Az IBM naplózó fájlrendszere

Naplózó fájlrendszerrel létfontosságú kiszolgálód mindig újra tud indulni.

A Linux rendszermagja folyamatosan új lehetőségekkel bővül, ezek egyike a naplózó fájlrendszerek támogatása. Az IBM JFS-e a Linuxhoz létező naplózó fájlrendszerek egyike. Ebben a cikkben a JFS működését és jellemzőit tárgyaljuk. Szót ejtünk arról, hogyan kell telepíteni és beállítani egy Linux-kiszolgálón, valamint milyen tapasztalatokra tettünk szert a használata során az Ericsson Research Labnél (Montrealban).

Egy fájlrendszer arra szolgál, hogy a merevlemezeken lévő felhasználói adatokat tárolja és kezelje. Biztosítja, hogy az adat, amit a lemezre írunk, teljesen megegyezik azzal, amit a lemezről visszaolvasunk. A fájlok tárolása mellett a fájlrendszer számos egyéb adatot is kezel, úgymint a rendelkezésre álló szabad terület mértékét, vagy az `i` csomópontok számát, amelyeket a saját céljaira használ fel. A fájlrendszer efféle szerkezeteit kiegészítő adatoknak (metadata) szokás hívni, ezekbe a fájl tényleges tartalmán kívül minden egyéb beletartozik. A fájllal kapcsolatos adatokat, mint a fizikai elhelyezkedését vagy a méretét is a kiegészítő adatok között tárolják.

Naplózó fájlrendszerek a nem naplózó fájlrendszerek ellenében?

Egy naplózó fájlrendszer tartalmilag sokkal összefüggőbb, ugyanakkor a helyreállítása is jóval egyszerűbb, mint nem naplózó társainak. Ennek következtében naplózó fájlrendszerek esetén az újraindításhoz szükséges idő is rövidebb.

A nem naplózó fájlrendszerek ki vannak téve annak a veszélynek, hogy esetleg megsérülnek, mivel egy-egy logikai fájlművelet gyakorta több I/O műveleten keresztül kerül elvégzésre, így egy váratlan leállás esetleg a folyamat kellős közepén szakítja meg a műveletsort. Például egy egyszerű állományba történő íráshoz a következő műveletek szükségesek:

- Terület lefoglalása a fájlrendszeren.
- A területek mutatóinak a frissítése.
- A fájlméret frissítése.
- Az adat tényleges kiírása.

Ha a rendszert ezeknek a műveleteknek az elvégzése közben szakítjuk meg, akkor a nem naplózó fájlrendszer egy köztes, nem összefüggő állapotban marad. Ilyen esetekben ezek a fájlrendszerek az `fsck` nevű eszközre bízzák, hogy feltárja a kiegészítő adatokban keletkezett hibákat (például a könyvtár- és lemezcímző szerkezeteket), és lehetőség szerint javítsa is őket. Azonban az `fsck` elég időigényes lehet, függően a lemez méretétől, a könyvtárak és a bennük található fájlok számától. Tehát egy nagyobb fájlrendszer esetén a naplózó tulajdonság elengedhetetlen, hiszen egy naplózó fájlrendszer kevesebb mint egy másodperc alatt képes újraindulni.

Bemutatkozik a JFS

A JFS-t úgy tervezték, hogy bármilyen üzemzavart követően gyorsan képes legyen helyreállni, ezenkívül felkészítették nagy lemezrészek és nagyméretű állományok kezelésére, azonban ugyanígy megbirkózik nagyszámú könyvtárakkal és fájlokkal is. Ahhoz, hogy megfeleljen ezeknek a követelményeknek, a JFS a másodperc törtrésze alatt képes helyreállni, úgy, hogy csak a kiegészítő adatokban bekövetkező változásokat naplózza. A JFS támogatja a 64-bites rendszereket is, akár petabájt (2^{50} bájt = 1024 terabájt) méretű fájlokkal és lemezrészekkel. Ráadásul a fájlrendszerben található összes szerkezet B+ fákra épül. A nagyobb teljesítmény elérése érdekében a fájlrendszerben az összes lehetséges helyen B+ fákat használnak a hagyományos lineáris fájlrendszerszerkezetek helyett.

Fájlrendszerek

Az állományok úgynevezett fokok sorozatában tárolódnak. Foknak nevezzük az egymást követő szomszédos blokkok sorozatát, ami a JFS-ben egy egységet jelent. A fokot teljes egészében egy csoport belsejében (és ezáltal egyetlen lemezrészen) találjuk meg. Nagyméretű fokok azonban több tárterületi egységre is kiterjedhetnek. Egy fok mérete bármekkora lehet 1 és 2^{24} -1 blokk között. Példának okáért a JFS 24-bites értéket használ a fok hosszának meghatározására. 4 k-s blokkméret esetén a legnagyobb fokméret 4 k 2^{24} -1 lehet, ami megközelítőleg 64 GB-tal egyenlő. Fontos, hogy ez a határ csak egyetlen fokra vonatkozik – a fájl teljes méretét ez egyáltalán nem befolyásolja. A fokokat egy B+ fa alapján tartják nyilván a nagyobb teljesítmény érdekében, legyen szó akár új fokok hozzáadásáról, akár fokok kereséséről stb. Általánosságban a JFS elhelyezési politikája (allocation policy) a lehető legfolyamatosabb tárterület-elrendezést próbálja meg kialakítani úgy, hogy egy állományhoz minél kevesebb fokot használjon fel, és ezek a fokok lehetőség szerint minél nagyobbak legyenek. Ennek következtében az I/O műveletek hatékonysága növekszik, ami a teljesítmény javulását is magával vonja.

A JFS története

Az IBM a Unix fájlrendszerét Journaled Filesystem, vagyis Naplózó fájlrendszer néven mutatta be, amelynek első változata az AIX 3.1-gyel jelent meg. Ezt a fájlrendszert az AIX-en most JFS1-nek hívják. Az elmúlt tíz év során ez volt az AIX elsődleges fájlrendszere, és

vásárlók millióinak a gépein található meg. 1995-ben kezdtek el dolgozni azon, hogy a fájlrendszer méretezhető legyen, és támogassa a többprocesszoros rendszereket. A másik célkitűzés az volt, hogy a fájlrendszer más operációs rendszerekre egyszerűbben átültethető legyen.
Történelmileg úgy alakult, hogy a JFS1 fájlrendszer erősen kötődött az AIX memóriakezelőjéhez. Az effajta kialakítás jellemző a zárt forrású operációs rendszerekre, illetve azokra a fájlrendszerekre, amelyek csak egyféle operációs rendszert kívánnak támogatni.
Az új Journaled Filesystemet, amelyen a Linux-változat is alapul, először 1999 áprilisában az OS/2 Warp Server for eBusiness nevű termékükhöz adták, amelyet sokévnyi tervezés, kódolás és kipróbálás előzött meg. Az OS/2 Warp Client nevű termék is tartalmazta a fájlrendszert, amit 2000 októberében adtak ki. Ezzel párhuzamosan még 1997-ben a JFS fejlesztői csapatának néhány tagja visszatért az AIX operációs rendszer fejlesztői csoportjába, és elkezdtek dolgozni azon, hogy az új JFS az AIX-on is működjön. 2001 májusában Enhanced Journaled Filesystem (JFS2) néven kiadták a naplózó fájlrendszer második változatát az AIX 5L-hez. Mindeközben 1999 decemberében felfedték az OS/2-höz adott JFS forráskódját, és ezzel egy időben elkezdődött a JFS átültetése Linuxra.

Tapasztalatok a nyílt forrású projekttel

1999 decemberében három naplózó fájlrendszer is fejlesztés vagy átültetés alatt volt Linuxra. Az `ext2`-höz naplózási képességeket adtak, és a létrejövő új fájlrendszert `ext3`-nak nevezték. XFS fájlrendszerüket az SGI az elkezdte átültetni az Irixről. A harmadik fájlrendszer fejlesztését ***Hans Reiser*** kezdte el, így ennek a neve ReiserFS lett. De Linuxon e fájlrendszerek egyike sem volt teljes egészében működőképes 1999-ben. Az IBM hitt abban, hogy a JFS egy nagyon fejlett és markáns fájlrendszer, és átültetésével értékesebbé tehetnék a Linuxot.

Felvették tehát a kapcsolatot a vezető linuxos fájlrendszer-fejlesztőkkel, és tisztázták hogy lehetséges egy újabb naplózó fájlrendszer hozzáadása.
A Linux alapvető szellemisége, hogy meg kell adni a választás lehetőségét, így ennek fényében elfogadták az újabb naplózó fájlrendszer ötletét.
Az IBM 1999 decemberében kezdte el átültetni a fájlrendszert Linuxra, és 2000 februárjára már előálltak az első forráskóddal. Ez az első változat tartalmazta a bemutató forráskódot, támogatta a kötetek befűzését és leválasztását, és lehetővé tette az `ls` parancs használatát a JFS-lemezrészen.

A JFS telepítése különálló lemezrészre

A JFS a 2.5.6-os fejlesztői változat óta része a rendszermagnak, de az ***Alan Cox***-féle 2.4.x-ac rendszermagok is tartalmazzák a 2002 februárjában kiadott 2.4.18-pre9-ac4 változattól kezdődően. Alan rendszermagfoltjai a 2.4.x-es sorozathoz a ➲ http://www.kernel.org címen érhetők el. A 2.4-es rendszermagot külön is letöltheted, és saját magad hozzáadhatod a JFS-foltokat. E cikk írásának időpontjában a legújabb rendszermag a 2.4.18-as, a legújabb JFS-változat pedig az 1.0.20-as. A cikk további részében ezekkel fogunk dolgozni. A JFS-folt a JFS honlapjáról tölthető le. Szükséged lesz a JFS segédeszközeire (`jfsutils-1.0.20.tar.gz`) és a fájlrendszerfoltokra (`jfs-2.4.18-patch` és `jfs-2.4-1.0.20.tar.gz`) is.
Több Linux-terjesztés már eleve tartalmazza a JFS-t: legújabb változataiban a Turbolinux, a Mandrake, a SuSE, a Red Hat és a Slackware mind mellékeli a JFS-t.

A rendszermag felkészítése a JFS-re

Ha a fentebb felsorolt terjesztések bármelyikét használod, nem lesz szükséged a rendszermag foltozására. Csupán annyit kell tenned, hogy a rendszermagot le kell fordítanod JFS-támogatással. Elsőként töltsd le a hivatalos Linux-rendszermagot. Ha a rendszereden már létezik a */usr/src/linux* könyvtár, akkor nevezd át, így nem fogja felülírni a linux-2.4.18-as forrásfa. Miután letöltötted a rendszermagot tartalmazó `linux-2.4.18.tar.gz` fájlt, mentsd a */usr/src* könyvtárba, és csomagold ki. Így létre fog jönni egy új */usr/src/linux* könyvtár.
Következő lépésként töltsd le a JFS segédeszközeit és a 2.4.18-as rendszermaghoz tartozó foltot. Hozz létre egy könyvtárat a JFS forráskódjának */usr/src/jfs1020* néven, és ebbe a könyvtárba töltsd le a foltot. Ezt követően lépj be a 2.4.18-as rendszermag könyvtárába, és helyezd fel a foltot:

```
% cd /usr/src/linux
% patch -p1 < /usr/src/jfs1020/jfs-2.4-18-
  ➥patch
% cp /usr/src/jfs1020/jfs-2.4-1.0.20.tar.gz.
% tar zxvf jfs-2.4-1.0.20.tar.gz
```

A rendszermag beállításakor a `make menuconfig` vagy `make config` parancsot követően a ***Filesystems*** almenüben (`CONFIG_JFS_FS=y`) engedélyezd a JFS-t. Arra is lehetőséged nyílik, hogy a JFS-t különálló bővítményként állítsd be. Ebben az esetben elég csak a bővítményeket lefordítanod és feltelepítened:

```
% make modules && make modules_install
```

Máskülönben, ha a JFS-t közvetlenül a rendszermagba akarod befordítani, a teljes rendszermagot újra kell fordítanod:

```
% make dep && make clean && make bzImage
```

Majd fordítsd le és telepítsd a bővítményeket is, ha vannak fordítandó bővítményeid:

```
% make modules && make modules_install
```

Végül telepítsd fel az új rendszermagot:

```
% cp arch/i386/boot/bzImage /boot/bzImage-jfs
% cp System.map /boot/System.map-jfs
% ln -s /boot/System.map-jfs /boot/System.map
```

Ne feledd lefuttatni a LILO-t sem! (Ha még sosem fordítottál rendszermagot, olvasd el a Kernel-HOWTO-t, hogy megtudd, hogyan kell.)
Ha a rendszermag fordításával és telepítésével elkészültél, a JFS kiegészítéseit is le kell fordítanod és telepítened kell. Másold a `jfsutils-1.0.20.tar.gz` fájlt a */usr/src/jfs1020* könyvtárba, majd:

```
% tar zxvf  jfsutils-1.0.20.tar.gz
% cd jfsutils-1.0.20
% ./configure
% make && make install
```

Ha ezzel is megvagy, a következő lépés, hogy egy JFS-lemezrészt hozz létre. A következő példában mi egy tartalék lemezrészt használunk erre a célra; a következőkben megtudod, hogyan kell átköltöztetni egy meglévő lemezrészt JFS-re.
Ha még van a lemezeden felosztatlan hely, hozz létre egy új lemezrészt az `fdisk` paranccsal. A mi rendszerünk tartalmaz egy */dev/hdb3* tartalék lemezrészt, így mi ezt alakítottuk JFS fájlrendszerűvé. Miután létrejött az új lemezrész, indítsd újra a rendszeredet, hogy megbizonyosodj arról, hogy az új lemezrész képes a JFS fogadására.
A JFS létrehozásához add ki a következő parancsot:

```
% mkfs.jfs /dev/hdb3
```

Miután a fájlrendszer létrejött, be kell fűzni a rendszerbe. Ehhez először létre kell hozni egy könyvtárat, ahova a fájlrendszert be lehet fűzni. Legyen ez a könyvtár a */mnt/jfs*, majd adjuk ki a `mount` parancsot:

```
% mount -t jfs /dev/hdb3 /mnt/jfs
```

Ha a fájlrendszert sikeresen befűzted, készen állsz rá, hogy kipróbáld a JFS-t.
A fájlrendszer lecsatolásához az `umount` parancsot kell kiadni az előzőleg létrehozott csatlakozási pontra:

```
% umount /mnt/jfs
```

Átállás ext2-ről JFS-re

Az előző szakaszban bemutattuk, hogyan hozzunk létre JFS fájlrendszert egy meglévő tartalék lemezrészen. Most azt szemléltetjük, hogyan lehet átalakítani egy tetszőleges fájlrendszert, például az `ext2`-t JFS-re. Megnézzük, miként lehet összeismertetni a Linuxot egy JFS-lemezrésszel, majd hogyan tehetjük ezt a lemezrészt a gyökérfájlrendszerré.
Milyen elrendezés szükséges egy JFS-gyökérfájlrendszerhez? Az átállási folyamathoz egy üres lemezrészre lesz szükségünk. Tételezzük fel, hogy a */dev/hda5* tartalmazza a jelenlegi `ext2` gyökérfájlrendszert. Mi a */dev/hda6*-ot használjuk JFS-gyökérfájlrendszerként. Először az `ext2` lemezrész tartalmát átmásoljuk a JFS-lemezrészre. Ezt követően, ha nem akarod megtartani az `ext2`-es fájlrendszert, anélkül leformázhatod, hogy Linux-rendszeredet elveszítenéd.
Hogy a JFS-re épülő gyökérfájlrendszert használhasd létre a */dev/hda6*-on, kövesd az előzőekben leírt utasításokat, hogy rendszermagod tartalmazza a JFS-bővítményt. Ahhoz, hogy erről az új lemezrészről el lehessen indítani a rendszeredet, Linux-telepítésedet egy az egyben át kell másolnod az új fájlrendszerre. Ennek legegyszerűbb módja: másolj át minden állományt a JFS-lemezrészre. Elsőként fűzd be a fájlrendszert:

```
mount -t jfs /dev/hda6 /jfs
```

Majd másolj át minden fájlt az `ext2` fájlrendszerről a JFS-re:

```
% cd /
% cp -a bin etc lib boot dev home usr var
  ➥[ ] /jfs
```

A */proc* és a */tmp* különleges bánásmódot igényel:

```
% mkdir /jfs/proc
% chmod 555 /jfs/proc
% mkdir /jfs/tmp
% chmod 1777 /jfs/tmp
```

Nagyon fontos, hogy a */proc*-ot és a */tmp*-t a megfelelő jogosultságokkal hozd létre. Az 1777-es jogosultság azt jelenti, hogy az ilyen könyvtárban csak a fájlok és a könyvtárak egyéni tulajdonosai nevezhetik át és törölhetik a fájlokat vagy a könyvtárak tartalmát, vagy a rendszergazda. Az utolsó lépés magában foglalja a */etc/lilo.conf* és */etc/fstab* fájlok megváltoztatását. Elsőként a *lilo.conf*-ot változtatjuk meg, hogy rendszerünket a JFS fájlrendszeren lévő rendszermaggal lehessen indítani. Figyeld meg, hogy a gyökérfájlrendszer az első bejegyzésben különbözik. Mindez azért van így, mert a kötet, amiről a rendszerünket el szeretnénk indítani, nem a */dev/hda5/boot* könyvtárban van, hanem a */dev/hda6/boot* könyvtárban:

```
image=/boot/vmlinuz-jfs
       label=jfs-kernel
       read-only
       root=/dev/hda6
```

Végül a */etc/fstab* fájlon keresztül Linux-rendszerünknek meg kell mondanunk, hogy milyen fájlrendszereket fogunk használni. Változtassuk meg a következő sort:

```
/dev/hda5     /      ext2   defaults   1      1
```

hogy így nézzen ki:

```
/dev/hda6     /      jfs    defaults   1      1
```

Most újraindíthatod a rendszeredet, és betöltheted a `jfs-kernel` rendszermagot – így a Linux a JFS gyökérfájlrendszerrel fog felállni.
Meghibásodást követően a napló önműködően visszajátssza a műveleteket. A megszokott `fsck` üzenetek helyett a JFS naplójának üzeneteit fogod látni. A napló visszajátszása elengedhetetlen, ha a fájlrendszer sérült.

Az Ericsson Research Lab tapasztalatai a JFS-sel

Az Ericsson Research Open Systems Labjének egyik feladata az, hogy olyan rendszereket tervezzen, hozzon létre és mérjen fel, amelyek távközlési alkalmazások alapjául szolgálhatnak. Az ilyen távközlési rendszereknek nagyon magas és szigorú követelményeknek kell megfelelniük a méretezhetőség, a megbízhatóság és a magas rendelkezésre állás terén. Folyamatosan működniük kell, függetlenül bármilyen eszköz- vagy programhibától, ugyanakkor lehetővé kell tenniük, hogy hiba esetén a rendszerfelügyelők a szolgáltatás megszakítása nélkül cserélhessék ki a meghibásodott eszközöket vagy programelemeket. Ezért biztosítani kell a lehető legnagyobb megbízhatóságot és a lehető legmagasabb rendelkezésre állást, amire gyakran csak mint öt-kilences megbízhatósági szintre hivatkoznak (azaz: 99,999%-ban üzemképes).
Az ehhez hasonló távközlési programokat úgy fejlesztették ki, hogy a frissítést a szolgáltatás szüneteltetése nélkül lehetővé tegyék, emellett hibatűrők és tartalék hálózati kapcsolatokkal rendelkeznek, hogy szerencsétlenségek – akár földrengés – esetén is megfelelően működjenek.
Bár a rendszer tervezésekor mindent elkövetnek, hogy mindenféle esetlegesen felmerülő hibát megakadályozzanak,

Követelmények a naplózó fájlrendszerekkel szemben

- Gyors és megbízható hiba-helyreállítási képesség: a kiegészítő adatok helyreállítása (visszaforgatásukkal vagy újbóli alkalmazásukkal), miután a fájlrendszert egy hibás leállást követően befűzzük. A helyreállításhoz szükséges idő nem növekedhet a lemez felosztásának méretével arányosan.
- Legyen méretezhető: a fájlrendszernek méretezhetőnek kell lennie, vagyis támogatnia kell a nagy lemezrész- és fájlméreteket egyaránt (beleértve a véletlenszerű hozzáférést), valamint a nagyszámú fájlokat.
- Nagy teljesítmény biztosítása: a naplózási képesség hozzáadása nem bírhat jelentős hatással az alapvető műveletekre, se az olvasásra (beleértve a véletlenszerű hozzáférést), se az írásra.
- Befejezés: miután egy művelet sikeresen befejeződött, a műveletet nem lehet visszaforgatni (vagyis a tranzakció befejezettnek tekintendő, miután a vezérlés visszakerült a felhasználóhoz).
- Megfelelő eszközök: fájlrendszer létrehozása, a fájlrendszer helyreállítása rendszerleállás után, mentés (dump) és a fájlrendszer visszaállítása mentésből, töredezettségmentesítés (néhány fájlrendszer esetén ez az eszköz szükségtelen, mivel a fájlrendszer belső folyamatai végzik a töredezettségmentesítést).
- Rugalmasság: lehetővé kell tenni a fájlrendszer átméretezhetőségét akár különböző fizikai lemezrészeken keresztül is, miközben a fájlrendszer használatban van (az LVM felületének felhasználásával).
- Megbízható műveletvégzés: a fájlrendszer működésének minden körülmények között megbízhatónak kell lennie.

mindig van egy picinyke esély, hogy egy processzor (vagy egy kiszolgáló csomópont) téveszteni fog. Ebben a nagyon kivételes esetben képesnek kell lennünk újraindítani és újra üzemképes állapotba hozni ezt a részt, hogy amilyen gyorsan csak lehet, a lehető legkisebb kieséssel újra ki tudja szolgálni a kéréseket.
A naplózó fájlrendszerek iránti érdeklődésünk a távközlési Linux-rendszereken annak köszönhető, hogy az ilyen fájlrendszerek képesek a gyors újraindulásra. Egy esetleges meghibásodás esetén a naplózó fájlrendszer a fájlrendszert a lehető leggyorsabban képes újra összefüggővé tenni, ezáltal jóval megbízhatóbb, mint az egyéb fájlrendszertípusok.
Az IBM JFS-sel való kísérletezést valamikor 2000 elején kezdtük el. A JFS-csapat nagyon segítőkész volt, és képviselőjük, ***Steve Labs*** 2001 januárjában meglátogatta a laboratóriumunkat. Azóta szorosan követjük a JFS fejlesztését és kiszolgálóink mindig a legfrissebb változatot futtatják.
Első JFS-telepítéseinknek egy 1U lemeztömb adott otthont, melyben egy 500 MHz-es Celeron processzor dolgozott, 256 MB RAM-mal és 20 GB-os IDE-lemezekkel. Ezek az eszközök biztosították számunkra a munkakörnyezetet, amelyben a JFS-t kipróbálhattuk, és alkalmazásaink használata során tapasztalatokra tehettünk szert. Amióta a JFS 1.0.0-s változata 2001 júniusában megjelent, úgy döntöttünk, hogy kísérleti Linux-rendszerünkre is JFS-t telepítünk (*2. kép*).
Linux-rendszereink úgy lettek megtervezve, hogy rövidtranzakció-alapú kéréseket szolgáljanak ki. A JFS biztosít egy napló-alapú, bájtszintű fájlrendszert, amellyel elsősorban a tranzakció-középpontú rendszereket célozzák meg, ez illik a mi rendszerünkre is a legjobban.
Távközlési szempontból a JFS előnye, hogy a fájlrendszert nagyon összefüggő szerkezeti egységekben tárolja, könnyen helyreállítható, és sokkal gyorsabban képes újraindulni, mint nem naplózó társai, a hagyományos Unix-fájlrendszerek.
A rendszer meghibásodása esetén a többi fájlrendszer rendszerint könnyen megsérül. Az ilyen fájlrendszerek egy helyreállító eszközt tesznek szükségessé, mint amilyen az `fsck` is. Az ilyen helyreállító eszközök végigfésülik a fájlrendszer kiegészítő adatait, és kijavítanak minden hibát, amire a fájlrendszer szerkezetében rábukkannak. Ez a művelet azonban rendkívül időpazarló tevékenység, amelynek során könnyedén hiba léphet fel, és a legszerencsétlenebb esetben rossz helyre vagy rosszul állít vissza adatokat. A távközlési rendszerek nem engedhetnek meg maguknak olyan műveleteket, amelyek tovább növelik a kiesést.
A JFS-sel meghibásodás esetén a fájlrendszer egy összefüggő állapotba kerül vissza, úgy, hogy a fájlrendszer a megfelelő tranzakciókhoz megismétli a naplóban rögzült műveleteket. Az ilyen naplóalapú rendszereknél a helyreállításhoz szükséges idő lényegesen kevesebb, mivel a helyreállítás során nem kell az összes bejegyzést átfésülni, hanem csakis azokat, amelyek a leállás során működőek voltak.

Összegzés

A JFS-nek kulcsszerepe van, mivel egy esetleges rendszerleállást követően nagyon gyors fájlrendszer-újraindítást eredményez. A JFS-csapat elsődleges és legfontosabb célja, hogy megbízható, magas rendelkezésre állású fájlrendszert hozzon létre. A JFS-csapatnak nagy szerepe van a JFS Linuxra való átültetésében. A teljesítmény szempontjából sok különféle mérést alapul véve a JFS megint csak győztesen kerül ki. A JFS Project honlapjáról a developerWorksön további érdekességeket tudhatsz meg.

Köszönetnyilvánítás

Köszönet az Ericsson Research Open Systems Lab csapatának, hogy támogatták a Linuxszal és nyílt forrású rendszerekkel folytatott munkánkat.

A cikkhez kapcsolódó anyagok a 45. CD Magazin/JFS könyvtárában találhatóak.

Linux Journal 2003. január, 103. szám

Steve Best (sbest@us.imb.com)
Austinban (Texas), az IBM Linux Technology Centerben dolgozik. Jelenleg a Linux naplózó fájlrendszer projekten munkálkodik.

David Gordon (gordd00@dmi.usherb.ca)
A Québecben lévő Sherbrook Egyetem Számítástechnikai tanszékének hallgatója, épp most készül megszerezni a diplomáját. Az Ericsson Research Labjében segédfelügyelőként dolgozik.

Ibrahim Haddad (Ibrahim.Haddad@Ericsson)
Az Ericsson Corporate Unit of Research kutatója Montrealban. Harmadik nemezedékbeli vezeték nélküli IP-hálózatok felépítésével foglalkozik. Ibrahim képviseli az Ericssont az OSDL (Open Source Development Lab) műszaki alcsoportjában. Jelen pillanatban a Concordia Egyetem DrSc jelöltje.