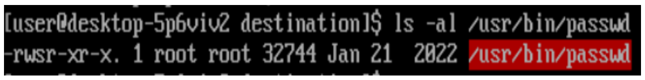
Sisteme de operare

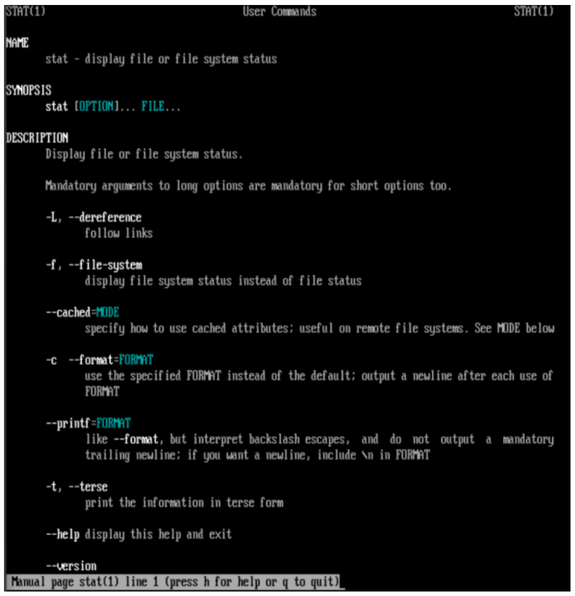
Tema 4

Exercițiul 1

Rulați toate exercițiile prezentate și încercați să înțelegeți cum lucrează. Vedeți paginile de manual pentru funcțiile utilizate.

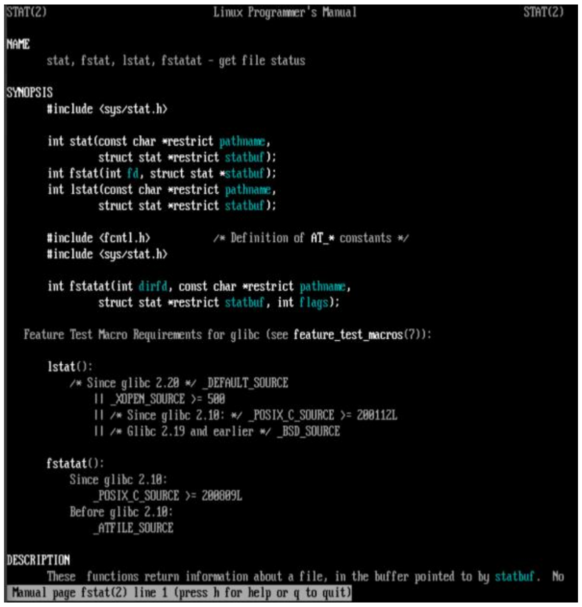


```
[user@desktop-5p6viv2 destination]$ ls -al /usr/bin/passwd
-rwsr-xr-x. 1 root root 32744 Jan 21  2022 /usr/bin/passwd
```



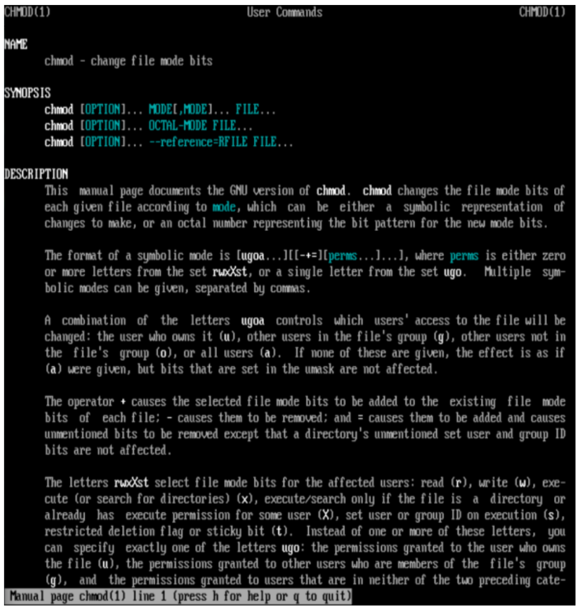
```
STAT(1)                          User Commands                          STAT(1)

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

       -c  --format=FORMAT
              use the specified FORMAT instead of the default; output a newline after each use of
              FORMAT

       --printf=FORMAT
              like --format, but interpret backslash escapes,  and  do  not  output  a  mandatory
              trailing newline; if you want a newline, include \n in FORMAT

       -t, --terse
              print the information in terse form

       --help display this help and exit

       --version
 Manual page stat(1) line 1 (press h for help or q to quit)
```

Funcția stat.



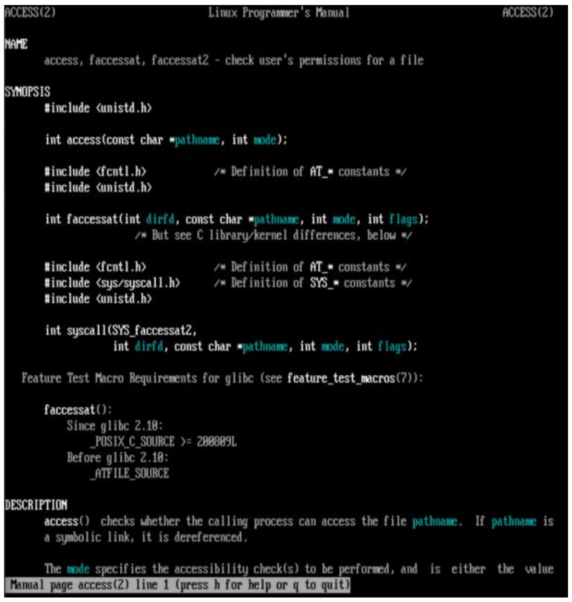
```
STAT(2)                    Linux Programmer's Manual                   STAT(2)

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

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():
           /* Since glibc 2.20 */ _DEFAULT_SOURCE
               || _XOPEN_SOURCE >= 500
               || /* Since glibc 2.10: */ _POSIX_C_SOURCE >= 200112L
               || /* Glibc 2.19 and earlier */ _BSD_SOURCE

       fstatat():
           Since glibc 2.10:
               _POSIX_C_SOURCE >= 200809L
           Before glibc 2.10:
               _ATFILE_SOURCE

DESCRIPTION
       These  functions return information about a file, in the buffer pointed to by statbuf.  No
 Manual page fstat(2) line 1 (press h for help or q to quit)
```

Funcția fstat și funcția lstat.



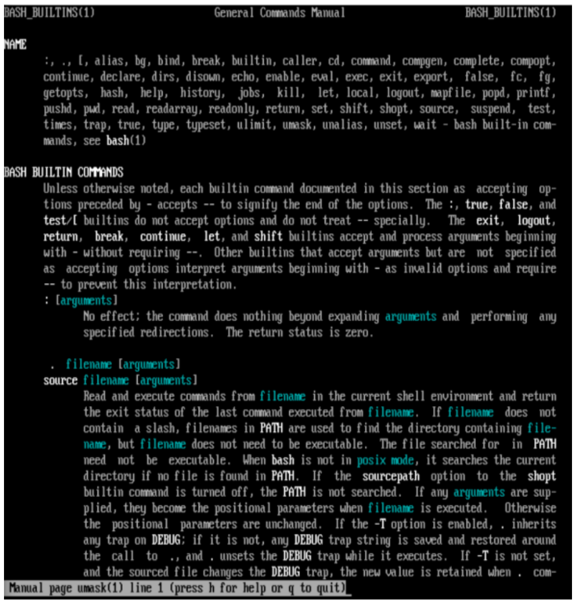
```
CHMOD(1)                         User Commands                        CHMOD(1)

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  sym-
       bolic 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 all users (a).  If none of these are given, the effect is as if
       (a) were given, but bits that are set in the umask are not affected.

       The operator + causes the selected file mode bits to be added to the  existing  file  mode
       bits  of  each file; - causes them to be removed; and = causes them to be added and causes
       unmentioned bits to be removed except that a directory's unmentioned set user and group ID
       bits are not affected.

       The letters rwxXst select file mode bits for the affected users: read (r), write (w), exe-
       cute (or search for directories) (x), execute/search only if the file is  a  directory  or
       already  has  execute permission for some user (X), set user or group ID on execution (s),
       restricted deletion flag or sticky bit (t).  Instead of one or more of these letters,  you
       can  specify  exactly one of the letters ugo: the permissions granted to the user who owns
       the file (u), the permissions granted to other users who are members of the  file's  group
       (g),  and  the permissions granted to users that are in neither of the two preceding cate-
 Manual page chmod(1) line 1 (press h for help or q to quit)
```

Funcția chmod.



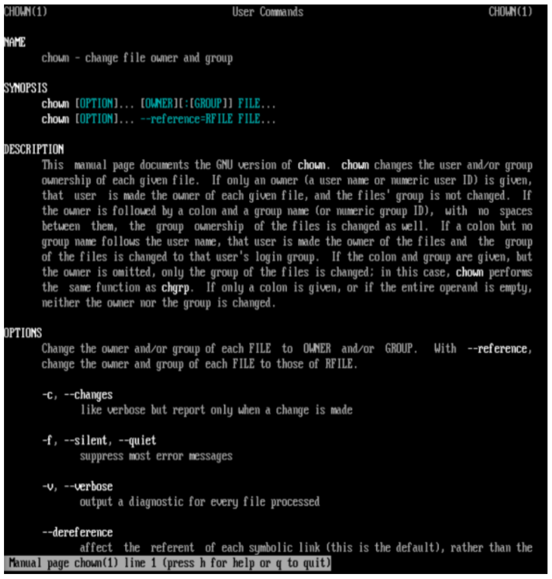
```
ACCESS(2)                    Linux Programmer's Manual                   ACCESS(2)

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

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,
                   int dirfd, const char *pathname, int mode, int flags);

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

       faccessat():
           Since glibc 2.10:
               _POSIX_C_SOURCE >= 200809L
           Before glibc 2.10:
               _ATFILE_SOURCE

DESCRIPTION
       access()  checks whether the calling process can access the file pathname.  If pathname is
       a symbolic link, it is dereferenced.

       The mode specifies the accessibility check(s) to be performed, and  is  either  the  value
 Manual page access(2) line 1 (press h for help or q to quit)
```

Funcția access.



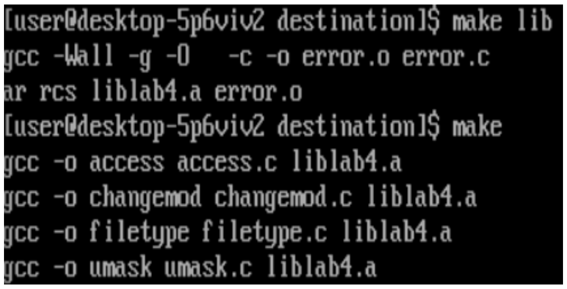
```
BASH_BUILTINS(1)             General Commands Manual             BASH_BUILTINS(1)

NAME
       :, ., [, alias, bg, bind, break, builtin, caller, cd, command, compgen, complete, compopt,
       continue, declare, dirs, disown, echo, enable, 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 com-
       mands, see bash(1)

BASH BUILTIN COMMANDS
       Unless otherwise noted, each builtin command documented in this section as  accepting  op-
       tions 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,
       return,  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 interpret arguments beginning with - as invalid options and require
       -- to prevent this interpretation.
       : [arguments]
              No effect; the command does nothing beyond expanding arguments and  performing  any
              specified redirections.  The return status is zero.

        .  filename [arguments]
       source filename [arguments]
              Read and execute commands from filename in the current shell environment and return
              the exit status of the last command executed from filename.  If filename  does  not
              contain  a slash, filenames in PATH are used to find the directory containing file-
              name, but filename does not need to be executable.  The file searched for  in  PATH
              need  not  be  executable.  When bash is not in posix mode, it searches the current
              directory if no file is found in PATH.  If  the  sourcepath  option  to  the  shopt
              builtin command is turned off, the PATH is not searched.  If any arguments are sup-
              plied, they become the positional parameters when filename is executed.  Otherwise
              the  positional  parameters are unchanged.  If the -T option is enabled, . inherits
              any trap on DEBUG; if it is not, any DEBUG trap string is saved and restored around
              the  call  to  ., and . unsets the DEBUG trap while it executes.  If -T is not set,
              and the sourced file changes the DEBUG trap, the new value is retained when .  com-
 Manual page umask(1) line 1 (press h for help or q to quit)
```

Funcția umask.



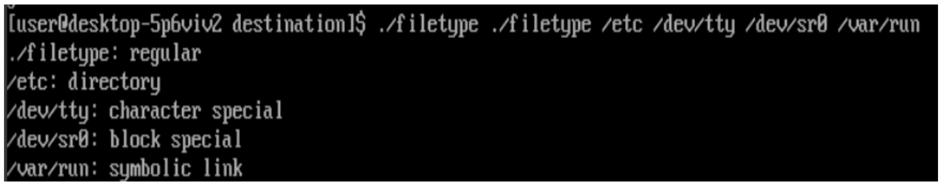
```
CHOWN(1)                          User Commands                          CHOWN(1)

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.

OPTIONS
       Change the owner and/or group of each FILE  to  OWNER  and/or  GROUP.   With  --reference,
       change the owner and group of each FILE to those of RFILE.

       -c, --changes
              like verbose but report only when a change is made

       -f, --silent, --quiet
              suppress most error messages

       -v, --verbose
              output a diagnostic for every file processed

       --dereference
              affect  the  referent  of each symbolic link (this is the default), rather than the
 Manual page chown(1) line 1 (press h for help or q to quit)
```

Funcția chown.



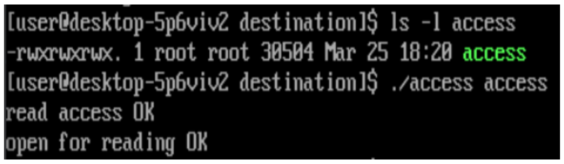
```
[user@desktop-5p6viv2 destination]$ make lib
gcc -Wall -g -O   -c -o error.o error.c
ar rcs liblab4.a error.o
[user@desktop-5p6viv2 destination]$ make
gcc -o access access.c liblab4.a
gcc -o changemod changemod.c liblab4.a
gcc -o filetype filetype.c liblab4.a
gcc -o umask umask.c liblab4.a
```

Am creat librăria liblab4.a .



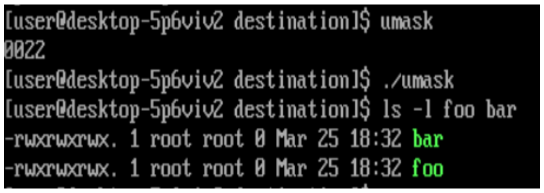
```
[user@desktop-5p6viv2 destination]$ ./filetype ./filetype /etc /dev/tty /dev/sr0 /var/run
./filetype: regular
/etc: directory
/dev/tty: character special
/dev/sr0: block special
/var/run: symbolic link
```

Am afișat tipul fiecărui fișier pus în linai de comandă.



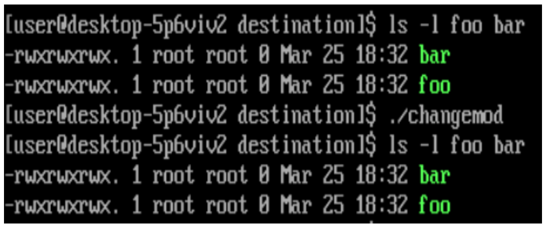
```
[user@desktop-5p6viv2 destination]$ ls -l access
-rwxrwxrwx. 1 root root 30504 Mar 25 18:20 access
[user@desktop-5p6viv2 destination]$ ./access access
read access OK
open for reading OK
```

Verifică dreptul de acces și citire din fișierul dat.



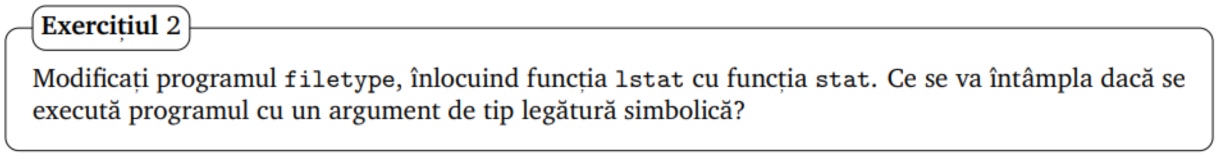
```
[user@desktop-5p6viv2 destination]$ umask
0022
[user@desktop-5p6viv2 destination]$ ./umask
[user@desktop-5p6viv2 destination]$ ls -l foo bar
-rwxrwxrwx. 1 root root 0 Mar 25 18:32 bar
-rwxrwxrwx. 1 root root 0 Mar 25 18:32 foo
```

Creează fișierule foo și bar dolosind măsti diferite pentru drepturile de acces.



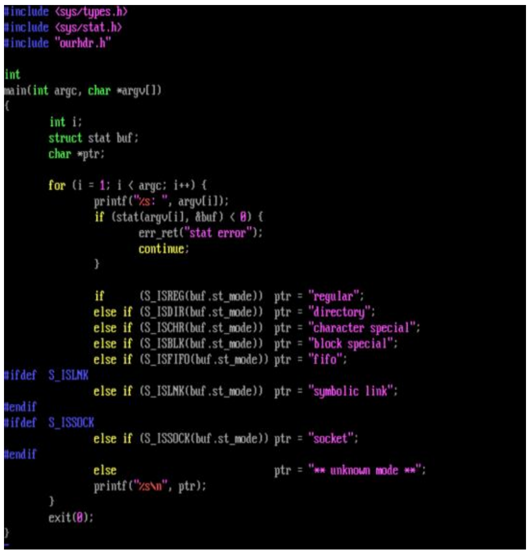
```
[user@desktop-5p6viv2 destination]$ ls -l foo bar
-rwxrwxrwx. 1 root root 0 Mar 25 18:32 bar
-rwxrwxrwx. 1 root root 0 Mar 25 18:32 foo
[user@desktop-5p6viv2 destination]$ ./changemod
[user@desktop-5p6viv2 destination]$ ls -l foo bar
-rwxrwxrwx. 1 root root 0 Mar 25 18:32 bar
-rwxrwxrwx. 1 root root 0 Mar 25 18:32 foo
```

Schimbăm drepturile de acces pentru fișierele foo și bar.w2edc



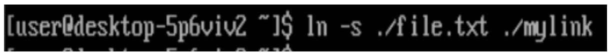
Exercițiul 2

Modificați programul `filetype`, înlocuind funcția `lstat` cu funcția `stat`. Ce se va întâmpla dacă se execută programul cu un argument de tip legătură simbolică?



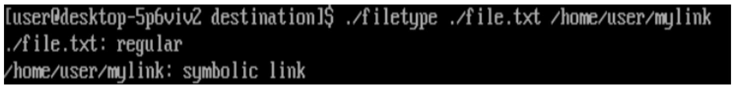
```c
#include <sys/types.h>
#include <sys/stat.h>
#include "ourhdr.h"

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

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

Am înlocuit lstat cu stat în programul filetype.



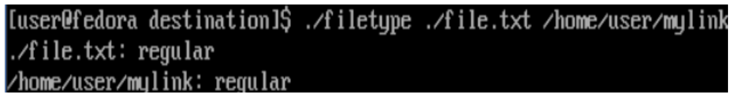
```
[user@desktop-5p6viv2 ~]$ ln -s ./file.txt ./mylink
```

Am creat o legătură simbolică pentru fișierul file.txt .



```
[user@desktop-5p6viv2 destination]$ ./filetype ./file.txt /home/user/mylink
./file.txt: regular
/home/user/mylink: symbolic link
```

Înainte de modificare, am folosit programul filetype cu fișierul file.txt și cu argumentul de tip legătură simbolică al fișierului.

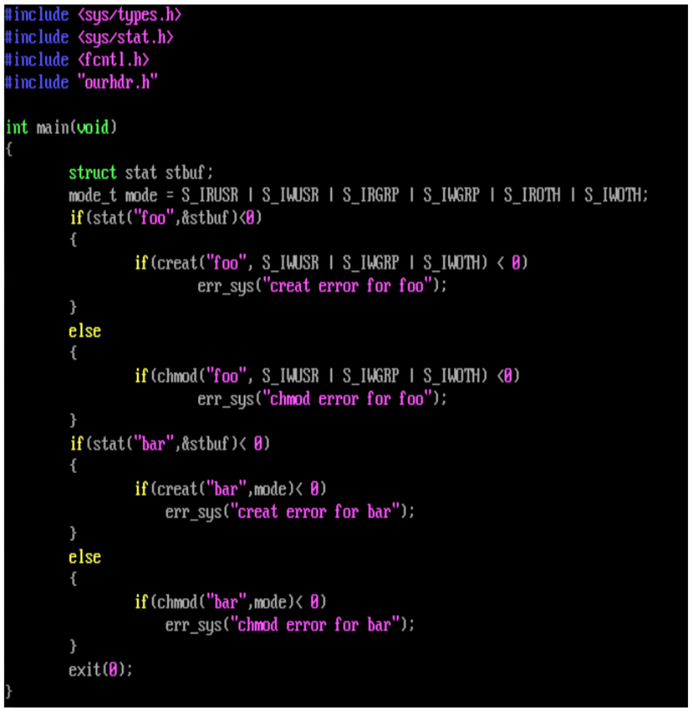


```
[user@fedora destination]$ ./filetype ./file.txt /home/user/mylink
./file.txt: regular
/home/user/mylink: regular
```

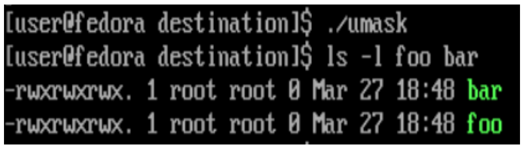
După modificarea aceasta, dacă folosim comanda de mai sus va apărea la ambele regular, deoarece la mylink se va lua fișierul de care este legat și nu el în sine.

Exercițiul 3

Modificați programul `umask` în așa fel încât să nu se mai cheme apelul de sistem `umask`; spargeți programul în două programe, `umask1` și `umask2`, care se ocupă separat de cele două fișiere. Realizați același efect folosind doar comanda bash `umask`, înaintea apelării fiecăruia.

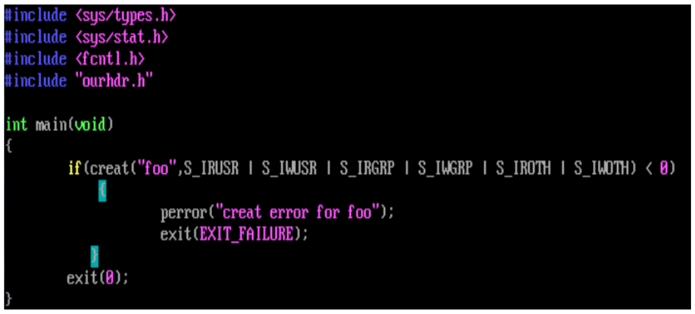


```c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include "ourhdr.h"

int main(void)
{
        struct stat stbuf;
        mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH;
        if(stat("foo",&stbuf)<0)
        {
                if(creat("foo", S_IWUSR | S_IWGRP | S_IWOTH) < 0)
                        err_sys("creat error for foo");
        }
        else
        {
                if(chmod("foo", S_IWUSR | S_IWGRP | S_IWOTH) <0)
                        err_sys("chmod error for foo");
        }
        if(stat("bar",&stbuf)< 0)
        {
                if(creat("bar",mode)< 0)
                    err_sys("creat error for bar");
        }
        else
        {
                if(chmod("bar",mode)< 0)
                    err_sys("chmod error for bar");
        }
        exit(0);
}
```

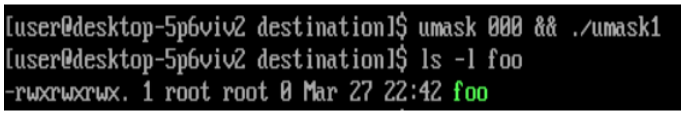


```
[user@fedora destination]$ ./umask
[user@fedora destination]$ ls -l foo bar
-rwxrwxrwx. 1 root root 0 Mar 27 18:48 bar
-rwxrwxrwx. 1 root root 0 Mar 27 18:48 foo
```

Am modificat programul umask în așa fel încât să nu se mai cheme apelul de sistem umask.

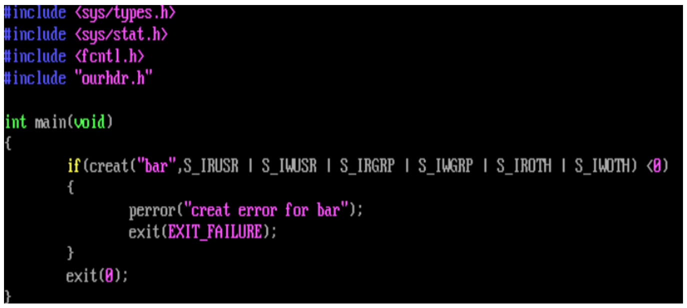


```c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include "ourhdr.h"

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

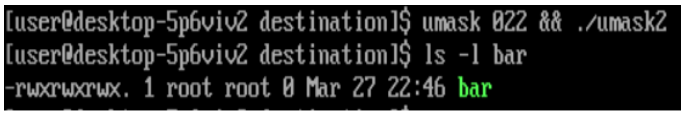


```
[user@desktop-5p6viv2 destination]$ umask 000 && ./umask1
[user@desktop-5p6viv2 destination]$ ls -l foo
-rwxrwxrwx. 1 root root 0 Mar 27 22:42 foo
```

Programul umask1.



```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)
        {
                perror("creat error for bar");
                exit(EXIT_FAILURE);
        }
        exit(0);
}
```

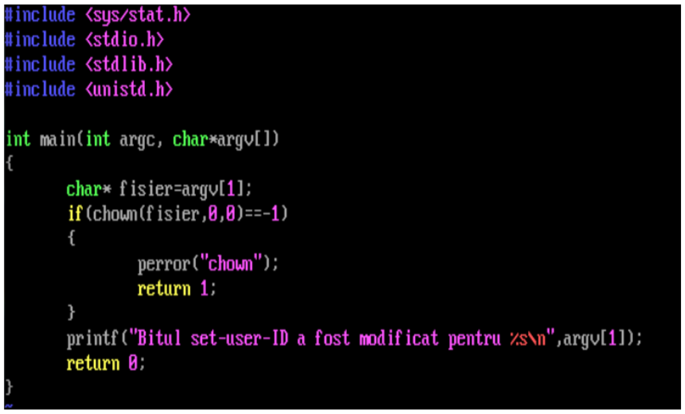


```
[user@desktop-5p6viv2 destination]$ umask 022 && ./umask2
[user@desktop-5p6viv2 destination]$ ls -l bar
-rwxrwxrwx. 1 root root 0 Mar 27 22:46 bar
```

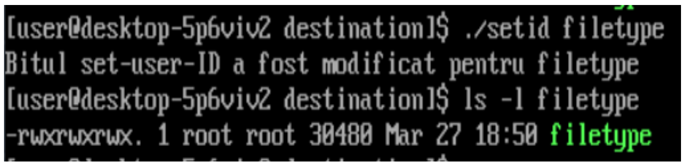
Programul umask2.

Exercițiul 4

Creați un program C care modifică bitul set-user-ID al unuia din executabilele folosite la laborator, unde numele acestuia este trimis ca parametru.



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

int main(int argc, char*argv[])
{
        char* fisier=argv[1];
        if(chown(fisier,0,0)==-1)
        {
                perror("chown");
                return 1;
        }
        printf("Bitul set-user-ID a fost modificat pentru %s\n",argv[1]);
        return 0;
}
```



```
[user@desktop-5p6viv2 destination]$ ./setid filetype
Bitul set-user-ID a fost modificat pentru filetype
[user@desktop-5p6viv2 destination]$ ls -l filetype
-rwxrwxrwx. 1 root root 30480 Mar 27 18:50 filetype
```