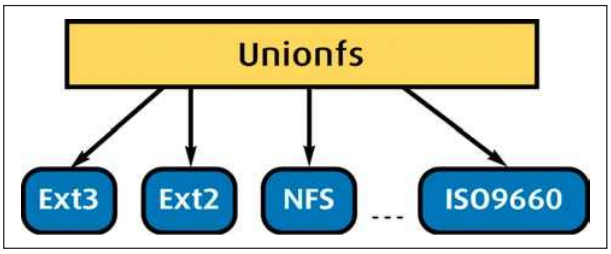
Unionfs: eggyé váló fájlrendszerek

Az Unionfs több könyvtárat is képes egyetlen nézetben egyesíteni. A továbbiakban az Unionfs alkalmazásait, valamint működésének néhány érdekes kérdését tárgyaljuk.

Az egymáshoz kapcsolódó, ám különböző fájlokat a könnyebb kezelés miatt sokszor érdemes elkülönítve tárolni. A felhasználók viszont inkább együtt szeretnék látni őket. Ilyenkor a rendszergazda egy egyesítéssel elérheti, hogy fizikailag elkülönítve maradjanak, logikailag azonban mégis egyetlen nézetben legyenek elérhetők a fájlok. Az egybevont könyvtárak halmazát uniónak nevezzük, minden fizikai könyvtár ennek egy ága. Amint az *1. ábrán* is látható, a ***Unionfs*** egyszerre több fájlrendszer felett vagy adott fájlrendszer több könyvtára felett alkot újabb réteget. Ezt a rétegezéses megoldást vermelésnek nevezzük, az internetes források között bővebben is lehet olvasni róla. A ***Unionfs*** a rendszermag felé egy fájlrendszer felületet mutat, míg az alatta üzemelő fájlrendszerek őt a rendszermag ***VFS***-ének látják. Mivel a ***Unionfs*** fájlrendszerként látszik a rendszermag számára, bármely felhasználói alkalmazás, illetve a rendszermag felől az NFS-kiszolgáló is igénybe veheti szolgáltatásait. A ***Unionfs*** elfogja az alacsonyabb szintű fájlrendszerekkel végzett műveleteket, és ezek módosításával képes az egységes nézet biztosítására. A korábbi vermelhető fájlrendszerekkel ellentétben a ***Unionfs*** valóban legyezőként terült szét az alsóbb rétegek felett, és nagy számú alsóbb szintű ágat képes közvetlenül elérni.



1. ábra Egy unió számos alsóbb szintű ágból áll, amelyek tetszőleges típusú fájlrendszerrel üzemelhetnek

Az Unionfs szemantikája és használata

Unionfs alatt minden ágnak van egy elsőbbségi szintje, precedenciája. A magasabb elsőbbségi szintű ágak elnyomják az alacsonyabb szintűeket. A ***Unionfs*** könyvtárakkal dolgozik. Ha egy könyvtár két ágban is megtalálható, akkor a ***Unionfs***-ben megjelenő könyvtár tartalma és jellemzői a két könyvtár kombinációjából tevődnek össze. A ***Unionfs*** magától eltávolítja a kettős könyvtárbejegyzéseket, így a felhasználókat nem zavarják meg a kettős fájl- és könyvtárnevek. Ha egy fájl két ágban is jelen van, akkor a ***Unionfs*** alatt látható fájl tartalma és jellemzői a magasabb elsőbbségi szintű ágban találhatóval egyeznek meg, az alacsonyabb szintű ágban lévő fájlt a rendszer elnyomja.
Példaként tegyük fel, hogy két könyvtárat egyesítünk, ezek a */**Gyümölcsök*** és a */**Zöldségek***:

```
$ ls /Gyümölcsök
Alma Paradicsom
$ ls /Zöldségek
Répa Paradicsom
$ cat /Gyümölcsök/Paradicsom
Növénytanilag gyümölcs vagyok.
$ cat /Zöldségek/Paradicsom
A kertészek zöldségként kezelnek.
```

A ***Unionfs*** használatba vételéhez először le kell fordítanunk a ***Unionfs*** modult, majd be kell töltenünk. Következő lépésként, mint minden más fájlrendszert, a ***Unionfs***-t is be kell fűzni. Az egyéb fájlrendszerekkel ellentétben a ***Unionfs*** nem egy eszköz, hanem a befűzésekor megadott könyvtárak fölé helyezkedik el. Unió létrehozásához a következőképpen kell befűznünk a ***Unionfs***-t:

```
# mount -t unionfs -o dirs=/Gyümölcsök:/Zöldségek
➥> none /mnt/egészséges
```

A fenti példában a befűzéskor átadott `dirs` értékkel adjuk meg az uniót alkotó könyvtárakat. Mivel a ***Unionfs*** eszközt, meghajtót nem használ, a `none` (semmi) helyőrzőt használjuk. Utolsóként a `/mnt/egészséges` értéket adjuk meg, amely az egyesített nézet helye. Most a */**mnt/egészséges*** könyvtár három fájlt tartalmaz: `Alma`, `Répa` és `Paradicsom`. Mivel a `/Gyümölcsök` könyvtárat a `/Zöldségek` előtt adtuk meg, a `/mnt/egészséges/Paradicsom` fájl tartalma a „`Növénytanilag gyümölcs vagyok`”. Ha a `dirs=` átadott érték elemeit megfordítjuk, akkor a `/mnt/egészséges/Paradicsom` tartalma az „`A kertészek zöldségként kezelnek`” lesz. (Egyébként ez utóbbi felel meg az ***USA Legfelsőbb Bírósága*** a témában 1893-ban hozott döntésének.)
A folyamat rekurzív. Ha a `Gyümölcsök` könyvtárban lenne

egy Zöld nevű alkönyvtár, amely tartalmazná a Zöldcitrom fájlt, a Zöldségek könyvtárban pedig volna egy szintén Zöld nevű alkönyvtár, benne a Saláta fájllal, az eredmény a következő lenne:

```
$ ls /mnt/egészséges
Alma Répa Zöld/ Paradicsom
$ ls /mnt/egészséges/Zöld
Zöldcitrom Saláta
```

A *Unionfs* számos különböző célra használható. Alkalmas például arra, hogy több kiszolgálóról egyesítsük a kezdőkönyvtárakat, vagy több *ISO* képfájlt összevonva egységes képet alkossunk egy terjesztésről. Hasonlóan, az *Unionfs* másolás írás közben szemantikával alkalmas CD-lemezek tartalmának foltozására, amivel forráskód kezelésére vagy pillanatfelvételek készítésére is megfelel.

Egyesített kezdőkönyvtárak

Egy-egy ügyfélgép gyakran több különböző *NFS*-kiszolgálóról fűzi be a kezdőkönyvtárat. Szerencsétlen megoldás, hiszen ilyenkor mindegyik kiszolgálóhoz külön befűzési pont tartozik, ami kényelmetlen a felhasználó számára. A legjobb az lenne, ha mindegyik kezdőkönyvtárat ugyanarról a helyről, például a */home* könyvtárból lehetne elérni. Az automatikus befűzők egy része szimbolikus hivatkozásokat használ, az egyesítés látszatát keltve. *Unionfs* alatt ilyen hivatkozásokra nincs szükség. Két különálló könyvtár teljesen egyszerűen egyesíthető egyetlen nézetben. Tegyük fel, hogy két fájlrendszerünk van, ezek /alcid és /pingvin névvel vannak befűzve. A */home* könyvtárban a következő módon egyesíthetjük őket:

```
# mount -t unionfs -o dirs=/alcid,/pingvin
➥> none /home
```

Így a /alcid és a /pingvin kezdőkönyvtára egyaránt elérhető a /home könyvtárból.
Az Unionfs támogatja a többes olvasási-írási ágakat, vagyis a felhasználói fájlokat nem kell egyik könyvtárból átmozgatni a másikba. Ebben is eltér a korábbi egyesítő megoldásoktól, például a 4.4-es *BSD*-ben található *Union Mounts* rendszertől, ezek általában csak egy olvasási-írási ágat támogatnak.

Terjesztések ISO képfájljainak egyesítése

A legtöbb *Linux*-terjesztés *ISO* képfájlok és különálló csomagok formájában egyaránt elérhető. Az *ISO* képfájlok kényelmes megoldást nyújtanak, hiszen közvetlenül fel lehet írni őket CD-lemezre, letölteni és külön tárolni csak néhány fájlt kell. Ha viszont hálózaton keresztül kell telepítenünk egy gépet, sokszor jó volna, ha az egyes csomagokat egyetlen könyvtárban látnánk. A hurokeszköz segítségével az *ISO* képfájlokat ugyan be tudjuk fűzni különböző könyvtárakba, ám a hálózati telepítéshez ez az elrendezés nem felel meg, hiszen az összes fájlnak egyetlen faszerkezetben kell lennie. Éppen ezért sok helyen az *ISO* képfájlokból és az egyes csomagfájlokból egyaránt fenntartanak egy másolatot, ami viszont a tárhely és a sávszélesség pazarlását, valamint a felügyeleti feladatok szaporodását eredményezi.

A *Unionfs* segítségével egy könnyed huszárvágással oldhatjuk meg a problémát, ha képzetesen egyesítjük az *ISO* képfájlok csomagokat tartalmazó könyvtárait.
Az alábbi példában két könyvtárat fűzünk be, ezek a /mnt/lemez1 és a /mnt/lemez2. A befűzési parancs a következő:

```
# mount -t unionfs -o dirs=/mnt/lemez1,/mnt/lemez2
➥> none /mnt/egyesített-terjesztés
```

Ezután a /mnt/egyesített-terjesztés könyvtárat *NFS*-en vagy *FTP*-n keresztül használhatjuk a hálózati telepítésekhez.

Másolás írás közben típusú uniók

Az iménti példában az unió minden ága csak olvasható volt, ennél fogva magát az uniót is csak olvasni lehetett. A *Unionfs* képes csak olvasható és írható-olvasható ágak egyesítésére is. Ilyenkor az unió maga írható-olvasható lesz, és az *Unionfs* másolás írás közben megoldást használva annak a látszatát kelti, hogy a csak olvasható ágak fájljainak és könyvtárainak módosítására is van lehetőségünk. Ezzel a módszerrel például CD-lemez tartalmát tudjuk frissíteni, foltozni. Ha a CD-ROM-meghajtót a /mnt/cdrom alá fűztük be, valamint van egy /tmp/cdfolt könyvtárunk is, akkor a következő parancsot adjuk ki:

```
# mount -t unionfs -o dirs=/tmp/cdfolt,/mnt/cdrom
➥> none /mnt/foltozott-cdrom
```

A /mnt/foltozott-cdrom könyvtáron keresztül nézve úgy tűnik, mintha írni lehetne a CD-lemezt, de természetesen az írások a /tmp/cdfolt könyvtárba történnek. Egy csak olvasható ág írása egy fölémásolásnak nevezett műveletet eredményez. Ha egy csak olvasható fájlt írásra nyitunk meg, akkor a fájlrendszer átmásolja egy magasabb elsőbbségi szintű ágba. Szükség esetén a *Unionfs* magától létrehozza a szülőkönyvtárak szerkezetét is.
A CD-lemezes példában az alacsonyabb szintű fájlrendszer kötelező érvénnyel betartja a csak olvashatóságot, az *Unionfs* pedig figyelembe veszi ezt. Lehetnek olyan esetek is, amikor az alacsonyabb szintű fájlrendszer ugyan írási-olvasási hozzáférést enged, ám az *Unionfs* számára meg akarjuk tiltani az ág módosítását. Lehet például egy águnk, amely eredeti rendszermag forráskódokat tartalmaz, és dönthetünk úgy, hogy emellett egy másik ágat használunk a helyi módosítások tárolására. A *Unionfs*-en keresztül az eredeti ágat csak olvashatóvá is tehetjük, hasonlóan az előző példa CD-lemezéhez. Ehhez csak az =ro kapcsolót kell hozzáadjunk a dirs befűzési átadott értékhez. Tegyük fel, hogy a /home/cpw/linux könyvtár üres, és a /usr/src/linux tartalmazza a Linux rendszermag forráskódjának könyvtárait. Az alábbi paranccsal a *Unionfs*-t forráskód-változatkövető rendszerré alakíthatjuk:

```
# mount -t unionfs -o
➥> dirs=/home/cpw/linux:/usr/src/linux=ro
➥> none /home/cpw/linux-src
```

A parancs kiadása után úgy fogjuk érzékelni, mintha egy teljes *Linux* forráskód lenne a /home/cpw/linux-src

könyvtárban, ám bármilyen módosítást is hajtunk végre rajta, a megváltozott vagy új fájlok valójában a `/home/cpw/linux` könyvtárba kerülnek.
Egy egyszerű módosítással elfedő befűzést is választhatunk, ekkor a `/home/cpw/linux` könyvtárat egységes nézetben látjuk:

```
# mount -t unionfs -o
➥> dirs=/home/cpw/linux:/usr/src/linux=ro
➥> none /home/cpw/linux
```

Megvalósítás

Unionfs alatt a legtöbb fájlrendszerbeli művelet a magasabb elsőbbségi szintű ágak felől az alacsonyabb elsőbbségi szintűek felé halad. A ***LOOKUP*** (keresés) például a szülőt tartalmazó legmagasabb elsőbbségi szintű ággal kezd, innen halad az alacsonyabb szintűek felé. A keresések alatt az ***Unionfs*** a későbbi műveletek céljaira gyorstárazza az adatokat.
A ***CREATE*** (létrehozás) a szülőkönyvtárat tartalmazó legmagasabb elsőbbségi szintű könyvtárban kísérli meg a fájl létrehozását. A ***CREATE*** művelet a gyorstárazott keresési adatokra támaszkodik, így biztosítva, hogy közvetlenül a megfelelő ággal dolgozzon, így lényegében a magasabb ágak felől mozog az alacsonyabbak felé.
A ***Unionfs*** számos módszert alkalmaz a csak olvasható ágak látszólagos módosíthatóságának fenntartására, miközben a közönséges, ***UNIX***-ra jellemző szemantikát is megőrzi. Ha egy fájl létrehozása közben hiba történik, akkor hibakezelést kell végezni. A hibakezelés a legalacsonyabb elsőbbségi szintű ág felől a magasabbak felé halad. A szülőt tartalmazó legmagasabb elsőbbségi szintű ággal kezdve az ***Unionfs*** mindegyik magasabb szintű ágban megkísérli a fájl létrehozását. Végül, ha a művelet a legmagasabb szintű ágban a teljes unióra nézve sikertelen, az ***Unionfs*** hibát jelez a felhasználó felé.
A ***CREATE*** művelettel ellentétben az ***UNLINK*** (leválasztás) mindig a legalacsonyabb elsőbbségi szintű ágtól halad a legmagasabb felé. Mivel az utolsó leválasztandó objektum a legmagasabb fontossági szintű objektum, a felhasználó szemszögéből semmi sem változik, amíg az ***Unionfs UNLINK*** művelete teljesen be nem fejeződik. A legbonyolultabbak a részleges hibák. Ha egy köztes fájlt nem sikerül eltávolítani, és az ***Unionfs*** egyszerűen törli a legmagasabb fontossági szintű fájlt, akkor az alacsonyabb fontossági szintű fájl válik láthatóvá a felhasználó számára. A hibahelyzetek kezelésére az ***Unionfs*** egy különleges, magas elsőbbségi szintű fájlt használ, ennek neve ***whiteout*** (kb. kihagyás). Ha az ***Unionfs*** kihagyás fájllal találkozik, akkor úgy viselkedik, mintha a kérdéses fájl egyik alacsonyabb elsőbbségi szintű ágban sem létezne. Az ***Unionfs*** belső folyamatai például az ***F*** nevű fájlhoz a kihagyás fájlt egy nulla bájtos ***.wh.F*** nevű fájl formájában hozzák létre.
Visszatérve a leválasztás művelethez: ha egy köztes leválasztás végrehajtása sikertelen, akkor a legmagasabb elsőbbségi szintű fájlt az ***Unionfs*** nem törli, hanem a megfelelő kihagyás névre kereszteli át.
A műveletek gondos sorrendezése két dolgot eredményez. Az első, hogy a unixos szemantika még hibák felmerülésekor és csak olvasható ágak kezelésekor is sértetlen marad. A felhasználó által látott állapot mindaddig érintetlen marad, amíg a művelet el nem ér a legmagasabb elsőbbségi szintű ágig. A művelet sikerességét vagy sikertelenségét az ebben az ágban kapott eredmény határozza meg. A kihagyás fájlok használatával adott fájlt még akkor is lehet törölni, ha az csak olvasható ágban található. A második fontos hatás, hogy ha nem történik hiba, akkor a fájlok és könyvtárak azokban az ágakban maradnak, ahol eredetileg is voltak. Ez fontos szempont, hiszen a ***Unionfs*** egyik fő célja a fájlok különböző helyeken tartása.

A fájlok törlésének szemantikája

Alapesetben az ***Unionfs*** az összes ágból törli a megadott fájl vagy könyvtár példányait, ezt a módot ***DELETE_ALL*** (teljes törlés) módnak nevezzük. A teljes törlés mellett az ***Unionfs*** további két törlési módot is támogat, ezek a ***DELETE_WHITEOUT*** (kihagyás törlése) és a ***DELETE_FIRST*** (első törlése). A kihagyás törlése kívülről szemlélve az alapmódhoz hasonlóan működik, ám az unió összes vonatkozó fájljának törlése helyett egy kihagyás fájlt hoz létre. Ennek az az előnye, hogy az alacsonyabb elsőbbségi szintű fájlok az alsóbb szintű fájlrendszereken keresztül elérhetők maradnak. Az első törlése mód szakít a hagyományos unixos szemantikával, és csak az unió legmagasabb elsőbbségi szintű bejegyzését távolítja el, aminek eredményeként az alacsonyabb elsőbbségi szintű bejegyzések láthatókká válnak. Mindezeket a módokat a ***RENAME*** (átnevezés) művelet is segítségül hívja, ez ugyanis egy létrehozásból és egy azt követő törlésből áll.
Az első törlése művelet használatához a felhasználónak ismernie kell az unió összetevőit. A mód elsősorban akkor használható jól, ha a ***Unionfs***-t a korábbi rendszermag forráskódfás példához hasonlóan forráskód változatainak követésére használjuk. Ha a ***/home/cpw/linux*** könyvtárban megváltoztatunk egy fájlt, akkor a rendszer felmásolja a magasabb elsőbbségi szintű ágba. Ha a fájlt a normál teljes törléssel távolítjuk el, az ***Unionfs*** egy kihagyás fájlt hoz létre a legmagasabb elsőbbségi szintű ágban – a csak olvasható alacsonyabb elsőbbségi szintű ágat ugyanis nem tudja módosítani. Az eredeti, az alacsonyabb elsőbbségi szintű ágban lévő forrásfájl ezzel elérhetetlenné válik, így a forrásból kell bemásolni az unióba, ez viszont aligha fogadható el egy változatkövető rendszertől. Pontosan ezek azok a helyzetek, amikor az első törlése mód jól jön. A törlési módot befűzéskor adhatjuk meg, például:

```
# mount -t unionfs -o
➥> dirs=/home/cpw/linux:/usr/src/linux=ro,
➥> delete=first none /home/cpw/linux
```

Most, ahogy korábban is, ha módosítunk egy a `/home/cpw/linux` könyvtárban lévő fájlt, a `/usr/src/ linux` könyvtár tartalma érintetlen marad. Ha meggondoljuk magunkat, és nem akarjuk megtartani a változásokat, akkor egyszerűen töröljük a fájlt, ekkor újra láthatóvá válik az eredeti változat.

Unionfs pillanatfelvételek

Az ***unionctl*** segédprogrammal az ***Unionfs*** ágait üzem közben is meg lehet változtatni. Az unió bármelyik pontjára új ágat csatolhatunk, tetszőleges ágat kivehetünk, vala-

mint írható-olvasható ágakat csak olvashatóvá változtathatunk, illetve fordítva. Rugalmasságának köszönhetően az ***Unionfs*** a fájlrendszer pillanatfelvételének elkészítésére is használható. A következő példában a `/usr` könyvtárról készítünk pillanatfelvételt, miközben telepítünk egy új csomagot:

```
# mount -t unionfs -o dirs=/usr none /usr
```

Ezen a ponton a ***Unionfs*** egyetlen írható-olvasható ággal rendelkezik, és ez a `/usr`. Minden művelet továbbkerül az alsóbb szintű fájlrendszerhez, minden úgy zajlik, mintha a ***Unionfs*** jelen se lenne.
A pillanatfelvétel elkészítése két lépésből áll. Az első a pillanatfelvétel fájlok helyének megadása, amely egy ág – ez legyen például a `/pillfelv/0` – hozzáadásával történik:

```
# unionctl /usr –add /pillfelv/0
```

Ekkor a ***Unionfs*** a `/usr` könyvtárba szánt új fájlokat a `/pillfelv/0` könyvtárban hozza létre, miközben a `/usr` könyvtár alkönyvtáraiba mentett fájlok továbbra is a `/usr` könyvtár alá kerülnek. A látszólagos ellentmondás abból a szabályból fakad, hogy a fájlokat a legmagasabb elsőbbségi szintű olyan ágban kell létrehozni, amelyben a szülő létezik. Az unió gyökérkönyvtárának, vagyis a `/usr` könyvtárnak a fájljai esetében a szülő mindkét ágban létezik. Mivel a `/pillfelv/0` a magasabb elsőbbségi szintű ág, az új fájlok és könyvtárak fizikailag a `/pillfelv/0` könyvtárban jönnek létre. A `/pillfelv/0` viszont üres, ezért, ha létrehoznánk egy fájlt a `/usr/local` könyvtárban, akkor a legmagasabb elsőbbségi szintű szülő a `/usr` ágban lenne.
Az áttelepítés teljessé tételéhez az eredeti `/usr` ágat csak olvashatóvá kell tenni. Ismét az `unionctl`-t használjuk az ágak módosítására:

```
# unionctl /usr –mode /usr ro
```

Most, köszönhetően annak, hogy az ***Unionfs*** a `/usr` könyvtárat csak olvashatónak látja, minden írási művelet ténylegesen a `/pillfelv/0` könyvtárra irányítódik. Több pillanatfelvételt is készíthetünk, ha létrehozunk egy újabb ágat, például `/pillfelv/1` névvel, és a `/pillfelv/0` ágat csak olvashatóként jelöljük meg.
Az első pillanatfelvételt a `/usr` könyvtáron keresztül látjuk. Mindegyik pillanatfelvétel egy alapkönyvtárból és további, az eltéréseket növekményi formában tartalmazó könyvtárakból áll. Ha megadott pillanatfelvételre vagyunk kíváncsiak, akkor csupán egyesítenünk kell az első pillanatfelvételt és a növekményes változásokat. Ha például a `/usr` és a `/pillfelv/0` könyvtárak tartalmából összeálló pillanatfelvételt akarjuk elérni, az ***Unionfs***-t a következőképpen kell befűznünk:

```
# mount -t unionfs -o ro,dirs=/pillfelv/0:/usr
➥> none /mnt/pillfelvétel
```

Ebben az esetben az ***Unionfs*** maga is csak olvashatóként kerül befűzésre, az alsóbb szintű könyvtárakat tehát nem lehet módosítani.

A pillanatfelvételen végrehajtott módosítások helyességének ellenőrzése után a következő lépés gyakran a pillanatfelvétel beolvasztása az alapba. A ***Unionfs*** letölthető csomagjában egy `snapmerge` nevű parancsfájl is szerepel, ez – a pillanatfelvétel könyvtárában lévő fájlokat rekurzívan az alapkönyvtárba másolva – a növekményes ***Unionfs*** pillanatfelvételeket képes összeolvasztani az alapkönyvtárral.
A másolási eljárás befejezése után az új és a módosított fájlok hiánytalanul rendelkezésünkre állnak. A végső lépés a fájltörlések kezelése, amely a kihagyás fájlok listájának létrehozásával és a megfelelő fájlok törlésével történik. Maguk a kihagyás fájlok szintén törlésre kerülnek, így nem terhelik feleslegesen a fát.

Összefoglalás

A ***Unionfs*** rekurzívan egyesít akár nagyobb számú könyvtárat vagy ágat is egyetlen képzetes nézetben. Hatékony, legyezőszerű szerkezete révén a ***Unionfs*** számos alkalmazási területen állja meg helyét. A ***Unionfs*** például ***ISO*** képfájlok egyesítésére, összevont ***/home*** könyvtár biztosítására és rengeteg további célra használható. Az ***Unionfs*** másolás írás közben szemantikájának köszönhetően változatkövetésre, pillanatfelvételek készítésére és CD-lemezek foltozására is alkalmas. Az ***Unionfs*** teljesítményét 2.4-es ***Linux*** alatt vizsgáltuk. Fordítási feladatoknál egynégy ág kezelésekor az ***Unionfs*** miatti többletterhelés mindössze egy-két százalék volt. A nagy mennyiségű be- és kiviteli műveletet végző folyamatoknál a többletterhelés egy ág esetén 10, négy ág esetén pedig 12 százalék körüli volt.

Köszönetnyilvánítás

Hálával tartozom ***Puja Gupta***, ***Harikesavan Krishnan***, ***Mohammad Zubair*** és ***Jay Dave*** barátomnak, akik szintén a ***Unionfs*** fejlesztői csapatába tartoznak. Külön szeretnék köszönetet mondani ***Mohammadnek***, amiért segített összeállítani az írásom alapjául szolgáló szoftverkörnyezetet.

Linux Journal 2004. december, 128. szám

Charles P. Wright (cwright@cs.stonybrook.edu) a Stony Brook Egyetem informatikus doktorandusz hallgatója. Charles operációs rendszerekkel kapcsolatos kutatásokat vezet, amelyek fő témái a fájlrendszerek, a biztonság és a bővíthetőség. A Stony Brook Linux Felhasználói Csoport (LUGSB) aktív tagja.

Erez Zadok (ezk@cs.stonybrook.edu) a Stony Brook Egyetem informatikai karán dolgozik, a Linux NFS and Automounter Administration (Sybex, 2001) című könyv szerzője, a FiST vermelhető sablonrendszer létrehozója és karbantartója, továbbá az Am-utils (más néven Amd) automatikus befűző fő karbantartója. Erez operációs rendszerekkel kapcsolatos kutatásokat vezet, ezek fő témái a fájlrendszerek, a biztonság, a sokoldalúság és a hordozhatóság.