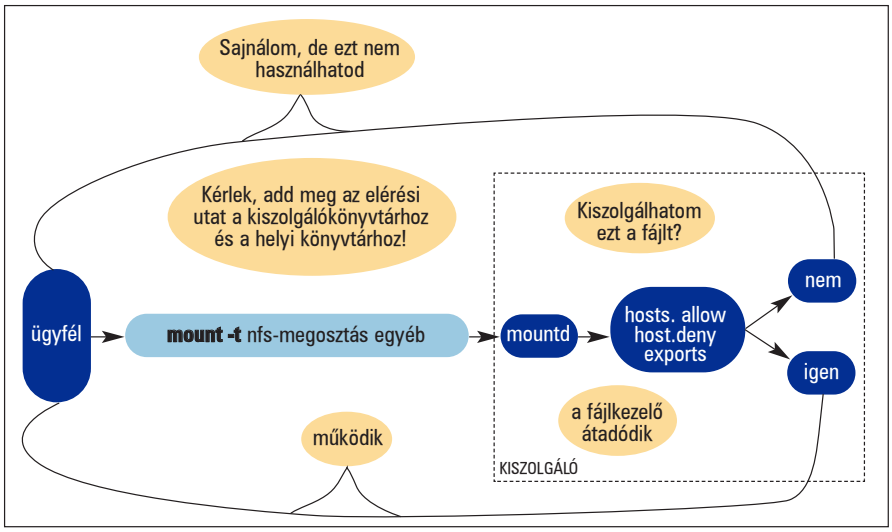
Fájlmegosztás NFS-sel

Olexij és Denis az NFS hálózati fájlrendszert mutatja be.

Előfordulhat, hogy több gépünk van, és szeretnénk a lemezterületeket, a különböző eszközöket vagy valamelyik CD-meghajtót megosztani. Erre használatosak a hálózati fájlrendszerek, például az NFS, amellyel az állományok és a források hálózati megosztása könnyen megvalósítható.
Az NFS használatával úgy dolgozhatunk egy távoli gép fájljaival, mintha a saját gépünkön lennének; a például tulajdonképpen a hálózat bármely gépéről ugyanazt a könyvtárat használhatjuk saját könyvtárként. Nagy előnye még, hogy nem kell több gépre másolgatni a fájljainkat, így nagy lemezterületeket takaríthatunk meg.
Az NFS-t a Sun Microsystems mérnökei vezették be 1985-ben. Igaz, régi, de még mindig jó, ráadásul folyamatosan fejlesztik, javítgatják. A Sun mostanában a linuxos NFS négyes kiadásán dolgozik. Mi most ugyanennek a harmadik kiadását ismertetjük – egy felhasználó számára ugyanis az egyes kiadások között nincs sok különbség.
Az NFS-t sem az ügyfél-, sem a kiszolgálógépre nem túl bonyolult telepíteni. Bemutatunk néhány egyszerű lehetőséget, de vigyázat: a legtöbb parancsot rendszergazdaként kell végrehajtani! Nézzük, hogyan is működik! Az NFS a legismertebb szolgáltatás, amely távoli eljáráshívást (RPC) használ. Például legyen a kiszolgáló neve ***tiger***, és a saját könyvtárak legyenek a ***/home*** alatt. ***Blackcat*** nevű gépünkről adjuk ki a következő parancsot:

```
mount -t nfs tiger:/home /home
```

A `mount` a fenti parancs alapján távoli eljáráshívás segítségével rákapcsolódik a kiszolgálógép `rpc.mountd` démonjára. A kiszolgáló ellenőrzi, hogy a ***blackcat*** jogosult-e hozzáférni a ***/home*** könyvtárhoz. Ha igen, akkor egy fájlkezelőt küld vissza, amelynek segítségével elérhetjük a ***/home*** könyvtár tartalmát. Ha nem szabad, hibaüzenetet ír ki. A kettőspont már jelzi a parancsban, hogy távoli fájlrendszert akarunk befűzni, így a `-t nfs` kapcsolót akár el is hagyhatjuk. Ha megkapjuk a fájlkezelőt, máris megszelídítettük a tigrist. A befűzés folyamata az *1. ábrán* látható.
Amikor a ***blackcat*** fájlt kér a hálózati fájlrendszerről, a rendszermag távoli eljáráshívást intéz az NFS-démon felé, amely általában az `rpc.nfsd`. A hívás tartalmazza a fájl nevét, felhasználó- és csoportazonosítóját, a démon ez alapján dönt, hogy kiszolgálja-e a kérést.



1. ábra Egy NFS-megosztás befűzése

Az NFS-kiszolgáló beállítása

Ha linuxos gépet szeretnénk NFS-kiszolgálóként használni, az `rpc.mountd` programot kell futtatnunk. A program segíti a befűzés folyamatát, majd a hívást átadja az `rpc.nfsd` programnak, amely minden további kiszolgálófeladatot elvégez. Az RPC-protokoll támogatja, hogy a kiszolgálógép rendszermagja teljesítményadatokat adjon ki, ami nagyban segíti az `rpc.lockd` és az `rpc.rquotad` munkáját. A 2.2.18-as és ennél újabb rendszermagok esetében az `rpc.nfsd` magától elindítja az `rpc.lockd` programot, tehát nem nekünk kell külön megtennünk. Tárkorlátok figyeléséhez az `rpc.quotad` démont kell futtatnunk.
Az eljáráshívás előnye azt jelenti, hogy az `inetd` internet-kiszolgáló ***/etc/inetd.conf*** fájljába nem kell beleírnunk.
Az NFS egy másik programot használ, a kapu-hozzárendelőt, (portmapper), amely segít megtalálni a hálózat NFS-szolgáltatásait. Az első védelmi vonalat, amely rendszerünket vigyázza, a ***/etc/hosts.allow*** és a ***/etc/hosts.deny*** nevű fájlok alkotják. Amennyiben a beérkező kérelemnek megfelelő bejegyzés szerepel a host.allow fájlban, a kérelem továbbjut. Ha itt nincs, az ellenőrzés a host.deny vizsgálatával folytatódik. Ha itt szerepel egy illeszkedő bejegyzés, a kérést a kiszolgáló elutasítja. Ha sem a ***/etc/hosts.allow***, sem a ***/etc/hosts.deny*** fájlban nincs megfelelő bejegyzés, a kérelmet akkor is továbbítja.
Ha a kiszolgálót például egy oktatási intézmény részére telepítjük, jobban oda kell figyelnünk a biztonságra. A hallgatók sok-sok gonoszságot művelhetnek: ellophatják mások jelszavát, „I love you” tárgyú leveleket küldözgethetnek vagy trójai falovakkal ajándékozhatják meg társaikat. Az NFS, főleg a régebbi kiadások, jókora biztonsági hézagot jelenthetnek, tehát jobb, ha a beállításkor odafigyelünk.
Az NFS súgóoldalán többet is megtudhatunk erről. Igaz, általában elég, ha az összes gépet letiltjuk, és csak néhányat engedélyezünk. A következő sorokat tegyük a ***/etc/hosts.deny*** nevű fájlba:

```
portmap:ALL
lockd:ALL
mountd:ALL
rquotad:ALL
statd:ALL
```

Ezek után senki nem tud az NFS-kiszolgálóra kapcsolódni, de mivel ezt nyilván nem akarjuk, nézzük a */etc/hosts.allow* állományt. A fájl szerkezete a következő:

```
dØmonlista: ... felhasznÆl @gØp-minta
```

Ha például a ***blackcat*** (`192.168.16.13`) és a ***tomcat*** (`192.168.16.24`) gépeknek engedélyezni akarjuk a hálózati fájlrendszerhez való hozzáférést, a következő sorokat adjuk a */etc/hosts.allow* fájlhoz:

```
portmap  :   192.168.16.13  192.168.16.24
lockd    :   192.168.16.13  192.168.16.24
mountd   :   192.168.16.13  192.168.16.24
rquotad  :   192.168.16.13  192.168.16.24
statd    :   192.168.16.13  192.168.16.24
```

Ha egy adott alhálózaton lévő összes gépet engedélyezni akarjuk, egy ilyenfajta lista meglehetősen hosszúra nőhet, ezért az egyszerűség kedvéért a szóban forgó gépek címét a következőképpen is megadhatjuk: az `192.168.16.0/255.255.255` számsorozat hozzáadásával például az összes olyan gépet engedélyezzük, amelynek IP-címe `192.168.16.0` és `192.168.16.255` közé esik.
Ez idáig csak nagyon általános lehetőségeket említettünk. Az NFS-szolgáltatás beállítófájlja (/etc/exports) az eddigieknél egyedibb. A fent említett két gép számára megnyitjuk a */home* és a */usr/doc* könyvtárakat:

```
/home     192.168.16.11(rw,root_squash)
➥192.168.16.24(rw)
/usr/doc 192.168.16.11(ro,root_squash)
➥192.168.16.24(ro)
```

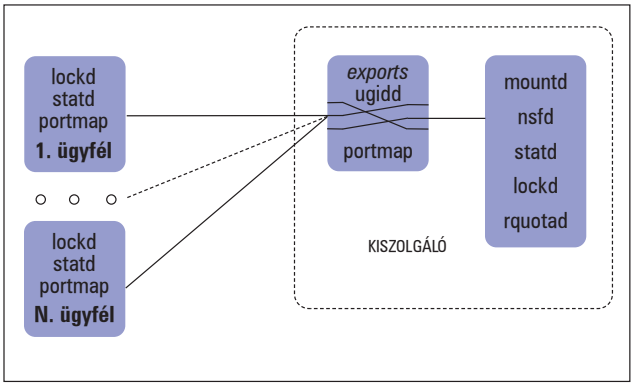
A fájl szerkezete egyszerű. A bal oldali mezőbe kerül a könyvtár neve, majd a gép IP-címe (zárójelben a jogosultságokkal). Fontos, hogy az IP-cím után nincs szóköz! A fenti példában írási és olvasási engedélyünk van a */home* könyvtárra, de a */usr/doc* könyvtárra csak olvasási jogunk.
A `root_squash` érték ahhoz kell, hogy megakadályozzuk az ügyfélgépről történő rendszergazdai hozzáférést. Ez azt jelenti, hogy hiába szerezte meg egy felhasználó, mondjuk egy hallgató az ügyfélgép rendszergazdai jelszavát, nem tud rendszergazdaként dolgozni a kiszolgálón lévő fájlokkal. Ez az alapértelmezett beállítás. A `squash` szó itt azt jelenti, hogy a rendszergazda csak a nobody nevű felhasználó jogait birtokolja.
Ha jól átgondoljuk a rendszergazdai hozzáférés példáját, észrevehetjük, hogy valamit a felhasználó- és csoportazonosítókkal kapcsolatban is tennünk kell. Ha ugyanis egy felhasználónak különböző felhasználóazonosítója (UID) és csoportazonosítója (GID) van az ügyfélgépen, mint a kiszolgálón, előfordulhat, hogy nincs joga minden állományát elérni, ellenben más felhasználók fájljaihoz teljes jogokat szerezhet.
Elkerülhetjük ezt a gondot, ha fájljainkra állandó felhasználó- és csoportazonosító hozzárendelést állítunk be. A következő példában ezt a */etc/nfs/home_map.cats* fájlban állítjuk be:

```
map_static=/etc/nfs/home map.cats
```

A hozzárendelést szabályozó fájl, a ***home_map.cats*** így nézzen ki:

```
# SzabÆlyok a gØpekhez:
#     kiszolgÆl          gyfØl
uid    0-60           -     # nobody lesz
uid    1061-1080   1041     # hozzÆrendelj k
                            # a 61-80 a 41-60-hoz
```

Az ügyfélen és a kiszolgálón lévő démonok adatcseréjét a *2. ábra* mutatja.



2. ábra Démonok közötti adatátvitel

Ha hálózatunkon működik NIS-kiszolgáló, a rendszer képes a NIS-szolgáltatást is használni az UID-GID kezeléséhez. Ekkor mást kell beállítanunk: `map_nis=kingdom.cat`.
Ez jobb megoldás, mert az NFS-kiszolgáló a szükséges adatokat a NIS `kindgom.cat` körzetnevének segítségével szerzi meg.
A NIS használatakor csoportokat is megadhatunk. Ha például a fájlhoz hozzáadjuk a `@macskatulajdonosok` értéket, akkor a `macskatulajdonosok` csoportjába tartozó felhasználók neve hozzáadódik az engedélyezettek listájához.
Miután az összes fájlt megfelelően beállítottunk, indítsuk el a szükséges alkalmazásokat. Mivel az NFS kapu-hozzárendelőt használ, először ezt kell indítanunk. Az újabb Linux-kiadásokban a `portmap` vagy az `rcp.portmap` rendszerindításkor elindul, amit a következő parancssorral könnyen leellenőrizhetünk:

```
ps axu | egrep portmap
```

Ha a `portmap` fut, a szabványos kimenetre valami ilyesmit ír ki:

```
daemon 99     0.0 0.3 1132 500 ?      S Jul11
➥0:02 /sbin/portmap
tiger  27837 0.0 0.3 1264 492 pts/0 R 17:03
➥0:00 egrep portmap
```

Az első sor azt mutatja, hogy a `portmap` nevű alkalmazás fut. Ha a kapu-hozzárendelő nem fut, indítsuk el saját magunk. Legtöbbször a */sbin* könyvtárban található, de ha nem, nézzük meg a */usr/sbin* könyvtárban. Ha ott sem lenne, akkor a programot telepítenünk kell, amely a ***netbase*** vagy a ***nkitb*** nevű csomagban van.
Ha a kapu-hozzárendelő már fut, indítsuk el a szükséges démonokat, pontosan ebben a sorrendben: `rpc.mountd`,

`rpc.nfsd`, `rpc.statd`, `rpc.lockd` és `rpc.quotad`. Az újabb kiadásokban megfelelő héjprogramokkal indíthatjuk őket. Debian alatt például a `/etc/init.d/nfs-kernel start` parancsot használhatjuk. Ha ezek a csomagok véletlenül hiányoznának, a fenti kiadásokból kölcsönvett programokat módosíthatjuk, vagy akár magunk is megírhatjuk őket. Az `rpc.quotad` démon csak akkor kell, ha a felhasználók lemezterület-felhasználását ellenőrizni akarjuk. Újabb rendszermagok esetében – ha kell – az `rpc.nfsd` démon hívja az `rpc.lockd`-t, de ha külön elindítjuk, akkor sem lesz semmi gond.
Miután minden programot elindítottunk, hajtassuk végre a következő parancsot: `rpcinfo -p`. Az utasítás kiírja futó programjainkat a kiadásuk számával és egyéb hasznos adatokkal együtt. A kapu-hozzárendelő (portmapper), a `mountd` és az `nfsd` mindenképp fusson, hogy a hálózati fájlrendszer-kiszolgálót használni tudjuk. A Linux NFS HOGYAN-ja segítséget nyújthat, ha véletlenül elakadnánk.

Az NFS-ügyfél beállítása

Győződjünk meg róla, hogy van-e a rendszerünkben NFS-támogatás. Nézzünk bele a */proc* könyvtárba:

```
tomcat>    cat /proc/filesystems
           ext2
nodev      proc
nodev      devpts
           iso9660
nodev      nfs
```

Az itt szereplő fájlrendszertípusokat támogatja a rendszermagunkkal. Ha a `nodev nfs` hiányzik, az azt jelenti, hogy az NFS-fájlrendszert támogató modul hiányzik. Ez esetben rendszergazdaként a következő paranccsal próbálkozzunk: `modprobe nfs`. Ha modul megtalálható, a paranccsal telepítettük is, így ha a */proc/filesystems* fájlt újra kiíratjuk, a kívánt sor szerepelni fog. Ha rendszerünk az NFS-t nem támogatja, nincs mese, rendszermagot kell fordítani, de ennek ismertetése túlmutat a jelen írás keretein.
Ha létezik NFS-támogatás, akkor a hálózati fájlrendszereket a példában vázolt módon tudjuk befűzni. A parancs szerkezete a következő:

```
mount -t nfs kiszolgÆl :k nyvtÆr_elØrØsi_œtja
➥helyi k nyvtÆr kapcsol k
```

A `kiszolgÆl` helyére az NFS-kiszolgáló gép nevét írjuk, a `k nyvtÆr_elØrØsi_œtja` helyébe pedig a befűzni kívánt könyvtár teljes elérési útját.
Nagyjából ennyi az egész, de hogy hálózati fájlrendszerünk befűzése jobban működjön, nézzük meg, milyen lehetőségeket tudunk beállítani. Később biztosan hasznunkra válik:

- `rsize=n` és `wsize=n`: az olvasásra és írásra vonatkozóan beállítja az egyszerre elküldött és fogadott adatok mennyiségét – ebben a sorrendben. Az alapértelmezés rendszermagkiadástól függ, és általában 1024 bájt pozitív egész szám többszöröse. Minél nagyobb adatmennyiséget tud egyszerre feldolgozni, annál hamarabb végez, így jó, ha nagyobb értéket állítunk be. 4096-ra vagy 8192-re állítva sokat javulhat a szolgáltatás sebessége.
- `timeo=n` és `retrans=n`: az első érték azt mutatja, hogy az NFS-ügyfél hány tizedmásodpercenként próbálkozzon újra csatlakozni a kiszolgálóra, ha a hálózat vagy a kiszolgáló épp nem érhető el. Az alapértelmezés 7. Ha a próbálkozási idő elérte a 60 másodpercet, megkétszerezi ezt a számot és újraindítja a kérelmet. A második szám azt mutatja, hogy ez utóbbit, mármint a megkétszerező-újraindító folyamatot az NFS-ügyfél hányszor tegye meg. Alapértelmezés szerint ez az érték 3.
- `hard` és `soft`: beállítja, hogy mit tegyen az NFS-ügyfél, ha nem éri el az NFS-kiszolgálót. Ha a `hard`-ot választjuk, a következő válasz mellett tovább próbálkozik: `server not responding, still trying`. Így ha a kiszolgáló újra elérhetővé válna, a folyamat onnan folytatódik, ahol abbamaradt. Ha viszont a `soft`-ot választjuk, akkor az NFS-ügyfél be- és kimeneti hibát jelez, ami hibás fájlokat eredményezhet. Vigyázzunk tehát ezzel a lehetőséggel.
- `intr`: megszakíthatjuk az NFS-hívást, ami akkor hasznos, ha a kiszolgáló sokáig nem érhető el.

Az `intr` és `hard` az ajánlott beállítás, és mivel a `hard` az alapértelmezett, elég az `intr`-t megadni. Ezekkel az értékekkel valószínűleg javítottunk az NFS-kapcsolat teljesítményén.
Hálózati fájlrendszert rendszerindításkor is befűzhetünk, ha a */etc/fstab* fájlt módosítjuk. Ennek minden sora legalább négy, de általában hat mezőből áll:

1. eszköznév,
2. befűzési pont,
3. a fájlrendszer típusa,
4. kapcsolók,
5. dump,
6. az ellenőrzés sorrendje.

```
tiger:/nfs1/home /home nfs rw,hard,intr 0 0
```

Az első érték a befűzendő eszközt azonosítja. Példánkban a tiger nevű NFS-kiszolgáló */nfs1/home* könyvtára lesz a */home* könyvtár. A negyedik mezőben a fájlrendszer típusát adjuk meg, ami jelen esetben `nfs`. Az ötödik mezőt a régi dump adatmentő rendszer számára tartják fenn, a hatodik mező pedig megadja, hogy az eszköz a rendszer főkönyvtára (1), egyéb ellenőrziendő könyvtár (2), vagy nem ellenőrizendő (0). Indítsuk el még az `rpc.lockd` és az `rpc.statd` démonokat, amit héjprogramok segítségével tehetünk meg. Ezennel beállítottunk egy működő NFS-ügyfelet.

Hálózati fájlrendszerünk tervezése

A kérdés az, hogy miféle adatokat tehetünk a hálózati fájlrendszerre? A válasz egyszerű: mindenfélét, így hálózatunkon lemeznélküli gépek is működhetnek. A rendszer tervezésénél figyelembe kell vennünk, hogy kétféle fájlról beszélhetünk: léteznek megosztható és nem megosztható adatok. A második típusba tartozó adatokat a saját gépünkön kell tartani – például az eszközfájlokat nem lehet megosztani.
Kisebb rendszer esetében a */home* könyvtárat tehetjük a kiszolgálógépre. Közepes és nagy rendszerek építésekor a */home* könyvtárat feloszthatjuk. A felhasználói könyvtárak mérete nem állandó, de ezt tárkorlátrendszerrel szabályozni tudjuk.
Mi a helyzet más könyvtárak esetében? Néhányuk állandó méretű, némelyik folyamatosan változik. A megosztható állandó méretű könyvtárak közül a */usr* és a */opt* említhető meg, de az állandó méretűek közül például nem osztható meg például a */etc* és a */boot*. A folyamatosan változó könyvtárak közül megosztható a */var/mail* és a */var/spool/news*, de nem lehet megosztani a */var/run* és a */var/lock* könyvtárakat.
Sokan a */usr/bin* és a */bin* könyvtárakat nem teszik hálózati fájlrendszerre, mert ekkor lassabban lehet elérni őket. Úgy gon-

doljuk, egy átlagos felhasználó nem veszi észre a különbséget egy hálózati fájlrendszerről hívott program és egy helyi lemezről indított alkalmazás között. Így tehát majdnem minden fájl feltehető az NFS-kiszolgáló lemezeire, hogy hely jusson MP3-fájljaink vagy akár egy másik operációs rendszer számára. Igaz, akármit tehetünk a hálózati fájlrendszerre, a szokás mégis az, hogy az NFS-kiszolgálón a felhasználói könyvtárakat, a leírásokat és a súgórendszert tartjuk. A ritkábban használatos programokat is nyugodtan a kiszolgálóra tehetjük, így például a */opt* és a */usr/local* könyvtárak tartalmát. A */usr/doc* és a */usr/share* könyvtárakat úgyszintén megoszthatjuk.
Miféle kapcsolókat kell használnunk a `mount`-tal ezekben az esetekben? Mint említettük, a felhasználóikönyvtárakra az `intr` és a `hard` beállítás ajánlott. Így biztosak lehetünk benne, hogy nem vész el adat. A */usr/doc* könyvtárak esetében viszont nem biztos, hogy ez a megfelelő beállítás. Ha például a súgórendszert próbáljuk elérni, de a kiszolgáló nem működik, feleslegesen terheljük a hálózatot. Ez esetben jobb a `soft` beállítás, miáltal a folyamat a hibaüzenet megjelenésével megszakad. Hasonlóan járjunk el a többi, nem túl fontos fájl esetében is. Leveleinket inkább más szolgáltatásokkal érjük el, használjunk IMAP-ot, POP-ot vagy NNTP-t.

Az NFS-kapcsolat sebességének javítása

Állítsuk be a legmegfelelőbb `rsize` és `wsize` értékeket. Az alapértelmezett beállítás nem biztos, hogy jó, előfordulhat ugyanis, hogy hálózati kártyánk a nagy adattömböket nem tudja kezelni. Régi alkatrészekkel vagy rendszermagokkal ez gyakran előfordul. Ha újabb kártyánk van, nagyobb érték beállításával próbálkozzunk. Esetenként a kapcsolat sebességét négy-ötszörösére növelhetjük, ezért jó, ha kísérletezünk.
A sebesség mérésének legegyszerűbb módja, ha létrehozunk egy fájlt a kiszolgáló lemezére és megmérjük a létrehozás idejét. Bármilyen adatot tartalmazhat a fájl, csak a kísérlet kedvéért hozzuk létre, így akár csupa nulla is lehet. Hozzuk létre a `dd` parancs segítségével, és a bemeneti fájl a */dev/zero* legyen. A parancs az időt is méri.
Mekkora legyen az ideiglenes fájlunk? Legyen kétszer vagy háromszor akkora, mint a kiszolgálógép memóriája. Ha a gépben 2 MB RAM van, 4 MB legyen. Előfordulhat, hogy – ha túl sok memória van a kiszolgálógépben – nem tudunk ekkora fájlt létrehozni, de 256 MB legtöbbször elég. A `df` paranccsal megnézhetjük, hogy mennyi szabad hely van a lemezen.
Az ügyfélgépen dolgozva a hálózati fájlrendszert újra az 1024-es `rsize` és `wsize` méretekkel fűzzük be. Futtassuk le a következő parancsot:

```
time dd if=/dev/zero of=/home/tempo/verbatim
➥bs=16k count=16384
```

A */dev/zero* fájlt használjuk bemenetként, így olvasása nem rontja el a kapott adatot. Parancsunk 16 384 darab 16 KB-os nulla bájttömböt küld, így 16 000×16 384/1024 kilobájt jön létre, ami 256 MB-nak felel meg. Jegyezzük meg az így kapott időt. Majd mérjük meg az olvasás és a */dev/null*-ra küldés idejét is:

```
time dd if=/home/tempo of=/dev/null bs=16k
```

Jegyezzük meg ezt az időt is, majd az ideiglenes fájlt letörölhetjük:

```
rm /home/tempo/ verbatim
```

Ezt ismételjük meg háromszor, majd számoljuk ki az átlagos olvasási idő/írási idő hányadost. Válasszuk le az NFS-fájlrendszert:

```
mount /home
```

Ezután fűzzük be újra, most 2048-as `rsize` és `wsize` értékekkel. Majd ismételjük meg a fenti eljárást egészen 32 768-as `rsize` és `wsize` értékekig.
Lássuk az eredményt! Minél gyorsabb az adatátvitel, annál kevesebb a mért idő. A legmegfelelőbb értékeket állítsuk be a */etc/fstab* fájlban, majd számításainkat tegyük biztonságos helyre.
Rövidebbé tehetjük az eljárást, ha újabb hálózati kártyánk és hármas kiadású NFS-rendszerünk van. Ez esetben induljunk 32 768-ról, és csökkentsük a számot. A kettes kiadással ne próbálkozzunk, ez ugyanis csak a régebbi rendszermagokkal, Solarisszal vagy SunOS-sel működik.
Úgy találtuk, hogy 2.2.19-es rendszermaggal a megfelelő érték 4096, ezért próbáljuk ki a 2048-as és 8192-es értékeket. Az újabb, 2.4.6-os rendszermag esetében ez az érték 8192 volt, de azt tapasztaltuk, hogy a régebbi rendszermag sokszor jobban teljesít. Igaz ez nagyban attól függ, hogy milyen alkatrészeket használunk. Mindenképpen vigyázzunk, ha NFS-kiszolgáló gépünkre új rendszermagot fordítunk.
TCP/IP segítségével kapcsolódó ügyfelek esetén a legjobb az 1024, vagy az annál is kisebb érték. Akkor is működik, ha az ügyfél és a kiszolgáló között útválasztó van.

Az NFS-kiadásokról néhány szóban

Miközben a `mount` végzi feladatát, az NFS ellenőrzi a protokollváltozatokat. Néha a következő hibaüzenet jelenik meg: „a kiszolgáló NFS kiadása régebbi, mint az ügyfélgépen lévő rendszermag által támogatott”.
A hálózati fájlmegosztás gondolata olyan régi, mint maga a hálózat, így több megoldás is kínálkozik. Az NFS teljesítményének javításához váltsunk a 2000-ben bemutatott 4-es kiadásra, vagy röviden NFSv4-re. Az NFS feltalálója, a Sun Microsystems támogatja a linuxos NFSv4 projektet. Megtartották az eddigi kiadások összes lehetőségét, de a biztonságot a megbízható RPC-folyamatok bevezetésével fokozták. A régebbi kiadások másik gyenge pontja az volt, hogy nem támogatták a soknyelvűséget. Az új kiadásban ezt is megoldották. Gyorstárak alkalmazásával a fájlműveletek sebességét is növelni tudták. A megfelelő sebesség eléréséért folytatott kísérletezést is elfelejthetjük, ugyanis az új kiadás magától megteszi. Nem bánjuk meg, ha erre állunk át, bár a linuxos NFSv3 néhány NFSv4-es lehetőséget is támogat.

Linux Journal 2002. január, 93. szám

Olexij Tykhomyrov
(tiger@ff.dsu.dp.ua) 1994 óta Linuxot használ. A Dnepropetrovski Nemzeti Egyetem Kísérleti Fizikai Tanszékén dolgozik, ahol fizikát és kommunikációt tanít. Rajong a fiáért, Misáért, aki Tigrisnek szólítja, mert a diákok állítólag félnek tőle. Tigris szeret úszni és utazni.

Denis Tonkonog,
Tigris volt tanítványa ugyanott dolgozik, és szintén kedveli az utazást. Hobbija a puskával halászás. Barátai Fekete Macskának szólítják, nem tudni, miért. E-mailt a következő címen vár: denis@ff.dsu.dp.ua.