A fájlleírók megzabolázása

Előfordul, hogy munkánk során egy-egy olyan hibával találkozunk, amelynek kiküszöböléséhez bele kell néznünk a rendszermag forrásaiba.

© Kiskapu Kft. Minden jog fenntartva

Tudtam, hogy a linuxos szakik gyorsabban raknak össze új fájlformátumokat és lemezterület-típusokat, mint ahogy Carlie az ingyenes italjegyeket osztogatta a Linux Journal partiján.

Minden egy telefonhívással kezdődött. „Tudnának olyan Linux-rendszert építeni, amely képes befűzni és olvasni egy DEC-meghajtót, és elérhetővé tenni az adatokat NT-munkaállomások számára Sambán keresztül?” A hívó a General Dynamics munkatársa volt. Ez a cég gyártja sok egyéb között az US haditengerészet atom-tengeralattjáróinak többségét, mi pedig egy haszonelvű linuxos vállalkozás vagyunk, a lehető leggyorsabban szerettem volna visszahívni.

Ekkor már jó néhány éve foglalkoztunk linuxos feladatok megoldásával, úgyhogy éppen a szakterületünkbe vágott a dolog. A Cosmos Engineering Company 1984 óta szállít rendelésre készülő számítógéprendszereket az ipar szereplői számára. 1996-ban kezdtünk el kizárólag Linux-rendszerekkel foglalkozni. Ugyanebben az évben mutattuk be a „Linux on a Disk” („Linux egy lemezen”) nevű rendszerünket. A következő évben a RedHat Hardware Partners alapítóivá váltunk.

A DEC-meghajtók Quantum 9 GB tárkapacitású SCSI-2 egységek voltak, OSF lemezrészekkel és UFS fájlrendszerrel. Bár akkor még nem voltam biztos benne, tudtam, hogy a linuxos szakik gyorsabban raknak össze új fájlformátumokat és lemezterület-típusokat, mint ahogy Carlie az ingyen italjegyeket osztogatta a Linux Journal találkozóján. Például a 2.2.15 és a 2.3.99pre9 között a támogatott idegen lemezterület-típusok száma háromról tizenötre nőtt. Az UFS a 2.0.xx változattól található meg a rendszermagban. Csakhogy az UFS-nek több változata is létezik, nem? Ezeknek a száma szintén tovább emelkedett a 2.4.0-használható-1 felé menetelés közben.

Így megnéztem a rendszermag pillanatnyi fejlesztői fáját, ez akkoriban a 2.3.99pre9 volt. Hét különféle UFS-típust támogatott ez a rendszer, bár a DEC-et külön nem említették, valamint az OFS-lemezrész támogatása is friss jövevény volt. Ezen felbuzdulva visszahívtam *John Loefflert* a General Dynamics Electronic Systemsnél. Egy óvatos igennel kezdtem, majd megkérdeztem: „Kaphatnánk egy meghajtót kipróbálásra, hogy pontos választ adhassunk?” Egy FedEx kézbesítéssel később felépítettem a legújabb fejlesztői rendszermagot, mely minden olyan fájlrendszert és lemezrész-típust támogatott, ami kicsit is hasonlított a számunkra szükségeshez.

Ahogy azt előre sejtettem, az OSF-lemezrész és a csak olvasható UFS-típusú Sun fájlrendszer támogatásával befűzött meghajtó működőképesnek látszott. Az ellenőrzőlemezen mindössze három fájl volt: rc.local, hosts és egy viszonylag nagy tároló, az oilpatch.tar. Ahogy kérték, elfaxoltuk nekik a két kisebbik fájl tartalmát, mintegy bizonyítékul, hogy a Cosmos Engineering Company el tudja vállalni a szóban forgó feladatra felkészített Cosmos 500 linuxos kiszolgálók összeállítását.

A kért gépek mindegyike négy, 9 gigás SCSI DEC OSF meghajtót képes kezelni, és lehetővé teszi a hálózaton kapcsolódó NT-kiszolgálóknak az adatok olvasását. Ez a feladat később tovább bővült a fájlok és könyvtárak törlésének lehetőségével. Rendben, semmi gond, a 2.4.0-test1 rendszermag rendelkezik kísérleti UFS-írás képességekkel. A gép semmi különös dolgot nem tartalmazott. Minden kiszolgáló egy 500 MHz-es Pentium III-as gép volt, ASUS i404BX ATX alaplapra ültetve, 128 MB memóriával egy közepes ATX toronyházban. A 18 GB IBM 2Ultra SCSI rendszerű meghajtóra RedHat Linux 6.1 került. Az egyetlen dolog, ami egyedivé tette a rendszert, hogy képes volt olvasni az idegen lemezformátumot. Ehhez kellett egyedi rendszermagot készíteni.

Meglehetősen lenyűgöző volt a papírmunka, ami egy olyan horderejű céggel történő megállapodáshoz kellett, amely rombolókhoz szokott alkatrészeket szállítani. Soha nem készítettünk olyan szerződést, amelyben ennyi kormányzati szabályozás szerepelt volna. No igen, olyat sem, amiben külön szabályozás lett volna arra, miképpen kell kezelni a fennmaradó összeget, amennyiben a költségek meghaladják az ötszázezer dollárt.

Az első ellenőrzőlemez, valamint számos azt követő, elég nyilvánvalóan valótlan adatokkal volt megtöltve. Soha nem tudtuk, mi a valódi adat és mi nem. Bármi is volt, az biztos, hogy sok volt belőle, és igencsak arra törekedtek, hogy a házon belüli adatforgalom abban a mederben folyjék, amit ők készítettek. Tudtuk, hogy a végfelhasználó az amerikai kormány, és a munkák során kiderült, hogy ez a hadsereget jelentette. Később, amikor segítettem a General Dinamicsnak megoldani egy SCSI lezárási gondot, elmondták, hogy a SCSI-meghajtók fémdobozokba vannak zárva, azokat pedig rázkódásmentes sínekben rögzítették. Elmondták, hogy ilyet használnak mindenhol. A „mindenhol” ebben az esetben az atom-tengeralattjárókat, rombolókat és egyéb fegyverrendszereket jelentette. Azt is megtudtam, hogy *Dr. Lee* üldözése közben, Los Alamosban eltűnt egy nukleáris titkokat tartalmazó táskagép merevlemeze azóta, sok figyelmet fordítottak kormányzati körökben a nemzet katonai adatainak biztosítására. Lehet, hogy ennek semmi köze nem volt a magától értetődő tömeges adatvándorláshoz. Ez azonban pusztán csak eszmefuttatás. Nem tudom, hogy a jövőben mire fogják a kiszolgálóinkat használni, de nem is akarom tudni. Amikor a General Dinamics-szal beszéltem, folyamatosan az volt az érzésem, hogy elmondhatják ugyan nekem, de akkor, sajnos kénytelenek lesznek… Nos, ugye sejtik, mire gondolok.

A szerződésnek megfelelően elkezdtük összeállítani és ellenőrizni az első kiszolgálókat. A munka azonban zátonyra futott, amikor kiderült, hogy a DEC-meghajtóról a hosszabb fájlok másolásával már gond van. A fájlok 96 kB-ra csonkolódtak, és ennek nem lett volna szabad megtörténnie. Miért pont 96 kB? Ezt szerettem volna kideríteni. Az eset további érdekessége volt, hogy a hosszú fájlokat – például a rendszermagtárolót – sértetlenül fel lehetett másolni a DEC-meghajtóra, majd visszamásolni. Ha azonban a felmásolás után a rendszert leállítottuk, majd újraindítás után másoltuk vissza a DEC-meghajtóról, akkor az eredmény helyes hosszúságú állomány volt ugyan, de a belsejében csak szemetet találtunk. Gyanítottam, hogy a gond a fájlleírók (inodes) memóriabeli kezelése és lemezen történő tárolása körül lehet.

© Kiskapu Kft. Minden jog fenntartva

Így megkezdtem hosszú utazásomat Leíróországban. Előzőleg már hallottam a fájlleírókról, és tudtam, hogy a fájlokhoz tartozó adatszerkezetek, ezenkívül mindig megfelelő mennyiségben szabadnak kell lenniük. Kaptál már valaha „no space on the device” (nincs szabad hely az eszközön) üzenetet, miközben 4 GB szabad helyed volt még? Ez történik, ha nincs elég szabad leíró. A leírók azért elég ködösek voltak nekem. A következő néhány héten azonban többet megtanultam róluk, mint amennyire valaha is emlékezni fogok.

Ezek a kicsiny adatszerkezetek határozzák meg a fájlok tulajdonságait. Az ext2 fájlrendszeren egy leíró 128 bit hosszú. Kényelmes helyzetben vagyok, ugyanis minden, ami ezután következik, egyaránt igaz a Linux ext2 rendszerére és a DEC UFS fájlrendszerre. A leíró valójában egy adattábla, amely megadja a fájl tulajdonosát, engedélyeit, készítésének és módosításának adatait, a fájlméretet, és ami a legfontosabb, a fájl adatainak lemezen elfoglalt helyét. Tulajdonképpen mindent, kivéve a fájlnevet. A fájlnév teljességgel mellékes a fájl szempontjából, csupán a mi kedvünkért van ott. Tulajdonképpen egy könyvtárbejegyzés. Mondjuk, ha szeretnénk meghallgatni egy dalt, és a The_Train_They_Call_the_City_of_New_Orleans.mp3-ra kattintunk, ez egy könyvtárbejegyzés, mely arra a leíróra mutat, ami tudja, hol van az a dal, és ki játszhatja le. Egy leíróra mutathat több könyvtárbejegyzés is, ezeket hívjuk közvetlen vagy egyenes hivatkozásnak (hard link). Egy fájlnak lehet egynél több neve is, de csak egyetlen leírója. Ez az, amiért a Sun berkein belül a fiúk azt mondják: „A leíró a fájl.” Minden fájlnak, ideértve az eszközöket, a könyvtárbejegyzéseket és közvetett hivatkozásokat (soft links), szüksége van leíróra.

Egy-egy lemezrészen viszont csak korlátozott számú fájlleíró van. Amikor a fájlrendszert létrehozzuk, meg kell határoznunk a készítendő leírók számát, és ez már nem változik, mindig ugyanannyi marad. Ha kifutunk a keretből, mert túl sok pici fájlt készítünk egy olyan lemezterületen, melynek kis leírótáblája van, akkor bizony trutyiban vagyunk, még akkor is, ha a szabad hely cicabájtokra rúg. A rendszert, amelyik egyik lemezterületén kifutott a szabad helyből, még megtévesztheted egy másik lemezterületre mutató közvetett hivatkozással. Azzal a rendszerrel azonban nem tehetsz semmit, amelyik kifogyott a szabad leírókból. Legalábbis addig, amíg le nem törölsz valamit, ezáltal fölszabadítva egyet. A közvetett hivatkozás – ez valójában egy fájl, mely egy másik könyvtárbejegyzésre mutat – is igényel leírót.

A leírók olyan adatszerkezetek, amelyek általában a lemezen tárolódnak, és a felhasználáshoz vagy módosításhoz a memóriába kerülnek.

Az adatszerkezetbe pillantva könnyű megérteni, miért is olyan lényeges a 96 kB.

A leíróban tárolt adat első blokkja foglalkozik az idővel, a dátummal, a tulajdonjogokkal stb. Ezután, a 0x28 eltolási címen találjuk azt, ami a lényeg – a fájl adatainak lemezen elfoglalt helyét. Említettem már, hogy különféle leírótípusok tartoznak a különféle fájltípusokhoz, és hogy most csak a DEC UFS szabványos fájlokhoz tartozó fájlleírókról beszélünk?

Az ext2 vagy DEC UFS 32 bites fájlrendszer adatszerkezetében egy négybájtos szó mutat a fájlrendszer adatblokkjainak egyikére. E rész első 48 bájtja tizenkét négybájtos mutatót tartalmaz a fájl adatainak első tizenkét blokkjára. E fájlrendszer esetében minden adatblokk 8 kB hosszú. Ezeket a blokkokat egyenes blokkoknak (direct blocks) nevezik, mivel a címük közvetlenül a leírókban tárolható. Ez egyúttal azt is jelenti, hogy a kisméretű állományok (96 kB vagy kisebbek) gyorsabban érhetők el. Nagyobb fájlok esetén, a leíróban található 13. címmező már nem egy 8 kilobájtos adatblokkra mutat, hanem egy 8 kilobájt méretű címtáblára, azaz egy olyan címmezőre, ami 2048 újabb 8 kB-os adatblokkra mutat. Ezeket áttételes blokkoknak (indirect blocks) nevezik. Még nagyobb fájlok esetében a leíró 14. címmezője egy olyan 8 kB-os blokkra mutat, amelyben mind a 2048 címmező újabb 8 kB-os címblokkra mutat. Ezeket kétszeresen áttételes blokkoknak nevezik. Ez valóban nagy fájlméretet tesz lehetővé.

Az volt a gondom, hogy a Linux csak az egyenes blokkokat tudta olvasni a DEC fájl adataiból. Minden olyan fájlolvasási kísérlet, ahol az áttételes blokkokat is használni kellett volna, „elérési kísérlet az eszköz végén túlra” („attempt to access beyond end of device”) hibajelenséget okozott.

Két dologra lett volna szükségem a feladat megoldásához.

1. A lemezen elhelyezkedő adat szerkezetének megértésére, különös tekintettel a leírók szerkezetére, valamint a címadat tárolására.
2. Világosan átlátni, hogy az operációs rendszer mit tesz azzal az adattal, amit a lemezen talál, és legfőképpen miképpen olvassa a fájlleírót.

A kettő közül egyikkel sem voltam tisztában, viszont segítségképpen ott volt a Linux-közösség. Már a munka megkezdésénél tájékoztattam a felhasználói csoportom (Linux Users Los Angeles) tagjait arról, hogy milyen kihívással kell szembenéznem. Ennek hatására számos ember osztotta meg velem megérzéseit és javaslatait. *Christopher Smith* még ennél is tovább ment, küldött nekem egy levelet: „Ha szándékodban áll kinyuvasztani az egyik ilyen meghajtót és a SCSI vezérlőt, esetleg elszórakozhatok vele tengernyi szabadidőmben.” Átvittem neki egy kiszolgálót és a General Dynamicstól kapott egyik próbameghajtó másolatát.

Segítsége felbecsülhetetlennek bizonyult a feladat megoldásában. *Dan Kegel* azt ajánlotta, hogy írjak a rendszermag listájára, levele megadta a bátorságot a cikk elküldéséhez. Ő mutatta meg a linux-fsdevel (linuxos fájlrendszerek fejlesztésével foglalkozó) levelezőlistát is. Igen sok segítséget kaptam mindkét listáról.

Peter Swain írta: „Úgy tűnik, az áttételes blokkkezelés zavarodott meg, az „endian” vagy a 64 bitesség miatt. Lehet, hogy a DEC egy nem szabványos megvalósítást használ, de te biztos használhatod, ha máshogy nem, akkor egy befűzési kapcsolóval.” *Jim Nance* szintén hasonlóan gondolkodott: „Úgy hangzik, mint valami 32/64 bit hiba, esetleg valamilyen bájtsorrend a gond.” Azt is javasolta, hogy vizsgáljuk meg a rendszermag felhasználói módba ültetésével, és elküldte nekünk, hogy hol tudunk ebbe az irányba elindulni. *Peter Rival* (Tru64 QMG Performance Engineering) azt ajánlotta, hogy lépjünk kapcsolatba *Marcus Barrow*-val (Mission Critical Linux), akit a következőképpen jellemzett: „az MCL-hez pártolása előtt ő volt a helyi UFS-nagyszaki”.

Marcus Barrow e szavakkal válaszolt a levelemre: „Örömmel vetnék rá egy pillantást.” „Ne írj mindhárom próbalemezre Linux alól, ugyanis a fájlrendszered meghibásodhat” – figyelmeztetett. Természetesen akkor már klónozott meghajtókat használtunk. Marcus a tapasztaltkról is részletesen írt: Először azt gondoltam, hogy a gondokat a 8k/1k blokk/darab kezelése okozza. Második körben még jobban összezavarodtam. Nem értem, miért tudod olvasni az egyenes blokkokat, és miért nem az áttételeseket. Különösen, hogy a Linux képes a saját áttételes blokkjait olvasni.

Lye Seaman rámutatott: „A feladat lényegesen egyszerűbb lehetne, ha meglennének a megfelelő fájlok az idegen OS /usr/include/fs/* könyvtárából… Biztos megvannak valahol, valakinek a múzeumában.” A támogatás, amit ennek a két listának a tagjai nyújtottak, fontos szerepet játszott abban, hogy végül be tudtuk fejezni ezt a munkát.

Megvolt továbbá a rendszermag forráskódja, és tulajdonképpen ez tette lehetővé a megoldást. A szabad program azt jelenti, hogy követhető, mi zajlik a rendszerben és ennek köszönhetően módosítható is. Kinyomtattam a lényeges fájlokat, a linux/fs/ufs/inode.c-t

és hozzá tartozó társait, majd felütöttem a könyveket. A kutatómunka során találtam pár kitűnő anyagot a Szabad Forrás Közösségében, ezek bemutatták a fájlrendszereket, valamint hogy milyen programokat kedvelnek. (Lásd Javasolt oktatóanyagok.)
Elméletben már értettem, hogy mit csinál az OS, hiszen birtokomban volt a forráskód, amit futtattam, és nyitva állt a lehetőség, hogy megváltoztassam és új futtatható állományokat készítsek. Ahhoz, hogy megértsem, miképpen épül fel a fájlrendszer, tudnom kellett olvasni a lemezen található adatot, és meg kellett értenem, hogyan szerveződik, valamint meg kellett néznem, miképpen épül fel ténylegesen egy fájlleíró a célmeghajtón. Ki akartam olvasni egyet, és látni, hogy mi módon mutat az áttételes blokk a fájl hiányzó adataira, ha egyáltalán oda mutat.
Ekkorra már a leírók elég valóságosnak tűntek számomra. Kiterveltem, hogy elkapok egyet, nevezetesen a rakoncátlan oilpatch.tar leíróját. Mivel be tudtam fűzni a lemezterületet, majd pedig újra tudtam fűzni írható-olvasható módon a rendszermag kísérleti változatával, meg tudtam változtatni az oilpatch.tar jogait. Meg is tettem. Tudtam, hogy az általam keresett résznek valahol a lemez elején kell elhelyezkednie, ezért kimásoltam ezt a részt egy fájlba a következő paranccsal:

```
dd if=/dev/sda of=root-patch
  ➥bs=1k count=100000
```

Ezután ismét befűztem a DEC lemezterületet, megváltoztattam az oilpatch.tar tulajdonosát rendszergazdáról (ID 0) senkire (nobody) (ID 99), majd leválasztottam, és készítettem egy újabb fájlt ezzel a paranccsal:

```
dd if=/dev/sda of=nobody-patch
  ➥bs=1k count=100000
```

A két fájl közti különbségkeresés kiderítette, hogy egy bájtérték bizonyos eltolási értéknél a fájlokban 0-ról 99-re változott. A leírón belül a négybájtos fájltulajdonos-azonosító ismert helyen van, így a helyes értékekre beállított `dd` parancs a következő lehet:

```
dd if=/dev/sda
  ➥of=inodecopy_for_oilpatch.tar
  ➥bs=128 count=1
  ➥skip=offset_from_front
```

Ez kiírja a fájlleírót egy állományba, amit így részletesen megvizsgálhattam. Ki tudtam nyomtatni, elolvastam a rejtett üzeneteit és megtaláltam a fájladatokat. Szerencsére az oilpatch.tar könnyen átlátható szöveges állománynak bizonyult. A szöveg darabjai úgy illettek egymáshoz, mint egy kirakójáték darabkái, ez pedig igen hasznos, ha a helyes sorrendet szeretnénk megállapítani.
A következőket állapítottam meg: az első 12 mutató a fájl első 12 adatblokkjára mutatott, a blokk első négy bájtja a 13. mutatóra mutatott, ami pedig a fájl 13. blokkjára. A mutatók blokkjának következő négy bájtja a fájladatok 14. blokkjára mutatott és így tovább.
Nem akadt sehol semmi furcsaság a működésben: semmi vad „nagy endian, kis endian” gond, sehol egy váratlan bitcsere. Minden nagyon szép és következetes. Pontosan olyan volt, ahogyan én is megvalósítottam volna. Valójában a Linux is ezt teszi az ext2 fájlrendszerben, valamint ezt várja a rendszermag UFS kódja is, viszont ez nemigen segített megmagyarázni, tulajdonképpen miért nem működik. Sok időt töltöttem a hiba keresésével, ami nem is létezik. Lassan levontam a következtetést, hogy a hiba egyáltalán nem a rendszermag UFS-kódjában van. A leírót úgy olvasta be, ahogy azt kell. A DEC UFS-rendszert ufs_type=sun-ként lehet beolvasni. Igazság szerint az UFS-kód már jó ideje közkézen forgott, ezért nehezen tartottam elképzelhetőnek, hogy hibás. A hiba mélyebben gyökeredzik: a Linux virtuális fájlrendszer (VFS), valamint a 2.4.0-test1 rendszermag és UFS-kód közötti kapcsolattartási hibában, amit a felsőbb szintű kód megváltoztatása okozott. Az első áttételes blokk mutatójának átadása történt hibásan, ez már a VFS-hez is rosszul került el, és ez rontotta el az UFS-kódot.
Az elképzelés ellenőrzése egyben a hiba kiküszöbölését is jelentette számomra. Először a 2.3.99-es majd a 2.4.-test sorozatú rendszermagokat használtam, mivel ezekben megtalálható az OSF lemezrészkezelés, mely a 2.2 változatból hiányzik. Ha igazam volt, és az UFS-kód csak mostanában romlott el, a 2.2 rendszermag hiba nélkül fogja olvasni a DEC-meghajtót, feltéve, ha tudja kezelni a lemez felosztási tábláját. Az OSF-felosztást kezelő kód visszaültetése 2.4-es rendszermagról 2.2.16 forrásba igen könnyű volt – még számomra is –, és az eredményül kapott 2.2.16-os rendszermag mindent hibátlanul hajtott végre a DEC OFS-meghajtókon, beleértve a nagy fájlok naphosszat tartó írását és olvasását. Mivel a 2.2.16 jobb választás egy ipari környezetben, ráadásul én is ilyen feladatra szántam, továbbléphettünk, és be tudtuk fejezni a munkát. Ezután elkezdhettük szállítani a 2.2.16-os rendszermaggal ellátott gépeket a General Dynamics és ügyfelei számára.
Természetesen megjelentettem a 2.4.0-test1-ben fellelhető hibát a linuxkernel és a linux-fsdevel listán. Írtam, hogy „sajnos, találtam egy hibát”. *Alan Cox* azonban így válaszolt: „Egy hiba felfedezése mindig jó hír. Nagy valószínűséggel ez a hiba nem derült volna ki a 2.4.0 kiadása előtt.”
Nagyjából ennyi a történet.
A munka során a későbbiekben még néhány apróbb megoldásra váró feladat akadt. Ilyen volt, amikor az első rendszerünk már működött, akadt még egy kis gond azzal, hogy a Windows 2000 munkaállomások a hosszú fájlneveket kezelik, de a sambás fiúk felkészültek a Microsoft legújabb trükkjeire, így a legújabb Samba-változatokra frissítés megoldotta a gondot.
A továbbiakban rögzített kapcsolatokat kellett létrehoznunk a meghatározott SCSI-azonosítójú DEC-meghajtók és a befűzési pontok közt, a telepített DEC-meghajtók számától függetlenül. Mivel a Linux a SCSI merevlemezeket sda, sdb, sdc neveken szereti emlegetni, olyan sorrendben, ahogy megtalálja őket, a lemezek befűzését egy kisebb programmal kellett megoldani. Az eredeti Tekram vezérlők nem igazán örültek a külső rázkódásmentesített kereteknek, így Adaptecre cseréltük azokat. Ezek természetesen átlagos kihívásoknak számítanak ebben a szakmában. A valódit a fájlleírók megrendszabályozása jelentette.

© Kiskapu Kft. Minden jog fenntartva

Javasolt oktatóanyagok

Rémy Cardtól a Design and Implementation of the Second Extended File System (Laboratoire MASI–Institut Blaise Pascal) ➲ card@masi.ibp.fr Theodore Ts'o (Massachusetts Institute of Technology) ➲ tytso@mit.edu és Stephen Tweedie (University of Edinburgh) ➲ sct@dcs.ed.ac.uk ➲ http://khg.redhat.com/HyperNews/get/fs/ext2intro.html

David A. Ruslingtól a The Linux Kernel ➲ david.rusling@arm.com, Chapter 9: The File System ➲ http://www.linuxdoc.org/LDP/tlk/tlk.html

Clay J. Claiborne, Jr. (cjc@linuxbeach.net) a Cosmos Engineering Company munkatársa. Harminc éve dolgozik a számítógépiparban. 1995 óta Linux-rajongó, 1996-ban megalkotta a Linux on Disket. Ezenkívül Linux-oktató a Los Angeles City College-ben, valamint a Linux Users Los Angeles (LULA) elnöke.