# Laboratorul 04. Rootfs

## Root file system

Pentru ca sistemul să fie inițializat corect după pornirea kernel-ului, este necesar ca toate script-urile și executabilele necesare pentru a porni daemon-ul de inițializare, *init*, și restul proceselor *user-space*, să existe în anumite locații în sistemul de fișiere. Acest sistem minimal de fișiere necesar la inițializare poartă numele de **root file system** sau **rootfs**. În cazul sistemelor Unix, *rootfs*-ul are ca rădăcină directorul / și înglobează o serie de directoare ce contin restul de fișiere necesare. De obicei, în directorul / nu se află niciun fișier, doar subdirectoare. Aceste subdirectoare sunt organizate în funcție de fișierele pe care le conțin:

- `/bin` - conține programe și comenzi necesare la inițializare ce pot fi folosite apoi de către un utilizator neprivilegiat în timpul unei sesiuni
- `/sbin` - conține programe asemănătoare cu cele din /bin ca utilitate, dar care pot fi utilizate în general doar de utilizatorul privilegiat *root*
- `/etc` - conține fișierele de configurare specifice sistemului
- `/lib` - conține bibliotecile folosite de programele din *rootfs* (cele din `/bin` și `/sbin`)
- `/dev` - conține referințe la toate dispozitivele periferice; aceste referințe sunt în general fișiere speciale
- `/boot` - conține fișierele necesare bootării și imaginea kernel-ului; este posibil ca acestea să fie păstrate pe un sistem de fișiere separat de *rootfs*
- `/tmp` - conține fișiere temporare și este de obicei curățat la repornirea sistemului
- `/opt` - conține în principal aplicațiile third-party
- `/proc`, `/usr`, `/mnt`, `/var`, `/home` - sunt directoare care, în general, reprezintă puncte în care se pot monta alte sisteme de fișiere pentru a stoca log-uri, biblioteci și alte aplicații și programe folosite de către utilizatori.

Standardul Unix recomandă ca *rootfs*-ul să aibe dimensiuni relativ mici, deoarece orice problemă sau corupere a acestuia poate împiedica inițializarea corectă a sistemului. Dacă, totuși, rootfs-ul devine corupt, există modalități prin care poate fi remediat. O soluție des folosită este de a monta acest sistem de fișiere într-un alt sistem, funcțional, pentru a putea fi explorat și verificat.

În cazul sistemelor embedded, de multe ori spațiul de stocare pe care îl avem la dispoziție este deja limitat: < 32MB. De cele mai multe ori, o bună parte din acest spațiu este ocupat de imaginea kernel-ului. De aceea, în majoritatea cazurilor, *rootfs*-ul pentru aceste sisteme este fie compus doar din minimul de programe necesar pentru inițializare, fie este stocat pe un server accesibil prin rețea și montat de către kernel la inițializare, folosind protocolul NFS.

## NFS

**N**etwork **F**ile **S**ystem este un protocol dezvoltat de Sun Microsystems în 1984, ce permite unui client să acceseze fișiere aflate pe o altă mașină din rețea. Deoarece protocolul poate rula peste TCP, acesta poate fi folosit fără probleme atât în retele mici (LAN-uri) cât și în rețele extinse (WAN).

În sistemele Unix, există o serie de configurări ce trebuie făcute, atât pe server, cât și pe client, pentru a putea folosi NFS:

- server-ul trebuie să ruleze un daemon NFS ce pune fișierele la dispoziția clienților
- tot server-ul stabilește și ce va fi exportat; directoarele și fișierele exportate sunt menținute într-o listă și se folosește un fișier de configurare pentru a stabili proprietățile acestora (drepturi asupra fișierelor, vizibilitate etc.)

- este necesară configurarea mecanismelor ce asigură securitatea în rețea (în general firewall-uri), pentru a permite accesul clienților pe server
- client-ul trebuie configurat pentru a accesa datele folosind utilitare precum `mount`; dacă totul este configurat corect, client-ul va putea naviga și folosi sistemul de fișiere de pe server ca pe orice sistem de fișiere local.

- Daemon-ul NFS se numește `nfsd`.
- Fișierul de configurare pentru export-uri permanente este `/etc/exports`. Utilitarul folosit pentru export-uri temporare se numeste `exportfs`.
- Clientul folosește fișierele exportate cu ajutorul lui `mount`.
- De cele mai multe ori montarea sistemelor de fișiere de către client este automatizată folosind `/etc/fstab`

# Crearea unui rootfs. Utilitare

Uneori vom fi în situația în care nu avem acces la un server și imaginea de care dispunem pentru sistemul nostru nu este satisfăcătoare. În aceste condiții va trebui să ne creăm propriul *rootfs* pe care să îl scriem în memoria sistemului embedded.

## Formatul imaginii RaspberryPi

Imaginea recomandată pentru RaspberryPi, Raspbian, conține două partiții: o partiție FAT folosită pentru boot și o partiție `ext4` pentru *rootfs*. Fiecare dintre aceste partiții începe de la un anumit offset în cadrul imaginii. Atunci când dorim să creăm o imagine nouă pentru RaspberryPi, trebuie să ne asigurăm că respectăm aceste partiții și formatele lor. Pașii pe care trebuie să îi urmăm sunt:

- Stabilirea dimensiunii imaginii și inițializarea acesteia cu zero-uri
- Crearea tabelei de partiții și a celor două partiții necesare
- Formatarea partițiilor cu formatul corespunzător
- Pentru popularea rootfs-ului, putem fie să montam partiția și să copiem manual directoarele și fișierele, fie putem să copiem o partiție întreagă de pe o altă imagine

Pentru fiecare dintre acești pași există utilitare ce ne ajută să realizăm operațiile necesare.

## dd

Pentru copierea, și eventual convertirea, unui fișier, la nivel de byte, se poate folosit utilitarul *dd*.. Folosind *dd*, putem de asemenea genera fișiere de anumite dimensiuni și le putem stabili conținutul. În cazul nostru, *dd* este util pentru a inițializa o imagine și pentru a copia în acea imagine conținutul care ne interesează. Dintre parametrii lui *dd*, cei mai des utilizați sunt:

- `if` - fișierul de intrare; dacă nu se specifică acest parametru, se va citi de la *standard input*
- `of` - fișierul de ieșire; ca și la `if`, dacă nu este specificat, se scrie la *standard output*
- `count` - numărul de blocuri de input ce vor fi copiate
- `bs` - numărul de bytes dintr-un bloc

Un exemplu de utilizare a lui `dd` pentru a inițializa cu zerouri un fișier de o dimensiune exactă:

```
$ dd if=/dev/zero of=<file> bs=1k count=4096
```

Observați valorile lui `count` si `bs`. Se vor copia în total 4096 de blocuri de câte 1KB fiecare, rezultând o dimensiune de 4096KB. Fișierul de intrare `/dev/zero` este un fișier special din care se pot citi oricâte

caractere ASCII NUL (0x00).

## fdisk

Pentru manipularea tabelei de partiții (crearea sau ștergerea partițiilor) se poate folosi utilitarul *fdisk*. Doar tabele de partiții DOS și disklabel-uri SUN și GNU sunt recunoscute și pot fi manipulate cu *fdisk*. Utilitarul primește ca parametru device-ul a cărui tabelă de partiții dorim să o modificăm. Un exemplu de apel este:

```
$ fdisk <device>
```

Rezultatul acestei comenzi este un prompt nou în care putem folosi tasta `m` pentru a afișa un meniu cu opțiunile disponibile. Opțiuni utile pentru crearea și ștergerea de partiții sunt:

- `p` - afișează tabela curentă
- `d` - șterge o partiție
- `n` - crează o partiție nouă
- `w` - salvează tabela și iese.

Trebuie reținut că în mod normal *fdisk* operează cu sectoare și nu cu octeți. Atunci când adăugam o partiție, dacă specificăm doar dimensiunea ei și omitem unitatea de măsură (K, M, G etc.), fdisk va considera numărul specificat în sectoare.

- Dimensiunea standard a unui sector de disc este `512 bytes`. Aceasta poate fi schimbată folosind opțiunea `-b`.
- Există două tipuri de partiții: `primary` și `extended`. Un device nu poate avea mai mult de 4 partiții primare.

## mkfs

Crearea unei partiții nu este suficient pentru a putea stoca fișiere. Avem nevoie ca partiția să conțină și un sistem de fișiere. Utilitarul folosit pentru a crea (formata) sisteme de fișiere Linux este *mkfs*. Parametrii pe care îi primește *mkfs* sunt device-ul ce trebuie formatat și tipul sistemului de fișiere dorit, specificat cu parametrul `-t`.

Comanda

```
$ mkfs -t ext2 /dev/fd0
```

va crea pe device-ul *fd0* un sistem de fișiere *ext2*. În Linux, utilitarul *mkfs* este împărțit în câte un executabil pentru fiecare tip de sistem de fișiere suportat. Pentru a le vedea pe cele disponibile, deschideți un terminal, scrieți `mkfs` și apasați tab.

## mount

În sistemele Unix, sistemul de fișiere este unul arborescent unde directoarele pot fi considerate noduri, iar fișierele frunze. Utilitarul *mount* ne permite să atașăm unui arbore existent un alt subarbore (sistem de fișiere). Apelată fără niciun parametru, această comandă va afișa toate device-urile montate, locația în care sunt montate și opțiunile cu care au fost montate.

Formatul general al comenzii este `mount -t <type> <device> <mount path>`, unde <*type*> este același ca la *mkfs*.

# Exerciții

1. Dorim ca partiția de *rootfs* a RaspberryPi-urilor noastre să fie în formatul *ext2*. Pentru aceasta, ne vom crea propria imagine.

- Partiția de *boot* a RaspberryPi are dimensiunea de 100MB. Pentru partiția de *rootfs* trebuie să creați o imagine de 6GB.

- Creați un fișier de dimensiunea necesară pentru imaginea completă.
- Creați tabela de partiții și partițiile necesare.
- Creați sistemul de fișiere asociat fiecărei partiții.

- Imaginea hard disk-ului (*rootfs*-ul) modificată pentru a rula pe platforma Versatile PB este disponibilă aici [https://drive.google.com/open?id=0B0lgiPZNMMyvOTFMakFuY1N2Q1E]. Aceasta va fi folosita ca referinta.
- Deoarece *mkfs* nu poate lucra cu porțiuni de imagine, va trebui să folosiți *dd* pentru a face fișiere separate pentru fiecare partiție, pe care să le copiați apoi în imagine la offset-urile corecte. Copiați-le în ordinea *boot*, *rootfs*. Dupa ce ati copiat *boot*, puteti observa ca fisierul a fost trunchiat, dar isi va reveni dupa copierea *rootfs*.

- Folosiți *dd* pentru a crea un fișier de o anumită mărime, inițializat cu 0.
- Folosiți *fdisk* pentru a crea o tabelă de partiții.
- Puteti folosi informatiile din tabela de partitii pentru a gasi numarul de blocuri ai fiecarai partitii, precum si offsetul lor in cadrul diskului virtual.
- Studiați opțiunea *seek* a comenzii *dd*.
- Atentie! cand copiati folosind *dd* intre fisiere, recomandam sa nu folositi paremetrul *count*.

2. Deoarece ne-am propus să modificăm doar formatul partiției, nu și conținutul *rootfs*-ului, pe acesta îl vom copia din imaginea inițială. Este important ca ownerii și permisiunile fișierelor originale să fie păstrate și în noua imagine.

Folosind *file* puteți afla rapid informații extrase din header-ul unui fișier: tabela de partiții a unei imagini în cazul nostru.

- Opțiunea `-o offset=<offset>` a comenzii *mount* permite montarea un sistem de fișiere care începe la un anumit offset pe *device*. *mount* așteaptă valoarea <*offset*> în **bytes**.
- Studiați opțiunea `-a` a comenzii *cp*.
- Doar utilizatorul *root* are dreptul de a crea fișiere cu alt *owner* decât *self*. Utilitarul *sudo* vă permite să rulați comenzi ca utilizatorul *root*. With great power comes great responsibility.

3. Acum că imaginea noastră este gata, vrem să vedem că funcționează. Bootați sistemul in qemu folosind aceasta imagine de kernel si rootfs-ul nou creat.

- Pentru qemu va trebui sa folositi board-ul versatilepb si CPU arm1176 la fel ca in laboratorul 3.
- Kernel-ul modificat pentru a rula pe platforma Versatile PB este disponibil aici [https://drive.google.com/open?id=0B0lgiPZNMMyvaEtfN3V4VVBxRjg].
- Folosiți string-ul `root=/dev/sda2` pentru linia de comandă a kernel-ului.
- La prima pornire distribuția Raspbian va rula un utilitar de configurare. Opțiunile implicite sunt suficiente pentru a porni sistemul.

## Resurse

- Soluție laborator (disponibilă începând cu 05.11.2015)