Laborator 5

Inode-urile contin metadate despre un fisier sau director si exemple de astfel de date permisiuni de acces, proprietar, dimensiune, timestamp urile de creare/modificare/acces si un pointer catre blocurile de date effective ale fisierului sau directorului.

Avantaje: permit eficienta in gestionarea fisierelor si directoarelor deoarece o structura de inode poate continue toate informatiile relevante despre un fisier in timp ce numele de fisier este stocat intr un director

Un hardlink este o legatura directa la un inode in sistemul de fiseire. Arunci cand se creaza un hard link catre un fisier existent va avea acelasi continut si inode cu originalul. Orice fisier indifferent de cate hard link uri are, v a avea un singur continut si un singur inode. Cand se creaza un hard link sistemul de fisiere creaza o noua intrare de director care face referire la acelasi inode ca si fiseirul original. Prin urmare st_nlink v a fi incrementat pt inode ul asociat acestui fisier indicand ca exista un alt hardlink catre acelasi fisier.

Hard link urile sunt fosoite adesea pentru acces rapid la un fisier din mai multe locatii in sistemul de fisiere si pentru a economisii spatiu pe disk atunci cand se doreste crearea mai multor referinte la acelasi continut. Nu poti face hardlink spre directoare.

Atunci cand un program sau utilizator acceseaza link ul symbolic sistemul de op. urmeaza calea specificata in fisierul de legatura simbolica pt a gasi fisierul sau directorul real.

Utilitate link symbolic: pt a creea referinte felxibile catre fisiere si directoare in sistemul de fisiere ele pot traversa limitele sistemului de fisiere si pot face referire la fisiere sau directoare din alte locatii chiar si de pe alte dispositive sau partitii. Totusi o legatura simbolica poate devenii inutila daca fisierul sau directorul catre cale face referire este sters sau mutat deoarece nu exista o verificare automata a validitatii link ului symbolic.

OBS: link urile simbolice pot creea bucle sau referinte circulare daca nu sunt gestionate corespunzator.

Parcurgere laborator

```
biancapinghireac@vbox:~/SO/lab5/src$ make lib
gcc -Wall -g -O   -c -o error.o error.c
gcc -Wall -g -O   -c -o pathalloc.o pathalloc.c
ar rcs liblab5.a error.o pathalloc.o
```

```
biancapinghireac@vbox:~/SO/lab5/src$ make
gcc -o ftw4 ftw4.c liblab5.a
gcc -o unlink unlink.c liblab5.a
gcc -o zap zap.c liblab5.a
gcc -o mycd mycd.c liblab5.a
gcc -o cdpwd cdpwd.c liblab5.a
gcc -o devrdev devrdev.c liblab5.a
biancapinghireac@vbox:~/SO/lab5/src$ tree
.
├── cdpwd
├── cdpwd.c
├── devrdev
├── devrdev.c
├── error.c
├── error.o
├── ftw4
├── ftw4.c
├── liblab5.a
├── Makefile
├── mycd
├── mycd.c
├── ourhdr.h
├── pathalloc.c
├── pathalloc.o
├── unlink
├── unlink.c
├── zap
└── zap.c
```

Crearea librariei

```
biancapinghireac@vbox:~/SO/lab5/src$ df /home
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/sda3       19919872 7898580  11446988  41% /home
```

Comanda **df /home** afiseaza informatii despre spatiul de disc disponibil si utilizat pentru sistemul de fisiere pe care este montat directorul **/home**.

Coloanele reprezinta:

Filesystem: Numele sistemului de fisiere sau al dispozitivului de stocare (in acest caz, **/dev/sda3**)

1K-blocks: Numarul total de blocuri de 1K disponibile pe sistemul de fisiere

Used: Numarul de blocuri de 1K care sunt deja utilizate

Available: Numarul de blocuri de 1K care sunt disponibile pentru utilizare

Use%: Procentul din spatiul total care este utilizat

Mounted on: Punctul de montare al sistemului de fișiere (în acest caz, **/home**)

```
biancapinghireac@vbox:~/SO/lab5/src$ cp ftw4.c tempfile
biancapinghireac@vbox:~/SO/lab5/src$ df /home
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/sda3       19919872 7898580  11446988  41% /home
```

Copiaza fisierul ftw4.c in tempfile, dar fisierul este prea mic sa se observe modificari

```
biancapinghireac@vbox:~/SO/lab5/src$ echo mini >tempfile
biancapinghireac@vbox:~/SO/lab5/src$ df /home
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/sda3       19919872 7898580  11446988  41% /home
```

Fisierul este suprascris, dar la fel este prea mic pentru a observa modificari

```
biancapinghireac@vbox:~/SO/lab5/src$ ./unlink
file unlinked
biancapinghireac@vbox:~/SO/lab5/src$ df /home
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/sda3       19919872 7898580  11446988  41% /home
biancapinghireac@vbox:~/SO/lab5/src$
```

(**unlink** sterge intrarea fisierului din sistemul de fisiere, dar daca fișierul este deschis, datele raman accesibile pana cand toate referintele sunt inchise – comanda din unlink.c)

```
biancapinghireac@vbox:~/SO/lab5/src$ cp unlink tempfile
biancapinghireac@vbox:~/SO/lab5/src$ ls -l tempfile
-rwxr-xr-x. 1 biancapinghireac biancapinghireac 22248 Apr 16 10:40 tempfile
biancapinghireac@vbox:~/SO/lab5/src$ ls -lu tempfile
-rwxr-xr-x. 1 biancapinghireac biancapinghireac 22248 Apr 16 10:40 tempfile
```

Copiaza unlink in tempfile

Ls -l tempfile : afiseaza date despre fisier(permisiuni, data ulimei schimbari)

Ls -lu temfile : afiseaza datele despre fisier si data ultimei accesari

```
biancapinghireac@vbox:~/SO/lab5/src$ date
Wed Apr 16 10:41:25 AM EEST 2025
```

Date :ata si ora curenta

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

int main(int argc, char *argv[])
{
        int             i;
        struct stat     statbuf;
        struct utimbuf  timebuf;

        for (i = 1; i < argc; i++) {
                if (stat(argv[i], &statbuf) < 0) {      /* fetch current times */
                        err_ret("%s: stat error", argv[i]);
                        continue;
                }
                if (open(argv[i], O_RDWR | O_TRUNC) < 0) {      /* truncate */
                        err_ret("%s: open error", argv[i]);
                        continue;
                }
                timebuf.actime  = statbuf.st_atime;
                timebuf.modtime = statbuf.st_mtime;
                if (utime(argv[i], &timebuf) < 0) {             /* reset times */
                        err_ret("%s: utime error", argv[i]);
                        continue;
                }
        }
        exit(0);
}
```

(fisierul zap.c)

Codul programului zap:

Deschide si trunchiaza fișierul: Foloseste **open** cu **O_RDWR | O_TRUNC** pentru a deschide si trunchia fisierul, ceea ce inseamna ca fisierul va fi golit de continut

Actualizeaza timpii de acces si modificare: Folosește **utime** pentru a reseta timpii de acces si modificare la valorile originale obtinute cu **stat**

```
biancapinghireac@vbox:~/SO/lab5/src$ ./zap tempfile
biancapinghireac@vbox:~/SO/lab5/src$ ls -l tempfile
-rwxr-xr-x. 1 biancapinghireac biancapinghireac 0 Apr 16 10:40 tempfile
biancapinghireac@vbox:~/SO/lab5/src$ ls -lu tempfile
-rwxr-xr-x. 1 biancapinghireac biancapinghireac 0 Apr 16 10:40 tempfile
biancapinghireac@vbox:~/SO/lab5/src$ ls -lc tempfile
-rwxr-xr-x. 1 biancapinghireac biancapinghireac 0 Apr 16 10:41 tempfile
```

ls -lc afiseaza timpul ultimei modificari a metadatelor fisierului (ctime), de aceasta chiar daca utime este schimbat sa fie cel anterior, sa nu arate data modificarii, ctime se schimba

```
biancapinghireac@vbox:~/SO/lab5/src$ ./ftw4 /usr
can't read directory /usr/lib/containers/storage/overlay-images/: Permission denied
can't read directory /usr/lib/containers/storage/overlay-layers/: Permission denied
can't read directory /usr/libexec/initscripts/legacy-actions/auditd/: Permission denied
can't read directory /usr/share/empty.sshd/: Permission denied
can't read directory /usr/share/mdadm/: Permission denied
can't read directory /usr/share/polkit-1/rules.d/: Permission denied
regular files  =  141672, 73.62 %
directories    =   13782,  7.16 %
block special  =       0,  0.00 %
char special   =       0,  0.00 %
FIFOs          =       0,  0.00 %
symbolic links =   36979, 19.22 %
sockets        =       0,  0.00 %
```

Programul afiseaza un rezumat al tipurilor de fisiere gasite în **/usr**:

regular files: 141,672 fisiere obisnuite, reprezentand 73.62% din total.

directories: 13,782 directoare, reprezentand 7.16% din total.

symbolic links: 36,979 legaturi simbolice, reprezentand 19.22% din total.

block special, char special, FIFOs, sockets: Nu au fost gasite astfel de fisiere.

```c
#include "ourhdr.h"

int main(void)
{
        char    *ptr;
        int     size;

        if (chdir("/var/spool") < 0)
                err_sys("chdir failed");

        ptr = path_alloc(&size);        /* our own function */
        if (getcwd(ptr, size) == NULL)
                err_sys("getcwd failed");

        printf("cwd = %s\n", ptr);
        exit(0);
}
```

Codul programului cdpwd:

chdir("/var/spool"): Schimba directorul curent la **/var/spool**

path_alloc(&size): Aloca memorie pentru a stoca calea directorului curent

getcwd(ptr, size): Obtine calea absoluta a directorului curent si o stocheaza in **ptr**

```
biancapinghireac@vbox:~/SO/lab5/src$ ./cdpwd
cwd = /var/spool
biancapinghireac@vbox:~/SO/lab5/src$ ls -l /var/spool
total 0
drwxr-x--x. 1 root abrt 140 Apr 15 19:35 abrt
drwx------. 1 abrt abrt   0 Sep 11  2024 abrt-upload
drwx--x---. 1 root lp     6 Oct 24 17:54 cups
drwxr-xr-x. 1 root root   0 Jul 17  2024 lpd
drwxrwxr-x. 1 root mail  38 Mar  5 10:59 mail
drwxr-xr-x. 1 root root   0 Sep 21  2024 plymouth
```

```c
#include <sys/types.h>  /* BSD: defines major() and minor() */
#include <sys/stat.h>
#include <sys/sysmacros.h>
#include "ourhdr.h"

int main(int argc, char *argv[])
{
        int             i;
        struct stat     buf;

        for (i = 1; i < argc; i++) {
                printf("%s: ", argv[i]);
                if (lstat(argv[i], &buf) < 0) {
                        err_ret("lstat error");
                        continue;
                }

                printf("dev = %d/%d", major(buf.st_dev),  minor(buf.st_dev));

                if (S_ISCHR(buf.st_mode) || S_ISBLK(buf.st_mode)) {
                        printf(" (%s) rdev = %d/%d",
                                        (S_ISCHR(buf.st_mode)) ? "character" : "block",
                                        major(buf.st_rdev), minor(buf.st_rdev));
                }
                printf("\n");
        }
        exit(0);
}
```

Codul programului devrdev:

lstat(argv[i], &buf): Obține informatii despre fisierul specificat

major(buf.st_dev), minor(buf.st_dev): Obtine numerele major si minor ale dispozitivului de stocare

S_ISCHR și S_ISBLK: Verifica daca fișierul este un dispozitiv de caracter sau bloc si afiseazs numerele major si minor ale dispozitivului

Număr Major

Identifică tipul de dispozitiv: Numărul major este folosit pentru a identifica driverul de dispozitiv asociat. Practic, spune sistemului de operare ce driver să folosească pentru a interacționa cu dispozitivul.

Număr Minor

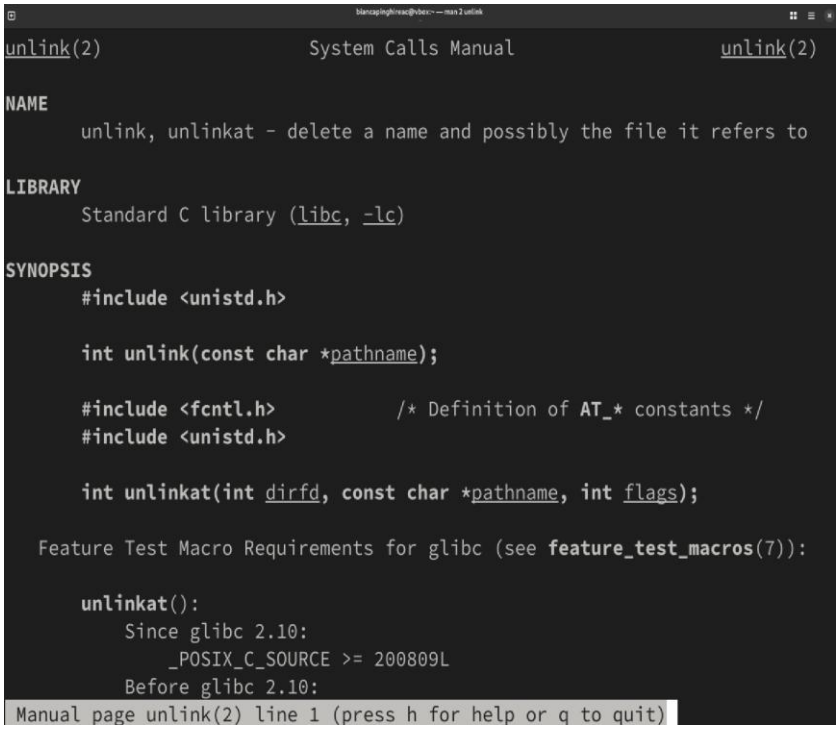
Identifică dispozitivul specific: Numărul minor este folosit pentru a diferenția între mai multe dispozitive care folosesc același driver. De exemplu, dacă ai mai multe hard disk-uri, toate ar putea avea același număr major, dar numere minore diferite.

```
biancapinghireac@vbox:~/SO/lab5/src$ ./devrdev / /dev/sr0 /dev/tty
/: dev = 0/39
/dev/sr0: lstat error: No such file or directory
/dev/tty: dev = 0/6 (character) rdev = 5/0
```

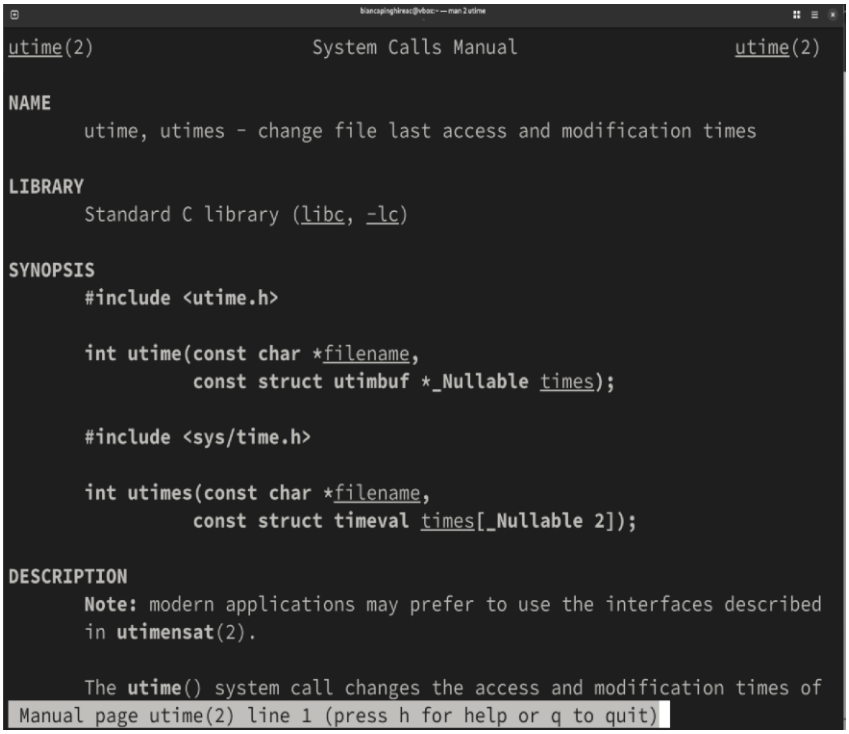
dev: Indică sistemul de fișiere unde este stocat fișierul de dispozitiv

rdev: Indică hard disk-ul specific pe care fișierul de dispozitiv îl controlează

Pagini de manual:

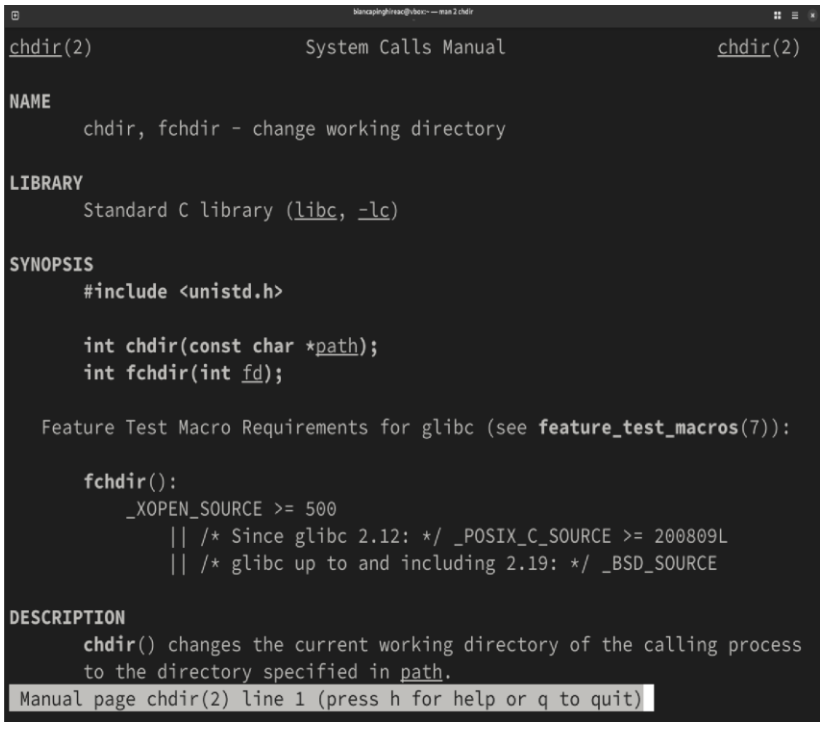


Unlink



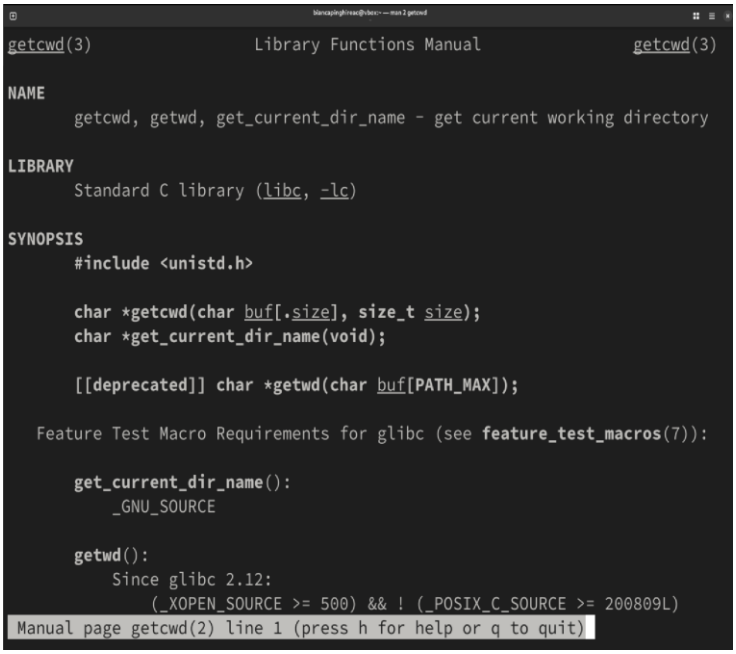
```
utime(2)                    System Calls Manual                    utime(2)

NAME
       utime, utimes - change file last access and modification times

LIBRARY
       Standard C library (libc, -lc)

SYNOPSIS
       #include <utime.h>

       int utime(const char *filename,
                 const struct utimbuf *_Nullable times);

       #include <sys/time.h>

       int utimes(const char *filename,
                 const struct timeval times[_Nullable 2]);

DESCRIPTION
       Note: modern applications may prefer to use the interfaces described
       in utimensat(2).

       The utime() system call changes the access and modification times of
 Manual page utime(2) line 1 (press h for help or q to quit)
```

Utime



```
chdir(2)                    System Calls Manual                    chdir(2)

NAME
       chdir, fchdir - change working directory

LIBRARY
       Standard C library (libc, -lc)

SYNOPSIS
       #include <unistd.h>

       int chdir(const char *path);
       int fchdir(int fd);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       fchdir():
           _XOPEN_SOURCE >= 500
               || /* Since glibc 2.12: */ _POSIX_C_SOURCE >= 200809L
               || /* glibc up to and including 2.19: */ _BSD_SOURCE

DESCRIPTION
       chdir() changes the current working directory of the calling process
       to the directory specified in path.
 Manual page chdir(2) line 1 (press h for help or q to quit)
```

Chdir



Getcwd

EXERCITIUL 2:

Comanda ls fara parametrii afiseaza toate fisierele din directorul curent(SO/lab5/src):

```
biancapinghireac@vbox:~/SO/lab5/src$ ls
cdpwd    devrdev.c  ftw4       Makefile  ourhdr.h     tempfile  zap
cdpwd.c  error.c    ftw4.c     mycd      pathalloc.c  unlink    zap.c
devrdev  error.o    liblab5.a  mycd.c    pathalloc.o  unlink.c
```

Comanda ls

```c
#include <stdio.h>
#include <dirent.h>
#include <string.h>
#include <stdlib.h>

int main(void){
        DIR *dir;
        struct dirent *curent;
        int count = 0;

        dir = opendir(".");
        if(dir==NULL){
                perror("eroare la deschiderea directorului");
                exit(10);
        }
        while((curent=readdir(dir))!=NULL){
                if(strcmp(curent->d_name, ".")!=0 && strcmp(curent->d_name,"..")!=0){
                        printf("%-13s",curent->d_name);
                        count++;
                }
                if(count%7 == 0 && count != 0){
                        printf("\n");
                        count = 0;
                }
        }
        printf("\n");
        closedir(dir);
        return 0;
}
```

Codul deschide directorul curent(in care se afla utilizatorul) si citeste pe rand fiecare componenta al acestuia, apoi o afiseaza pe ecran intr-un stil similar cu cel din comanda ls.

```
biancapinghireac@vbox:~/SO/lab5/src$ nano lsInC.c
biancapinghireac@vbox:~/SO/lab5/src$ gcc lsInC.c -o lsInC
biancapinghireac@vbox:~/SO/lab5/src$ ./lsInC
ftw4.c       mycd.c       Makefile     cdpwd.c      pathalloc.c  unlink.c     error.c
zap.c        devrdev.c    ourhdr.h     error.o      pathalloc.o  liblab5.a    ftw4
unlink       zap          mycd         cdpwd        devrdev      tempfile     lsInC.c
lsInC
```

EXERCITIUL 3:

```c
/* function type that's called for each filename */
typedef int     Myfunc(const char *, const struct stat *, int);

static Myfunc   myfunc;
static int      myftw(char *, Myfunc *);
static int      dopath(Myfunc *, char *);
```

```c
static int myftw(char *pathname, Myfunc *func)
{

        return dopath(func,pathname);
}
/*
 * Descend through the hierarchy, starting at "fullpath".
 * If "fullpath" is anything other than a directory, we lstat() it,
 * call func(), and return.  For a directory, we call ourself
 * recursively for each name in the directory.
 */
static int dopath(Myfunc* func,char* fullpath)
{
        struct stat     statbuf;
        struct dirent   *dirp;
        DIR             *dp;
        int             ret = 0;
        char            *ptr;

        if (lstat(fullpath, &statbuf) < 0)
                return func(fullpath, &statbuf, FTW_NS);        /* stat error */

        if (S_ISDIR(statbuf.st_mode) == 0)
                return func(fullpath, &statbuf, FTW_F); /* not a directory */

        /*
         * It's a directory.  First call func() for the directory,
         * then process each filename in the directory.
         */

        if ( (ret = func(fullpath, &statbuf, FTW_D)) != 0)
                return ret;
        if(chdir(fullpath)<0){ //change the directory
          return func(fullpath,&statbuf, FTW_DNR);
        }
```

```c
if ( (dp = opendir(".")) == NULL){ //directorul curent
        if(chdir("..")<0)
            perror("nu se poate intoarce");
        return func(fullpath, &statbuf, FTW_DNR);
}

while ( (dirp = readdir(dp)) != NULL) {
        if (strcmp(dirp->d_name, ".") == 0  ||
            strcmp(dirp->d_name, "..") == 0)
                    continue;       /* ignore dot and dot-dot */

        if ( (ret = dopath(func,dirp->d_name)) != 0)           /* recursive */
                break;                          /* time to leave */
}

if(chdir("..")<0){ //director parinte
        perror("can't chdir to ..");
        exit(10);
}
if (closedir(dp) < 0){ //director curent
        perror("closedir");
        exit(10);
}

return(ret);
```

Ftw4.c dupa modificare

```
biancapinghireac@vbox:~/SO/lab5(1)/src$ sudo time ./ftw4 /usr
[sudo] password for biancapinghireac:
regular files  =  141701, 73.63 %
directories    =   13782,  7.16 %
block special  =       0,  0.00 %
char special   =       0,  0.00 %
FIFOs          =       0,  0.00 %
symbolic links =   36980, 19.21 %
sockets        =       0,  0.00 %
0.11user 11.62system 0:22.88elapsed 51%CPU (0avgtext+0avgdata 1584maxresident)k
147936inputs+0outputs (0major+240minor)pagefaults 0swaps
```

Fisier original(foloseste calea absoluta: de la root pana la cel curent)

```
biancapinghireac@vbox:~/SO/lab5(1)/srbiancapinghireac@vbox:~/SO/lab5(1)/src$ gcc ftw4c.c -o ftw4c
biancapinghireac@vbox:~/SO/lab5(1)/src$ sudo time ./ftw4c /usr
regular files  =  141701, 73.63 %
directories    =   13782,  7.16 %
block special  =       0,  0.00 %
char special   =       0,  0.00 %
FIFOs          =       0,  0.00 %
symbolic links =   36980, 19.21 %
sockets        =       0,  0.00 %
0.08user 2.31system 0:02.85elapsed 83%CPU (0avgtext+0avgdata 1820maxresident)k
```

fisier modificat(foloseste calea relativa : folder/fisier.extensie)

Output ul arata 3 timpi:

real: actual elapsed time

user: CPU time in user mode

sys: CPU time in kernel mode

Din rezultate se deduce faptul ca folosirea cailor relative este mai rapida decat cea prin calea absoluta. Diferenta este una foarte mare intre cele doua, iar la parcurgeri de nivel mare(de exemplu al intregului sistem), folosirea caii absolute nu este recomandata.