקובץ

```
struct stat {

    dev_t     st_dev - ID of device containing file
    ino_t     st_ino - inode number
    mode_t    st_mode – protection & file type
    nlink_t   st_nlink - number of hard links
    uid_t     st_uid - user ID of owner
    gid_t     st_gid - group ID of owner
    off_t     st_size -total size, in bytes
    time_t    st_atime -time of last access
    time_t    st_mtime -time of last modification
    time_t    st_ctime - time of last status change
    and more…
};
```

ראה בשקופית הבאה פונקצית stat הממלאת מבנה זה

מידע על קובץ

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

int stat(**const char*** path, **struct stat*** buf);

stats the file pointed to by *path* and fills in *buf*.

int fstat(**int** fd, **struct stat*** buf);

is identical to **stat**(), except that the file to be stat-ed is specified by the file descriptor *fd*.

```
#include <stdio.h>
int fileno(FILE *stream); \\ returns fd
```

Streams provide a higher-level interface, layered on top of the primitive **file descriptor** facilities.

int stat(**const char*** path, **struct stat*** buf) ➔ **m=buf.st_mode**

מידע על קובץ – סוג הקובץ

The follow … using
it's ***st_m***

S_ISR
regul

S_ISC
chara

S_ISB
block

S_ISF
fifo (a … a

```
struct stat {

    dev_t     st_dev - ID of device containing file
    ino_t     st_ino - inode number
    mode_t    st_mode – protection & file type
    nlink_t   st_nlink - number of hard links
    …                    owner
    …                    of owner
    off_t     st_size -total size, in bytes
    time_t    st_atime -time of last access
    time_t    st_mtime -time of last modification
    time_t    st_ctime - time of last status change
    and more…
};
```

m= buf.st_mode

t2_1.c

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

int main (int argc, char* argv[])
{

        char * filename = argv[1];
        char msg[64];

        struct stat stat_p;

        if ( stat (filename, &stat_p) == -1 ) /* declare the 'stat' structure */

        {
        sprintf(msg," Error occurred attempting to stat %s\n",  filename); /*printf*/
                perror(msg);
                return 1;
        }

        printf("Stats for %s \n", filename);
        printf("File size is %d bytes\n", stat_p.st_size);

        if ( S_ISREG(stat_p.st_mode))
                printf("This is a regular file");

        if ( S_ISDIR(stat_p.st_mode))
                printf("This is a directory");

        return 0;
}
```

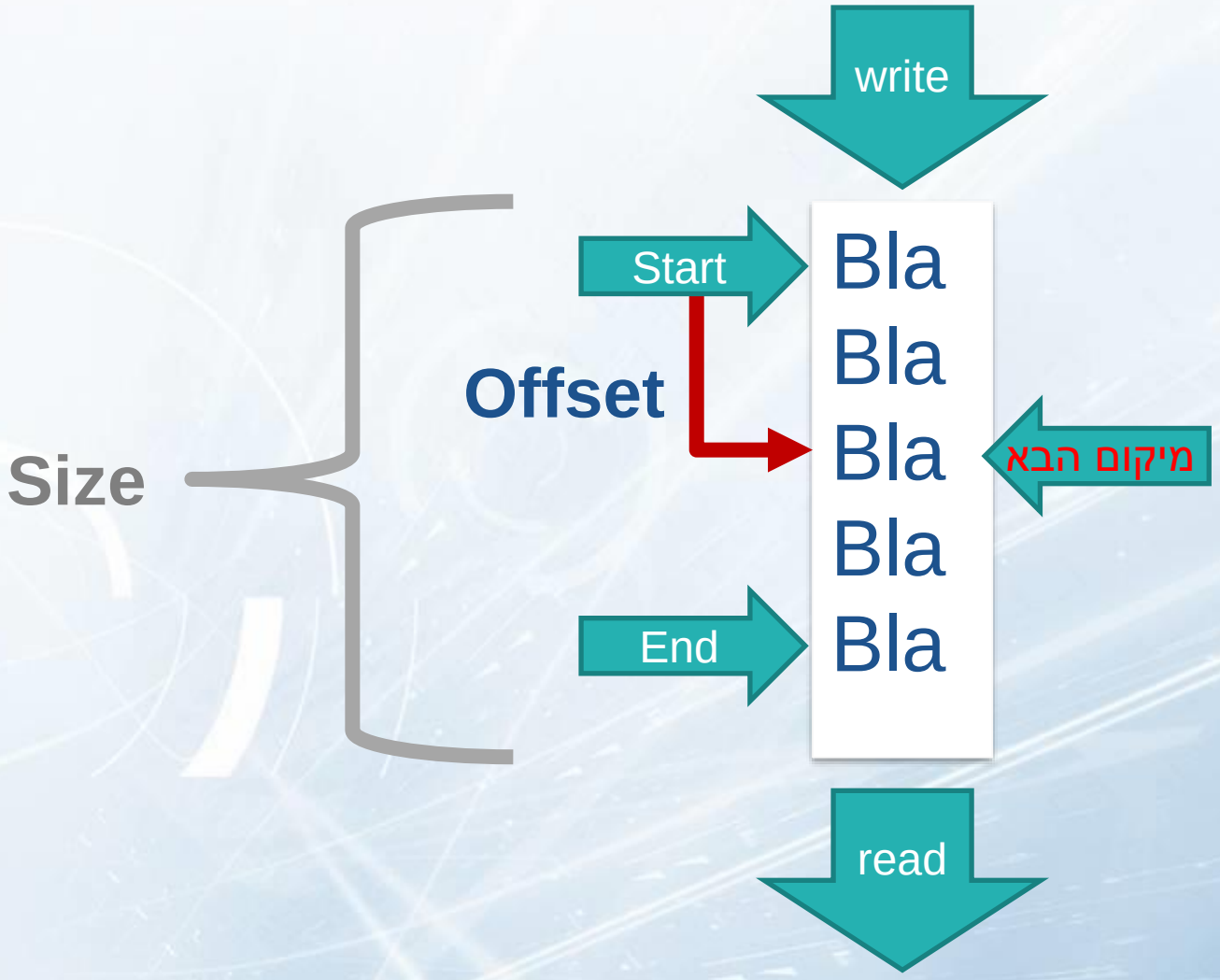
gcc t2_1.c → ./a.out t2_1_ex.txt

מידע על קובץ - הרשאות

- כדי לדעת מה ההרשאות של המשתמש, הקבוצה והאחרים אפשר לבצע & בין ה st_mode ובין אחד מהמוגדרים הבאים:

  - **S_IRWXU – ההרשאות של יוצר הקובץ**
  - **S_IRWXG – ההרשאות של הקבוצה אליה משוייך הקובץ**
  - **S_IRWXO – ההרשאות של היתר**

ניהול מידע בתוך קובץ



Read syscall

- int read(int fd, void* buff, size_t nbytes);
  - קריאת הנתונים מתבצעת עד שמספר הבתים המבוקשים נקראו או עד שסוף הקובץ הגיע.
  - הערך החוזר:
    - 1- : שגיאה
    - 0 : סוף הקובץ
    - מספר חיובי: כמות הבתים שנקראו בפועל

Write syscall

- int write(int fd, void* buff, size_t nbytes);

- כתיבת נתונים לקובץ המוצבע ע"י fd.
- מיקום הכתיבה הבאה לקובץ תמשיך מיד לאחר המיקום של הכתיבה הקודמת.
- אם הקובץ נפתח עם הדגל O_APPEND המידע ישמר בסוף הקובץ.

Close syscall

- int close(int fd);

- עקרונית כל קובץ שנשאר פתוח כאשר התהליך מסתיים יסגר ע"י מערכת ההפעלה.
- סגירת fd זה בעצם עדכון ל open file description שיש לו פחות מצביע אחד.
- כאשר המונה של מספר המצביעים של ה ofd מגיע לאפס אז ה ofd גם כן משתחרר.

T2_2.c – Copy file content

```
#include <stdio.h> #include <sys/fcntl.h>
        #include <errno.h> #include<stdlib.h>
#define SIZE10

main(int argc , char **argv)
{
        int fdin;           /* input file descriptor */
        int fdout;          /* out   file descriptor */
        char *buf[SIZE+1]; /* input (output) buffer */
        int readAmt;        /* how many chars were actually red */
        int writeAmt;       /* how many chars were actually written
        */

        fdin = open("example1.txt",O_RDONLY);
        if (fdin < 0) /* means file open did not take place */
        {
                perror("after open ");   /* text explaining why */
                exit(-1);
        }
```

```
/* create the file with write and read permissions */
fdout = open("example2.txt", O_CREAT | O_WRONLY,
0666);
if (fdout < 0)     /* means file open did not take place */
{
        perror("after create ");   /* text explaining why */
        exit(-1);
}

do {
        readAmt = read(fdin,buf,SIZE);
        /* why writting SIZE can be wrong...*/
        writeAmt = write(fdout,buf,readAmt);

        if (writeAmt < readAmt)
                printf("error reading\n");
}while ( (readAmt == SIZE) && (writeAmt == SIZE));
close(fdin);            /* free allocated structures */
close(fdout);           /* free allocated structures */
}
```

משפחת dup

```
#include <unistd.h>
int dup(int oldfd);
int dup2(int oldfd, int newfd);
```

ה syscall הללו יוצרות עוד עותק של *oldfd*

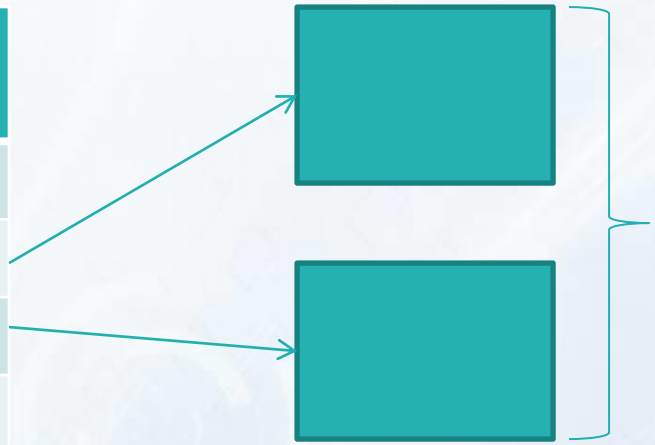
Dup- מחפשת ברשימת ה fd את המקום הפנוי הראשון ומעתיקה אליו את ה oldfd.

Dup2- סוגרת את oldfd ומעתיקה במקומו את newfd

ערך חוזר- כשלון: 1-, הצלחה: ה fd שהוחלף(newfd)

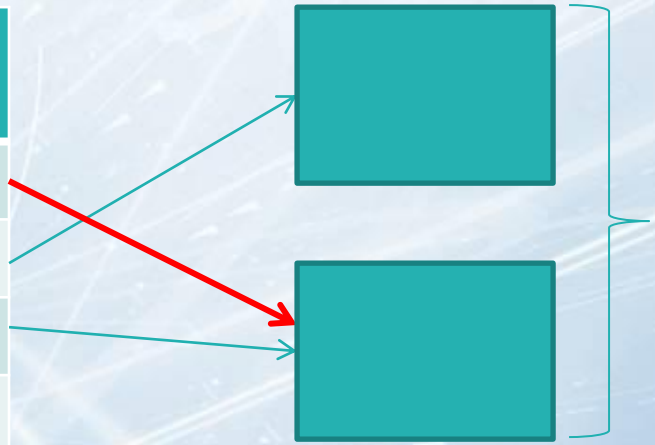
Dup flow - שכפול

File descriptors
3 - free
4 - occupied
5 - occupied
6- free



Calling dup(**5**);

File descriptors
3
4
5
6



int dup2(int oldfd, int newfd)

Dup2 flow – הזזת מיקום

File descriptors
0 - occupied
1 - occupied
2 - occupied
3- occupied

File

Open file descriptors

Calling dup2(**3,1**);

File descriptors
0
1
2
3

Open file descriptors

בדר"כ משמש להסטה מהתקן אחד להתקן אחר

T2_3.c –standard output to file

```
#include <stdlib.h> #include <stdio.h> #include <fcntl.h>

int main(int argc, char **argv) {
        int pid, status;
        int newfd; /* new file descriptor */
        if (argc != 2) {
                fprintf(stderr, "usage: %s output_file\n", argv[0]);
                exit(1);
        }
        if ((newfd = open(argv[1], O_CREAT|O_TRUNC|O_WRONLY, 0644)) < 0) {
                perror(argv[1]); /* open failed */
                exit(1);
        }
        printf("This goes to the standard output.\n");
        printf("Now the standard output will go to \"%s\".\n", argv[1]);

        /* this new file will        become the standard output */
        /* standard output is file descriptor 1, so we use dup2 */
        /* to copy the new file descriptor onto file descriptor 1 */
        /* dup2 will close the current standard output */

        dup2(newfd, 1); //Form here any printout will go to the file instead to the screen till the termination of this program

        printf("This goes to the standard output too.\n");
        exit(0);
}
```

מידע על תיקייה

```
#include <sys/types.h>
#include <dirent.h>
```

כדי לקבל מצביע על תיקייה יש להשתמש ב:

```
DIR* opendir(const char* pathname);
```

כדי לסגור תיקייה יש להשתמש ב:

```
int closedir(DIR* dp);
```

מידע על תיקייה

```
struct dirent {
ino_t          d_ino;        /* inode number */
off_t          d_off;        /* offset of that directory entry */
unsigned short d_reclen;     /* length of this record */
unsigned char  d_type;       /* type of file */
char           d_name[256];  /* filename */
};
```

כדי לקבל מידע על נתון בתיקייה יש להשתמש ב:

struct dirent* readdir(**DIR*** dp);

כל קריאה לפונקציה תחזיר מידע על הנתון הבא בתיקייה עד שלבסוף יחזור NULL.