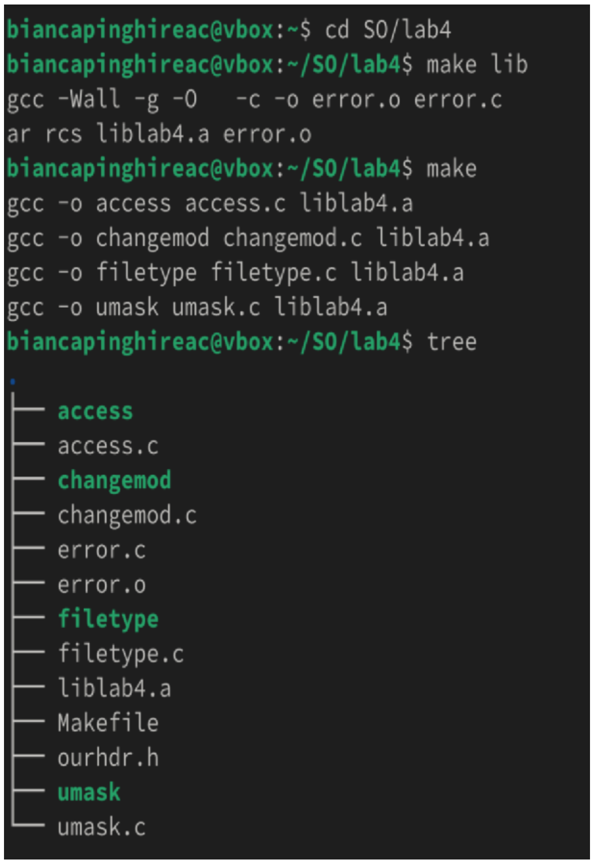
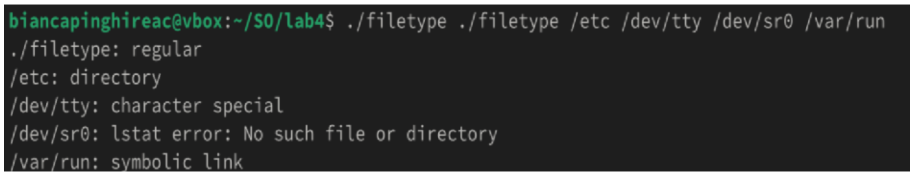
Laborator 4

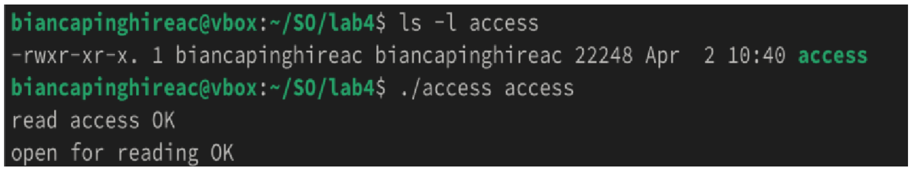


```
biancapinghireac@vbox:~$ cd SO/lab4
biancapinghireac@vbox:~/SO/lab4$ make lib
gcc -Wall -g -O   -c -o error.o error.c
ar rcs liblab4.a error.o
biancapinghireac@vbox:~/SO/lab4$ make
gcc -o access access.c liblab4.a
gcc -o changemod changemod.c liblab4.a
gcc -o filetype filetype.c liblab4.a
gcc -o umask umask.c liblab4.a
biancapinghireac@vbox:~/SO/lab4$ tree
.
├── access
├── access.c
├── changemod
├── changemod.c
├── error.c
├── error.o
├── filetype
├── filetype.c
├── liblab4.a
├── Makefile
├── ourhdr.h
├── umask
└── umask.c
```



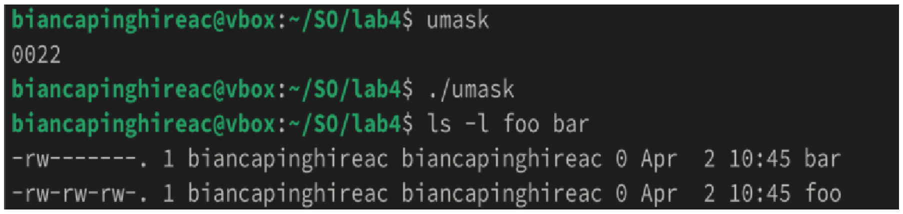
```
biancapinghireac@vbox:~/SO/lab4$ ./filetype ./filetype /etc /dev/tty /dev/sr0 /var/run
./filetype: regular
/etc: directory
/dev/tty: character special
/dev/sr0: lstat error: No such file or directory
/var/run: symbolic link
```

Filetype.c – arata tipul fisierului dat in linia de comanda



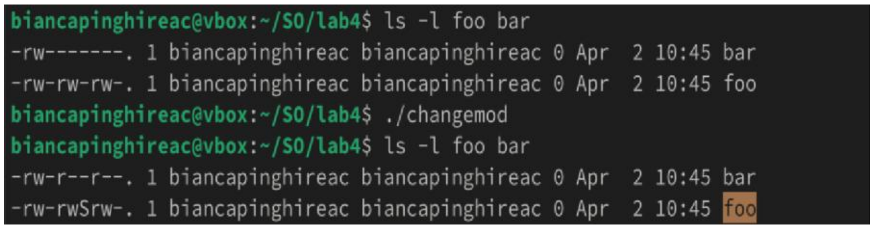
```
biancapinghireac@vbox:~/SO/lab4$ ls -l access
-rwxr-xr-x. 1 biancapinghireac biancapinghireac 22248 Apr  2 10:40 access
biancapinghireac@vbox:~/SO/lab4$ ./access access
read access OK
open for reading OK
```

Access.c – verifica drepturile de citire si acces al fisierului dat

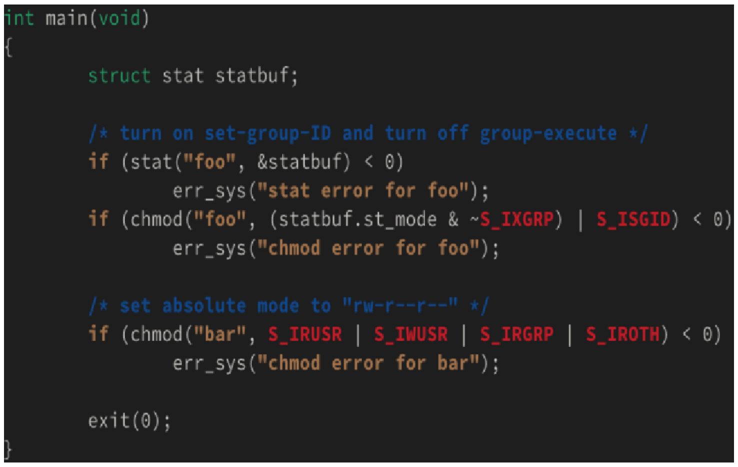


```
biancapinghireac@vbox:~/SO/lab4$ umask
0022
biancapinghireac@vbox:~/SO/lab4$ ./umask
biancapinghireac@vbox:~/SO/lab4$ ls -l foo bar
-rw-------. 1 biancapinghireac biancapinghireac 0 Apr  2 10:45 bar
-rw-rw-rw-. 1 biancapinghireac biancapinghireac 0 Apr  2 10:45 foo
```

Se creeaza fisierele ‘foo’ și ‘bar’ dolosind masti diferite pentru drepturile de acces si anume: 000 (umask(0)) si 066



```
biancapinghireac@vbox:~/SO/lab4$ ls -l foo bar
-rw-------. 1 biancapinghireac biancapinghireac 0 Apr  2 10:45 bar
-rw-rw-rw-. 1 biancapinghireac biancapinghireac 0 Apr  2 10:45 foo
biancapinghireac@vbox:~/SO/lab4$ ./changemod
biancapinghireac@vbox:~/SO/lab4$ ls -l foo bar
-rw-r--r--. 1 biancapinghireac biancapinghireac 0 Apr  2 10:45 bar
-rw-rwSrw-. 1 biancapinghireac biancapinghireac 0 Apr  2 10:45 foo
```

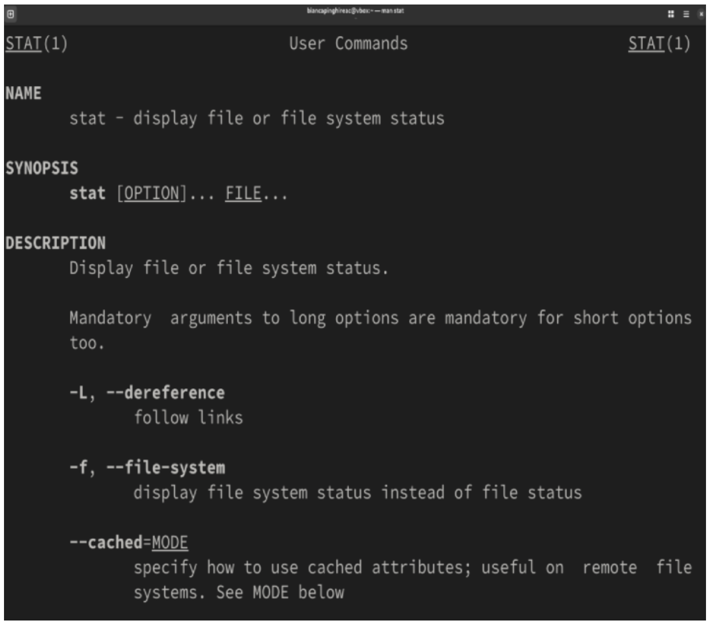


```
int main(void)
{
        struct stat statbuf;

        /* turn on set-group-ID and turn off group-execute */
        if (stat("foo", &statbuf) < 0)
                err_sys("stat error for foo");
        if (chmod("foo", (statbuf.st_mode & ~S_IXGRP) | S_ISGID) < 0)
                err_sys("chmod error for foo");

        /* set absolute mode to "rw-r--r--" */
        if (chmod("bar", S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH) < 0)
                err_sys("chmod error for bar");

        exit(0);
}
```

Schimbam drepturile de acces ale fisierelor in:

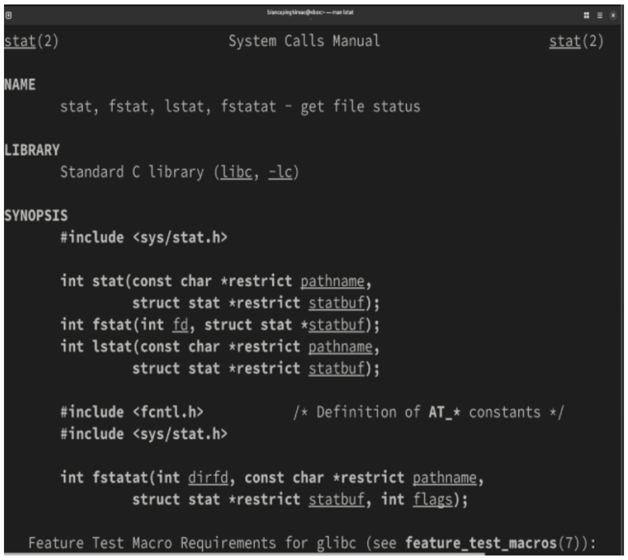
1. Pentru "foo": activeaza SGID(set-group-ID) si dezactivează execuția pentru grup (rezulta in rwS)
2. Pentru "bar": seteaza permisiunile la rw-r--r--

Pagini de manual:



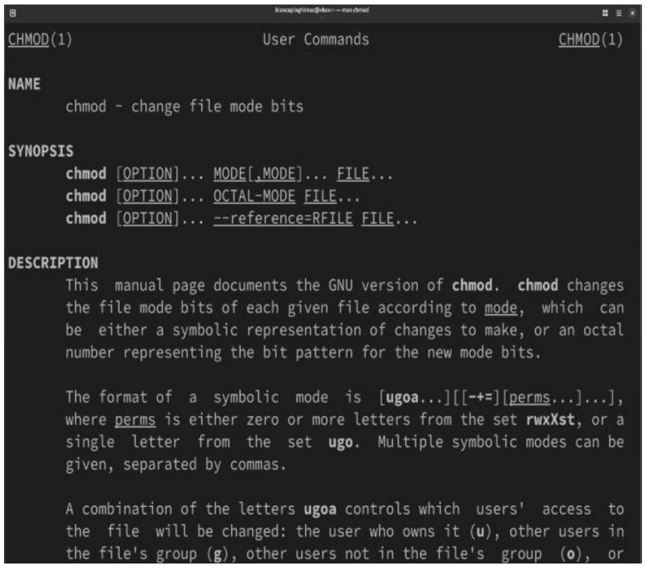
```
STAT(1)                      User Commands                      STAT(1)

NAME
       stat - display file or file system status

SYNOPSIS
       stat [OPTION]... FILE...

DESCRIPTION
       Display file or file system status.

       Mandatory  arguments to long options are mandatory for short options
       too.

       -L, --dereference
              follow links

       -f, --file-system
              display file system status instead of file status

       --cached=MODE
              specify how to use cached attributes; useful on  remote  file
              systems. See MODE below
```

Stat



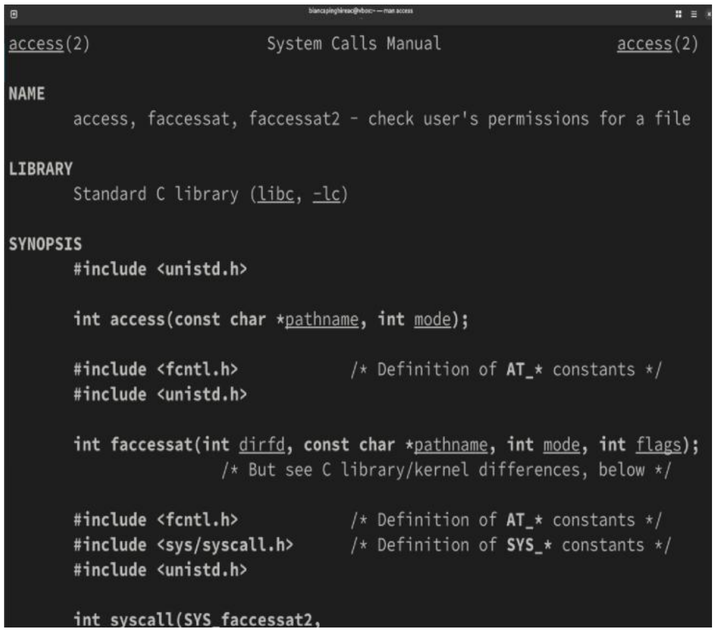
```
stat(2)                    System Calls Manual                    stat(2)

NAME
       stat, fstat, lstat, fstatat - get file status

LIBRARY
       Standard C library (libc, -lc)

SYNOPSIS
       #include <sys/stat.h>

       int stat(const char *restrict pathname,
                struct stat *restrict statbuf);
       int fstat(int fd, struct stat *statbuf);
       int lstat(const char *restrict pathname,
                struct stat *restrict statbuf);

       #include <fcntl.h>           /* Definition of AT_* constants */
       #include <sys/stat.h>

       int fstatat(int dirfd, const char *restrict pathname,
                struct stat *restrict statbuf, int flags);

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

Lstat



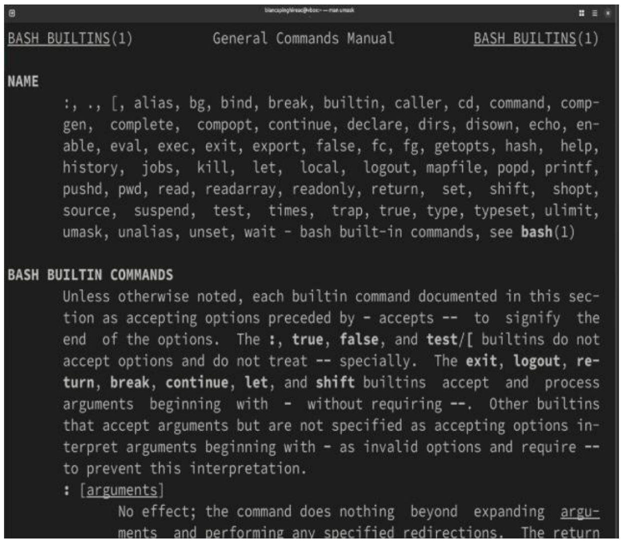
```
CHMOD(1)                     User Commands                    CHMOD(1)

NAME
       chmod - change file mode bits

SYNOPSIS
       chmod [OPTION]... MODE[,MODE]... FILE...
       chmod [OPTION]... OCTAL-MODE FILE...
       chmod [OPTION]... --reference=RFILE FILE...

DESCRIPTION
       This  manual page documents the GNU version of chmod.  chmod changes
       the file mode bits of each given file according to mode,  which  can
       be  either a symbolic representation of changes to make, or an octal
       number representing the bit pattern for the new mode bits.

       The format of  a  symbolic  mode  is  [ugoa...][[-+=][perms...]...],
       where perms is either zero or more letters from the set rwxXst, or a
       single  letter  from  the  set  ugo.  Multiple symbolic modes can be
       given, separated by commas.

       A combination of the letters ugoa controls which  users'  access  to
       the  file  will be changed: the user who owns it (u), other users in
       the file's group (g), other users not in the file's  group  (o),  or
```

Chmod



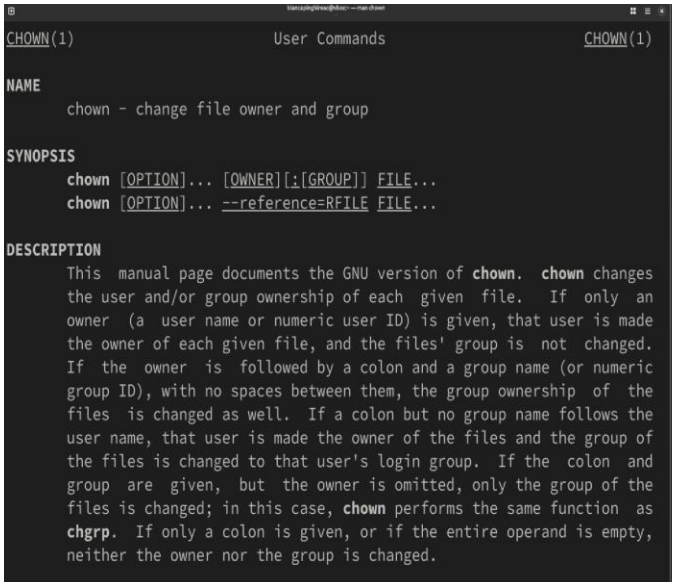
```
access(2)                  System Calls Manual                  access(2)

NAME
       access, faccessat, faccessat2 - check user's permissions for a file

LIBRARY
       Standard C library (libc, -lc)

SYNOPSIS
       #include <unistd.h>

       int access(const char *pathname, int mode);

       #include <fcntl.h>            /* Definition of AT_* constants */
       #include <unistd.h>

       int faccessat(int dirfd, const char *pathname, int mode, int flags);
                      /* But see C library/kernel differences, below */

       #include <fcntl.h>            /* Definition of AT_* constants */
       #include <sys/syscall.h>      /* Definition of SYS_* constants */
       #include <unistd.h>

       int syscall(SYS_faccessat2,
```

Access



```
BASH_BUILTINS(1)          General Commands Manual          BASH_BUILTINS(1)

NAME
       :, ., [, alias, bg, bind, break, builtin, caller, cd, command, comp‐
       gen,  complete,  compopt, continue, declare, dirs, disown, echo, en‐
       able, eval, exec, exit, export, false, fc, fg, getopts, hash,  help,
       history,  jobs,  kill,  let,  local,  logout, mapfile, popd, printf,
       pushd, pwd, read, readarray, readonly, return,  set,  shift,  shopt,
       source,  suspend,  test,  times,  trap, true, type, typeset, ulimit,
       umask, unalias, unset, wait - bash built-in commands, see bash(1)

BASH BUILTIN COMMANDS
       Unless otherwise noted, each builtin command documented in this sec‐
       tion as accepting options preceded by - accepts --  to  signify  the
       end  of the options.  The :, true, false, and test/[ builtins do not
       accept options and do not treat -- specially.  The exit, logout, re‐
       turn, break, continue, let, and shift builtins  accept  and  process
       arguments  beginning  with  -  without requiring --.  Other builtins
       that accept arguments but are not specified as accepting options in‐
       terpret arguments beginning with - as invalid options and require --
       to prevent this interpretation.
       : [arguments]
              No effect; the command does nothing  beyond  expanding  argu‐
              ments  and performing any specified redirections.  The return
```

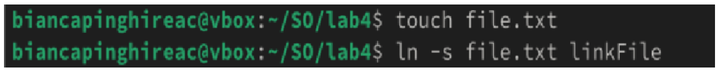
Umask



```
CHOWN(1)                        User Commands                        CHOWN(1)

NAME
       chown - change file owner and group

SYNOPSIS
       chown [OPTION]... [OWNER][:[GROUP]] FILE...
       chown [OPTION]... --reference=RFILE FILE...

DESCRIPTION
       This  manual page documents the GNU version of chown.  chown changes
       the user and/or group ownership of each  given  file.   If  only  an
       owner  (a  user name or numeric user ID) is given, that user is made
       the owner of each given file, and the files' group is  not  changed.
       If  the  owner  is  followed by a colon and a group name (or numeric
       group ID), with no spaces between them, the group ownership  of  the
       files  is changed as well.  If a colon but no group name follows the
       user name, that user is made the owner of the files and the group of
       the files is changed to that user's login group.  If the  colon  and
       group  are  given,  but  the owner is omitted, only the group of the
       files is changed; in this case, chown performs the same function  as
       chgrp.  If only a colon is given, or if the entire operand is empty,
       neither the owner nor the group is changed.
```

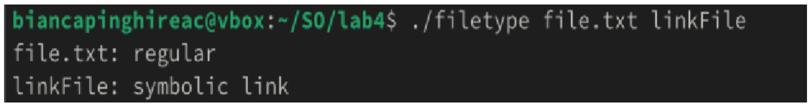
Chown

EXERCITIUL 2:

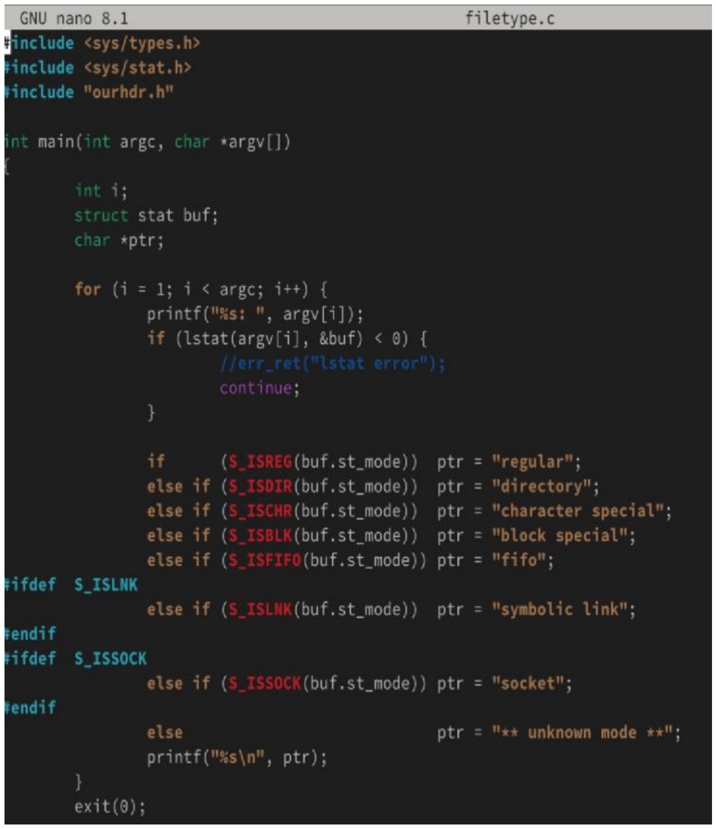


```
biancapinghireac@vbox:~/SO/lab4$ touch file.txt
biancapinghireac@vbox:~/SO/lab4$ ln -s file.txt linkFile
```

Facem symbolic link (este un tip special de fisier care contine o referinta (sub forma unei cai) catre un alt fisier sau director) spre un fisier file.txt de tip regular

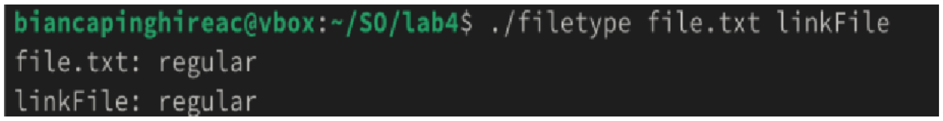


```
biancapinghireac@vbox:~/SO/lab4$ ./filetype file.txt linkFile
file.txt: regular
linkFile: symbolic link
```



```
GNU nano 8.1                                          filetype.c
#include <sys/types.h>
#include <sys/stat.h>
#include "ourhdr.h"

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

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

                if      (S_ISREG(buf.st_mode))  ptr = "regular";
                else if (S_ISDIR(buf.st_mode))  ptr = "directory";
                else if (S_ISCHR(buf.st_mode))  ptr = "character special";
                else if (S_ISBLK(buf.st_mode))  ptr = "block special";
                else if (S_ISFIFO(buf.st_mode)) ptr = "fifo";
#ifdef  S_ISLNK
                else if (S_ISLNK(buf.st_mode))  ptr = "symbolic link";
#endif
#ifdef  S_ISSOCK
                else if (S_ISSOCK(buf.st_mode)) ptr = "socket";
#endif
                else                            ptr = "** unknown mode **";
                printf("%s\n", ptr);
        }
        exit(0);
```

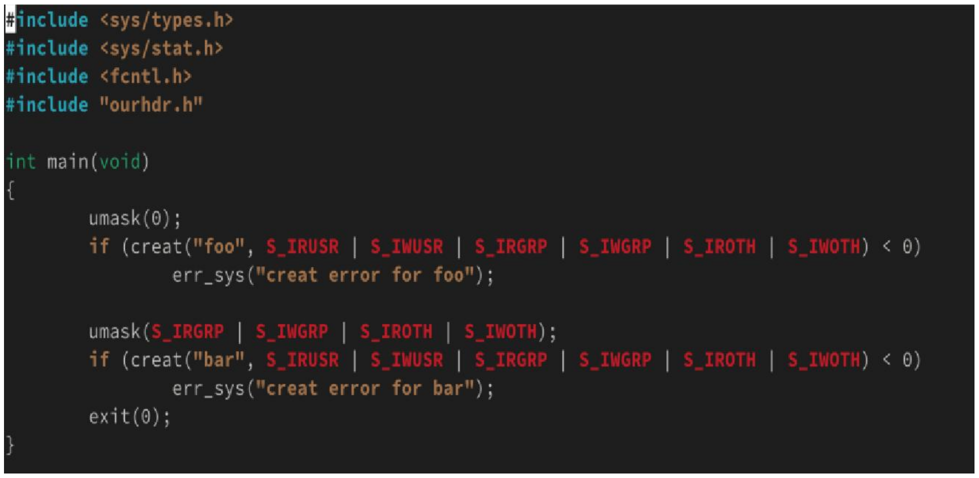
Filetype.c cu lstat permite identificarea acestor fisiere link symbolic



```
biancapinghireac@vbox:~/SO/lab4$ ./filetype file.txt linkFile
file.txt: regular
linkFile: regular
```

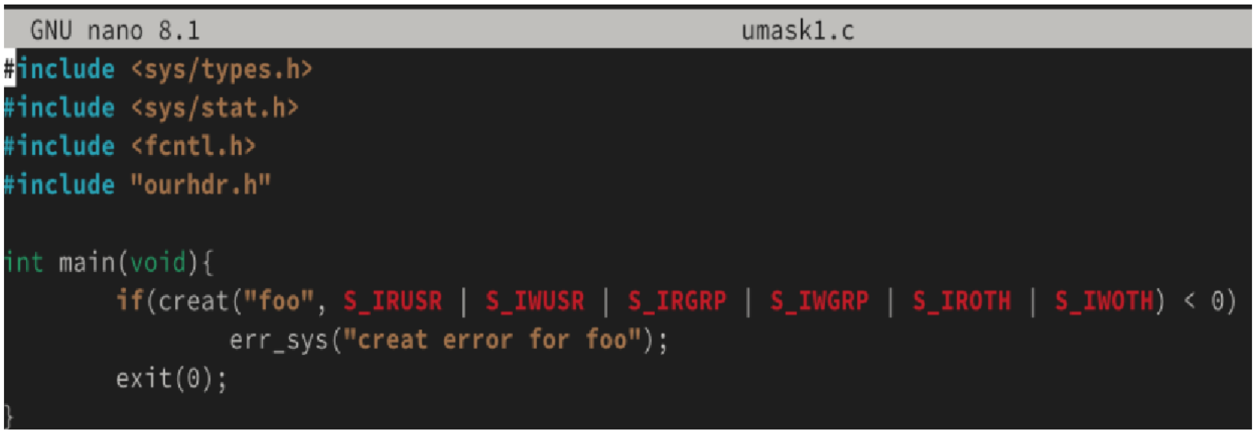
Fisierul modificat cauta tipul fisierului spre care arata fisierul link symbolic

EXERCITIUL 3:



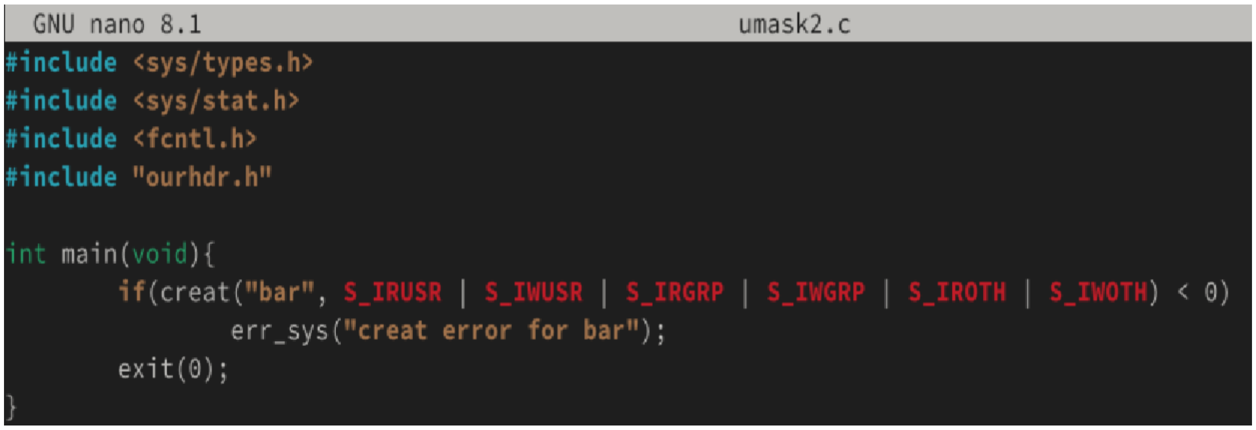
```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include "ourhdr.h"

int main(void)
{
        umask(0);
        if (creat("foo", S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH) < 0)
                err_sys("creat error for foo");

        umask(S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH);
        if (creat("bar", S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH) < 0)
                err_sys("creat error for bar");
        exit(0);
}
```

(umask.c original)



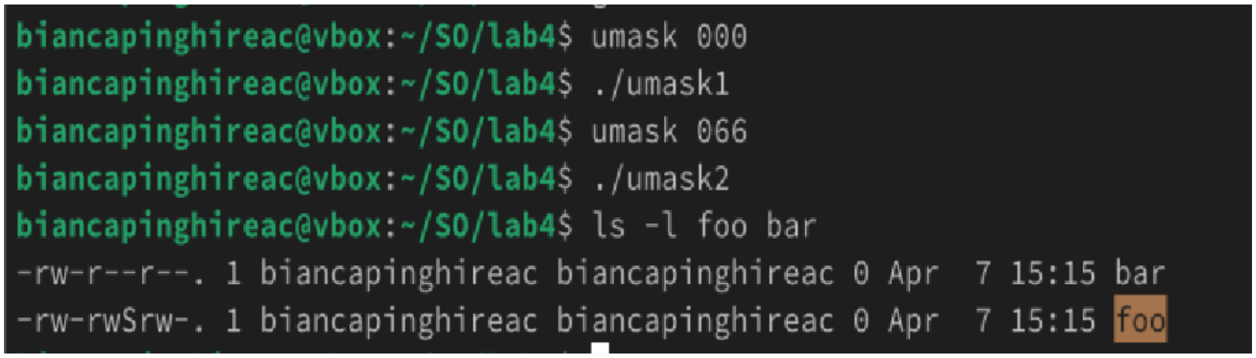
```
  GNU nano 8.1                                    umask1.c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include "ourhdr.h"

int main(void){
        if(creat("foo", S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH) < 0)
                err_sys("creat error for foo");
        exit(0);
}
```

Foo – permisiuni 666



```
  GNU nano 8.1                                    umask2.c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include "ourhdr.h"

int main(void){
        if(creat("bar", S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH) < 0)
                err_sys("creat error for bar");
        exit(0);
}
```

Bar – permisiuni 666



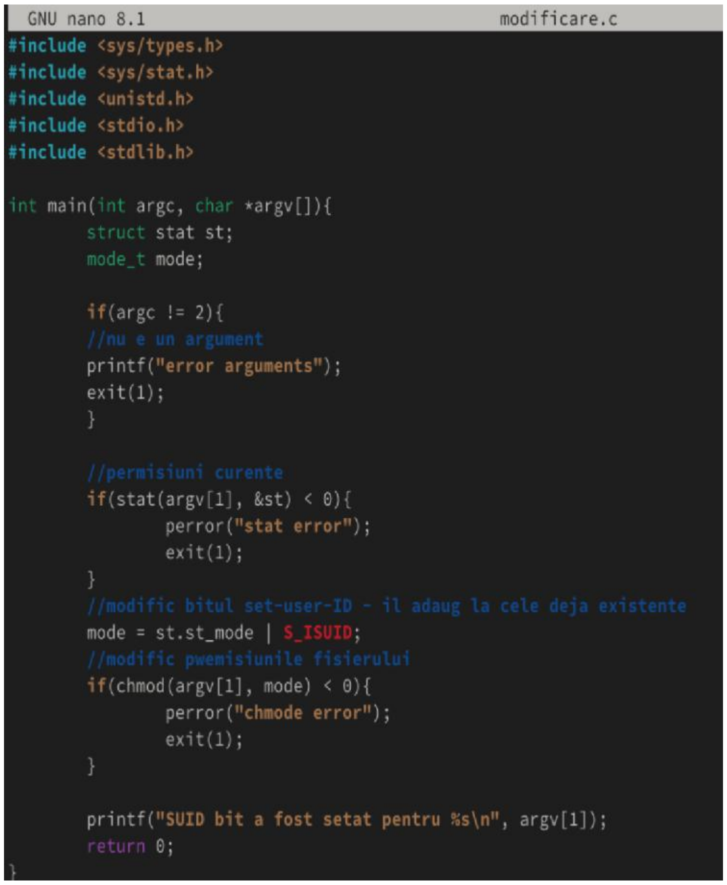
```
biancapinghireac@vbox:~/SO/lab4$ umask 000
biancapinghireac@vbox:~/SO/lab4$ ./umask1
biancapinghireac@vbox:~/SO/lab4$ umask 066
biancapinghireac@vbox:~/SO/lab4$ ./umask2
biancapinghireac@vbox:~/SO/lab4$ ls -l foo bar
-rw-r--r--. 1 biancapinghireac biancapinghireac 0 Apr  7 15:15 bar
-rw-rwSrw-. 1 biancapinghireac biancapinghireac 0 Apr  7 15:15 foo
```

Umask 000 – nu neaga nimic (S – nu are drept de executie, dar set-user-ID este setat)

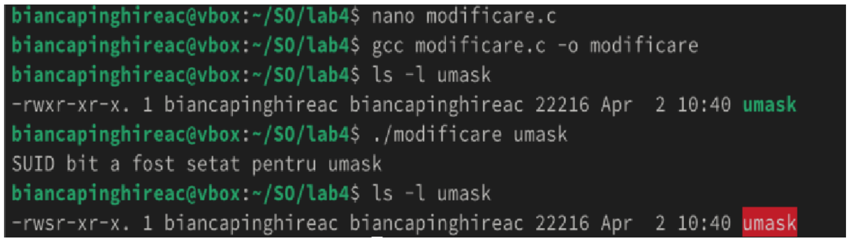
Umask 066 – neaga read si write

EXERCITIUL 4:

Metoda fara chown:

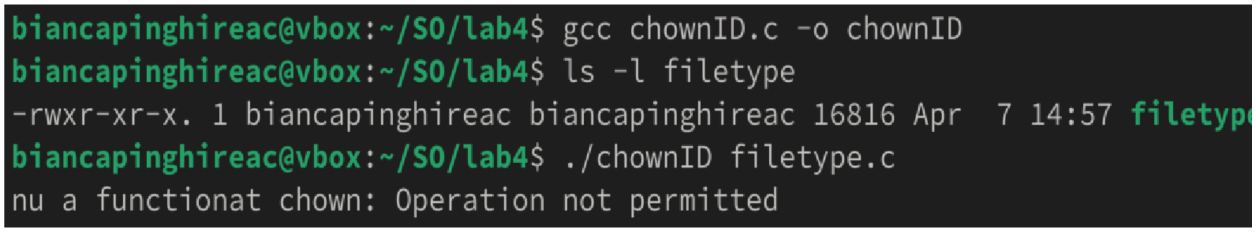


```
 GNU nano 8.1                                        modificare.c
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]){
        struct stat st;
        mode_t mode;

        if(argc != 2){
        //nu e un argument
        printf("error arguments");
        exit(1);
        }

        //permisiuni curente
        if(stat(argv[1], &st) < 0){
                perror("stat error");
                exit(1);
        }
        //modific bitul set-user-ID - il adaug la cele deja existente
        mode = st.st_mode | S_ISUID;
        //modific pwemisiunile fisierului
        if(chmod(argv[1], mode) < 0){
                perror("chmode error");
                exit(1);
        }

        printf("SUID bit a fost setat pentru %s\n", argv[1]);
        return 0;
}
```

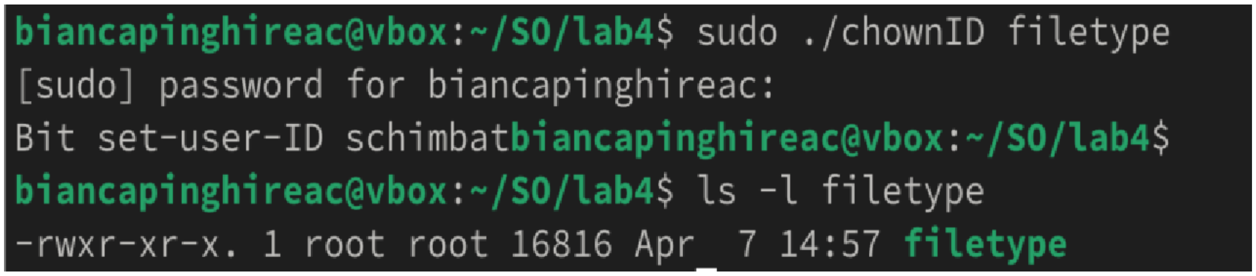

```
biancapinghireac@vbox:~/SO/lab4$ nano modificare.c
biancapinghireac@vbox:~/SO/lab4$ gcc modificare.c -o modificare
biancapinghireac@vbox:~/SO/lab4$ ls -l umask
-rwxr-xr-x. 1 biancapinghireac biancapinghireac 22216 Apr  2 10:40 umask
biancapinghireac@vbox:~/SO/lab4$ ./modificare umask
SUID bit a fost setat pentru umask
biancapinghireac@vbox:~/SO/lab4$ ls -l umask
-rwsr-xr-x. 1 biancapinghireac biancapinghireac 22216 Apr  2 10:40 umask
```

Se poate observa acum 's' care inseamna ca bitul de set-user-ID este setat si se poate executa, daca ar fi fost 'S' set-user-ID ul este setat, dar nu are drept de executie.

Metoda cu chown:


```
biancapinghireac@vbox:~/SO/lab4$ gcc chownID.c -o chownID
biancapinghireac@vbox:~/SO/lab4$ ls -l filetype
-rwxr-xr-x. 1 biancapinghireac biancapinghireac 16816 Apr  7 14:57 filetyp
biancapinghireac@vbox:~/SO/lab4$ ./chownID filetype.c
nu a functionat chown: Operation not permitted
```

Initial da eroare deoarece nu avem permisiunea de a schimba set-user-ID ul


```
biancapinghireac@vbox:~/SO/lab4$ sudo ./chownID filetype
[sudo] password for biancapinghireac:
Bit set-user-ID schimbatbiancapinghireac@vbox:~/SO/lab4$
biancapinghireac@vbox:~/SO/lab4$ ls -l filetype
-rwxr-xr-x. 1 root root 16816 Apr  7 14:57 filetype
```

Dar daca apelam cu `sudo` avem permisiune complete si putem efectua schimbarea