

Capitol 12: Implementarea sistemelor de fișiere

Sumar curs anterior (Dispozitive de intrare/ieșire)

placa de rețea este folosită pentru transferul datelor de la mai multe aplicații (de la mai mulți socketi)

placa de rețea și fiecare socket au buffere de transmitere (TX) și recepție (RX)

primirea pachetelor generează o întrerupere, procesorul apoi preia pachetele și le pune în memorie, apoi sunt transmise socket-ului

fiecare socket TCP are asociat un 5-tuplu: adresă sursă, port sursă, adresă destinație, port destinație, protocol; pe baza pachetului sursă al pachetului primit se demultiplexează socketul care trebuie să primească pachetul

în cazul UDP un pachet transferat din user space duce la transferul unui pachet (datagram)

în cazul TCP se copiază într-un buffer al nucleului de unde este transferat la momentul potrivit

un apel de transmitere (TX, send) se blochează dacă bufferul nucleului este plin

un apel de primire (RX, receive) se blochează dacă bufferul nucleului este gol

pachetele sunt transmise la dimensiunea frame-ului plăcii de rețea (MTU: Maximum Transmission Unit), 1500 octeți

pentru a mări viteza există suport pentru segmentation offload în care un segment mai mare este compartimentat în frame-uri

pentru prelucrarea datelor pentru mai multe conexiuni avem varianta asincronă sau varianta multithread/multiproces, varianta asincronă este eficientă dar e dificil de programat, este nevoie de menținerea unei stări a fiecărei conexiuni

Stocarea datelor și sistemul de fișiere

aplicațiile și sistemul de operare folosesc memoria pentru reținerea codului și datelor, procesorul pentru prelucrare și I/O pentru comunicare cu exteriorul

pornirea unei aplicații sau a sistemului de operare necesită un suport persistent (pentru reținerea executabilelor, bibliotecilor, imagini de kernel)

aplicațiile folosesc date (configurare, multimedia, baze de date) care necesită un suport persistent (non-volatile)

dispozitivele de stocare (storage) sunt dispozitive I/O care asigură suport persistent, în general discuri (CD/DVD/Blu-ray, HDD, SSD, USB flash drive), NVRAM; cartelele perforate (punch cards) sau bandă magnetică erau forme de stocare

datele sunt organizate în fișiere pe dispozitivele de stocare pentru a fi accesate de aplicații

fișierele sunt forme de compartimentare a datelor; uzual fișierele sunt structurate ierarhic

modul de organizare a datelor pe un suport persistent reprezintă sistemul de fișiere

sistemul de fișiere expune o interfață aplicațiilor

interfața este în linii mari generică, aceleași tipuri de operații sunt expuse de toate sistemele de fișiere, indiferent de implementarea acestora
este responsabilitatea sistemului de operare să ofere o interfață comună care să ascundă complexitatea și diversitatea sistemelor de fișiere
+ diagramă cu discuri, sisteme de fișiere, sistem de operare, interfață comună
un sistem de fișiere urmărește să asigure o structură ierarhică (pentru organizarea informațiilor în directoare, subdirectoare, fișiere), performanță (să fie rapid) și scalabilitate (să putem crea multe fișiere, fișiere mari)

Reminder: Interfața sistemului de fișiere

interfața sistemului de fișiere definește operațiile și structurile pentru lucrul cu fișiere
interfața cuprinde descriptori de fișiere și operațiile: deschidere, citire, scriere, căutare, închidere, creare, ștergere, mapare
fișierele sunt identificate de nume, un fișier deschis e identificat de un descriptor de fișier
deschiderea unui fișier oferă un descriptor de fișier procesului apelant
un descriptor de fișier este un index în tabela de descriptori de fișiere a unui proces care referă o structură de fișier deschis: permisiuni de deschidere, cursor de fișier, pointer la structura de fișier de pe disc
structura de fișier de disc (numită și FCB, File Control Block, inode pe Unix) este copiată în memorie de pe disc
+ *diagramă cu FCB pe disc, FCB în memorie, structură de fișier deschis, tabelă de descriptori de fișiere, descriptor de fișier*
structura de fișier deschis dispare din memorie când dispar toate referințele către aceasta (când se închide un fișier)
FCB dispare din memorie când nu mai sunt fișiere deschise care o referă
FCB dispare de pe disc la operații de ștergere (rm, del)

File Control Block (FCB)

FCB este metadata unui fișier: informații despre fișier
este o structură persistentă care rezidă pe suportul persistent; este copiată în memorie atunci când este deschis un fișier și când se creează o structură de fișier deschis care o referă
se numește inode sau i-node (index node) pe sistemele de fișiere specifice sistemelor de operare Unix (ext4, HFS, AppleFS)
FCB conține:
* identificator (inode number, ino)
* controlul accesului (ownership, permissions)
* timpi de acces (timestamps)
* tip de fișiere
* dimensiunea datelor
* pointeri către date
+ folosire comandă stat pentru a afișa metadatele unui fișier
pointerii sunt către blocuri de date

există pointeri către blocuri de date care conțin pointeri către blocuri de date (indirectare)
există pointeri către blocuri de date care conțin pointeri către blocuri de date care conțin
pointeri către blocuri de date (indirectare dublă)
există pointeri către ... (indirectare triplă)

Numele unui fișier. Dentry-uri. (Hard) link-uri

În general, un FCB nu conține numele unui fișier
numele unui fișier este reținut într-o structură separată numită dentry (directory entry)
un dentry conține referință către inode (inode number) și numele fișierului
un dentry se mai numește un link (sau un hard link)
pot exista mai multe dentry-uri care referă același inode, adică mai multe hard link-uri, util
pentru a plasa un fișier în mai multe locuri
un FCB conține și numărul de link-uri
+ demo cu folosirea `ln` pentru crearea de link-uri hard și `ls -li` pentru afișarea de informații
un fișier este șters când nu mai are link-uri
+ demo cu recuperare de fișier șters din `/proc`
comanda `rm`, apelul `remove()` nu șterg un fișier ci șterg un link (un dentry)
apelul de sistem aferent este `unlink()/unlinkat()`
+ demo cu strace -e `unlinkat` `rm a.txt`

Directoare

dentry-urile sunt reținute în blocurile de date ale directoarelor
un director este un fișier ale cărui blocuri de date conțin dentry-uri; pot fi dentry-uri de fișiere
sau de subdirectoare
tipul unui fișier poate fi: obișnuit (regular) sau director (directory) sau alte tipuri
comanda `ln` creează un dentry nou într-un bloc de date al unui director, dentry ce referă un
inode/FCB existent
comanda `rm/unlink` șterge/invalidează un dentry într-un bloc de date al unui director
dacă un FCB nu mai are un dentry care îl referă este șters împreună cu blocurile de date ale
FCB-ului
există un director rădăcină, acesta este marcat corespunzător în sistemul de fișiere
ierarhia sistemului de fișiere este construită din parcurgerea dentry-urilor din directorul
rădăcină, dentry-uri referă alte FCB-uri de tip director, care conțin dentry-uri către FCB-uri de
tip fișier și director și așa mai departe
un director trebuie să permită și urcatul în ierarhia, cu ajutorul comenzii `cd ..`
un director conține un dentry cu numele `..`, dentry ce referă FCB-ul directorul părinte
de asemenea, un director conține un dentry cu numele `.`, auto-referențiere, de unde `./a.out`
sau `cp path/to/file .`
de aceea, un director fără subdirectoare are două link-uri: dentry-ul din directorul părinte cu
numele directorului și `.` (auto-referința)
pe cazul general, un director cu N subdirectoare are $N+2$ link-uri: dentry-ul din directorul
părinte cu numele directorului și `.` (auto-referința) și intrările `..` din fiecare subdirector
+ demo cu directoare și număr de link-uri de directoare

dimensiunea unui director (numărul de blocuri ocupate) este proporțional cu numărul de intrări din director (număr de dentry-uri): cu cât mai multe fișiere/directoare conține, cu atât dimensiunea unui director (numărul de blocuri ocupate) este mai mare
+ demo cu dimensiunea unui director

Alte tipuri de intrări

fișierele (indicate de FCB) pot fi fișiere obișnuite (regular files) sau directoare sau alte tipuri de fișiere

symbolic link-urile (symlink) sunt FCB-uri al căror conținut este o cale, un șir; șirul este interpretat și rezultă un dentry și FCB-ul corespunzător

un symlink poate fi "dangling" dacă șirul nu este o cale validă (nu se rezolvă la dentry și FCB)

comanda readlink rezolvă șirul de tip cale a unui FCB de tipul symlink

un symlink este un inode, un hard link este un dentry

putem crea symlink-uri la intrări din alte partiții (sisteme de fișiere montate), dar nu hard link-uri; două sisteme de fișiere diferite au fișiere diferite cu același inode, nu ar funcționa un hard link între sisteme de fișiere, nu ar face diferența

+ demo cu stat pe un symbolic link

alte intrări în sistemul de fișiere sunt socket-uri UNIX, named pipes (FIFO), char device, block device

aceste intrări au FCB dar nu au date; sunt doar interfețe de comunicare inter-proces (socket-uri UNIX, named pipes) sau de interacțiune cu resurse (hardware) expuse de sistemul de operare (char device, block device)

Gestiunea spațiului

când creăm un hard link (dentry) se ocupă un slot dintr-un bloc de date al unui director ocuparea înseamnă validarea/activarea acelei intrări; putem spune că se alocă un bloc nou atunci când nu mai sunt slot-uri în blocurile curente

ștergerea unui link înseamnă invalidarea/dezactivarea acelei intrări (uzual se pune -1 sau 0 în câmpul ino al inode-ului)

în mod similar există o zonă dedicată pentru inode-uri; în acea zonă se validează/activează sau invalidează/dezactivează inode-uri în momentul creării sau ștergerii unui inode

un bloc de date este valid dacă este referit de un inode (și unul singur)

atunci când se scrie dincolo de limita curentă a fișierului se alocă un bloc de date nou, adică se referă acel bloc în cadrul inode-ului; când se trunchiază un fișier se "dezalocă" acel block, dispare referința din inode

Disponerea informațiilor pe spațiul de stocare

un sistem de fișiere ocupă spațiu pe un disc în urma formatării: secțiuni din partiție conțin informațiile de stocare

sistemul de fișiere cuprinde un superbloc: acesta conține informații despre celelalte secțiuni (metadate despre metadate)

în mod uzual există o zonă care reține informații despre inode-urile valide/activate/ocupate (inode map) și o zonă care reține informații despre blocurile valide/activate/ocupate (data map); aceste zone sunt uzual bitmap-uri: bit 0 înseamnă inode sau bloc liber, bit 1 înseamnă inode sau bloc ocupat

o zonă dedicată reține inode-urile și altă zonă reține blocurile
inode-urile au pointeri (numere de blocuri) către blocurile corespunzătoare
inode-urile/blocurile valide sunt marcate astfel în bitmap-ul corespunzător

+ diagramă cu dispunerea informațiilor pe spațiul de stocare

când se creează un fișier, se găsește un loc liber (bitul 0 în inode map) se marchează în inode bitmap bitul 1 și se completează inode-ul corespunzător în zona de inode-uri

când se eliberează un fișier, se marchează bitul 0 în inode map

de obicei la eliberare nu se completează cu zero inode-ul eliberat sau blocurile eliberate;

ceea ce înseamnă că se pot recupera date la nevoie sau se pot exfiltra date mai vechi de un atacator care are acces la disc

Operații cu sistemul de fișiere

un sistem de fișiere se află pe o partiție, în urma operației de formatare

formatarea creează superblocul și celelalte zone, creează inode-ul rădăcină

formatarea "raw" se asigură de zeroizarea partiției

un sistem de fișiere este montat pentru a fi folosit; montarea înseamnă legarea inode-ului rădăcină la o cale din sistemul de fișiere

sistemul de fișiere rădăcină este montat în /, alte sisteme de fișiere sunt montate în căi din sistemul de fișiere rădăcină

demontarea înseamnă dezactivarea căii unde a fost montat, sistemul de fișiere devine inaccesibil

în cazul unei întreruperi bruște este posibil să existe date inconsecvente: marcaje în data map, dar pointerii încă prezenți în blocul de date al fișierului, inode-uri marcate ca șterse dar intrări în valide de tip dentry; o astfel de soluție este rezolvată prin filesystem check (fsck)
pentru acces mai rapid la informații, este util ca blocurile de date să fie într-o zonă apropiată; operația se cheamă defragmentare

Sumar

aplicațiile au nevoie de persistență pentru executabile și pentru datele folosite

spațiul de stocare persistent (discuri, NVRAM) este formatat cu un sistem de fișiere

sistemul de operare asigură interfață generică peste mai multe sisteme de fișiere, organizare ierarhică cu fișiere și directoare

interfața de sistem de fișiere cuprinde operații și descriptori de fișiere

un descriptor de fișier este un index în tabela de descriptori de fișiere a procesului, intrare care referă o structură de fișier deschis, care referă FCB în memorie

FCB (file control block) este metadata unui fișier și conține pointeri către datele fișierului

numele fișierului nu se găsește în FCB ci într-o structură numită dentry; pot exista mai multe dentry-uri (numite hard link-uri) la un fișier

un dentry conține numele fișierului și indexul inode-ului

un director este un fișier ale cărui blocuri de date conțin dentry-uri, adică intrările în acel director

un director are o referință .. către directorul părinte, și o referință . către sine

un director are N+2 hard link-uri/nume: numele său (în dentry-ul părintelui), referința către sine și referințele .. ale celor N subdirectoare

alte tipuri de intrări în sistemul de fișiere sunt symbolic link-urile, socketii UNIX, FIFO-uri, char devices, block devices

symbolic link-urile conțin o cale, fiecare symbolic link este un inode; fiecare hard link este un dentry

pe disc datele sunt ținute în blocuri de date într-o zonă de blocuri, inode-urile într-o zonă de inode-uri

o zonă numită inode map și alta numită data map rețin ce blocuri și ce inode-uri sunt ocupate

superblocul este o zonă care definește unde se găsesc zonele unui sistem de fișiere formatat

pentru folosirea unui sistem de fișiere, se formează o partiție și apoi se montează într-un punct de montare (un director existent)

sistemul de fișiere rădăcină e montat înaintea altor sisteme de fișiere în /