Fák a Reiser4 fájlrendszerben (1. rész)

Az új ReiserFS felépítése – fák, csomópontok, kulcsok és cikkelyek.

Az adatszervezés egyik módja az, ha fákban helyezzük el őket. Ha egy számítógépen adatokat rendezünk, rendszerint úgynevezett halmokban, vagyis csomópontokban helyezzük el őket, amelyek mindegyike tartalmazza a többi halom nevét (vagyis mutatóikat). A csomópontok némelyike mutatókkal rendelkezik, ezeket végigböngészve általában azt találjuk, hogy más hasonló csomópontokra mutatnak.
Minket elsősorban a szervezési rész érdekel, így a dolgokra akkor bukkanhatunk rá, ha keressük őket. A fa egy szervezési módszer, amely ennek megfelelő tulajdonságokkal rendelkezik. Ennek következtében a fát a következők alapján határozzuk meg:

1. A fa csomópontok sokaságából áll, melyeket a gyökércsomópont köt össze; és nulla vagy több csomóponttal rendelkezik, amelyeket részfáknak hívunk.
2. A részfák mindegyike önállóan is fát alkot.
3. A fa egyetlen pontja sem mutat a gyökércsomópontra, és a csomópontból minden egyes nem gyökércsomópontra egyetlen mutató irányul.
4. A gyökércsomópont az összes részfájára, vagyis a részfákhoz tartozó gyökércsomópontokra rendelkezik mutatóval.

Az egyetlen pontból álló, mutatók nélküli csomópontot is fának nevezzük, mivel az egy gyökércsomópont. Hasonlóképpen fának nevezzük az egymásból egyenesen kiinduló, elágazások nélküli csomópontokat is.

A meghatározás sarokpontjai

Érdekes dolog azon vitatkozni, hogy vajon a *véges* is a fák meghatározásai közé tartozik-e. Fákat többféle módon határozhatunk meg, és a legjobb meghatározás mindig a célfeladattól függ. ***Donald Knuth*** professzor is többféle meghatározást használ a fákra. Elsődleges meghatározásaként egy olyan fát mutat be, amelynek egyáltalán nincsenek mutatói, se élei, se összekötővonalai, se egyebei – egyszerűen csak csomópontokból áll. Knuth a fát véges számú csomópontok halmazaként határozza meg. Az Interneten a végtelen fákról is lehet találni írásokat. Úgy gondolom, sokkal megfelelőbb a végest mint tulajdonságot használni, semmint beleerőszakolni a meghatározásba – bár kutatásaimban csak véges fákkal foglalkozom.
Ha fákkal foglalkozunk, gyakran találkozhatunk az él fogalmával. A mutató egyirányú, ami azt jelenti, hogy a mutató követhető abból a csomópontból, ahonnan kiindul, de a folyamat visszafelé nem lehetséges. Nem lehetséges egy csomópontból visszakövetni, hogy melyik másik csomópontból hivatkoznak rá. Ennek megfelelően az él kétirányú, vagyis mindkét irányból nyomon követhető.
Az alábbiakban három különböző fameghatározást sorolunk fel, melyekben az a különös, hogy matematikai szempontból mégis egyenlők. Az általam felvázolt meghatározásnak azonban nem felelnek meg, mivel az él nem egyezik meg a mutatóval. Mindhárom meghatározás esetén legyen egyetlen él, amely ugyanazt a két csomópontot köti össze:

- Élekkel összekötött csúcsok (vagyis pontok) esetében az élek száma pontosan eggyel kevesebb, mint a csúcsok száma.
- Csúcsok élekkel összekötött halmaza, mely nem tartalmaz kört (körnek nevezzük a csúcsból önmagába visszahajló összekötővonalat).
- Csúcspontok csoportját, amelyeket élek kötnek össze, és két csúcspont csak egyetlen útvonalon érheti el egymást.

Ezekben a meghatározásokban a fáknak nincsen egységes gyökerük, és az ilyen fákat szabad fáknak hívjuk.
Az általam használt meghatározás egy gyökérrel rendelkező fát takar, nem egy szabad fát, amelyben nincsen kör sem, és pont eggyel kevesebb mutatóval rendelkezik, mint ahány csomópontja van, emellett minden csomópont bármely másikat csak egyetlen útvonalon érhet el.

Gráf vagy fa?

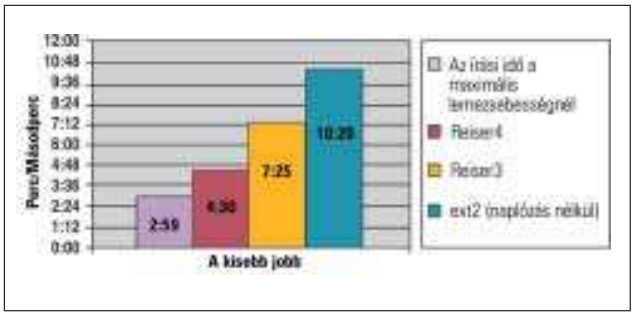
Vedd sorba azokat a feladatokat, amelyeket inkább gráf segítségével oldanál meg, és azokat, amiket fa segítségével valósítanál meg. Egy fában a fa minden egyes csomópontjához a gyökérből kiindulva csak egyetlen útvonal létezik, ezenkívül a fa minimális követelménnyel rendelkezik a mutatók számát illetően, hogy minden csomópontot össze lehessen kapcsolni. Ez teszi a fát egyszerű és hatékony szerkezetté. A fák akkor használatosak, amikor kevésbé bonyolult és hatékony szerkezetre van szükségünk, ahol nem jelent gondot, hogy minden csomópontot csak egyetlen útvonalon érhetünk el. A Reiser4 fákat és gráfokat egyaránt használ. Fákat olyankor, ha a fájlrendszer választja ki a szükséges szerkezetet (ezt hívjuk tárolási rétegnek, amelynek egyszerűnek és hatékonynak kell lennie), és gráfokat, amikor a felhasználó választja ki a szükséges szerkezetet (a tartalmi szinten, melynek sokoldalúnak kell lennie, hogy a felhasználó azt tehesse, amit csak akar).

Kulcsok

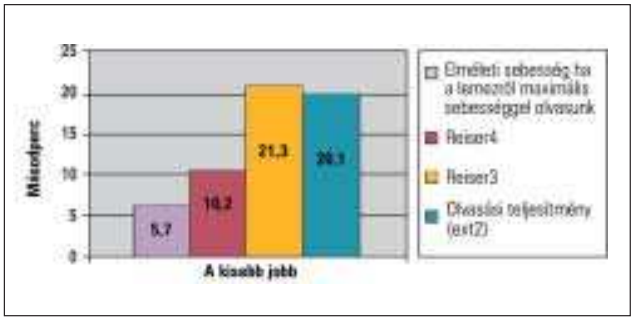
A fában minden elemhez kulcsot rendelünk, ennek alapján találjuk meg a keresett elemeket, használatuk pedig óriási rugalmasságot biztosít a dolgok rendezésekor. Ha a kulcsok rövidek, akkor kevés adat birtokában is megtalálhatjuk, amit keresünk. Azt is behatárolja egyúttal, hogy milyen adatok alapján kereshetjük meg a dolgokat.
Ez a határ korlátozza a kulcs használhatóságát, emiatt rendelkezünk egy tárolási réteggel, ami a dolgokat a kulcsok alapján keresi meg, és egy tartalmi réteggel, ami nagyon gazdag elnevezési rendszerrel rendelkezik (erre cikkünk 2. részében térünk ki). A tárolási réteg csak a tároló megszervezésére használ kulcsokat, így növelve a teljesítményt, ugyanakkor a tartalmi réteg olyan nevekkel dolgozik, amelyeknek a felhasználó számára jelentése van. Talán felteszed a kérdést, hogy vajon ez a megfelelő elrendezés-e, olyan, ami biztosítja a szabadságot, hogy újabb dolgokat adjunk a tárolási réteghez, így növelve teljesítményét – ugyanakkor a tartalmi réteg elnevezéseinek kezelésekor el kell viselnünk a mellékhatásait.

A részfa kiválasztása

A gyökérnél kezdjük a keresést, mivel innen az összes csomópont elérhető. De hogyan dönthetjük el, hogy a gyökérből elindulva melyik részfán folytassuk az utunkat? A részfákra a gyökérben található mutatók segítségével juthatunk el. Minden egyes részfára irányuló mutatóhoz tartozik egy bal oldali körülhatároló kulcs. A részfákra irányuló mutatók és maguk a részfák is baloldali körülhatároló kulcsuk szerint vannak rendezve. Egy részfa mutatójának a bal oldali körülhatároló kulcsa megegyezik a részfában található legkisebb kulccsal. Ennek a jobb oldali körülhatároló kulcsa nagyobb, mint a részfa legnagyobb kulcsa, és ez a csomópont következő alfájának a bal oldali kulcsa is egyben.



1. ábra Írási teljesítmény (30 másolata 2.4.18-as rendszermagnak)



2. ábra Olvasási teljesítmény
(a 2.4.18-as rendszermag forráskódját beolvasva)

Az egyes részfák csak olyan dolgokat tartalmaznak, amelyeknek vagy a mutatójukhoz tartozó bal oldali körülhatároló kulcsa legalább egyenlő, vagy a jobb oldali körülhatároló kulcsa legfeljebb ilyen nagyságú. Ha a részfában nincsenek többszörösen előforduló elemek, akkor minden részfa csak olyan dolgokat tartalmaz, amelynek a kulcsai kisebbek a jobb oldali körülhatároló kulcsánál. Ha nincsenek kettőződések, akkor egy csomópont részfákra irányuló mutatóinak és azok körülhatároló kulcsainak alapján egyértelműen eldönthető, hogy melyik ág tartalmazza a keresett dolgot.
A többszörösen előforduló kulcsokról egy másik alkalommal lesz szó. Most csak annyit jegyeznék meg, hogy először a részfák között a kettőzött kulcsú elem egyik kulcsát kell megtalálni, majd az ehhez tartozó párokat kell egyesével végigvizsgálni, mígnem megtaláljuk a megfelelő elemet. A kettőzött kulcsokkal kisebb méretű kulcsokhoz jutunk, így alkalomadtán egyetértésre juthatunk a kulcsok méretét és az ilyen nem hatékony keresések mennyiségét illetően. A fa minden csomópontjában a rendezés a csomópontokon belül zajlik. Ennek alapján az egész fát kulcsok szerint rendezzük, így bármely kulcs birtokában tudjuk, hol keressük a kulcshoz tartozó elemek egyikét.

Csomópontméret

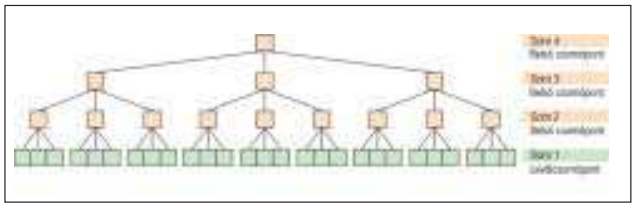
A csomópontok méretét egyformának választjuk, így könnyebb meghatározni a csomópontok közti szabad területet, mivel az a csomópont méretének valahányszorosával lesz egyenlő. Ilyen módon nem okoz gondot, ha rendelkezünk ugyan hellyel, de a rendelkezésre álló hely túlságosan kicsi egy csomópont tárolásához. Emellett a merevlemezek olyan csatolófelülettel rendelkeznek, ami feltételezi az egyenlő méretű szakaszokat, ami kényelmesen illeszkedik a hibajavító műveletekhez.
Ha nem fontos, hogy a csomópontok egyforma méretűek legyenek, mert mondjuk elférnek a RAM-ban, érdemes lehet egy pillantást vetnünk az átugró listákra.
A Reiser4 csomópontjai alapesetben a lapmérettel egyeznek meg, amely – ha Linuxot használsz Intel processzorral – 4 k (4096 bájt). Semmilyen tapasztalati bizonyíték nem támasztja alá, hogy ez a méret jobban megfelelne, mint bármely másik; az egyetlen indok, amiért mégis ezt használjuk, az az, hogy a Linux ezt teszi a legegyszerűbben programozhatóvá. Az igazat megvallva: inkább csak nem volt még időnk egyéb méretekkel kísérletezgetni.

Takarékoskodjunk a hellyel

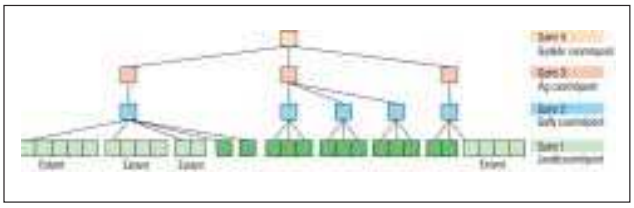
Ha a csomópontok mérete megegyezik, hogyan tároljunk nagyobb dolgokat? Darabokra vagdossuk őket: az így létrejövő darabokat cikkelyeknek (item) hívjuk. A cikkely mérete pont akkora, hogy elférjen egyetlen csomópontban.
A hétköznapi fájlrendszerek az állományokat teljes szakaszokban tárolják. Ez körülbelül annyit jelent, hogy fájlonként egy fél szakaszt elveszítünk. Ha a fájl jóval kisebb, mint a szakasz mérete, akkor a veszteség a fájl méretének sokszorosa is lehet. Ennek következtében nem ajánlatos szokványos adatbáziselemeket (címek, telefonszámok) tárolni ilyen hétköznapi fájlrendszeren, mivel az elfoglalt terület legalább 90 százaléka kárba vész. Azáltal, hogy a Reiser4-ben egyetlen csomópontban több dolgot is képesek vagyunk tárolni, az is lehetséges, hogy egy-egy szakaszt több kisméretű fájl között osszunk meg. Hatékonyságunk a kisméretű fájlok helykihasználását illetően megközelítőleg 94 százalék.
Ez a szám természetesen nem tartalmazza a fájlonkénti többlettárhelyet, ami függ az egyes fájlok méretétől, emiatt nehezen megbecsülhető.
A fájlok 4 k-s szakaszméretre igazítása azonban nagy fájlok esetén kifejezetten előnyös. Ha egy program közvetlenül szeretne egy fájllal dolgozni, tehát anélkül, hogy különféle rendszerhívásokhoz kellene folyamodnia, használhatja az `mmap()` függvényt, amellyel a fájl tartalma közvetlenül az alkalmazás címteréből válik elérhetővé. Bizonyos megvalósítási tényezőkből következően az `mmap()` megköveteli, hogy a fájl tartalma 4 k-s határra legyen igazítva. Ha az adatok már eleve 4 k-ra vannak igazítva, az nagymértékben felgyorsítja az `mmap()`-ot. A jelenlegi alapértelmezés szerint a Reiser4-ben a 16 k-nál nagyobb fájlok 4 k-s határra vannak igazítva. Jelenleg azonban még nem rendelkezünk elég tapasztalattal és adatokkal ahhoz, hogy meg tudjuk ítélni, a 16 k-s fájlméret valóban megfelelő-e.

Levelek, ágak, gallyak

Fás hasonlatunknál maradva, a levelek gyermekkel nem rendelkező csomópontok. A belső csomópontok alatt a gyermekkel rendelkező csomópontokat értjük.
A keresés a gyökérnél kezdődik, ezt követően bejár két belső csomópontot, majd egy levélen fejezi be az útját, mely egy adatokat tartalmazó csomópont, gyermekek nélkül. A cikkely



3. ábra Egy Magasság = 4; 4-szintű; Elkülönítés = 3; Kiegyensúlyozott fa



4. ábra Egy Reiser4 fa belső, vagyis ágcsomópontokkal

egy adattároló szerkezet, ami teljes egészében egy szakaszon belül helyezkedik el. Azt a csomópontot, ami cikkelyeket tartalmaz, formázott csomópontnak nevezzük. Ha dolgokat tárolunk egy fában, az egyes darabokat cikkelyekben és formázatlan levelekben tároljuk el. Formázatlan leveleknek hívjuk az olyan leveleket, amelyek semmilyen formázási adatot nem tartalmaznak, csakis adatot. Egyedül a levelek tartalmazhatnak formázatlan adatokat. A mutatók cikkelyekben tárolódnak, ennek következtében minden belső csomópont egyúttal szükségképpen formázott csomópont is.

A formázott csomópontokra irányuló mutatók különböznek a formázatlan levelekre irányulókról. A jobb megértés kedvéért vegyük a következő példát: a fokmutatók formázatlan levelekre hivatkoznak. Foknak nevezzük az egymást követő, szomszédos, szakaszon belüli vagy szám szerint egymást követő, egyazon dologhoz tartozó formázatlan leveleket. A fokmutató tartalmazza a kezdő szakasz számát és a hosszt. Mivel a fok csak egyetlen dologhoz tartozik, a fokhoz csak egyetlen kulcsot tárolhatunk, és még ezután is kiszámolhatjuk minden egyes bájtnak a kulcsát a fokon belül. Ha a fok legalább két szakasz hosszúságú, a fokmutatók sokkal tömörebbek, mint az általános csomópontmutatók.

A csomópontmutatók formázott csomópontokra irányuló mutatók. Jelen pillanatban még nem rendelkezünk a csomópontmutatók tömörített változatával, de hamarosan elkészülnek. Fontos, hogy fokmutatók esetén nem szükséges tárolni a hivatkozott csomópontok körülhatároló kulcsait, míg a csomópontmutatók használatakor ez muszáj. Valószínűleg a tömörített kulcsok a tömörített csomópontmutatókkal egyidőben kerülnek bemutatásra. Valaki talán azt várná, hogy a kulcsok tömörítve legyenek, csak azért, mert növekvő sorrendbe vannak rendezve. Szeretnénk, ha csomópont- és cikkelybővítményeink ilyen tulajdonságokkal valamikor a későbbiekben egyszerűen kiegészíthetővé válnának.

A gallyak a levelek szülői, és a fokmutatók csakis gallyakban léteznek. (Ez egy elég vitatott tervezési elmélet, amelyet a cikk következő részében még tárgyalunk.) Az ágak belső csomópontok, és nem egyeznek meg a gallyakkal.

Azt hihetnéd, hogy a gyökeret egyszerűen első szintnek is nevezhetnénk, de mivel a fa a felszínen nő, sokkal ésszerűbb első szintnek nevezni azt a részt, ahol a levelek is találhatók és az adatok tárolódnak. A fa magassága attól függ, hogy hány dolgot kell tárolnunk benne, és hogy a belső és gallycsomópontok hány gyermekkel rendelkeznek.

A cikkelyek típusai

A Reiser4 többféle típusú cikkelyt tartalmaz a különféle típusú adatok tárolásához:

`static_stat_data`: a tulajdonost, a jogosultságokat, az utolsó hozzáférés idejét, a létrehozás idejét, az utolsó módosítás idejét, a méretet és a fájlra mutató hivatkozások számát tartalmazza.

`cmpnd_dir_item`: a könyvtárbejegyzéseket és a fájlokra mutató kulcsokat tartalmazza.

Fokmutatók: lásd feljebb.

Csomópontmutatók: lásd feljebb.

Törzs: a fájlnak azon részét tartalmazza, ami nem elég nagy ahhoz, hogy egy különálló formázatlan levélben tárolódjon.

Egységek

Egységnek nevezzük azt az elemet, amelyet teljes egészében egy cikkelyben kell elhelyeznünk, anélkül, hogy több cikkely között szétdarabolnánk. Egy cikkely tartalmát gyakran egységekben egyszerűbb végigolvasni:

- A törzsben található cikkelyek esetén az egység a bájt.
- Könyvtárcikkelyek esetén az egység az egyes bejegyzéseket jelenti. A könyvtárbejegyzések tartalmazzák a fájl nevét és kulcsát (amelyek a gyakorlatban akár tömörítettek is lehetnek).
- Fokcikkelyek esetén az egység a fok. Fokcikkelyek csak az adott fájlhoz tartozó fokokat tartalmaznak.
- A `static_stat_data` számára az egész statisztikai adat egyetlen oszthatatlan, rögzített méretű mezőt képez.

Összegzés

Elmagyaráztam a Reiser4 fa alapvető szerkezeteit, de az igazán szórakoztató rész még csak most következik. Még nem magyaráztam el, hogy más kutatók hogyan alakítják a fáikat, és még azt sem tudod, hogy az elemek tartalma miért a fa alján helyezkedik el, miért szükséges a pontos elkülönítés, és miért van szükség különféle kiegyenlítőkre. Még csak nem is utaltam eddig arra, hogy miért jobbak a kiegyensúlyozott fák, és hogy miért a táncoló fa a legjobb. Amiről pedig aztán végképp nem beszéltem, az az, hogyan változik meg egy bonyolult és vitatott faszerkezet, amelyet ebben a cikkben is bemutattunk, úgy, hogy a Reiser4 kétszer gyorsabban olvasson, mint a Reiser3. Az erről szóló írás – helytől függően – a Linux Journal következő számában fog megjelenni.

Forrás

Donald E. Knuth A számítógép-programozás művészete

Linux Journal 2002. december, 104. szám

Hans Reiser (reiser@namesys.com)
Záró dolgozatát a nehéz tudományok és a számítástudomány filozófiájának különbözőségeiről írta, amelyre egy eredeti elnevezési rendszert is kifejlesztett. Ezt a rendszert még most is fejleszti, ahol is a Reiser4 képezi a tárolási réteget. 1993-ban Oroszországba utazott, hogy összeszedjen egy programozókból álló csapatot, és belefogjanak a ReiserFS fejlesztésébe. 1999 körül a rendszer olyannyira jól kezdett működni, hogy édesanyja felhagyott az unszolással, hogy a fia valamilyen jól fizető állást keressen egy nagy cégnél.