Laborator 12 - Implementarea sistemelor de fișiere

Nice to read

* TLPI - Chapter 14, File Systems
* TLPI - Chapter 15, File Attributes
* TLPI - Chapter 18, Directories and Links

Resurse utile

* [How inodes work?](https://www.youtube.com/watch?v=ymYZPtrvgec)
* [What chroot is really for?](https://lwn.net/Articles/252794/)

Device Nodes

Nucleul gestionează fiecare device hardware sau virtual prin intermediul unui device driver. Un *device driver* este o porțiune de cod din nucleu care implementează o serie de operații corespunzătoare acțiunilor de I/E asociate cu un device hardware. Procesele din spațiul utilizator (userspace) interacționează cu device driver-ul prin intermediul unor fișiere speciale denumite *device nodes*. API-ul (interfața de programare) oferită de device drivere este fixată, și include următoarele operații:

* open, close
* read, write
* ioctl, mmap

Unele device-uri sunt reale (mouse, tastatură, disc), altele sunt virtuale în sensul că nu au un device hardware asociat (e.g /dev/zero, /dev/null). După modul în care se accesează datele, device-urile sunt împărțite în două categorii:

* device de tip caracter, datele sunt procesate octet cu octet. În această categorie se înscriu: tastatura, linia serială, mouse-ul.
* device de tip bloc, datele pot fi procesate la nivel de bloc (e.g hard disk).

Fișierele device node se găsesc în /dev și au asociat un identificator format din *major ID* și *minor ID*.

Majorul și minorul sunt dați de coloanele 5 și 6 din output-ul ls -l, separate prin virgulă.

Puteți vizualiza majorii folosiți în sistem din fișierul /proc/devices.

Crearea unui device node se face folosind funcția [mknod](http://man7.org/linux/man-pages/man2/mknod.2.html)

int mknod(const char \*pathname, mode\_t mode, dev\_t dev);

În general informații despre fișiere, și în particular despre device node-uri se pot afla cu funcțiile din familia[stat](http://man7.org/linux/man-pages/man2/stat.2.html).

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

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

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

Toate aceste funcții completează informațiile despre un fișier în structura struct stat, care conține următoarele câmpuri:

struct stat {

dev\_t st\_dev; */\* ID of device containing file \*/*

ino\_t st\_ino; */\* inode number \*/*

mode\_t st\_mode; */\* protection \*/*

nlink\_t st\_nlink; */\* number of hard links \*/*

uid\_t st\_uid; */\* user ID of owner \*/*

gid\_t st\_gid; */\* group ID of owner \*/*

dev\_t st\_rdev; */\* device ID (if special file) \*/*

off\_t st\_size; */\* total size, in bytes \*/*

blksize\_t st\_blksize; */\* blocksize for file system I/O \*/*

blkcnt\_t st\_blocks; */\* number of 512B blocks allocated \*/*

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 \*/*

};

Sisteme de fișiere

Un *sistem de fișiere* este o colecție organizată de fișiere și directoare. Un sistem de fișiere este creat folosind comanda [mkfs](http://linux.die.net/man/8/mkfs). Din punct de vedere funcțional sistemele de fișiere se pot împărți în:

* sisteme de fișiere pentru disc (ext2, ext3, reiserfs, fat, ntfs, etc.)
* sisteme de fișiere pentru rețea (nfs, smbfs, ncp, etc.)
* sisteme de fișiere virtuale (procfs, sysfs, sockfs, pipefs, etc.)

Tipurile de sisteme de fișiere suportate de nucleu pot fi observate în fișierul /proc/filesystems.

daniel@debian$ cat /proc/filesystems

nodev sysfs

nodev proc

nodev ramfs

ext4

fuseblk

Pentru a putea fi folosit un sistem de fișiere trebuie atașat (montat) în ierarhia de directoare din sistem. Acest lucru se realizează cu comanda [mount(8)](http://linux.die.net/man/8/mount):

**mount** -t type device **dir**

sau apelul [mount(2)](http://man7.org/linux/man-pages/man2/mount.2.html):

int mount(const char \*source, const char \*target,

const char \*filesystemtype, unsigned long mountflags,

const void \*data);

Operația inversă, demontarea sistemului de fișiere din ierarhia de directoare se face cu comanda [umount(8)](http://man7.org/linux/man-pages/man8/umount.8.html):

**umount** {**dir**|device}...

sau apelul: [umount(2)](http://man7.org/linux/man-pages/man2/umount.2.html)

int umount(const char \*target);

Directoare și link-uri

Fiecare proces are două atribute legate de directoare:

* directorul rădăcina, determină punctul de unde căile absolute sunt interpretate.
* directorul curent, determină punctul de unde căile relative sunt interpretate.

Un director este stocat în sistemul de fișiere într-un mod similar cu un fișier obișnuit. Există două lucruri diferite:

* tipul din structura inode este diferit.
* conținutul este diferit. Un director conține un vector de nume de fișiere și inode-uri.

Link-uri simbolice (soft)

Un *link simbolic* (sau soft link), este un tip special de fișier al cărui conținut reprezintă numele altui fișier. Link-urile simbolice sunt create cu comanda ln -s sau cu apelul [symlink(2)](http://linux.die.net/man/2/symlink)

int symlink(const char \*oldpath, const char \*newpath);

Ștergerea unui link simbolic se face cu comanda unlink sau cu apelul [unlink(2)](http://linux.die.net/man/2/unlink)

int unlink(const char \*pathname);

Crearea și ștergerea directoarelor

Un director poate fi creat folosind comanda mkdir sau apelul [mkdir(2))](http://man7.org/linux/man-pages/man2/mkdir.2.html)

int mkdir(const char \*pathname, mode\_t mode);

Apelul [rmdir(2)](http://man7.org/linux/man-pages/man2/rmdir.2.html) șterge directorul specificat în argumentul pathname:

int rmdir(const char \*pathname);

De asemenea, pentru a șterge un fișier sau un director gol se poate folosi funcția [remove(3)](http://linux.die.net/man/3/remove)

int [remove](http://www.opengroup.org/onlinepubs/009695399/functions/remove.html)(const char \*pathname);

Citirea directoarelor

După cum am precizat mai sus, un director conține nume de directoare sau fișiere.

Apelul [opendir(3)](http://linux.die.net/man/3/opendir) deschide un director și întoarce un handle ce poate fi folosit mai târziu pentru a referi directorul.

DIR \*opendir(const char \*name);

DIR \*fdopendir(int fd);

Apelul [readdir(3)](http://linux.die.net/man/3/readdir) citește intrări succesive dintr-un stream de directoare (DIR).

struct dirent \*readdir(DIR \*dirp);

Apelul readdir întoarce un pointer la următoarea structură struct dirent streamul referit de dir:

struct dirent {

ino\_t d\_ino; */\* inode number \*/*

off\_t d\_off; */\* offset to the next dirent \*/*

unsigned short d\_reclen; */\* length of this record \*/*

unsigned char d\_type; */\* type of file; not supported*

*by all file system types \*/*

char d\_name[256]; */\* filename \*/*

};

Directorul curent al unui proces

Directorul curent al unui proces definește punctul de start pentru formarea căilor relative referite de procesul respectiv. Un proces nou creat moștenește directorul curent de la procesul părinte.

Directorul curent al unui proces poate fi determinat folosind apelul [getcwd(3)](http://linux.die.net/man/3/getcwd):

char \*getcwd(char \*cwdbuf, size\_t size);

* cwdbuf, trebuie alocat înainte de apel astfel încât să poată stoca cel puțin size octeți.
* după apel cwdbuf va conține calea absolută a directorului curent.

Schimbarea directorului curent

Apelul [chdir(2)](http://linux.die.net/man/2/chdir) schimbă directorul curent al procesului apelant către numele absolut sau relativ primit ca argument.

int chdir(const char \*path);

Schimbarea directorului rădăcina al unui proces

Fiecare proces are un director rădăcină reprezentând punctul de unde căile absolute sunt interpretate. În mod implicit, acesta este directorul rădăcina real al sistemului de fișiere. Un proces nou moștenește directorul rădăcină de la părintele său. Există situații (e.g pentru a ascunde o parte din sistemul de fișiere) în care este util pentru un proces să-și schimbe directorul rădăcină. Acest lucru se realizează folosind apelul [chroot(2)](http://linux.die.net/man/2/chroot)

int chroot(const char \*path);

Rezolvarea unei căi

Apelul [realpath(3)](http://man7.org/linux/man-pages/man3/realpath.3.html) dereferențiază link-ul simbolic primit ca argument și rezolvă toate referințele către '/.'și '/..' pentru a produce un șir de caractere conținând calea absolută.

char \*realpath(const char \*path, char \*resolved\_path);

dirname și basename

Apelurile [dirname(3)](http://linux.die.net/man/3/dirname) și [basename(3)](http://linux.die.net/man/3/basename) împart un șir de caractere reprezentând o cale în partea de director și partea de fișier.

char \*dirname(char \*path);

char \*basename(char \*path);

De exemplu:

path **dirname** **basename**

"/usr/lib" "/usr" "lib"

"/usr/" "/" "usr"

"usr" "." "usr"

"/" "/" "/"

"." "." "."

".." "." ".."

Exerciții

Exercițiu bonus - Completare feedback (1p)

Vă invităm să **evaluați activitatea echipei de SO** și să precizați punctele tari și punctele slabe și sugestiile voastre de îmbunătățire a materiei. Feedback-ul vostru este foarte important pentru noi să creștem calitatea materiei în anii următori și să îmbunătățim materiile pe care le veți face în continuare.

Găsiți **formularul de feedback** în partea dreaptă a paginii principale de SO de pe cs.curs.pub.ro într-un frame numit *“FEEDBACK”* (sau [click aici](http://cs.curs.pub.ro/2015/blocks/simplehtml/view.php?courseid=115&blockid=1468)). Trebuie să fiți inrolați la cursul de SO, altfel veți primi o eroare de acces.

Vă mulțumim!

Exercițiul 0 - Joc interactiv (2p)

* Detalii desfășurare [joc](http://ocw.cs.pub.ro/courses/so/meta/notare#joc_interactiv).

Linux (8p)

Pentru rezolvarea laboratorului descărcați arhiva de [lab12-tasks.zip](http://elf.cs.pub.ro/so/res/laboratoare/lab12-tasks.zip). Codul va fi scris în fișierulmini.c din directorul 1-mini/. Pentru fiecare exercițiu decomentați linia TODO corespunzătoare.

Exercițiul 1 (1p)

Folosiți comanda ls -l /dev și precizați două device node-uri de tip caracter și două device node-uri de tip bloc. Ce major și minor au?

Exercițiul 2 (1p)

Implementați comanda list <device\_node>, ce va primi ca argument un device node și va afișa pentru acesta tipul (c/b), identificatorii major, respectiv minor. Folosiți funcția [stat(2)](http://man7.org/linux/man-pages/man2/stat.2.html) pentru a obține o structură de tipulstruct stat din care veți extrage tipul device-ului (st\_mode) (hint: [S\_ISCHR, S\_ISBLK](http://www.gnu.org/software/libc/manual/html_node/Testing-File-Type.html#Testing-File-Type)) apoi din câmpulst\_rdev extrageți [major](http://man7.org/linux/man-pages/man3/makedev.3.html) și [minor](http://man7.org/linux/man-pages/man3/makedev.3.html). Nu uitați să decomentați linia marcată cu *#define TODO2*

Exercițiul 3 (1p)

Creați punctul de montare /mnt/my. Ca root, rulați comanda:

**mkdir** /mnt/my

Parcurgeți paginile de manual ale funcțiilor [mount](http://man7.org/linux/man-pages/man2/mount.2.html) și [umount](http://man7.org/linux/man-pages/man2/umount.2.html).

Folosiți comenzile mount și umount din executabilul mini pentru a monta discul /dev/sda1 în punctul de montare/mnt/my. Citiți secțiunea marcată cu TODO din fișierul mini.c. Pentru argumentul 4 și argumentul 5 al funcțieimount folosiți, respectiv, valorile 0 și NULL.

Testare: Rulați, ca root, comanda:

./mini

și apoi rulați comanda de montare în cadrul acestui shell:

**mount** /dev/sda1 /mnt/my ext4

Într-o altă consolă, într-un shell obișnuit, verificați rezultatele folosind comanda

**cat** /proc/mounts

Pentru demontare rulați comanda:

**umount** /mnt/my

Exercițiul 4 (1p)

Adăugați suport pentru comenzile symlink și unlink în programul mini. Urmăriți *TODO4*.

Pentru testare folosiți, în shell-ul aferent comenzii ./mini, comanda:

symlink /etc/**passwd** local-passwd

Ca să verificați, într-o altă consolă, în același director cu cel în care ați rulat comanda ./mini, folosiți

**ls** -l

Pentru a șterge symlink-ul folosiți comanda

**unlink** local-passwd

Pentru validare rulați din nou comanda

**ls** -l

Exercițiul 5 (1p)

Adăugați suport pentru comenzile mkdir și rmdir în programul mini. Urmăriți *TODO5*.

Ca al doilea argument pentru funcția mkdir folosiți (mode\_t) 0755.

Exercițiul 6 (2p)

Adăugați suport pentru comanda ls dirname/ în programul mini. Aceasta va trebui să afișeze recursiv toate directoarele și fișierele începând cu directorul dat ca parametru (puteți parcurge recursiv în adâncime arborele de fișiere). Urmăriți *TODO6*și demo-ul 5 de la [curs](http://ocw.cs.pub.ro/courses/so/cursuri/curs-12)

Exercițiul 7 (1p)

Adăugați suport pentru comenzile pwd și chdir în programul mini. Urmăriți *TODO7*.