Laborator 12 - Implementarea sistemelor de fișiere

Materiale Ajutătoare

- lab12-slides.pdf [http://elf.cs.pub.ro/so/res/laboratoare/lab12-slides.pdf]

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/]

Link-uri către secțiuni utile

Linux

- Device nodes
- Sisteme de fisiere
- Directoare și linkuri
- Link-uri simbolice (soft)
- Crearea și ștergerea directoarelor
- Citirea directoarelor
- Directorul curent al unui proces
- Schimbarea directorului curent
- Schimbarea directorului rădăcină al unui proces
- Rezolvarea unei căi
- dirname și basename

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(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` din 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

Completare feedback

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 acs.cs.pub.ro [https://acs.curs.pub.ro] într-un frame numit *"FEEDBACK"*. Trebuie să fiți înrolați la cursul de SO și să intrati pe pagina asociată seriei voastre (nu pe metaserie), altfel veți primi o eroare de acces.

Vă mulțumim!

Exercițiul 0 - Joc interactiv

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

Linux

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 fișierul `mini.c` din directorul `1-mini/`. Pentru fiecare exercițiu decomentați linia TODO corespunzătoare.

Exercițiul 1

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

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 tipul `struct 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 câmpul `st_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

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 funcției `mount` 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 ext3
```

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

```
cat /proc/mounts
```

Pentru demontare rulați comanda:

```
umount /mnt/my
```

Exercițiul 4

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

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

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

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

Soluții

Soluții exerciții laborator 12 [http://elf.cs.pub.ro/so/res/laboratoare/lab12-sol.zip]