lab 4 – fisiere

**4.2 conceptul de file descript**

SO asigneaza fiecarui fisier deschis un descriptor/identificator, reprezentat ca un intreg.

-- cand se face un apel de sistem ce creeaza sau deschide un fisier, SO returneaza un descriptor; descriptorii sunt unici la nivel de proces :

🡪primii 3 descriptori de fisiere sunt standard

*🡪****0: standard input*** *-> ex.* ***scanf***

*🡪****1: standard output****-> ex.* ***printf***

***🡪2: standard error ->*** *ex.* ***error***

* **Apeluri de sistem ce creeaza descriptori: creat, open, fcntl, dup, pipe**

**4.2.2 creare, deschidere, inchidere fisiere**

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

**int open(const char \*pathname, int flags);** // deschidere fisier

**int open(const char \*pathname, int flags, mode\_t mode);** // deschidere fisier

**int creat(const char \*pathname, mode\_t mode);** // creare fisier

**int close(int fd);** // inchidere fisier

**4.2.2.1 open**

**fd1 = open("/usr/include/stdlib.h", O\_RDONLY);**

open -> var1. 🡪 **pathname** = numele/calea spre fisier ( absoluta sau relativa );

* **flags** : felul in care se deschide fisierul : citire,scriere…

var2**. ++ mode** = **folosit DOAR la crearea unui fisier nou si indica permisiunile de access**

flags = masca de biti ce poate fi ;

🡪**Argumente FLAGS**

** O\_RDONLY** - deschide fisierul pentru citire;

** O\_WRONLY** - deschide fisierul pentru scriere;

** O\_RDWR -** deschide fisierul pentru citire si scriere;

** O\_APPEND** - deschide fisierul pentru scriere, dar daca avem ceva scris in el, se va scrie doar dupa sfarsitul fisierului, prin adaugare 🡺 deci nu sunt suprascrise datele

** O\_CREAT** – in cazul in care fisierul nu exista, se va crea

** O\_EXCL** – daca folosim O\_CREAT si fisierul exista deja**, apelul de sistem va esua**

** O\_TRUNC** – daca sifierul are deja continut acesa se va sterge;

**🡪Argumente MODE**

* **S\_IRUSR, S\_IWUSR, S\_IXUSR** - read, write, execute pentru **owner**-ul fisierului;

🡪 **S\_IRGRP, S\_IWGRP, S\_IXGRP** - read, write, execute pentru **membrii grupului** din care face parte

owner-ul fisierului;

🡪 **S\_IROTH, S\_IWOTH, S\_IXOTH** - read, write, execute pentru restul utilizatorilor.

-- pt mode se poate folosi reprezentarea ca nr octal a permisiunilor de acces (eg. 0644) ~ rezultatul e un descriptor de fisier valid (in caz de succes) sau -1 (esec);

**4.2.2.2 creat**

**fd2 = creat("copy\_stdlib.h", S\_IRUSR | S\_IWUSR | S\_IRGRP | S\_IROTH);**

* **!!!!** Creat este similar cu a apela open cu flag-urile **: O\_WRONLY|O\_CREAT|O\_TRUNC**

creat -> analog open – pathname si flags au aceeasi semantica

**4.2.2.3 close** 🡪 int close(int fd);

**close -> inchiderea fisieului al carui descriptor a fost primit ca parametru -- 0 = succes,-1 else**

**4.2.2.4 Exemplu**

#include <stdio.h>

#include <stdlib.h>

int main()

{

int fd1 = -1;

int fd2 = -1;

**fd1 = open("/usr/include/stdlib.h", O\_RDONLY);** // flagul O\_RDONLY **deschide** fisierul pentru citire

if(fd1 == -1){

perror(" nu se poate deschide fisierul");

return 1;

}

**fd2 = creat("copy\_stdlib.h",S\_IRUSR | S\_IWUSR | S\_IRGRP | S\_IROTH);** //creaza un fisier cu drepturi de

citire/scriere pt useri si de scriere pentru grup/others

if(fd2 == -1){

perror(" nu se poate deschide fisierul");

**close(fd1);** // deja avem descriptori de fisier deschisi si trebuie inchisi inainte de a iesi din fct

return 1;

}

// instr..

**close(fd1);**

**close(fd2);**

} // -->Functia **perror** va afisa string-ul primit ca argument, urmat de motivul ultimei erori

**4.2.3 citire scriere**

**#include <unistd.h>**

**ssize\_t read(int fd, void \* buf, size\_t count);**

**ssize\_t write(int fd, const void \*buf, size\_t count);**

**4.2.3.1 read**

**ssize\_t read(int fd, void \* buf, size\_t count);**

= apel de sistem ce citeste de la pozitia curenta din fisierul deschis **fd** un nr count de octeti pe care ii scrie in bufferul **buf**;

**--> dupa citire, pozitia curenta in fisier va avansa cu numarul de octeti cititi**

--> read returneaza numarul de octeti cititi in caz de succes sau -1 altfel

**Obs: e posibil ca numarul de octeti cititi sa nu corespunda cu valoarea parametrului count ; putem incerca sa citim 8 octeti dar pana la finalul fisierului sa fie numai 3 🡺 read va returna valoarea 3**

---> **daca se returneaza 0** => **pozitia curenta in fisier era la sfarsitul fisierului inaintea executiei apelului read si prin urmare nu s=a citit nimic**

**4.2.3.1 write**

**ssize\_t write(int fd, const void \*buf, size\_t count);**

= scrie in fd un numar count de octeti din bufferul buf

🡪 dupa scriere, pozitia curenta va avanga cu un numar de octeti scrisi

🡪 **daca returneaza o val mai mica decat paramentrul count inseamna ca discul e plin si nu s-au mai putut scrie toate datele**

**4.2.3.1 continuare exemplu**

#define BUFF\_SIZE 64

Int main(){

**int err = 0;**

**unsigned char buff[BUFF\_SIZE];**

.......... // deschidere / creare fisier

**for(;;){** // bucla infinita

**size = read(fd1,buff,BUFF\_SIZE);** // citeste din buffer buff BUFF\_SIZE (64) octeti

if(size<0){

// read error 🡪 nu pot citi din fisier

perror("Could not read from input file");

err=1;

break;

}else if(size==0){

// ma aflu la sfarsitul fisierului 🡪 nu am citit nici un octet

break;

}else{

size = write(fd2,buff,size); // scrie dib buffer size

if(size <= 0){

perror("Could not write to output file"); 🡪 nu se poate efectua scrierea (disk plin)

err=2;

break;

}

} }

**4.2.4 deplasarea cursorului in fisiere**

**-- lseek - modifica pozitia cursorului**

#include <sys/types.h>

#include <unistd.h>

**off\_t lseek(int fd, off\_t offset, int whence);**

**🡪 fd –** descriptor de fisier

🡪 **offset** - deplasament / adresa de unde porneste

🡪 **wheace** - indicator ce spunde de unde se calculeaza deplasamentul

poate fi :

🡪 **SEEK\_SET** - deplasamentul se considera de la **inceputul fisierului;**

 **SEEK\_CUR** - deplasamentul se considera de la **pozitia curenta;**

 **SEEK\_END** - deplasamentul se consider a de la **finalul fisierului**

* **lseek 🡪 val returnata rep pozitia absoluta a cursorului in fisier**

**4.2.4.1 determinarea dimensiunii unui fisier**

-- ne deplasam pana la final si aflam pozitia cursorului

... int fd;

off\_t size; // depinde de tipul fisieruli -- suficient un long int de obicei

fd = open(..);

if(fd===-1){...}

**size = lseek(fd,0,SEEK\_END);** // se cere deplasarea cursorului exact la final si apelul returneaza pozitia cursorului in fisier dupa deplasare

printf("the file size is: %d\n" ,size);

....

**4.2.4.2 copierea cifrelor la finalul unui fisier**

-- citeste un fisier caracter cu caracter scriind cifreele la finalul fisierului

int fd;

off\_t size,crtPos,i;

char c=0;

..

fd=open(...,O\_RDWR);

if(fd==-1){...}

size = lseek(fd,0,SEEK\_END); // aflu dimensiunea fisierului ca sa stiu cand sa ma opresc

lseek (fd,0,SEEK\_SET);

for(i=0;i<size;i++){ // din forul asta stiu cand sa ma opresc... (ca eu tot adaug in fisier si pot intra intr-o bucla infinita)

if(read(fd,%c,1)!=1){

perror("reading error");

break;

}

if(c>='0'&&c<='9'){

crtPos = lseek(fd,0,SEEK\_CUR); //retin unde am ramas

lseek(fd,0,SEEK\_END); // ma duc la final cu cursorul

if(write(fd,&c,1)!=1){

perror("writing error");

break;

}

lseek(fd,crtPos,SEEK\_SET);

}

}

close(fd);

**4.3.1 intrebari recapitulative**

**1. ce apel de sistem foloseste intern printf?**

**r: write**

**2. apeluri de sistem folosite pt crearea unui fisier?**

🡪 creat (fd,”path..”);

🡪open(fd,”path”,O\_WRONLY|O\_CREAT|O\_TRUNC);

**3. ce returneaza apelurile de sistem read si write?**

nr de octeti cititi/scrisi in caz de succes sau -1 in caz de eroare

-- 0 pt read=> pozitia cursorului se afla la sfarsitul fisierului

-- -1 pt write => eroare -- daca se returneaza o val mai mica decat cea data inseamna ca discul era deja plin

**4. cum aflu dimensiunea unui fisier deja deschis utilizand lseek?**

--- **size=lseek(fd,0,SEEK\_END);** -- cursorul se deplaseaza de la inceputul la finalul fisierului

5.

1 int fd = open("file.txt", O\_WRONLY);

2 lseek(fd, 80, SEEK\_SET); **// plasez cursorul la 80**

3 write(fd, buff, 50); **// incepand cu octetul 80 scriu 50 de octeti (deci 20 se vor suprascrie) 🡪 dim 140**

4 close(fd);

-- file.txt=100 octeti si buff = 50 ... dim finala a fisierului?

--> r : 130 ?